



# A fast method for binary programming using first-order derivatives, with application to topology optimization with buckling constraints

PA Browne, CJ Budd, NIM Gould, HA Kim, JA Scott

November 2011

**©2011 Science and Technology Facilities Council**

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

RAL Library  
STFC Rutherford Appleton Laboratory  
R61  
Harwell Oxford  
Didcot  
OX11 0QX

Tel: +44(0)1235 445384  
Fax: +44(0)1235 446403  
email: [libraryral@stfc.ac.uk](mailto:libraryral@stfc.ac.uk)

Science and Technology Facilities Council reports are available online at: <http://epubs.stfc.ac.uk>

**ISSN 1358- 6254**

Neither the Council nor the Laboratory accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations.

# A fast method for binary programming using first-order derivatives, with application to topology optimization with buckling constraints

Philip A. Browne<sup>1,2</sup>, Christopher J. Budd<sup>1</sup>, Nicholas I. M. Gould<sup>3,4</sup>,  
H. Alicia Kim<sup>5</sup> and Jennifer A. Scott<sup>3,4</sup>

## ABSTRACT

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 weight minimisation 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.

---

<sup>1</sup> Department of Mathematical Sciences, University of Bath, BA2 7AY, England, EU.  
Email: p.a.browne@bath.ac.uk, cjb@maths.bath.ac.uk.

<sup>2</sup> This research was supported through the EPSRC Industrial CASE studentship MCA 09-2008/2009 at the University of Bath in conjunction with the STFC Rutherford Appleton Laboratory.

<sup>3</sup> Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, EU.  
Email: nick.gould@stfc.ac.uk, jennifer.scott@stfc.ac.uk.  
Current reports available from “<http://www.numerical.rl.ac.uk/reports/reports.shtml>”.

<sup>4</sup> This work was supported by the EPSRC grants EP/E053351/1 and EP/I013067/1.

<sup>5</sup> Department of Mechanical Engineering, University of Bath, BA2 7AY, England, EU.  
Email: H.A.Kim@bath.ac.uk .

Computational Science and Engineering Department  
Atlas Centre  
Rutherford Appleton Laboratory  
Oxfordshire OX11 0QX  
November 18, 2011

# 1 Introduction

The problem considered in this paper is to minimize the weight of an elastic structure whilst maintaining the integrity of the structure 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. The goal of optimization is to determine which elements should contain material and which should be void of material. This is usually represented by a binary variable where a value of 1 is assigned to an element with material and a value of 0 to an element containing no material, thus resulting in 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 and in some cases took over one week of CPU time on a modern server to compute the solution. Whilst these methods do find global minima, they suffer from exponential growth in the computation time as the number of variables increases.

Beckers [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.

Achtziger 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 stiffness of the material in the element. 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 optimum is attained.

By far the most popular optimization method to update the structure is the Method of Moving Asymptotes (MMA) [15]. MMA requires the function values and values of the derivatives at the current iteration in order to progress to a new iteration with lower objective function. There are many examples of MMA being very efficient at solving topology optimization problems with different objectives/constraints (see, for example [16]). 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 a single modeshape corresponding to the critical load, i.e. 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.

Semidefinite programming methods have been developed specifically to deal with such eventualities. Kocvara [17], and in conjunction with Stingl [18], has applied such methods to topology optimization problems. More recently, along with Bogani [19], 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. With these techniques they were able to solve a non-discrete problem with 5000 variables in about 35 minutes on a standard pc.

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 [20], Pedersen [21] and Neves et al. [22] 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 assigning zero 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).

Due to the dimensionality of the problems, 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 linear elasticity analysis. With this in mind, the proposed fast binary descent method we introduce will try to reduce the number of derivative calculations required.

The remainder of this paper is organised as follows. In Section 2, we formulate the topology

optimization problem to include a 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 a buckling constraint

Let  $\Omega$  be the design domain containing the elastic structure that is discretised using a finite-element mesh  $\mathcal{T}$ . A load  $f$  is applied to  $\Omega$ , and this induces displacements  $u$ , which are the solution to the equilibrium equations of linear elasticity

$$Ku = f \quad (1)$$

where  $K$  is the finite-element stiffness matrix. Compliance, defined as the product  $f^T u$ , is a measure of external work done on the structure and adding an upper bound  $c_{\max}$  ensures that the structure remains stiff.

The minimisation of weight subject to a compliance constraint is a well-studied problem, see for example [16]. However, it has long been observed that structures optimized for minimum weight or compliance are prone to buckling [23].

