



SIXTH FRAMEWORK PROGRAMME

FP-6 STREP 30717 PLATO-N (Aeronautics and Space)

PLATO-N

A PLAtform for Topology Optimisation incorporating Novel, Large-Scale, Free-Material Optimisation and Mixed Integer Programming Methods

D29 – Efficient Methods for solving mixed integer stress constrained problems

PLATO-N Public Report PU-R-5-2008

Authors:

N. Nguyen Canh

M. Stolpe

Due date of deliverable: October 1, 2008

Actual submission date: October 1, 2008

Start date of project: October 1, 2006

Duration: 36 Months

Organization name of lead contractor for this deliverable: DTU

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Dissemination level: Public

Efficient methods for solving discrete topology design problems in the PLATO-N project

Nguyen Canh Nam* Mathias Stople†

October 1, 2008

Abstract

This paper considers the general multiple load structural topology design problems in the framework of the PLATO-N project. The problems involve a large number of discrete design variables and were modeled as a non-convex mixed 0–1 program. For the class of problems considered, a global optimization method based on the branch-and-cut concept was developed and implemented. In the method a large number of continuous relaxations were solved. We also present an algorithm for generating cuts to strengthen the quality of the relaxations. Several heuristics were also investigated to obtain efficient algorithms. The branch and cut method is used to solve benchmark examples which can be used to validate other methods and heuristics.

Key words: Topology optimization, Branch and cut, Stress constraints, Reformulations, Relaxations, Heuristics.

1 Introduction

We present a special purpose method for solving mixed 0-1 structural topology optimization problems that have arisen in the PLATO-N project ([9, 17]) to global optimality. These problem were solved by a finitely convergent non-linear branch and cut method, which is an exact algorithm consisting of a combination of cutting plane method with a branch-and-bound algorithm.

Mixed 0-1 optimization problems are often solved by a convergent deterministic branch-and-bound algorithm, since this guarantees global optimality [18]. In such a method a large number of continuous relaxations are solved. Branch-and-bound method have been successfully used in several cases of topology design optimization, see [4, 23, 21] and references

*Department of Mathematics, Technical University of Denmark (DTU), Matematiktorvet, Building 303 S, DK – 2800 Kgs. Lyngby, Denmark, *E-mail:* *N.N.Canh@mat.dtu.dk*

†Department of Mathematics, Technical University of Denmark (DTU), Matematiktorvet, Building 303 S, DK – 2800 Kgs. Lyngby, Denmark, *E-mail:* *M.Stolpe@mat.dtu.dk*

therein. In our study of the structure of the problem considered, suitable relaxations are proposed. Tight lower bounds given by relaxations means that the algorithm can be accelerated significantly. In this work, we used various reformulation techniques presented in [17, 9], which are suitable for implementation in a nonlinear branch-and-cut framework for solving the class of problem considered, and these give different relaxations with their lower bounds that will allow the solutions of these problems in reasonable computational times.

But branch-and-bound approaches result in slow convergence for the class of problems considered. One way to strengthen the representation of the original problems is by introducing valid inequalities and cuts. A pure branch and bound approach can be considerably accelerated by employing of a cutting plane scheme, either just at the top of the tree, or at every node of the tree, resulting in a branch-and-cut approach, [19]. The cuts lead to a considerable reduction in the size of the tree. We therefore investigate types of cuts which may advance the optimization, namely the recently proposed Combinatorial Benders' cuts (see [10, 15]) and a simple cut that we call a functional cut.

Finding good feasible points is important when trying to solve this class of problems to global optimality. Therefore, for large and/or hard problems, branch-and-cut can be used in conjunction with heuristics to obtain a good and possibly optimal solution. We present several heuristics which are suitable for implementation in a branch-and-cut framework for these classes of problem. These heuristics are a generalization of the ideas proposed in [7, 13, 12, 3].

The rest of the report is organized as follows: section 2 presents the problems considered in the project. Section 3 discusses about the relaxation and reformulation techniques and the branching procedures used. Section 4 is mostly about cutting planes. Section 5 motivates the proposed heuristics. Section 6 is devoted to design and implementation aspects. Numerical results are showed in section 7. Finally, we end the report with some conclusions and outlines of future research.

2 Problems considered

In practice, topology design problems are treated by discretizing the design domain into a large number of finite elements. The elements are used to describe the topology and to compute the objective function (weight, compliance, etc) and the constraints (bounds on displacements, stresses, etc).

We start with a set of n finite elements in two or three-dimensional design space with appropriate support conditions, see e.g. [5]. For a given vector $x \in \mathbb{B}^n$ of design variables the symmetric and positive semi-definite stiffness matrix of the structure in global coordinates is denoted by $K(x)$.

