INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 0000; **00**:1–20
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme

A fast method for solving binary programming problems using first order derivatives, with specific application to topology optimization with buckling constraints

P. A. Browne[1*], Prof. C. Budd[1], Prof. N. I. M. Gould[2],
Dr H. A. Kim[3] and Dr J. A. Scott[2]

[1]*Department of Mathematical Sciences, University of Bath, UK*
[2]*Numerical Analysis Group, Rutherford Appleton Laboratory, STFC, UK*
[3]*Department of Mechanical Engineering, University of Bath, UK*

SUMMARY

We present a method for finding solutions of large-scale binary programming problems where the calculation of derivatives is very expensive. We then apply this method to a topology optimization problem of minimisation of weight subject to compliance and buckling constraints. We derive an analytic expression for the derivative of the stress stiffness matrix with respect to the density of an element in the finite-element setting. Results are presented for a number of two-dimensional test problems. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Topology optimization, buckling, eigenvalue, derivative, structural optimization

1. INTRODUCTION

Topology optimization can be thought of in many ways. To an engineer, it may be thought of as placing material somewhere in a design space in order to attain an optimal value of a function that is important to the system. Mathematically it can be posed as finding an optimal domain on which a function of the solution of the underlying PDE is minimal.

We are interested in a conceptually straightforward problem: to minimize the weight of an elastic structure. However, we also want to maintain the integrity of the structure, and we do this by prescribing two constraints. The first ensures that the structure has a prescribed level of stiffness, and the second that the structure is not prone to buckling. The purpose of this paper is to present an algorithm that can provide a solution for problems such as these in a reasonable computing time.

To formulate the problem mathematically, the design space (region in which material is placed) is discretised using finite elements. Our goal is to determine which elements should contain material

*Correspondence to: Department of Mathematical Sciences, University of Bath, Bath, UK. E-mail: P.A.Browne@bath.ac.uk

and which should be void of material. If we associate a value of 1 to an element with material and a value of 0 to an element containing no material, this topology optimization problem is a binary programming problem.

Finding a global solution to binary programming problems is notoriously difficult. The methods for finding such minima can be broadly put into three categories: implicit enumeration, branch-and-bound and cutting plane methods. The most popular implementations involve hybrids of branch-and-bound and cutting plane methods. For a comprehensive description of these binary programming methods see, for example, Wolsey [1]. These methods were popular for structural optimization from the late 1960s through to the early 1990s. In 1994, Arora & Huang [2] reviewed the methods for solving structural optimization problems discretely.

In 1968, Toakley [3] applied a combination of cutting-plane methods and branch-and-bound to solve truss optimization problems. Using what is now known as the branch-and-cut method, this method was resurged in 2010 by Stolpe and Bendsøe [4] to find the global solution to a minimisation of compliance problem, subject to a constraint on the volume of the structure.

In 1980, Farkas and Szabo [5] applied an implicit enumeration technique to the design of beams and frames. Branch-and-bound methods have been used by, amongst others, John et al. [6], Sandgren [7, 8] and Salajegheh & Vanderplaats [9] for structural optimization problems.

In the latest of these papers, the number of variables in the considered problem was 100. Whilst these methods do find global minima of the problems, they suffer from exponential growth in the computation time as the number of variables increases.

Beckers 2000 [10] uses a dual method to find discrete solutions to structural optimization problems. In 2003, Stolpe and Svanberg [11] formulate a topology optimization problem as a mixed 0-1 program and solve using branch-and-bound methods.

Achtizger and Stolpe [12, 13, 14] have studied in detail the topology optimization of truss structures using branch and bound methods, and have been able to find global solutions to problems with over 700 bars in the ground structure.

To avoid the computational issues associated with binary programming, the traditional approach to topology optimization has been to relax the binary constraint and to look for a solution that varies continuously in $\mathbb{R}^n$. This is known as continuous relaxation of the problem. Physically, this relaxed variable can correspond to the density of the material in the element or the thickness of the material in the element (if the problem is $2-$dimensional). Nested analysis and design is then performed, meaning that the structural analysis of the current structure is carried out and appropriate derivatives calculated. These values are then fed to an optimization routine that updates the structure. The analysis is performed again and this process iterates until an optima is attained.

By far the most popular optimization method to update the structure is the Method of Moving Asymptotes (MMA) [15]. To function, MMA needs only the function values and values of the derivatives at that point. There are many examples of MMA being very efficient at solving topology optimization problems with different objectives/constraints.

The buckling load of a structure is found as the solution to an eigenvalue problem and a structure is said to buckle at the lowest positive eigenvalue, referred to as the critical load. Derivatives of this critical load are only well defined if there is only one modeshape corresponding to the critical load. Mathematically, this means we have a simple eigenvalue. Hence a direct bound on the critical load cannot be used as a constraint with MMA as the derivative is not well defined.

DOI: 10.1002/nme

Semidefinite programming methods have been developed specifically to deal with such eventualities. Kocvara [16], and in conjunction with Stingl [17], has applied such methods to topology optimization problems. More recently, along with Bogani [18], they have applied an adapted version of their semidefinite codes to find noninteger solutions to buckling problems. This made use of a reformulation of a semidefinite constraint using the indefinite Cholesky factorisation of the matrix, and solving a resulting nonlinear programming problem with an adapted version of MMA.

When a continuous relaxation approach is used in problems involving calculating the buckling modes (or harmonic modes) of a structure, unwanted numerical effects are introduced. Tenek and Hagiwara [19], Pedersen [20] and Neves et al. [21] all noted that spurious buckling (or harmonic) modes would be computed in which the buckling is confined to regions where the density of material is less than 10%. Whilst their proposed solution of having no stress stiffness (or mass in the harmonic analysis case) contributions from these elements can eradicate these spurious modes, this is not consistent with the underlying model of the structure. Indeed, if one were to consider a structure where a small fraction (less than 10%) of material was equidistributed throughout the design domain, the stress stiffness matrix would be the zero matrix, and as a result the critical load of the structure would be computed as infinite.

In this paper we introduce an efficient method for binary programming that is able to find a local minima of the topology optimization problem with buckling constraints. In doing so, we avoid the problem of spurious buckling modes and can find solutions to large two-dimensional problems ($\mathcal{O}(10^5)$ variables). However, the method does not guarantee the computed solution is a global minimiser.

Due to the dimensionality of the problems we wish to solve, and the complexity of derivative-free methods for binary programs, we will use derivative information to reduce this complexity. The efficiency of topology optimization methods involving a buckling constraint is severely hindered by the calculation of the derivatives of the buckling constraint. This calculation typically takes an order of magnitude more time than the structural analysis. With this in mind, the binary descent method we introduce will try to reduce the number of derivative calculations made to improve efficiency.