The critical buckling load of a structure is defined by the smallest positive value of  $\lambda$  corresponding to a nonzero  $v$  for which

$$(K + \lambda K_\sigma)v = 0. \quad (2)$$

In this equation,  $K$  is the symmetric finite-element stiffness matrix and  $K_\sigma$  is the symmetric stress stiffness matrix.  $(\lambda, v)$  is an eigenpair of the generalised eigenvalue problem (2).  $\lambda$  is referred to as the eigenvalue and  $v \neq 0$  the corresponding eigenvector (or mode shape). The critical load is  $\lambda$  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 the 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 to be solved becomes:

$$\min_x \sum x_i \quad (3a)$$

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

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

$$x \in \{0, 1\}^n \quad (3d)$$

$$K(x)u(x) = f \quad (3e)$$

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

## 2.1 Derivative calculations

To use the binary descent method (discussed in Section 3) we need an efficient way of calculating the derivative of the constraints with respect to the variables  $x_i$ . As will be seen in Section 4, the computation of derivatives of the buckling constraint (3c) is the bottleneck in our optimization algorithm, so it is imperative that we have an analytic expression for this. To calculate the derivatives, the binary constraints on the variables are relaxed and assume that the following holds

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

where  $K_{\ell}$  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  $\sigma_{\ell}$  on an element  $\ell$  is a  $3 \times 3$  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,  $E_\ell$  is a constant matrix of material properties and  $B_\ell$  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.

$$K_\sigma = \sum_{\ell=1}^n \int G_\ell^T \begin{bmatrix} \sigma_{11} & \sigma_{12} & 0 & 0 \\ \sigma_{12} & \sigma_{22} & 0 & 0 \\ 0 & 0 & \sigma_{11} & \sigma_{12} \\ 0 & 0 & \sigma_{12} & \sigma_{22} \end{bmatrix}_\ell G_\ell dV_\ell, \quad (4)$$

where  $G_\ell$  is a matrix containing derivatives of the basis functions that relates the displacements of an element  $\ell$  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

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

Note that  $\Theta$  is a linear operator. Using this, (4) becomes

$$\begin{aligned} K_\sigma &= \sum_{\ell=1}^n \int G_\ell^T \Theta(x_\ell E_\ell B_\ell u) G_\ell dV_\ell \\ &= \sum_{\ell=1}^n \int G_\ell(\xi)^T \Theta(x_\ell E_\ell B_\ell(\xi) u) G_\ell(\xi) dV_\ell \\ &\approx \sum_{\ell=1}^n \sum_j \omega_j G_\ell(\xi_j)^T \Theta(x_\ell E_\ell B_\ell(\xi_j) u) G_\ell(\xi_j) \end{aligned} \quad (5)$$

where  $\omega_j$  are the weights associated with the appropriate Gauss points  $\xi_j$  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  $\Theta$  with respect to  $x_i$ . Since  $\Theta$  is linear

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

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



Applying the chain rule to (5) we obtain

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

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  $x_i = 1, \dots, n$ , (6) must be computed. As (6) contains a sum over  $i = 1, \dots, n$ , it can be seen 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 Fast Binary Descent Method

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

$$\min_x e^T x \quad (7a)$$

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

$$x \in \{0, 1\} \quad (7c)$$

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

The objective function  $e^T x$  is a linear function of  $x$  that 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 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. The method will take discrete steps so that

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

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

Consider now variables  $x_i^k$  such that  $x_i^k = 1$  that we wish to change to  $x_i^{k+1} = 0$ . Since  $x_i^{k+1} - x_i^k = -1$ , 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 minimal, all the terms of  $\frac{\partial c(x^k)}{\partial x_i}$  need to be as small as possible. However, since there are multiple constraints, the variables for which the gradient of one constraint is small may have a large gradient for another constraint.

Assuming a feasible point such that  $c(x^k) > 0$  and ignoring the higher order terms,

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

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

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

or equivalently

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

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

Define the sensitivity of variable  $i$  to be

$$s_i(x^k) = \max_{j=1, \dots, m} \frac{\partial c_j(x^k)}{\partial x_i} / \max\{c_j(x^k), 10\epsilon\} \quad (9)$$

where  $\epsilon$  is the machine epsilon that guards against round off errors. For each variable,  $s_i(x^k)$  is the most conservative estimate of how the constraints will vary if the value of the variable is changed. 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.  $\beta_j$  is the point at which the line associated with the constraint  $c_j$  crosses the  $y$ -axis and so  $\beta_j = 1 - \frac{\partial c_j(x^k)}{\partial x_i} / c_j(x^k)$ . The amount that the normalised constraint  $c_j$  would change if the variable  $x_i^k$  were changed is then given by  $1 - \beta_j = \frac{\partial c_j(x^k)}{\partial x_i} / c_j(x^k)$ .

In this case the derivatives indicate that if the variable  $x_i^k$  were to be decreased, the second constraint is affected relatively more than the first constraint (as  $\max\{a, b\} = b$ ), and hence the sensitivity associated with this variable  $x_i$  is given the value  $s_i(x^k) = \frac{\partial c_2(x^k)}{\partial x_i} / c_2(x^k)$ .