We assume throughout that the stiffness matrix $K(x)$ depends linearly on the design variables x , i.e.,

$$K(x) := K_0 + \sum_{j=1}^n x_j K_j \quad (\text{A1})$$

where

$$x_j K_j \quad (\text{A2})$$

is the symmetric and positive semi-definite stiffness matrix of the j -th element and K_0 is a given symmetric positive semi-definite matrix (possibly equal to zero). The design variables are interpreted as

$$x_j = \begin{cases} 1 & \text{if the } j\text{-th element contains material, and} \\ 0 & \text{otherwise.} \end{cases}$$

Here, d denotes the number of degrees of freedom of the structure after the deletion of fixed nodal coordinate direction. We consider M static load cases where the loads are given by the vectors

$$f_1, \dots, f_M \in \mathbb{R}^d \setminus \{0\} \quad (\text{A3})$$

in global reduced coordinates. The elastic equilibrium equations for the structure subjected to the static external load vector f_k is assumed to be given by

$$K(x)u_k = f_k, \quad k = 1, \dots, M,$$

where $u_k \in \mathbb{R}^d$ denotes the nodal displacement vector corresponding to the external load f_k . The density ρ_j for the j -th element is assumed to be strictly positive, i.e.,

$$\rho_j > 0 \quad \forall j = 1, \dots, n, \quad (\text{A4})$$

and we assume that the weight (or volume) of the structure, described by x , is given by

$$\sum_{j=1}^n x_j \rho_j.$$

The first problem formulation which is considered for the branch-and-cut method is the following minimum weight problem with constraints on the

compliances, the nodal displacements, and the local stress

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^n x_j \rho_j \\
& \begin{array}{l} x \in \mathbb{R}^n, \\ u_1, \dots, u_M \in \mathbb{R}^d \end{array} && \\
\text{(WP)} \quad & \text{subject to} && \begin{aligned} & K(x)u_k = f_k \quad \forall k \\ & f_k^T u_k \leq \bar{\gamma}_k \quad \forall k \\ & u_k^T W_j u_k \leq \bar{\sigma}^2 \quad \forall k, \forall j : x_j = 1 \\ & \underline{u} \leq u_k \leq \bar{u} \quad \forall k \\ & Ax \leq b \\ & x \in \{0, 1\}^n. \end{aligned}
\end{aligned}$$

The compliances are bounded by the given scalars

$$\bar{\gamma}_k > 0 \quad \forall k = 1, \dots, M. \quad (\text{A5})$$

The stress constraints are described by the given symmetric and positive semi-definite matrices $W_j \in \mathcal{S}_+^d$ and the stress bound

$$\bar{\sigma} > 0 \quad . \quad (\text{A6})$$

The matrix $A \in \mathbb{R}^{m \times n}$ and the vector $b \in \mathbb{R}^m$ can be used to model demands on the connectivity of the structure. Within the branch-and-cut framework this pair is used to store the cut pool, i.e., a collection of linear valid inequalities and generated cuts.

The second problem formulation considered is the minimum compliance problem with a weight (or volume) constraint and constraints on the nodal displacements and the local stress

$$\begin{aligned}
& \text{minimize} && \frac{1}{2} \sum_{k=1}^M f_k^T u_k \\
& \begin{array}{l} x \in \mathbb{R}^n, \\ u_1, \dots, u_M \in \mathbb{R}^d \end{array} && \\
\text{(CP)} \quad & \text{subject to} && \begin{aligned} & K(x)u_k = f_k \quad \forall k \\ & \sum_{j=1}^n x_j \rho_j \leq V \\ & u_k^T W_j u_k \leq \bar{\sigma}^2 \quad \forall k, \forall j : x_j = 1 \\ & \underline{u} \leq u_k \leq \bar{u} \quad \forall k \\ & Ax \leq b \\ & x \in \{0, 1\}^n. \end{aligned}
\end{aligned}$$

If the local stress and nodal displacement constraints are removed from (WP) and (CP), two special cases are obtained. The resulting problem classes, which are interesting also in their own right, are relaxations of the original problems. Any relaxation of these problems also give rise to relaxations of (WP) and (CP), respectively. In the branch-and-cut framework,

we also consider the minimum weight problem with compliance constraints

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^n x_j \rho_j \\
& \begin{array}{l} x \in \mathbb{R}^n, \\ u_1, \dots, u_M \in \mathbb{R}^d \end{array} && \\
(\widetilde{\text{WP}}) \quad & \text{subject to} && \begin{array}{l} K(x)u_k = f_k \quad \forall k \\ f_k^T u_k \leq \bar{\gamma}_k \quad \forall k \\ Ax \leq b \\ x \in \{0, 1\}^n. \end{array}
\end{aligned}$$

and the minimum compliance problem with weight constraint

$$\begin{aligned}
& \text{minimize} && \frac{1}{2} \sum_{k=1}^M f_k^T u_k \\
& \begin{array}{l} x \in \mathbb{R}^n, \\ u_1, \dots, u_M \in \mathbb{R}^d \end{array} && \\
(\widetilde{\text{CP}}) \quad & \text{subject to} && \begin{array}{l} K(x)u_k = f_k \quad \forall k \\ \sum_{j=1}^n x_j \rho_j \leq V \\ Ax \leq b \\ x \in \{0, 1\}^n. \end{array}
\end{aligned}$$

3 A Branch-and-Bound algorithm

In what follows we describe two basic operation in a branch-and-bound algorithm that solve the problems considered to global optimality. These two operations are bound estimation and branching procedure.