The remainder of this paper is organised as follows. In Section 2 we formulate the topology optimization problem to include the buckling constraint. Section 3 motivates and states the new method which we use to solve the optimization problem. Section 4 then contains implementation details and results for a number of two-dimensional test problems. Finally in section 5 we draw conclusions about the proposed algorithm.

2. FORMULATION OF TOPOLOGY OPTIMIZATION TO INCLUDE BUCKLING CONSTRAINTS

Let Ω be the design domain containing the elastic structure that we discretise using a finite-element mesh $\mathcal{T}$. We apply a load f to Ω, and this induces displacements u, which are the solution to the equilibrium equations of linear elasticity

$$Ku = f \tag{1}$$

DOI: 10.1002/nme

where K is the finite element stiffness matrix. To prescribe stiffness of the system, we bound a quantity known as compliance. Compliance is defined as the product $f^T u$, which is a measure of external work done on the structure. We want to give this an upper bound $c_{\max}$ so that our structure remains stiff.

The minimisation of weight subject to a compliance constraint is a well-studied problem, see for example [22]. However, it has long been observed that structures optimized for minimum weight or compliance are prone to buckling [23].

The critical load of a structure is defined by the smallest positive value of λ corresponding to a nonzero eigenvector v for which

$$(K + \lambda K_\sigma)v = 0. \tag{2}$$

In this equation, K is the symmetric finite-element stiffness matrix and K_σ is the symmetric stress stiffness matrix. (λ, v) is an eigenpair of the generalised eigenvalue problem (2). λ is refered to as the eigenvalue and $v \neq 0$ the corresponding eigenvector (or modeshape). The critical load is then λ times the applied load f. Given a safety factor parameter $c_s > 0$, a bound of the form $\lambda \geq c_s$ is equivalent to the semidefinite constraint

$$K + c_s K_\sigma \succeq 0.$$

This means that all the eigenvalues of the system $(K + c_s K_\sigma)$ are non-negative. This happens only if $\sum_{i=1}^{M} v_i^T (K + c_s K_\sigma) v_i \geq 0$ where v_i are the M buckling modes that solve $(K + \lambda K_\sigma)v_i = 0$. If we let $x \in \{0,1\}^n$ represent the density of material in each of the elements of our mesh, with $x_i = 0$ corresponding to an absence of material in element i and $x_j = 1$ corresponding to element j being filled with material, the problem we wish to solve becomes:

$$\min_x \sum x_i \tag{3a}$$

$$\text{subject to} \qquad c_1(x) := c_{\max} - f^T u(x) \geq 0 \tag{3b}$$

$$c_2(x) := \sum_{i=1}^{M} v_i(x)^T (K(x) + c_s K_\sigma(x)) v_i(x) \geq 0 \tag{3c}$$

$$x \in \{0,1\}^n \tag{3d}$$

$$K(x)u(x) = f \tag{3e}$$

$$[K(x) + \lambda(x) K_\sigma(x)]v(x) = 0. \tag{3f}$$

2.1. Derivative calculations

To use the binary descent method that we will explain in Section 3 we need an efficient way of calculating the derivative of the constraints with respect to the variables x_i. As we will see later in the results section, the computation of derivatives of the buckling constraint (3c) is the bottleneck of our optimization algorithm, so it is imperitive that we have an analytic expression for this. To calculate the derivatives, we relax the binary constraints on our variables and assume the following holds

DOI: 10.1002/nme

$$K(x) = \sum_{\ell} x_\ell K_\ell,$$

where K_ℓ is the local element stiffness matrix. The derivative of this with respect to the density of an element x_i is given by

$$\frac{\partial K}{\partial x_i}(x) = K_i.$$

Calculating the derivative of the buckling constraint requires the derivation of an expression for $\frac{\partial K_\sigma}{\partial x_i}$. This quantity is nontrivial to compute, unlike the derivative of a mass matrix which would be in place of the stress stiffness matrix in structural optimization involving harmonic modes. The stress field σ_ℓ on an element ℓ is a 3×3 tensor with 6 degrees of freedom. This can be written in three dimensions as

$$\sigma_\ell = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{13} \\ \sigma_{23} \end{bmatrix}_\ell = x_\ell E_\ell B_\ell u,$$

which in two dimensions reduces to

$$\sigma_\ell = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix}_\ell = x_\ell E_\ell B_\ell u,$$

where u are the nodal displacements of the element, E_ℓ is a constant matrix of material properties and B_ℓ contains geometric information about the element. The indices $1, 2$ and 3 refer to the coordinate directions of the system.

We consider the two dimensional case, and note that all the following steps have a direct analogue in three dimensions. We write the stress stiffness matrix given in (2) as follows.

$$K_\sigma = \sum_{\ell=1}^{n} \int G_\ell^T \begin{bmatrix} \sigma_{11} & \sigma_{12} & 0 & 0 \\ \sigma_{12} & \sigma_{22} & 0 & 0 \\ 0 & 0 & \sigma_{11} & \sigma_{12} \\ 0 & 0 & \sigma_{12} & \sigma_{22} \end{bmatrix}_\ell G_\ell \mathrm{d}V_\ell, \tag{4}$$

where G_ℓ is a matrix containing derivatives of the basis functions that relates the displacements of an element ℓ to the nodal degrees of freedom [24] and n is the total number of elements in the finite-element mesh $\mathcal{T}$. Now define a map $\Theta : \mathbb{R}^3 \mapsto \mathbb{R}^{4\times 4}$ by

$$\Theta(\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}) := \begin{bmatrix} \alpha & \gamma & 0 & 0 \\ \gamma & \beta & 0 & 0 \\ 0 & 0 & \alpha & \gamma \\ 0 & 0 & \gamma & \beta \end{bmatrix}.$$

DOI: 10.1002/nme

Note that Θ is a linear operator. Using this, (4) becomes

$$\begin{aligned} K_\sigma &= \sum_{\ell=1}^{n} \int G_\ell^T \Theta(x_\ell E_\ell B_\ell u) G_\ell \, \mathrm{d}V_\ell \\ &= \sum_{\ell=1}^{n} \int G_\ell(\xi)^T \Theta(x_\ell E_\ell B_\ell(\xi) u) G_\ell(\xi) \, \mathrm{d}V_\ell \\ &\approx \sum_{\ell=1}^{n} \sum_{j} \omega_j G_\ell(\xi_j)^T \Theta(x_\ell E_\ell B_\ell(\xi_j) u) G_\ell(\xi_j) \end{aligned} \tag{5}$$

where ω_j are the weights associated with the appropriate Gauss points ξ_j that implement a chosen quadrature rule to approximate the integral. Differentiating the equilibrium equation (1) with respect to the density x_i yields

$$\frac{\partial K}{\partial x_i} u + K \frac{\partial u}{\partial x_i} = 0$$

and hence

$$\frac{\partial u}{\partial x_i} = -K^{-1} \frac{\partial K}{\partial x_i} u.$$

Now consider the derivative of the operator Θ with respect to x_i. Since Θ is linear

$$\begin{aligned} \frac{\partial \Theta(x_\ell E_\ell B_\ell u)}{\partial x_i} &= \Theta\Big(\frac{\partial}{\partial x_i} x_\ell E_\ell B_\ell u(x_i) \Big) \\ &= \Theta\Big(\delta_{i\ell} E_\ell B_\ell(\xi_j) u + x_\ell E_\ell B_\ell(\xi_j) \frac{\partial u}{\partial x_i} \Big) \end{aligned}$$

where $\delta_{i\ell}$ is the Kronecker Delta.