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

$$s_1 \leq s_2 \leq \dots \leq s_p \quad \forall s_i \text{ s.t. } x_1^k, x_2^k, \dots, x_p^k = 1 \quad (10)$$

$$s_{p+1} \geq s_{p+2} \geq \dots \geq s_n \quad \forall s_i \text{ s.t. } x_n^k, x_{n-1}^k, \dots, x_{p+1}^k = 0 \quad (11)$$

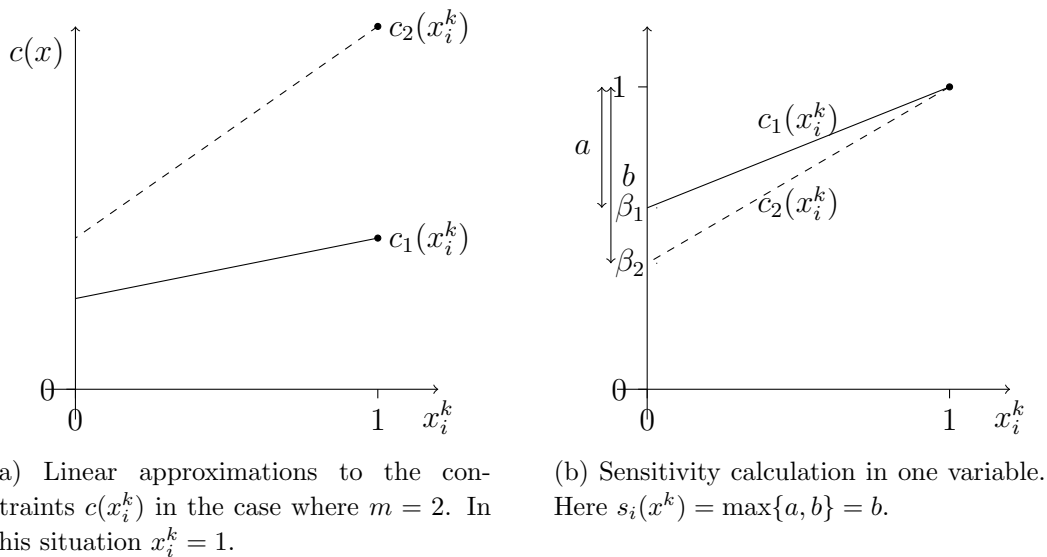


Figure 1: Sensitivity calculation in one variable for the case when  $m = 2$ .

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

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

or equivalently

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

To update the current solution, we consider the variables ordered so that (10) and (11) hold and find for some  $\alpha \geq 0$

$$L := \max_{1 \leq \ell \leq p} \ell \text{ s.t. } \alpha c_j(x^k) - \sum_{i=1}^{\ell} \frac{\partial c_j(x^k)}{\partial x_i} > 0 \quad \text{for all } j \in 1, \dots, m. \quad (12)$$

Then we decrease from 1 to 0 those variables  $x_1^k, \dots, x_L^k$  so as to reduce the objective function by a value of  $L$ .

However, there is the possibility that increasing variables from 0 to 1 could further reduce the objective function by reducing yet more variables from 1 to 0. This is tested by finding (or attempting to find)  $J > 0$  such that

$$J := \max_{0 \leq \ell \leq (p-L)/2} \ell \text{ s.t. } \sum_{i=1}^{\ell} \frac{\partial c_j(x^k)}{\partial x_{p+i}} - \sum_{i=1}^{2\ell} \frac{\partial c_j(x^k)}{\partial x_{L+i}} \geq 0 \quad \text{for all } j \in 1, \dots, m. \quad (13)$$

So the variables corresponding to the terms in the first sum are increased from 0 to 1 but for each of these two variables are decreased from 1 to 0, corresponding to the terms in the second summation. As there are more terms in the second summation the objective function improves whilst remaining a feasible solution. Hence the variables  $x_{L+1}^k, \dots, x_{L+2J}^k$  are decreased from 1

to 0 and the variables  $x_{p+1}^k, \dots, x_{p+J}^k$  are increased from 0 to 1. Note that in (12) and (13) the equations have to hold for each of the constraints  $j = 1, \dots, m$ .

The coefficient  $\alpha$  is a measure of how well the linear gradient information is predicting the change in the constraints. If the problem becomes infeasible, then the method has taken too large a step, so  $\alpha$  is reduced 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 that are as large as possible. If the step has been accepted for the previous two iterations without reducing  $\alpha$  then  $\alpha$  is increased to try and take larger steps and thus speed up the algorithm.

Note that if  $\alpha$  is too large and the solution becomes infeasible then  $\alpha$  is reduced and a smaller step is taken without recomputing the derivatives. Hence increasing  $\alpha$  by too much is not too detrimental to the performance of the algorithm. Based on experience,  $\alpha$  is reset to  $0.7\alpha$  when the solution becomes infeasible and  $\alpha$  is set to  $1.5\alpha$  when we want to increase it. These values appear stable and give good performance for most problems.

To ensure that at least one variable is updated,  $\alpha$  must be larger than a critical value  $\alpha_c$  given by

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

This guarantees that  $L \geq 1$  and at least one variable is updated. The upper bound  $\alpha \leq 1$  must also be enforced so that  $c(x^{k+1}) \geq 0$ .