3.1 Bound estimation

At each node along the search tree a relaxation need to be solved to obtain a lower bound. In consequence, a possibly large number of convex, or mildly non-convex, continuous problems are solved. These problem are obtained by relaxation and reformulation techniques.

3.1.1 Reformulation of first continuous relaxation

The continuous relaxation of $(\widetilde{\text{WP}})$ is obtained by relaxing the constraints $x \in \{0, 1\}^n$ to $x \in [0, 1]^n$. Reformulation techniques will be applied from the

resulting problem to obtain an all quadratic program as following, see [17]

$$\begin{aligned}
& \text{minimize} && \sum_{k=1}^M \begin{pmatrix} \alpha_k \\ z_k \end{pmatrix}^T \begin{pmatrix} \bar{\gamma}_k & f_k^T \\ f_k & K_0 \end{pmatrix} \begin{pmatrix} \alpha_k \\ z_k \end{pmatrix} \\
& \alpha_k \in \mathbb{R}, z_k \in \mathbb{R}^d, && + \eta^T b + \xi^T e \\
& \eta \in \mathbb{R}_+^m, \xi \in \mathbb{R}_+^n && \\
\text{(W-Q2P)} & \text{ subject to} && \sum_{k=1}^M z_k^T K_j z_k - a_j^T \eta - \xi_j \leq \rho_j \quad \forall j \\
& && \eta \geq 0, \xi \geq 0.
\end{aligned}$$

which fails to be convex but from a KKT point of (W-Q2P) we can get an optimal solution of the continuous relaxation as in the following lemma.

Lemma 1. *Let $(\alpha_1, z_1, \dots, \alpha_M, z_M, \eta, \xi)$ be a local or global minimizer of (W-Q2P) with $\alpha_k \neq 0$ for $k = 1, \dots, M$ and corresponding Lagrange multipliers $y \in \mathbb{R}_+^n$, $\tau \in \mathbb{R}_+^n$, and $\sigma \in \mathbb{R}_+^m$. Then (x, u_1, \dots, u_M) is optimal for continuous relaxation of (\widetilde{WP}) , where*

$$\begin{aligned}
x & := y + \underline{x} \\
u_k & := -z_k / \alpha_k \quad k = 1, \dots, M.
\end{aligned}$$

The proof of the Lemma 1 and additional results can be found in [17].

We get similar results when applying these techniques to the continuous relaxation of (\widetilde{CP}) . Nevertheless, the resulting all quadratic (C-Q2P) is a convex problem (\widetilde{CP}) , see [2, 3],

$$\begin{aligned}
& \text{minimize} && \frac{1}{2} \sum_{k=1}^M u_k^T K_0 u_k - \sum_{k=1}^n f_k^T u_k \\
& \alpha_k \in \mathbb{R}, z_k \in \mathbb{R}^d, && + \lambda V - r^T e + b^T \eta \\
& \eta \in \mathbb{R}_+^m, \xi \in \mathbb{R}_+^n && \\
\text{(C-Q2P)} & \text{ subject to} && \frac{1}{2} \sum_{k=1}^M u_k^T K_j u_k - \lambda \rho_j + r_j - a_j^T \eta \leq 0 \quad \forall j \\
& && r \leq 0, \lambda \geq 0, \eta \geq 0.
\end{aligned}$$

A lemma concerning the coupling between optimal solutions of (C-Q2P) and continuous relaxation of (\widetilde{CP}) follows

Lemma 2. *Let $(u_1, \dots, u_M, \lambda, \eta)$ be optimal for (C-Q2P) with corresponding Lagrange multipliers τ_j , $j = 1, \dots, n$, for the quadratic constraints. Then (x, u_1, \dots, u_M) is optimal for continuous relaxation of (\widetilde{CP}) , where*

$$x := \underline{x} + \tau$$

3.1.2 Second relaxation

The second class of relaxations of the discrete problem (WP) follows the approaches outlined in [8], [24], and [25]. The reformulation requires additional assumptions on the element stiffness matrices and a more precise definition of the stress constraints. We will not repeat all the main hypotheses and the reformulation procedure which are presented in [17]. The resulting relaxation is a convex program with linear objective function