Applying the chain rule to (5) we obtain

$$\begin{aligned} \frac{\partial K_\sigma}{\partial x_i} &\approx \sum_{l=1}^{n} \sum_{j} \omega_j G_\ell(\xi_j)^T \frac{\partial \Theta(x_\ell E_\ell B_\ell(\xi_j) u)}{\partial x_i} G_\ell(\xi_j) \\ \frac{\partial K_\sigma}{\partial x_i} &\approx \sum_{l=1}^{n} \sum_{j} \omega_j G_\ell(\xi_j)^T \Theta(\delta_{i\ell} E_\ell B_\ell(\xi_j) u - x_\ell E_\ell B_\ell(\xi_j) K^{-1} \frac{\partial K}{\partial x_i} u) G_\ell(\xi_j), \end{aligned} \tag{6}$$

where the approximation is due to the error in the quadrature rule used. This matrix can now be used to find the derivative of the buckling constraint which we require. For each variable $x_i = 1, \ldots, n$, (6) must be computed. As (6) contains a sum over $i = 1, \ldots, n$ one can see that computing $\frac{\partial K_\sigma}{\partial x_i}$ has computational complexity of $\mathcal{O}(n)$ for each i and hence computing (6) for all variables has complexity of $\mathcal{O}(n^2)$.

3. BINARY DESCENT METHOD

In this section, we motivate and describe the new method which we propose for solving the binary programming problem. If we solve the state equations (3e) and (3f) then problem (3) takes the

DOI: 10.1002/nme

general form

$$\min_x e^T x \tag{7a}$$

$$\text{subject to} \quad c(x) \geq 0 \tag{7b}$$

$$x \in \{0,1\} \tag{7c}$$

with $x \in \mathbb{R}^n$, $c \in \mathbb{R}^m$ and $e = [1,1,\ldots,1]^T \in \mathbb{R}^n$ and note that problem (3) is of this form. Typically m will be small (less than 10) and $m << n$. We also assume that $x^0 = e$ is an initial feasible point of (7). From now on we let k denote the current iteration, and x^k the value x on the k-th iteration.

The objective function $e^T x$ is a purely linear function of x this can be optimized by successively reducing the number of nonzero terms in x and we need not worry about errors in approximating this. However, the constraints are nonlinear functions of x and ensuring that (7b) holds is difficult. Accordingly, we now describe how a careful linearisation of the constraint equations can lead to a feasible algorithm. Taylor's theorem can then be used to approximate $c(x^k)$

$$c(x^{k+1}) = c(x^k) + \sum_{i=1}^{n} \frac{\partial c(x^k)}{\partial x_i}(x_i^{k+1} - x_i^k) + \text{ higher order terms}$$

where $\frac{\partial c(x^k)}{\partial x_i}$ is determined using the explicit derivative results of the previous section. Our method will take discrete steps so that

$$x_i^{k+1} - x_i^k \in \{-1,0,1\} \qquad \forall i = 1,\ldots,n,$$

and so we must assume that the higher order terms will be small, but later we will introduce a strategy to cope with when they are not.

If we now consider variables x_i^k such that $x_i^k = 1$ which we wish to change to $x_i^{k+1} = 0$. Then $x_i^{k+1} - x_i^k = -1$ and so for the difference in the linearised constraint functions

$$c(x^{k+1}) - c(x^k) = \sum_{i=1}^{n} \frac{\partial c(x^k)}{\partial x_i}(x_i^{k+1} - x_i^k)$$

to be minimized, we want all the terms of $\frac{\partial c(x^k)}{\partial x_i}$ to be as small as possible. However, we have multiple constraints, so the variables for which the gradient of one constraint is small may have a large gradient for another constraint.

Assume we are at a feasible point so that $c(x^k) > 0$. Ignoring the higher order terms, we can write

$$c(x^{k+1}) = c(x^k) + \sum_{i=1}^{n} \frac{\partial c(x^k)}{\partial x_i}(x_i^{k+1} - x_i^k). \tag{8}$$

DOI: 10.1002/nme

We have to ensure $c(x^{k+1}) > 0$, so

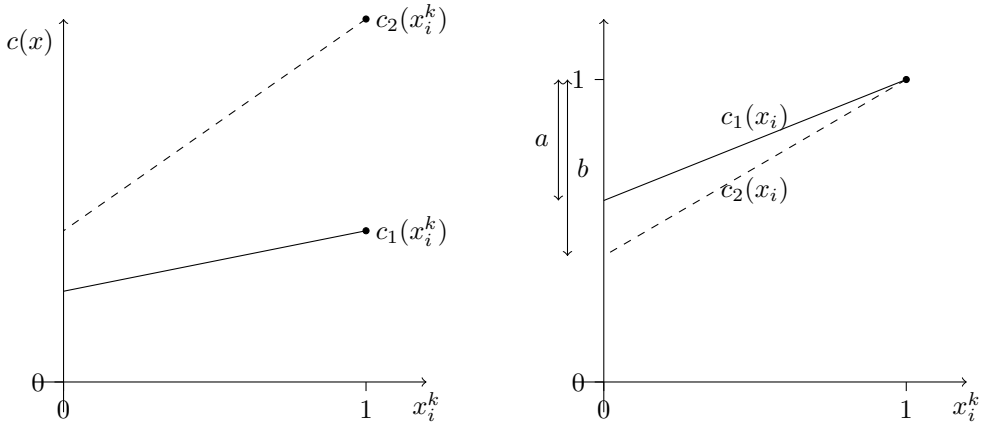
$$c(x^k) + \sum_{i=1}^{n} \frac{\partial c(x^k)}{\partial x_i}(x_i^{k+1} - x_i^k) > 0$$
$$\Leftrightarrow \qquad 1 + \sum_{i=1}^{n} \frac{\partial c_j^T(x^k)}{\partial x_i}/c_j(x^k)(x_i^{k+1} - x_i^k) > 0 \qquad \forall j = 1, \ldots, m.$$

If $x_i^{k+1} \neq x_i^k$ then each normalised constraint $c_j(x^k)$ is changed by $\pm\frac{\partial c_j(x^k)}{\partial x_i}/c_j(x^k)$.