If we cannot make any further progress with this algorithm, we stop. Making further progress would be far too expensive as we would have to switch to a different integer programming strategy and the curse of dimensionality for the problems that we wish to consider prohibits this. However, we believe the computed solution is good because if we try and improve the objective function by changing the variable for which the constraints are infinitesimally least sensitive, the solution becomes infeasible.

The fast binary descent algorithm is thus presented in Algorithm 1:

---

**Algorithm 1** Fast binary descent method
 

---

```

1: Initialise  $x^0$  and  $\alpha$ .
   Compute objective function (7a) and constraints (7b)
2: if  $x^0$  not feasible then
3:   Stop
4: else
5:   Compute derivatives  $\frac{\partial c(x^k)}{\partial x_i}$ 
6:   Sort  $s_i$  (9)
7:   Compute values  $L$  from (12) and  $J$  from (13)
8:   Update the variables  $x_i^k$  that correspond to  $L$  and  $J$  from (12) and (13)
9:   if no variables updated then
10:    {Algorithm has converged}
11:    return with computed solution
12:   end if
13:   Compute objective function and constraints from equations (7a) and (7b).
14:   if not feasible then
15:    {Reject update step}
16:    Reduce  $\alpha$ .
17:    GO TO 7
18:   else
19:    {Accept update step}
20:    Increase  $\alpha$  if desired
21:     $k = k + 1$ 
22:    GO TO 5
23:   end if
24: end if

```

---

## 4 Implementation and results

We consider optimising isotropic structures with Young's modulus 1.0 and Poisson's ratio 0.3. The design spaces are discretised using square bilinear elements on a uniform mesh.

The fast binary descent method 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 all corresponding eigenvectors of the critical load were found. 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 passed through a standard low-pass filter [28] with radius  $2.5h$  where  $h$  is the width of an element and ordered using HSL\_KB22, a heapsort [29] algorithm.

The codes were executed on a desktop with an Intel® Core™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. All reported times are wall-clock times measured using `system_clock`.

## 4.1 Short cantilevered beam

We consider a clamped beam with a vertical unit external force applied to the free side as shown in Figure 2. Figures 3 to 5 refer to the solutions found with the same design domain and material properties but with buckling and compliance constraints.

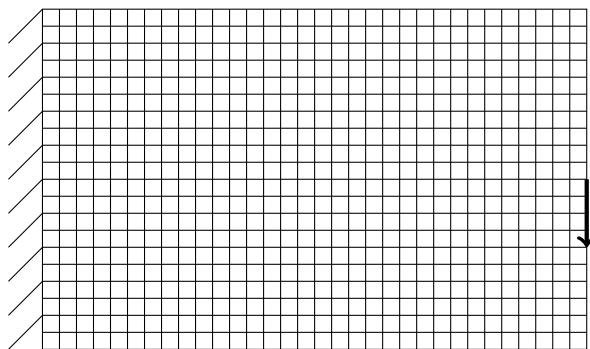


Figure 2: Design domain of a centrally loaded cantilevered beam. The aspect ratio of the design space is 1.6 and a unit load is applied vertically from the centre of the right hand side of the domain.



Figure 3: Solution found on mesh of  $80 \times 50$  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. The buckling constraint  $c_2$  is active and the compliance constraint  $c_1$  is not.

Figure 3 is the computed solution to the problem with parameters  $c_s = 0.9$  in (3c) and  $c_{\max} = 35$  in (3b). In this case the compliance constraint  $c_1(x^0)$  is large initially but the buckling constraint  $c_2(x^0)$  is small initially. We see that the method has produced a typical optimum grillage structure with 4 bars under compression and only 3 bars under tension. Note that in the upper bar near the point of loading there is a distinct corner in the computed

solution. This type of formation attracts high concentrations of strain energy and so if the problem were minimization of compliance then an optimization method would wish to avoid such situations. However, in this case optimization of this region is primarily dominated by the buckling constraint and the compliance is not the critical constraint.



Figure 4: Solution found on mesh of  $80 \times 50$  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 ( $c_s = 0.9$ ) but is allowed to be more flexible with  $c_{\max} = 60$  (i.e. the compliance constraint is not as restrictive). This results in 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. Thus optimization reinforced the lower bar to meet the buckling constraint.