$$\begin{aligned}
& \text{minimize} && \rho^T x \\
& \begin{array}{l} x \in \mathbb{R}^n, u_k \in \mathbb{R}^d \\ q_1 \in \mathbb{R}^{r_1}, \dots, q_n \in \mathbb{R}^{r_n} \end{array} && \\
& \text{subject to} && \sum_{j=1}^n B_j^T(q_k)_j = f_k \quad \forall k \\
& && x_j \underline{c} \leq (q_k)_j \leq x_j \bar{c} \quad \forall k \quad \forall j \\
& && E_j B_j u_k - (q_k)_j \geq (1 - x_j) \underline{c} \quad \forall k \quad \forall j \\
& && E_j B_j u_k - (q_k)_j \leq (1 - x_j) \bar{c} \quad \forall k \quad \forall j \\
& && f_k^T u_k \leq \bar{\gamma}_k \quad \forall k \\
& && (q_k)_j^T \hat{V}_j^T \hat{V}_j (q_k)_j \leq x_j \bar{\sigma}^2 \quad \forall k \quad \forall j \\
& && \underline{u} \leq u_k \leq \bar{u} \quad \forall k \\
& && Ax \leq b \\
& && x_j \in [\underline{x}_j, \bar{x}_j] \quad \forall j.
\end{aligned}$$

This relaxation has a large size both in terms of number of variables and constraints. But the size of the subproblems (both in term of variables and constraints) will decrease with increasing depth in a branch-and-bound search tree. By introducing the index sets $\mathcal{N} = \{1, \dots, n\}$, $\mathcal{N}_0 = \{j \in \mathcal{N} \mid \underline{x}_j = \bar{x}_j = 0\}$, and $\mathcal{N}_1 = \{j \in \mathcal{N} \mid \underline{x}_j = \bar{x}_j = 1\}$, i.e., the set of all design variables fixed to zero and one, respectively. We arrive at the condensed relaxation

$$\begin{aligned}
& \text{minimize} && \rho^T x \\
& \begin{array}{l} x \in \mathbb{R}^n, u_k \in \mathbb{R}^d \\ q_1 \in \mathbb{R}^{r_1}, \dots, q_n \in \mathbb{R}^{r_n} \end{array} && \\
\text{(RW)} & \text{subject to} && K(\underline{x})u_k + \sum_{j \notin \mathcal{N}_0 \cup \mathcal{N}_1} B_j^T(q_k)_j = f_k \quad \forall k \\
& && x_j \underline{c} \leq (q_k)_j \leq x_j \bar{c} \quad \forall k \quad \forall j \notin \mathcal{N}_0 \cup \mathcal{N}_1 \\
& && E_j B_j u_k - (q_k)_j \geq (1 - x_j) \underline{c} \quad \forall k \quad \forall j \notin \mathcal{N}_0 \cup \mathcal{N}_1 \\
& && E_j B_j u_k - (q_k)_j \leq (1 - x_j) \bar{c} \quad \forall k \quad \forall j \notin \mathcal{N}_0 \cup \mathcal{N}_1 \\
& && f_k^T u_k \leq \bar{\gamma}_k \quad \forall k \\
& && (u_k)^T W_j u_k \leq \bar{\sigma}^2 \quad \forall k \quad \forall j \in \mathcal{N}_1 \\
& && (q_k)_j^T \hat{V}_j^T \hat{V}_j (q_k)_j \leq x_j \bar{\sigma}^2 \quad \forall k \quad \forall j \notin \mathcal{N}_0 \cup \mathcal{N}_1 \\
& && \underline{u} \leq u_k \leq \bar{u} \quad \forall k \\
& && Ax \leq b \\
& && x_j \in [\underline{x}_j, \bar{x}_j] \quad \forall j.
\end{aligned}$$

When working with minimum compliance problem, we arrive at the relaxation (RC) as following

$$\begin{aligned}
& \underset{\substack{x \in \mathbb{R}^n, u_k \in \mathbb{R}^d \\ q_1 \in \mathbb{R}^{r_1}, \dots, q_n \in \mathbb{R}^{r_n}}}{\text{minimize}} && \frac{1}{2} \sum_{k=1}^M f_k^T u_k \\
\text{(RC)} \quad & \text{subject to} && K(\underline{x})u_k + \sum_{j \notin \mathcal{N}_0 \cup \mathcal{N}_1} B_j^T (q_k)_j = f_k \quad \forall k \\
& && x_j \underline{c} \leq (q_k)_j \leq x_j \bar{c} \quad \forall k \quad \forall j \notin \mathcal{N}_0 \cup \mathcal{N}_1 \\
& && E_j B_j u_k - (q_k)_j \geq (1 - x_j) \underline{c} \quad \forall k \quad \forall j \notin \mathcal{N}_0 \cup \mathcal{N}_1 \\
& && E_j B_j u_k - (q_k)_j \leq (1 - x_j) \bar{c} \quad \forall k \quad \forall j \notin \mathcal{N}_0 \cup \mathcal{N}_1 \\
& && \sum_{j=1}^n \rho_j x_j \leq V \\
& && (u_k)^T W_j u_k \leq \bar{\sigma}^2 \quad \forall k \quad \forall j \in \mathcal{N}_1 \\
& && (q_k)_j^T \hat{V}_j^T \hat{V}_j (q_k)_j \leq x_j \bar{\sigma}^2 \quad \forall k \quad \forall j \notin \mathcal{N}_0 \cup \mathcal{N}_1 \\
& && \underline{u} \leq u_k \leq \bar{u} \quad \forall k \\
& && Ax \leq b \\
& && x_j \in [\underline{x}_j, \bar{x}_j] \quad \forall j.
\end{aligned}$$