If we define a sensitivity for each variable i as

$$s_i(x^k) = \max_{j=1,\ldots,m} \frac{\partial c_j(x^k)}{\partial x_i}/c_j(x^k) \tag{9}$$

this gives us a quantity for each variable which is the most conservative estimate of how the constraints will vary if we change the value of the variable. In one variable, this has the form shown in Figure 1. Figure 1a shows the absolute values of the linear approximations to the constraints based on their values and corresponding derivatives. Figure 1b shows the calculation that we make based on normalising these approximations to compute which of the constraints would decrease the most if the variable x_i^k were changed.



(a) Linear approximations to the constraints $c(x_i^k)$ in the case where $m = 2$.

(b) Sensitivity calculation in one variable. Here $s_i(x^k) = \max\{a, b\} = b$.

Figure 1. Sensitivity calculation in one variable for the case when $m = 2$.

This sensitivity measure also provides an ordering so that if we choose to update variables in increasing order of this quantity, the change in the constraint values are minimised. Now for ease of notation, let us assume that the variables are ordered so that

$$s_1 \leq s_2 \leq \ldots \leq s_p \qquad \forall s_i \text{ s.t. } x_1^k, x_2^k, \ldots, x_p^k = 1$$
$$s_p \geq s_{p+1} \geq \ldots \geq s_n \qquad \forall s_i \text{ s.t. } x_n^k, x_{n-1}^k, \ldots, x_{p+1}^k = 0$$

DOI: 10.1002/nme

To be cautious, instead of requiring $c(x^{k+1}) \geq 0$, we allow for the effects of the nonlinear terms and so would content if instead $c(x^{k+1}) \geq (1-\alpha)c(x^k)$. This implies that

$$c(x^k) + \sum_{i=1}^{n} \frac{\partial c^T(x^k)}{\partial x_i}(x_i^{k+1} - x_i^k) \geq (1-\alpha)c(x^k),$$

i.e. we require

$$\alpha c(x^k) + \sum_{i=1}^{n} \frac{\partial c^T(x^k)}{\partial x_i}(x_i^{k+1} - x_i^k) \geq 0.$$

To update the current solution we can find

$$L := \max \ell \text{ s.t. } \alpha c(x^k) - \sum_{\substack{j=1 \\ x_j^k=1}}^{\ell} \frac{\partial c(x^k)}{\partial x_j} > 0 \tag{10}$$

where the lower summation condition $x_j^k = 1$ means we only consider those gradients corresponding to variables with the value 1. Then we change from 1 to 0 those variables $x_1^k, \ldots, x_L^k$ so as to reduce the objective function by a value of L.

However, there is the possibility that changing variables from $0 \mapsto 1$ could allow us to further reduce the objective function by changing yet more variables from $1 \mapsto 0$. We test for this situation by finding (or attempting to find) $J > 0$ such that

$$J := \max j \text{ s.t. } \sum_{\substack{i=1 \\ x_i^k=0}}^{j} \frac{\partial c(x^k)}{\partial x_i} - \sum_{\substack{i=1 \\ x_i^k=1}}^{2j} \frac{\partial c(x^k)}{\partial x_{L+i}} \geq 0 \tag{11}$$

So we can change variables the variables corresponding to the terms in the first sum from $0 \mapsto 1$ but we can then find more variables to change from $1 \mapsto 0$, corresponding to the terms in the second summation, so that the objective function improves whilst remaining a feasible solution.

The coefficient α is a measure of how well the linear gradient information is predicting the change in the constraints as we go from one iteration to another. If the problem becomes infeasible, then we know we have taken too big a step, so we must reduce α in order to take a smaller step. However, recall the goal of this method is to compute the gradients as few times as possible, and so we wish to take steps which are as large as possible. If the step has been accepted for the previous 2 iterations without reducing α then we choose to increase α in order to attempt to take larger steps and thus speed up the algorithm.

Note that if α is too large and the step we take becomes infeasible then we can reduce α and try a smaller step all without recomputing the derivatives. Hence increasing α by too much would not be too detrimental to the performance of the algorithm. After experimenting with different values to increase and decrease α in these cases, we have chosen to reset α to 0.7α when the problem becomes infeasible and we reset α to 1.5α when we want to increase it. These values appear stable and give good performance for most problems.

Prepared using nmeauth.cls
DOI: 10.1002/nme

To ensure that we try and update at least one variable, α must be larger than a critical value α_c given by

$$\alpha_c = \max_{j=1,\ldots,m} \{(\frac{\partial c_j(x^k)}{\partial x_1^k})/c_j(x^k)\}.$$

This guarantees that $L \geq 1$ and so we update at least 1 variable.

If we cannot make any further progress with this algorithm, we stop. When we have nonlinear constraints we cannot say for certain that we are at an optimal point, however making further progress would be far too expensive as we would have to switch to a different integer programming strategy and the dimension of the problems that we wish to consider prohibits this.

we now present the algorithm:

Algorithm 1 Fast binary descent method

Initialise x^0 and α.
Compute objective function (7a) and constraints (7b)
if x^0 not feasible **then**
  Stop
else
  Compute derivatives $\frac{\partial c(x^k)}{\partial x_i}$
  Sort s_i (9)
  Compute values L (10) and J (11)
  Update, based on L and J, the variables x_i^k
  if no variables updated **then**
    {We cannot change any variables}
    return with optimal solution
  end if
  Compute objective function and constraints.
  if not feasible **then**
    {Reject update step}
    Reduce α.
    GO TO 7
  else
    {Accept update step}
    Increase α if desired
    $k = k + 1$
    GO TO 5
  end if
end if

4. IMPLEMENTATION AND RESULTS

In our experiments, we consider optimising an isotropic material with Young's modulus 1.0 and Poisson ratio 0.3. We discretise the design spaces using square bilinear elements on a uniform mesh.

DOI: 10.1002/nme

Algorithm 1 has been implemented in `Fortran90` using the HSL mathematical software library [25] and applied to a series of two-dimensional structural problems. The linear solve for the calculation of displacements (1) used HSL_MA87 [26], a DAG based direct solver designed for shared memory systems. For the size of problems considered, HSL_MA87 has been found to be very efficient and stable. The first 6 buckling modes of the system (2) were computed as these were sufficient to ensure we found all corresponding eigenvectors of the critical load. These eigenpairs were calculated using HSL_EA19 [27], a subspace iteration code, preconditioned by the Cholesky factorisation already computed by HSL_MA87. The sensitivities were ordered using HSL_KB22, a heapsort algorithm.