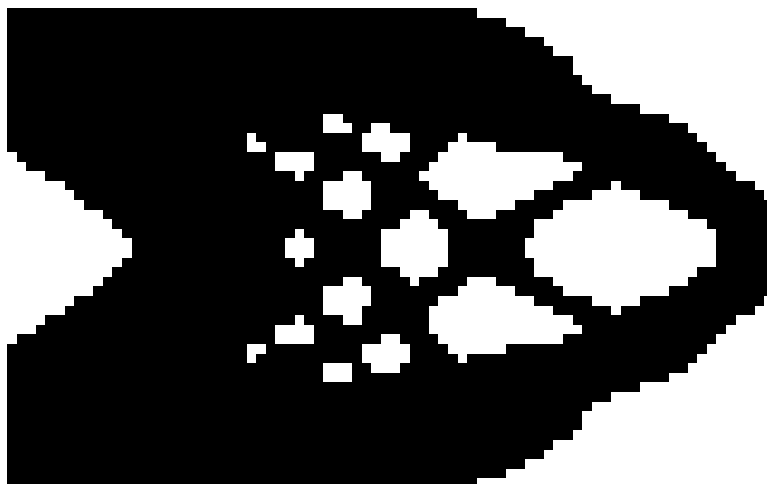


Figure 5: Solution found on mesh of  $80 \times 50$  elements. The buckling constraint is set to  $c_s = 0.1$  and the compliance constraint  $c_{\max} = 30$ . A volume of 0.692 is attained. Here the compliance constraint  $c_1$  is active and the buckling constraint  $c_2$  is not.

Figure 5 was obtained as the solution for a problem with  $c_s = 0.1$  and  $c_{\max} = 30$ . In this case the initial value of  $c_1(x^0)$  is close to 0. The computed solution has only the compliance constraint active and hence the computed solution is more symmetrical than the solutions shown in Figures 3 and 4.

From Figures 3 to 5 it is possible to see a clear difference in the topology of the resulting solution depending on the parameters  $c_s$  and  $c_{\max}$ . The history of the algorithm when applied to the problem solved in Figure 3 where  $c_{\max} = 35$  and  $c_s = 0.9$  is displayed in Figures 6 to 8.

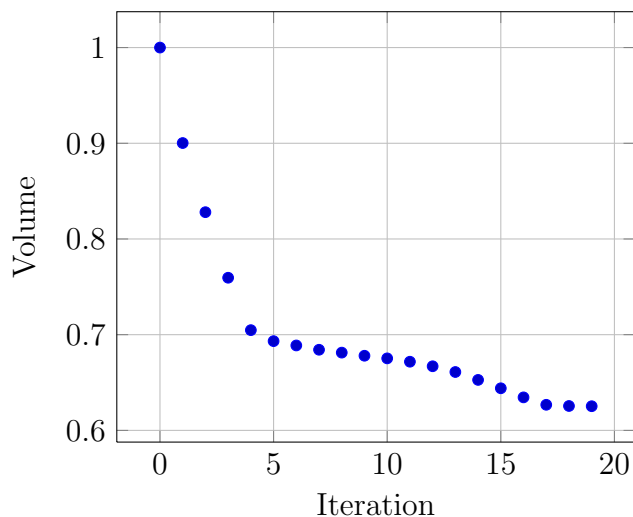


Figure 6: Volume - iterations of the fast binary descent method

The plot of the objective function against iteration number shown in 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 computation progresses. Until iteration 4 large steps have been made and thus the objective function is swiftly decreasing. When going to iteration 5 taking a large step would make the current solution infeasible so the method automatically decreases the step size and hence the decrease in the objective function is reduced.

Figure 7 shows that 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  $c_1$ . If this plot is compared with Figure 6 then the large changes in compliance can be seen to occur where there are large reductions in volume and similarly when there is a small change in the volume the change in compliance is also small.

Figure 8 shows the lowest 6 eigenvalues of the system as the 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. At iterations 5 and 8 we see that the eigenvalue constraint is close to being violated. The increase in the lowest eigenvalue at the subsequent steps corresponds to a local thickening of the structure around the place where the buckling is most concentrated. This shows that the method has re-introduced material in order to move away from the constraint boundary. The nonlinearity in  $c_2(x)$  is clear from the non-