3.2 Branching procedure using binary subdivision

Binary subdivision is apply evidently from the binary property of design variables. We discuss in detail about some proposed node selection rules and branching rules.

3.2.1 Node selection rules

Some standard node selection rules are proposed in this framework. These are Best-First, Depth-First and Breath-First. Each of these rule has its own advantages as well as disadvantages. We do not mention in detail here since they can be found in almost basic documents about branch-and-bound method.

3.2.2 Branching rules

Two branching rules which are proposed to use in the branch-and-bound algorithm are most infeasible branching and pseudocost branching. Some other advance heuristics can be found in [1].

Most infeasible branching

This still very common rule chooses the variable with the fractional part closest to 0.5, i.e., index $i^* = \min_i |x_i - 0.5|$ is chosen. The heuristic reason behind this choice is that this selects a variable where the least tendency can be recognized to which "side" (up or down) the variable should be rounded.

Pseudo cost branching

This is a sophisticated rule in the sense that it keeps a history of the success of variables on which already has been branched. This rule goes back to [6].

4 Cuts

A general property to achieve good convergence of Branch-and-Bound algorithms is to have a strong representation of the problem being solved. One way of obtaining this is by introducing cuts. Cuts are valid inequalities which cut away a given point, in our case a candidate design, which is not an optimal solution. We use here the recently proposed Combinatorial Benders' cuts as described in [10] and [15]; and a simple cut that we denote a functional cut.

4.1 Combinatorial Benders' Cuts

Combinatorial Benders' cuts are successfully used for discrete truss topology optimization problem in [21]. This class of cuts builds on the principle of deriving cuts from minimal sets of inconsistencies. Before discussing how these cut are generated we reformulate the equilibrium constraints of problem (WP) (or (CP)) as conditional linear constraints.

By introducing an additional continuous variables $(z_k)_j \in \mathbb{R}^d$ with the interpretation $(z_k)_j = x_j u_k$, the equilibrium equations for load f_k are rewritten as

$$K(x)u_k = \sum_{j=1}^n x_j K_j u_k = \sum_{j=1}^n K_j (z_k)_j = f_k$$

The bilinear terms $z_j = x_j u$ can be written as the set of conditional linear equalities

$$\begin{aligned} x_j = 0 &\rightarrow (z_k)_j = 0 \\ x_j = 1 &\rightarrow (z_k)_j = u_k \end{aligned}$$

Given a candidate design x , which may contain non-integer entries, we denote $\mathcal{N}_0(x)$ and $\mathcal{N}_1(x)$ the index-sets of design variables with value 0 and 1, respectively, i.e.,

$$\begin{aligned} \mathcal{N}_0(x) &:= \{j \in \{1, 2, \dots, n\} \mid x_j = 0\} \\ \mathcal{N}_1(x) &:= \{j \in \{1, 2, \dots, n\} \mid x_j = 1\} \end{aligned}$$

Consider the following linear system

$$\begin{aligned} \sum_{j=1}^n K_j (z_k)_j &= f_k \\ (z_k)_j - u_k &= 0 \quad \forall j \in \mathcal{N}_1(x), \\ (z_k)_j &= 0 \quad \forall j \in \mathcal{N}_0(x). \end{aligned} \tag{1}$$

If the linear system (1) is infeasible, at least one binary variable x_j ($j \in \mathcal{N}_1(x) \cup \mathcal{N}_2(x)$) has to be changed to remove, probably, the infeasibility. Nevertheless to get a strong cut, we look for an irreducible (or minimal) infeasible subsystem (IIS) involving the rows of the system defined by the index-sets $\mathcal{N}_0(x)$ and $\mathcal{N}_1(x)$. Denote such an IIS by \mathcal{S} . A linear inequality, called Combinatorial Benders' cut, is written as

$$\sum_{j \in \mathcal{S}: x_j=0} x_j + \sum_{i \in \mathcal{S}: x_j=1} (1 - x_j) \geq 1$$

which can be added to the formulation. For each candidate design x one or more cuts can be generated depending on how many IISs are found. Finding an IIS is in practice done by solving a linear program by Simplex method as described in [20].