The codes were executed on a desktop with an Intel® Core$^{\text{TM}}$2 Duo CPU E8300 @ 2.83Ghz with 2GB RAM running a 32-bit Linux OS and were compiled with the `gfortran` compiler in double precision.

4.1. Short cantilevered beam

We consider a clamped beam with a vertical external force applied to the free side as shown in Figure 2.

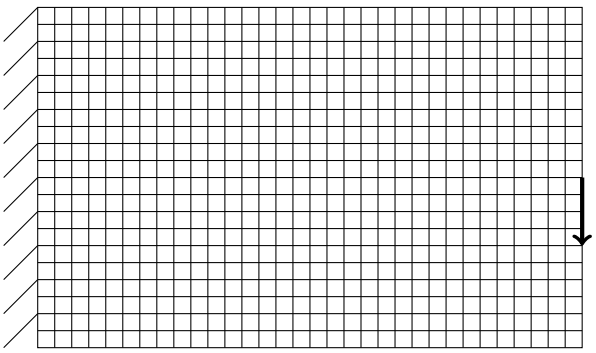


Figure 2. Design domain of a centrally loaded cantilevered beam. Here we have a width to height ratio of $8:5$ and a unit load acting vertically from the centre of the right hand side of the domain.

Figure 3. Solution found on mesh of 80×50 elements. The buckling constraint is set to $c_s = 0.9$ and the compliance constraint $c_{\max} = 35$. A volume of 0.6255 is attained. Similarly to Figure 4 the buckling constraint c_2 is active and the compliance constraint c_1 is not.

DOI: 10.1002/nme

Figure 3 is the computed solution to a problem that is allowed to be reasonably flexible (the compliance constraint $c_1(x^0)$ is large initially) but the buckling constraint is reasonably tight ($c_2(x^0)$ is small initially). We see that the method has produced a structure with 4 bars under compression and only 3 bars under tension.

Figure 4. Solution found on mesh of 80×50 elements. The buckling constraint is set to $c_s = 0.9$ and the compliance constraint $c_{\max} = 60$. A volume of 0.5535 is attained. Here the buckling constraint c_2 is active and the compliance constraint c_1 is not.

Figure 4 is the computed solution to a problem with the same buckling constraint as in Figure 3 but is allowed to be more flexible (the compliance constraint is not as restrictive). There is a clear asymmetry in the computed solution in which the lower bar is much thicker than the upper bar. This lower bar is under compression with this loading, and hence would be prone to buckling. The method has automatically designed the structure with more material in this compressed bar in order to keep the buckling load of the structure above the constraint.

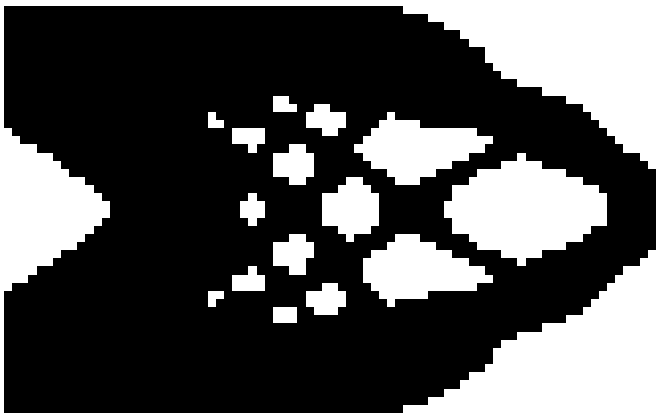


Figure 5. Solution found on mesh of 80×50 elements. The buckling constraint is set to $c_s = 0.1$ and the compliance constraint $c_{\max} = 30$. A volume of 0.692 is attained. Here the compliance constraint c_1 is active and the buckling constraint c_2 is not.

Figure 5 has a very small buckling constraint but a very tight compliance constraint ($c_1(x^0)$ is close to 0). The method has computed a solution that only has the compliance constraint active. In this calculation, the compliance constraint has been the major influencing factor and hence

DOI: 10.1002/nme

the computed solution is much more symmetrical than one when the buckling constraint is more prominent.

Figures 3 to 5 refer to the solutions found with the same design domain and material properties but with different values for the buckling and compliance constraints. We can see that there is a clear difference in the topology of the resulting solution depending on these qualities. We now display some of the history of the algorithm when applied to the problem solved in Figure 3 where $c_{\max} = 35$ and $c_s = 0.9$.

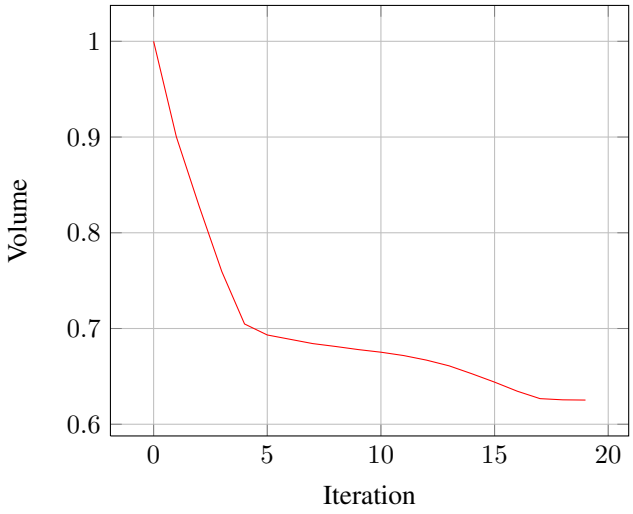


Figure 6. Volume - iterations of Binary method

Figure 6 is monotonically decreasing and so shows that the method as described in Section 3 is indeed a descent method. Note that in the initial stages of the computation large steps are made and this varies as the compuation progresses.

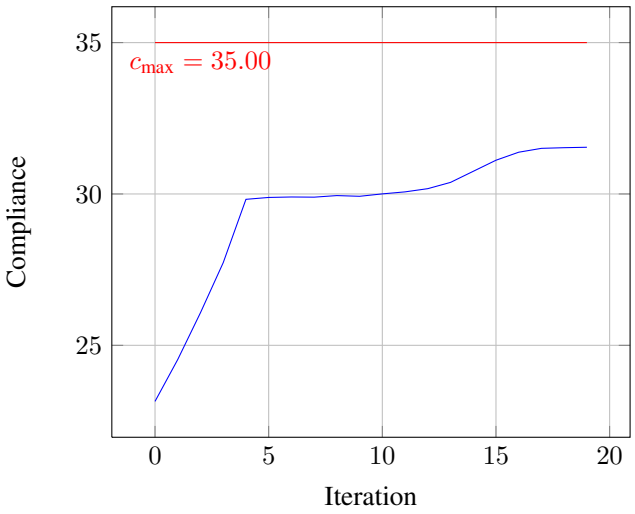


Figure 7. Compliance - iterations of the Binary method

DOI: 10.1002/nme

Figure 7 shows how the compliance constraint is inactive at the solution of this problem. Note that at all points the compliance of the structure is below the maximum compliance $c_{\max}$ and so the solution is feasible at all points with respect to c_1.