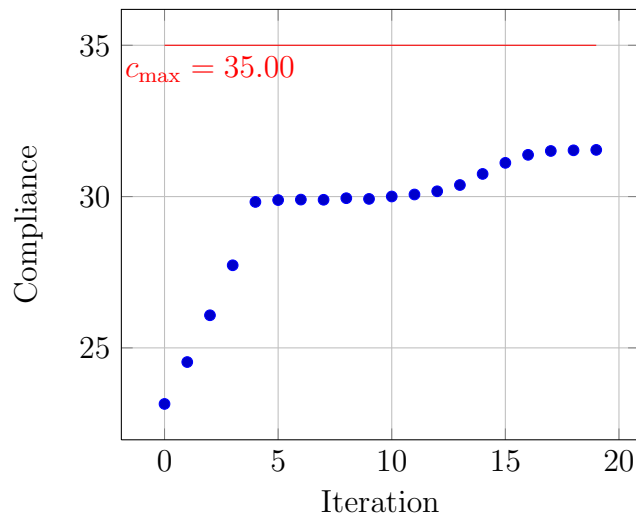


Figure 7: Compliance - iterations of the fast binary descent method

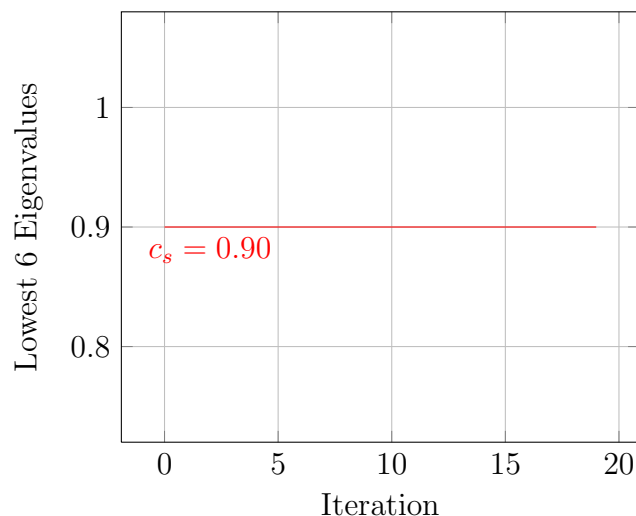


Figure 8: Eigenvalues - iterations of the fast binary descent method. Note that on the 20-th iteration the eigenvalue constraint is violated, thus the computed solution is at the 19-th iteration.

monotonic behaviour seen in Figure 8. Generally we do see the eigenvalues converging and that supports the intuitive optimality criteria of coincidental eigenvalues.

Figure 8, when viewed in combination with Figure 7 shows 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. 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. The problem solved in Figure 9b has  $c_s = 0.225$  and  $c_{\max} = 22.5$ . The problem solved in Figure 9c has  $c_s = 0.001$  and  $c_{\max} = 60$ .

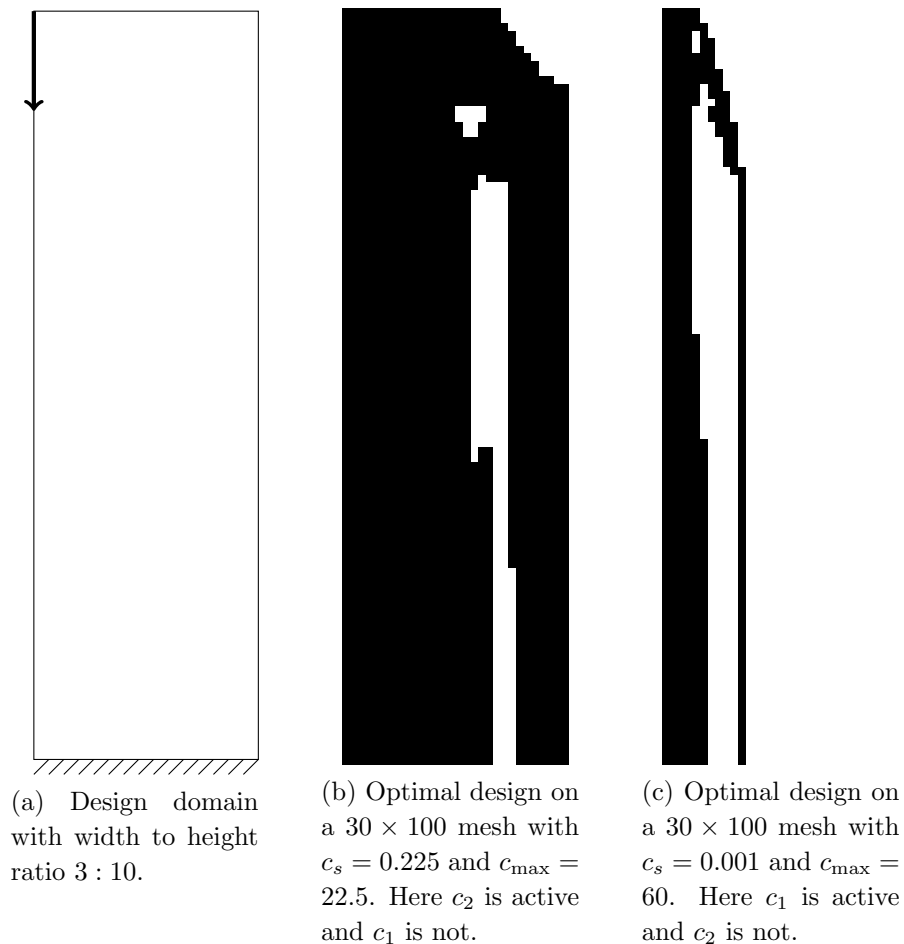


Figure 9: A column loaded at the side

In Figure 9c as the constraints are relaxed compared with the problem in Figure 9b, the computed solution has a significantly lower objective function. However, it follows the same structural configuration where the main compressive column directly under the load resists the buckling and the slender column on the side provides additional support in tension to reduce bending. In both of these structures the path of the optimization is driven by the first buckling mode.

### 4.3 Centrally loaded column

We consider a square design domain (Figure 10). A unit load is applied vertically downwards at the centre of the top of the design domain and the base is fixed.