We need to investigate the feasibility of k linear systems such as (1). This task should be done in an efficient way. In practice, for each load case a reduced system is considered. We work in the systems where all the zero variables are eliminated and all the one variables are grouped as unique variables, i.e., we work with the systems

$$\sum_{j \in \mathcal{N}_1(x)} K_j z_k + \sum_{j \notin \mathcal{N}_0(x) \cup \mathcal{N}_1(x)} K_j (z_k)_j = f_k \quad \forall k$$

Last but not least we might find, at some nodes of the search tree, a new cut stronger than some cuts existed in the cut pool. The new cut will be added into the pool in parallel with removing the weaker previous ones, we call removing the redundant cuts.

4.2 Functional cut

The goal of this type of cuts is to avoid a feasible known design $\hat{x} \in \{0, 1\}^n$ being found again even, in some cases, to assure that the new feasible design is not worse than the previous one. For a given design vector $\hat{x} \in \{0, 1\}^n$, a feasible design x is better than \hat{x} if and only if its corresponding objective function, of minimum weight problem, is smaller, i.e,

$$\rho^T x < \rho^T \hat{x}$$

Equivalently, this condition can be written as

$$\begin{aligned} \sum_{i: \hat{x}_i=0} \rho_i x_i + \sum_{i: \hat{x}_i=1} \rho_i (x_i - 1) &< 0 \\ \Rightarrow \sum_{i: \hat{x}_i=0} \lfloor \rho_i \rfloor x_i + \sum_{i: \hat{x}_i=1} \lceil \rho_i \rceil (x_i - 1) &< 0 \end{aligned}$$

Finally, by the integrality of the design variables, we arrive at the following inequality, called a functional cut,

$$\sum_{i:\hat{x}_i=0} \lfloor \rho_i \rfloor x_i + \sum_{i:\hat{x}_i=1} \lceil \rho_i \rceil (x_i - 1) \leq -1 \quad (2)$$

The functional cut (2) can be constructed only when a feasible design is available. If the densities ρ_i are integer, the new feasible design, if found, is always better than the current one. It should be noted that, with these cuts, we never find the same design twice.

5 Heuristics

Finding good feasible points is important when solving this class of problems by branch-and-cut methods since this gives possibilities to fathom, hopefully large, parts of the feasible set. In the present implementation, a good feasible point, and its corresponding upper bound, is also important for generating cuts and for fixing variables at node in the search tree. Hence several heuristics are implemented and used throughout the search.

5.1 Rounding heuristic

The rounding heuristic is based on a simple rounding of design variables obtained after solving the continuous relaxations (W-Q2P) and (C-Q2P) (or (RW) and (RC)). This simple rounding heuristic is used in [3] within a nonlinear branch-and-bound method for solving discrete truss topology optimization problems and the numerical experience indicated that the heuristic, although simple, often finds feasible point. We present in the following the algorithms for the problems without stress and displacement constraints.

It is well-known that the equilibrium constraints are more potentially satisfied if we can distribute as much material as possible (at least if the resulting structure is connected and can carry the external forces). Given the optimal solution to the natural continuous relaxation of (WP) associated with the node. The largest design variables are rounded to one and the remaining to zero in such a way that the volume is strictly less than the objective of the incumbent (if one exists). Let ub denote the upper bound, i.e., the objective value of the incumbent. If an incumbent is not yet known, ub can safely be chosen as $ub = \sum_j \rho_j \bar{x}_j$. This rounding gives a candidate design vector \tilde{x} . If $A\tilde{x} \leq b$ then an attempt is made to find displacement vectors satisfying the equilibrium equations. This rounding heuristic requires one assembly of the stiffness matrix and (at most) M attempts to solve the (linear) equilibrium equations. This heuristic is presented in detail in Algorithm 1.

Algorithm 1: A simple rounding heuristic for the minimum weight problem ($\widetilde{\text{WP}}$).

Given the vectors \underline{x} and \bar{x} and the upper bound ub .

Let $(x^*, u_1^*, \dots, u_M^*)$ denote the solution to the relaxation of ($\widetilde{\text{WP}}$).

Sort the entries in x^* in descending order.

Let the \mathcal{I} denote the indices after sorting.