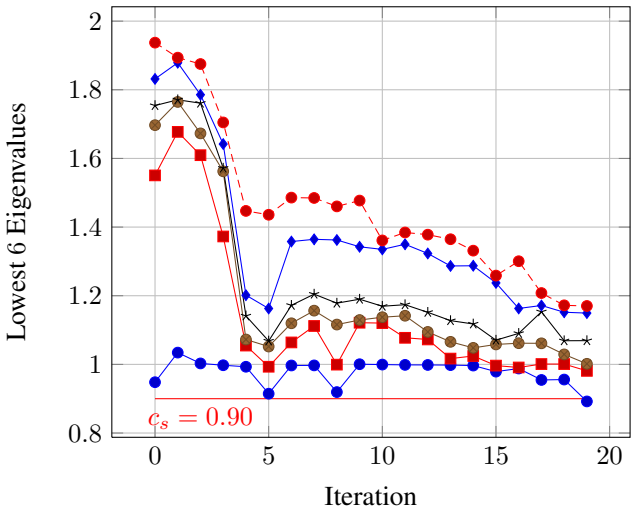


Figure 8. Eigenvalues - iterations of the Binary method

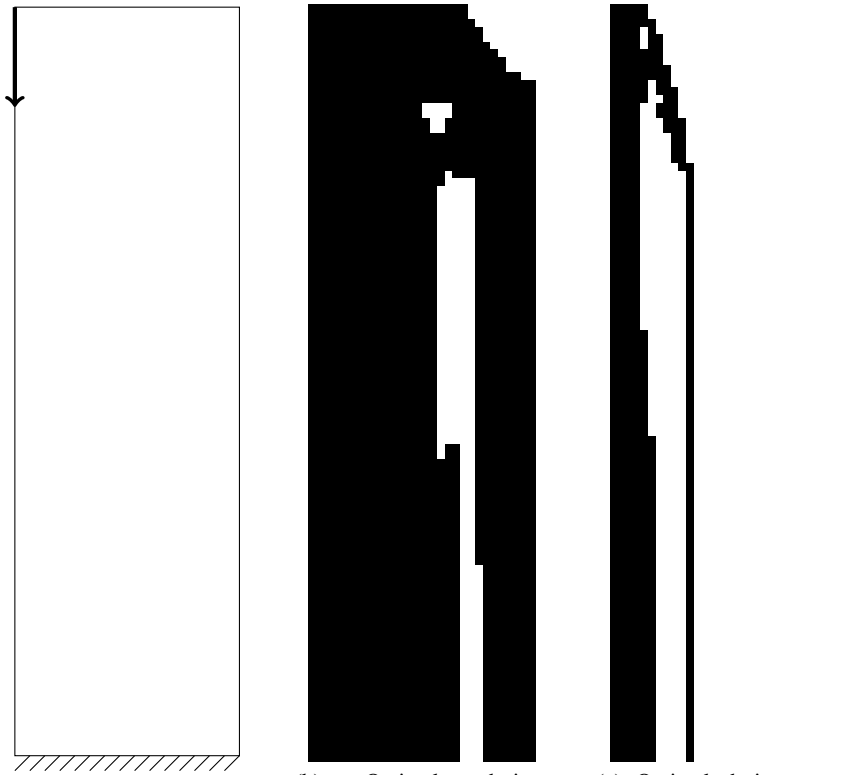
Figure 8 shows the lowest 6 eigenvalues of the system as binary descent method progresses. We see that on the 20-th iteration the lowest eigenvalue is below the constraint c_s and so the computed solution is at iteration 19. The nonlinearity in $c_2(x)$ is clear to see from Figure 8 as we can see that it is not behaving monotonically. When viewed in combination with Figure 7 we see that for the history of the algorithm the solutions are all feasible.

4.2. Side loaded column

In this section we consider a tall design space fixed completely at the bottom carrying a vertical load applied at the top corner of the design space. This type of loading would typically be representative of that experienced by a cross section of a wall supporting a roof. The design space is shown in Figure 9a and the computed solutions to this problem with differing constraints are shown in Figures 9b and 9c.

We can see from Figure 9b because of the compliance constraint we have a thick structure vertically underneath the applied load which carries the force directly to the ground. The buckling constraint has introduced the slender support in the opposite side of the design domain to the load akin to a buttress. We can see that this is carrying comparatively less load but is helping the main support to resist buckling. In Figure 9c as the constraints are relaxed compared with the problem in Figure 9b, the computed solution has a significantly lower objective function, but we still see the same thick support directly below the load and the more slender support to the side to help resist buckling.

Prepared using nmeauth.cls
DOI: 10.1002/nme



(a) Design domain with width to height ratio $3:10$.

(b) Optimal design on a 30×100 mesh with $c_s = 0.225$ and $c_{\max} = 22.5$. Here c_2 is active and c_1 is not.

(c) Optimal design on a 30×100 mesh with $c_s = 0.001$ and $c_{\max} = 60$. Here c_1 is active and c_2 is not.

Figure 9. A column loaded at the side

4.3. Centrally loaded column

As shown in Figure 10 we have a square design domain. The loading is vertically downwards at the top of the design domain and the base is fixed completely.

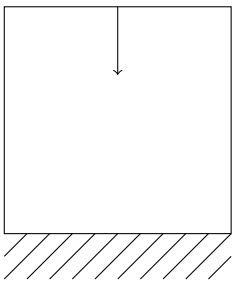


Figure 10. Design domain of model column problem. This is a square domain of side length 1 with a unit load acting vertically on the space at the midpoint of the upper boundary of the space.

Figure 11 through to Figure 14 show results of this problem for a specific mesh size with varying values of the different constraints. Figure 16 is the resulting solution when solved on the mesh with 200 elements.