Figures 11 to 14 present results for a mesh of  $60 \times 60$  elements for a range of values of the constraints. Figures 11 and 12 have  $c_s = 0.5$  with  $c_{\max} = 5$  and  $c_{\max} = 5.5$ , respectively. This small change in the compliance constraint results in two distinct configurations. Figure 12 with the higher compliance constraint achieves a lower volume and has the compliance constraint

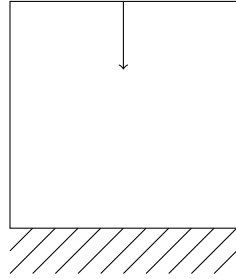


Figure 10: Design domain of model column problem. This is a square domain of side length 1 with a unit load acting vertically at the midpoint of the upper boundary of the space.

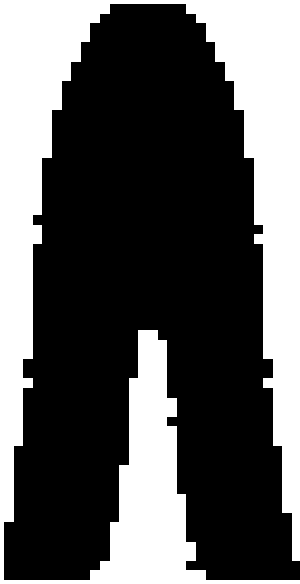


Figure 11: Solution computed on a mesh of  $60 \times 60$  elements. The buckling constraint is set to  $c_s = 0.5$  and the compliance constraint  $c_{\max} = 5$ . Here, the compliance constraint is active and the buckling constraint is inactive.



Figure 12: Solution computed on a mesh of  $60 \times 60$  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.

active as opposed to the buckling constraint which is active in Figure 11.

Distinct “*A-like*” structures have been found in Figures 13 and 14. These problems share the parameter  $c_{\max} = 8$  but vary in that they have  $c_s = 0.4$  and  $c_s = 0.1$ , respectively. The higher buckling constraint of Figure 13 leads to the development of thick regions in the centre of the supporting legs. These regions help to resist the first order buckling mode of the individual legs and are not seen in Figure 14 as the buckling constraint is lower. Figure 15 is the solution to a problem with the same parameters as the problem considered in Figure 14 but is solved on a much finer  $200 \times 200$  mesh.

From Figures 11 to 15 we see that the symmetry of the problem is not present in the

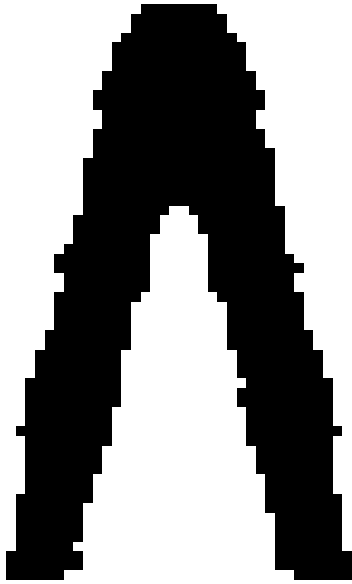


Figure 13: Solution computed on a mesh of  $60 \times 60$  elements. The buckling constraint is set to  $c_s = 0.4$  and the compliance constraint  $c_{\max} = 8$ . A volume of 0.276 is attained.

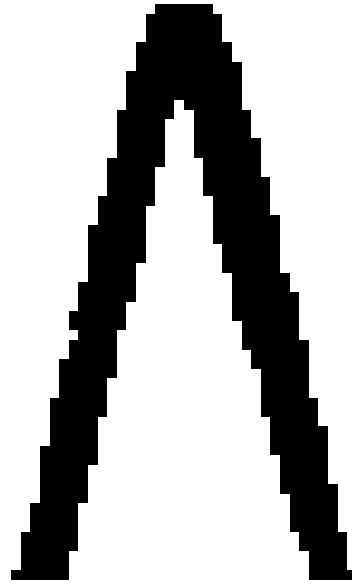


Figure 14: Solution computed on a mesh of  $60 \times 60$  elements. The buckling constraint is set to  $c_s = 0.1$  and the compliance constraint  $c_{\max} = 8$ . A volume of 0.183 is attained.

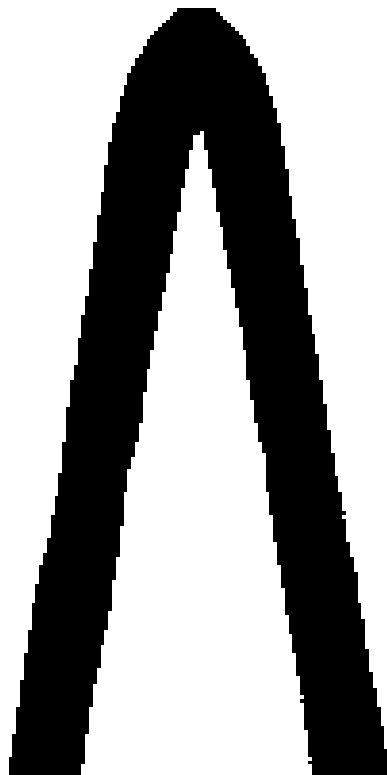


Figure 15: Solution computed on a mesh of  $200 \times 200$  elements. The buckling constraint is set to  $c_s = 0.1$  and the compliance constraint  $c_{\max} = 8$ . A volume of 0.1886 is attained. Compare with Figure 14.