Set $\mathcal{N}_0 = \mathcal{N}_1 = \emptyset$.

for $j = 1, \dots, n$ **do**

if $\sum_{j \in \mathcal{N}_1 \cup \{\mathcal{I}(j)\}} \rho_j < ub$ **then**

$\mathcal{N}_1 \leftarrow \mathcal{N}_1 \cup \{\mathcal{I}(j)\}$ and $\tilde{x}_j = 1$.

else

$\mathcal{N}_0 \leftarrow \mathcal{N}_0 \cup \{\mathcal{I}(j)\}$ and $\tilde{x}_j = 0$.

end

end

if $A\tilde{x} \leq b$ **then**

 Attempt to find vectors $\tilde{u}_1, \dots, \tilde{u}_M$ such that $K(\tilde{x})\tilde{u}_k = f_k$ and

$f_k^T \tilde{u}_k \leq \gamma_k$.

end

The proposed rounding heuristic can also be modified for the discrete minimum weight problem ($\widetilde{\text{WC}}$). The different thing is that the limit of volume of distributed material is not changed along the search tree. Moreover for compliance any any displacement vector satisfying equilibrium will suffice since compliance is constant for all displacement vectors satisfying equilibrium. This heuristic is presented in more detail in Algorithm 2.

Algorithm 2: A simple rounding heuristic for the minimum compliance problem ($\widetilde{\text{WC}}$).

Given the vectors \underline{x} and \bar{x} and the scalar $V > 0$.
Let $(x^*, u_1^*, \dots, u_M^*)$ denote the solution to the relaxation ($\widetilde{\text{WC}}$).
Sort the entries in x^* in descending order.
Let the \mathcal{I} denote the indices after sorting.
Set $\mathcal{N}_0 = \mathcal{N}_1 = \emptyset$.
for $j = 1, \dots, n$ **do**
 if $\sum_{j \in \mathcal{N}_1 \cup \{\mathcal{I}(j)\}} \rho_j \leq V$ **then**
 $\mathcal{N}_1 \leftarrow \mathcal{N}_1 \cup \{\mathcal{I}(j)\}$ and $\tilde{x}_j = 1$.
 else
 $\mathcal{N}_0 \leftarrow \mathcal{N}_0 \cup \{\mathcal{I}(j)\}$ and $\tilde{x}_j = 0$.
 end
end
if $A\tilde{x} \leq b$ **then**
 Attempt to find vectors $\tilde{u}_1, \dots, \tilde{u}_M$ such that $K(\tilde{x})\tilde{u}_k = f_k$ and
 $f_k^T \tilde{u}_k \leq \gamma_k$.
end

Similar rounding heuristics can also be applied to problems with stress constraints (WP) and (CP). After rounding the design variables to the candidate design vector \tilde{x} , suitable displacement vectors can be found by solving the continuous relaxation or its dual formulation again for fixed $x = \tilde{x}$. If multiple load conditions are considered then the problem of finding suitable displacement vectors decomposes into M smaller problems.

This type of heuristics are easy to implement, they are fast, and they use some knowledge of the mechanical background of the optimization problems. They can be periodically called while searching the branch-and-bound tree.

5.2 Hard fixing heuristic

A slightly more elaborate class of heuristics are based on hard fixing and diving, i.e., fixing an increasing number of variables and resolving the continuous relaxation. Given an optimal solution to one of the natural continuous relaxations of either (WP) or (CP) at some node in the search tree (for example the root node) some of the variables are rounded and fixed. The relaxation is then resolved. This procedure is iterated until the relaxation becomes infeasible, the objective function becomes worse than the current incumbent, or until a feasible point is found. If the variable fixing is appropriately done, this heuristic will terminate after a finite number of iterations (= relaxations solved). Similar strategies are suggested in [7] for linear mixed integer programs. For topology optimization problems we round all design variables with value ≥ 0.9 and the largest non-fixed variable to one

unless the new volume is greater than the incumbent, working with (WP), or this violates the weight constraint, working with (CP). Design variable with value ≤ 0.01 and the smallest non-fixed value is rounded to zero. This hard-fixing diving heuristic is presented in Algorithm 3. The reluctance to round variables to zero is based on the observation that the structure must be connected to carry the external loads.

Algorithm 3: A hard-fixing and diving heuristic for the discrete minimum weight topology optimization problem (WP).

Let $\underline{x} := 0$ and $\bar{x} := e$ and set $\mathcal{N}_0 := \mathcal{N}_1 := \emptyset$ and $\tilde{x} := 0$.

Let `done` := `false`.

while `not(done)` **do**

 Attempt to solve relaxation of (WP).

if (R-1) is infeasible **then**

`done` \leftarrow `true`.

else if the optimal value of (WP) $> ub$ **then**

`done` \leftarrow `true`.

else

 Let $(x^*, u_1^*, \dots, u_M^*)$ denote the optimal solution to the (WP).

if $x^* \in \{0, 1\}^n$ **then**

$(x^*, u_1^*, \dots, u_M^*)$ is a feasible point to (WP).