Prepared using nmeauth.cls
DOI: 10.1002/nme

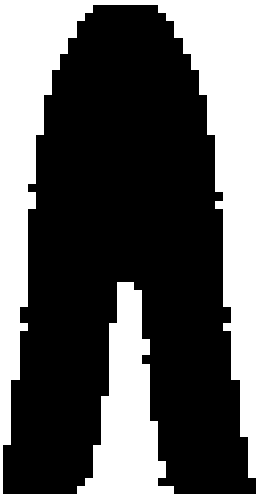


Figure 11. Solution found on mesh of 60×60 elements. The buckling constraint is set to $c_s = 0.5$ and the compliance constraint $c_{\max} = 5$. Here, the compliance constraint it active and the buckling constraint is inactive.

Figure 12. Solution found on mesh of 60×60 elements. The buckling constraint is set to $c_s = 0.5$ and the compliance constraint $c_{\max} = 5.5$. In this case, compared with Figure 11, the higher compliance constraint has led to a solution where this constraint is inactive and the buckling constraint is now active.

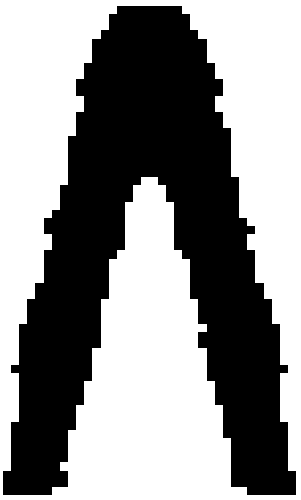


Figure 13. Solution found on mesh of 60×60 elements. The buckling constraint is set to $c_s = 0.4$ and the compliance constraint $c_{\max} = 8$. A volume of 0.276 is attained.

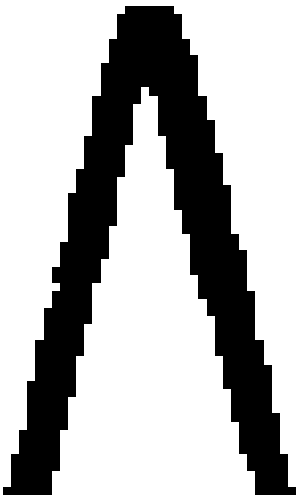


Figure 14. Solution found on mesh of 60×60 elements. The buckling constraint is set to $c_s = 0.1$ and the compliance constraint $c_{\max} = 8$. A volume of 0.183 is attained.

In all the solutions to this test problem shown in Figures 11 to 16 we can see that the symmetry of the problem is not present in the computed solution. As Rozvany [28] has shown, we do not necessarily expect the optimal solution to these binary programming problems to be symmetric. The asymmetry in the computed solutions shown in Figures 11 to 16 arise from (10) and (11). Only a subset of elements with precisely the same sensitivity values may be chosen to be updated and so the symmetry may be lost.

The first thing to note about the results is the problem size which the fast binary method has been able to solve in reasonable time. A computation on a two-dimensional mesh of 3×10^4 elements

DOI: 10.1002/nme

Problem size n	Objective	Derivative calculations	Analyses	Time (mins) to 3 s.f.	Proportion on ∇c_2
$30 \times 30 = 900$	0.266	11	26	0.421	0.623
$40 \times 40 = 1600$	0.229	12	22	1.10	0.782
$50 \times 50 = 2500$	0.213	11	21	2.29	0.857
$60 \times 60 = 3600$	0.183	26	31	6.73	0.901
$70 \times 70 = 4900$	0.187	24	28	11.6	0.931
$80 \times 80 = 6400$	0.185	21	24	18.1	0.948
$90 \times 90 = 8100$	0.184	20	22	28.5	0.948
$100 \times 100 = 10000$	0.184	18	23	40.6	0.966
$110 \times 110 = 12100$	0.188	19	21	61.2	0.973
$120 \times 120 = 14400$	0.187	18	20	84.5	0.978
$130 \times 130 = 16900$	0.184	19	23	119.	0.980
$140 \times 140 = 19600$	0.188	17	18	154.	0.984
$175 \times 175 = 30625$	0.173	20	22	386.	0.985
$180 \times 180 = 32400$	0.191	20	23	458.	0.989
$200 \times 200 = 40000$	0.188	21	24	734.	0.990
$317 \times 317 = 100489$	0.181	19	20	4229	0.996

Table I. Table of results for the centrally loaded column

in less than 8 hours on a modest desktop is a speed which would be useful to a structural engineer. This speed is attained because the number of derivative calculations appears to not be dependent on the number of variables. However, if we look at the final column, we see that the vast majority of the work is spent calculating the derivative of the stability constraint. However, we note that it is possible to parallelise this step. A massively parallel implementation should achieve near optimal speedup as no information transfer is required for the calculation of the derivative with respect to the individual variables. A solution to a problem with 10^5 variables was found using the algorithm described within 3 days which shows the $\mathcal{O}(n^2)$ behaviour and that larger problems can be solved if the user's time constraints allow.

DOI: 10.1002/nme

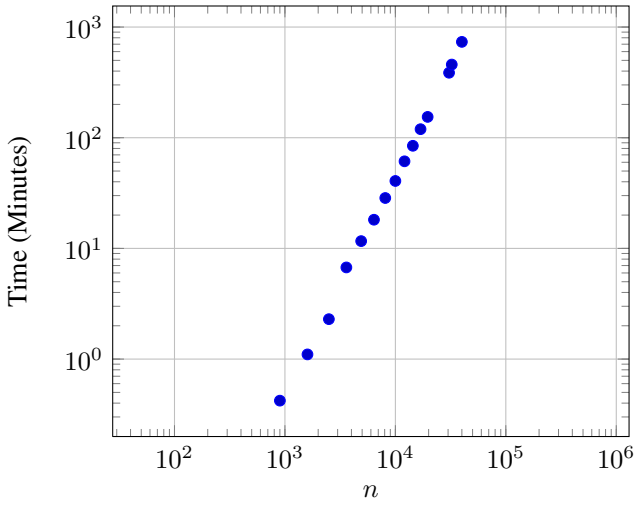


Figure 15. Log-log plot of time against the number of optimization variables. The gradient of this plot appears to be 2, suggesting that the time to compute the solution to a problem with n variables is $O(n^2)$.

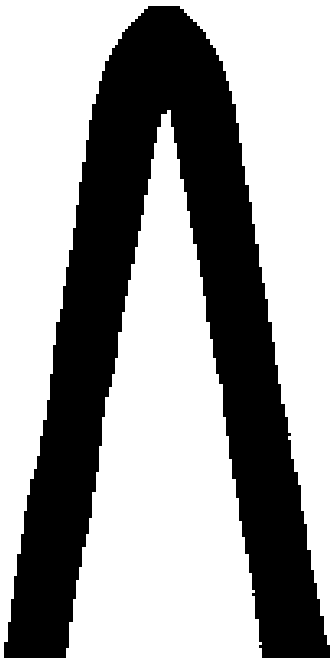


Figure 16. Solution found on mesh of 200×200 elements. The buckling constraint is set to $c_s = 0.1$ and the compliance constraint $c_{\max} = 8$. A volume of 0.1886 is attained. Compare with Figure 14.