Problem size $n$	Objective	Derivative calculations	Analyses	Time (mins) to 3 s.f.	Proportion of time on $\partial c_2/\partial x$
$30 \times 30 = 900$	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	4230	0.996

Table 1: Table of results for the centrally loaded column

computed solution. As Stolpe [30] and Rozvany [31] have shown, since we do not have continuous variables we do not necessarily expect the optimal solution to these binary programming problems to be symmetric. The asymmetry in the computed solutions arise from (12) and (13) as only a subset of elements with precisely the same sensitivity values may be chosen to be updated and so the symmetry may be lost.

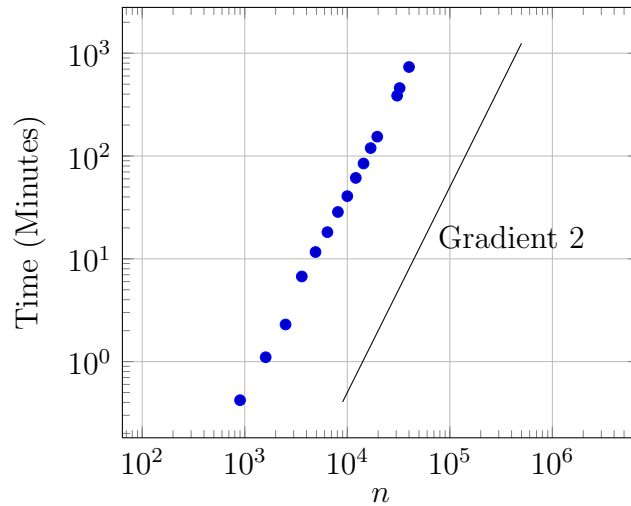


Figure 16: Log-log plot of time against the number of optimization variables. The gradient of this plot appears to be 2, suggesting that the time to compute the solution to a problem with  $n$  variables is  $O(n^2)$ .

Table 1 summarises the results obtained when solving the problem considered in Figures 14 and 15 but with varying mesh sizes. Note the problem size that the fast binary method has been able to solve. A computation on a two-dimensional mesh of  $3 \times 10^4$  elements took less than 8 hours on a modest desktop and  $4 \times 10^4$  elements took around 12 hours. This speed is attained because the number of derivative calculations appears to not be dependent on the number of variables. Figure 16 shows a log-log plot of the number of optimization variables against the wall-clock time taken to compute a solution. As the plot appears to have a gradient close to 2 this indicates that the time to compute a solution is  $\mathcal{O}(n^2)$ .

A detailed examination of the computational cost indicates that the vast majority of the computational cost is in the computation of the derivative of the buckling constraint (see the final column of Table 1). A massively parallel implementation of this step is possible and it is anticipated that it should achieve near optimal speedup as no information transfer is required for the calculation of the derivative with respect to the individual variables.

## 5 Conclusions

The main computational cost associated with topology optimization involving buckling is the calculation of the derivatives of the buckling load. We have employed an analytic formula for this but it still remains the most expensive part of the algorithm. To reduce the computational cost we have developed an algorithm that aims to minimise the number of these computations that are required. The method is a descent method that enforces feasibility at each step and thus could be terminated early and would still result in a feasible structure.

We have numerically shown that the algorithm scales quadratically with the number of elements in the finite-element mesh of the design space. This corresponds to the analytical result that the derivative of the stress-stiffness matrix with respect to each of the design

variables is an  $\mathcal{O}(n^2)$  operation. The numerical experiments demonstrate the efficiency of the method for binary topology optimization using compliance and buckling constraints.

## 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 section-use 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. Optimum 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] Bendsøe MP, Sigmund O. *Topology Optimization: Theory, Methods and Applications*. Springer, 2003.
- [17] 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>.
- [18] 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>.
- [19] 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.
- [20] 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.
- [21] Pedersen N. Maximization of eigenvalues using topology optimization. *Structural and Multidisciplinary Optimization* 2000; **20**(1):2–11.
- [22] 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>.
- [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] Huang X, Xie YM. *Evolutionary Topology Optimization of Continuum Structures*. Wiley, 2010.
- [29] Williams J. Algorithm 232: heapsort. *Communications of the ACM* 1964; **7**(6):347–348.
- [30] Stolpe M. On some fundamental properties of structural topology optimization problems. *Structural and Multidisciplinary Optimization* 2010; **41**(5):661–670.
- [31] Rozvany GIN. Authors reply to a discussion by Gengdong Cheng and Xiaofeng Liu of the review article On symmetry and non-uniqueness in exact topology optimization by George I.N. Rozvany (2011, *Struct Multidisc Optim* 43:297317). *Structural and Multidisciplinary Optimization* Sep 2011; **44**(5):719–721, doi:10.1007/s00158-011-0703-2. URL <http://www.springerlink.com/index/10.1007/s00158-011-0703-2>.