5. CONCLUSIONS

The main computational cost associated with topology optimization problems involving buckling is the calculation of the derivatives of the buckling load. We have written an analytic formula for this but it remains still the most expensive part of an algorithm. To reduce the computational cost we have developed an algorithm which tries to minimise the number of these computations that have to be made. This method does not guarantee finding the global minimum, but instead will find a point

DOI: 10.1002/nme

which can be thought of as a weak local minimum. That is, if the variable with lowest sensitivity is changed from 1 to 0 the structure will become infeasible.

We have shown that the algorithm appears to scale quadratically with the number of elements in the finite element mesh of the design space. We have been able to solve large problems, of the size that would typically be of use to an engineer in an industrial application.

REFERENCES

1. Wolsey L. *Integer programming*. Wiley-Interscience series in discrete mathematics and optimization, Wiley, 1998. URL http://books.google.com/books?id=x7RvQgAACAAJ.
2. Arora J, Huang M. Methods for optimization of nonlinear problems with discrete variables: a review. *Structural Optimization* 1994; **8**:69–85.
3. Toakley A. Optimum design using available sections. *Proc Amer Soc Civil Eng, J Struct Div,* 1968; **94**:1219–1241.
4. Stolpe M, Bendsøe MP. Global optima for the Zhou-Rozvany problem. *Structural and Multidisciplinary Optimization* 2010; **43**(2):151–164.
5. Farkas J, Szabo L. Optimum design of beams and frames of welded I-sections by means of backtrack programming. *Acta Technica* 1980; **91**(1):121–135.
6. John K, Ramakrishna C, Sharma K. Optimum design of trusses from available sectionsuse of sequential linear programming with branch and bound algorithm. *Engineering Optimization* 1988; **13**(2):119–145.
7. Sandgren E. Nonlinear integer and discrete programming for topological decision making in engineering design. *Journal of Mechanical Design* 1990; **112**(1):118–122, doi:10.1115/1.2912568. URL http://link.aip.org/link/?JMD/112/118/1.
8. Sandgren E. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design* 1990; **112**(2):223–229, doi:10.1115/1.2912596. URL http://link.aip.org/link/?JMD/112/223/1.
9. Salajegheh E, Vanderplaats G. Optimim design of trusses with discrete sizing and shape variables. *Structural Optimization* 1993; **6**:79–85.
10. Beckers M. Dual methods for discrete structural optimization problems. *International Journal for Numerical Methods in Engineering* 2000; (48):1761–1784.
11. Stolpe M, Svanberg K. Modelling topology optimization problems as linear mixed 0-1 programs. *International Journal for Numerical Methods in Engineering* Jun 2003; **57**(5):723–739, doi:10.1002/nme.700. URL http://doi.wiley.com/10.1002/nme.700.
12. Achtziger W, Stolpe M. Truss topology optimization with discrete design variables - Guaranteed global optimality and benchmark examples. *Structural and Multidisciplinary Optimization* Dec 2007; **34**(1):1–20, doi:10.1007/s00158-006-0074-2. URL http://www.springerlink.com/index/10.1007/s00158-006-0074-2.
13. Achtziger W, Stolpe M. Global optimization of truss topology with discrete bar areas - Part I: theory of relaxed problems. *Computational Optimization and Applications* Nov 2008; **40**(2):247–280, doi:10.1007/s10589-007-9138-5. URL http://www.springerlink.com/index/10.1007/s10589-007-9138-5.
14. Achtziger W, Stolpe M. Global optimization of truss topology with discrete bar areas - Part II : Implementation and numerical results. *Computational Optimization and Applications* 2009; **44**(2):315–341, doi:10.1007/s10589-007-9152-7.
15. Svanberg K. The method of moving asymptotes - a new method for structural optimization. *International Journal For Numerical Methods in Engineering* 1987; **24**:359–373.
16. Kočvara M. On the modelling and solving of the truss design problem with global stability constraints. *Structural and Multidisciplinary Optimization* Apr 2002; **23**(3):189–203, doi:10.1007/s00158-002-0177-3. URL http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s00158-002-0177-3.
17. Kočvara M, Stingl M. Solving nonconvex SDP problems of structural optimization with stability control. *Optimization Methods and Software* Oct 2004; **19**(5):595–609, doi:10.1080/10556780410001682844. URL http://www.informaworld.com/openurl?genre=article&doi=10.1080/10556780410001682844&magic=crossref||D404A21C5BB053405B1A640AFFD44AE3.
18. Bogani C, Kočvara M, Stingl M. A new approach to the solution of the VTS problem with vibration and buckling constraints. *8th World Congress on Structural and Multidisciplinary Optimization*, 1, 2009.

DOI: 10.1002/nme

19. Tenek LH, Hagiwara I. Eigenfrequency Maximization of Plates by Optimization of Topology Using Homogenization and Mathematical Programming. *JSME International Journal Series C* 1994; **37**(4):667–677.
20. Pedersen N. Maximization of eigenvalues using topology optimization. *Structural and Multidisciplinary Optimization* 2000; **20**(1):2–11.
21. Neves MM, Sigmund O, Bendsøe MP. Topology optimization of periodic microstructures with a penalization of highly localized buckling modes. *International Journal for Numerical Methods in Engineering* Jun 2002; **54**(6):809–834, doi:10.1002/nme.449. URL http://doi.wiley.com/10.1002/nme.449.
22. Bendsøe MP, Sigmund O. *Topology Optimization: Theory, Methods and Applications*. Springer, 2003.
23. Hunt G, Thompson J. *A General Theory of Elastic Stability*. Wiley-Interscience, 1973.
24. Cook RD, Malkus DS, Plesha ME. *Concepts and Applications of Finite Element Analysis*. John Wiley and Sons, 1989.
25. HSL(2011). A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk.
26. Hogg J, Reid J, Scott J. Design of a multicore sparse cholesky factorization using DAGs. *SISC* 2010; **32**:3627–3649.
27. Ovtchinnkov E, Reid J. A preconditioned block conjugate gradient algorithm for computing extreme eigenpairs of symmetric and hermitian problems. *Technical Report RAL-TR-2010-019*, RAL 2010.
28. Rozvany GIN. On symmetry and non-uniqueness in exact topology optimization. *Structural and Multidisciplinary Optimization* Sep 2010; **43**(3):297–317, doi:10.1007/s00158-010-0564-0. URL http://www.springerlink.com/index/10.1007/s00158-010-0564-0.

DOI: 10.1002/nme