`done` \leftarrow `true`.

else

forall $j \in \mathcal{N} \setminus (\mathcal{N}_0 \cup \mathcal{N}_1)$ **do**

if $x_j^* \geq 0.9$ and $\sum_{k \in \mathcal{N}_1 \cup \{j\}} \rho_k < ub$ **then**

$\mathcal{N}_1 \leftarrow \mathcal{N}_1 \cup \{j\}$ and $\underline{x}_j = \tilde{x}_j = 1$.

else if $x_j^* \leq 0.01$ **then**

$\mathcal{N}_0 \leftarrow \mathcal{N}_0 \cup \{j\}$ and $\bar{x}_j = \tilde{x}_j = 0$.

end

end

 Let $j = \arg \max_{j \in \mathcal{N} \setminus (\mathcal{N}_0 \cup \mathcal{N}_1)} x_j^*$.

if $\sum_{k \in \mathcal{N}_1 \cup \{j\}} \rho_k \leq ub$ **then**

$\mathcal{N}_1 \leftarrow \mathcal{N}_1 \cup \{j\}$ and $\underline{x}_j = \tilde{x}_j = 1$.

end

 Let $j = \arg \min_{j \in \mathcal{N} \setminus (\mathcal{N}_0 \cup \mathcal{N}_1)} x_j^*$.

$\mathcal{N}_0 \leftarrow \mathcal{N}_0 \cup \{j\}$ and $\bar{x}_j = \tilde{x}_j = 0$.

end

end

end

The heuristic in Algorithm 3 is, of course, computationally more expensive than the rounding heuristics presented in Algorithm 2 and 1. The main objective is to call this procedure only once at the root node of the

branch-and-bound tree (or even prior to calling the branch-and-cut or local branching methods) to find the first feasible point. The first incumbent is necessary in order to call the more advanced heuristics based on local branching and RINS. The simple rounding heuristics in Algorithms 2 and 1 are less likely to find feasible point when called at the root node. Topology optimization problems are notoriously well-known to produce solutions to the relaxations which contain many gray elements, i.e., solutions with many non integer design variables, and solutions which do not approximate the optimal discrete solution well.

5.3 RINS heuristics

Relaxation induced neighborhood search (RINS) is suggested in [12] as a heuristic to improve feasible points, \hat{x} , to mixed integer programs. At some node in the branch and bound tree the following steps are performed. The variables which have the same value in the optimal solution to the (current) relaxation and the incumbent are fixed. An objective cutoff based on the objective value of the incumbent is added as a constraint to the problem. The resulting reduced mixed integer program is solved by a branch-and-cut method. In [12] the search is not restricted to the part of the search tree currently under consideration.

It should notice that that sub-problem is also a difficult problem. So in our heuristics, we modify slightly the original idea. We first round the optimal solution, x^* , to the relaxation to get \tilde{x} . We then apply RINS from \tilde{x} . The idea is that we hope with \tilde{x} the number of integer variables in the sub-problems is much reduced if we take x^* . We also impose a node limit on the method for mixed integer sub-problems.

This heuristic requires that a feasible solution is known but can otherwise be invoked at every node in the branch-and-bound tree.

5.4 Local branching heuristic

Given a feasible reference point $(\hat{x}, \hat{u}_1, \dots, \hat{u}_M)$, for example the current incumbent, of the minimum weight problem (WP) the m -OPT neighborhood $\mathcal{N}(\hat{x}, m)$ of \hat{x} is the set of feasible designs of (WP) satisfying the additional local branching constraint

$$\Delta(x, \hat{x}) = \sum_{j:\hat{x}_j=1} (1 - x_j) + \sum_{j:\hat{x}_j=0} x_j \leq m. \quad (3)$$

The two sums in (3) count the number of binary variables in x which have flipped their values (with respect to \hat{x}) either from one to zero or from zero to one. Given the incumbent $(\hat{x}, \hat{u}_1, \dots, \hat{u}_M)$ the feasible set of (WP) can be partitioned by the disjunction

$$\Delta(x, \hat{x}) \leq m \text{ (left branch)} \quad \text{or} \quad \Delta(x, \hat{x}) \geq m + 1 \text{ (right branch)} .$$

The neighborhood size m should be sufficiently small such that the left problem, i.e., (WP) with the additional constraint $\Delta(x, \hat{x}) \leq m$ is much easier to solve than (WP) itself. The neighborhood size m should ideally also be chosen large enough such that the neighborhood $\mathcal{N}(\hat{x}, m)$ is likely to contain better points than the current incumbent. The left branch can be solved by, for example, a convergent branch-and-cut method developed for solving (WP). Notice that the local branching constraints can be included in the linear constraints $Ax \leq b$. If the optimal solution in the left branch improves on the incumbent this solution becomes the new incumbent. The approach can then be re-applied on the right branch.

Local branching method becomes exact if all branches are solved to global optimality. However solving the left and the right-most branches can be very time consuming. We propose therefore a heuristic as follows. When solving the left branches we impose a node limit. If the tactical solver reach the limit, we then reduce the neighborhood size. The right -most branch will not be solved in our heuristic. The heuristic local branching algorithm presented in Algorithm 4 resembles the algorithm described in [13].

Algorithm 4: A local branching heuristic for improving the incumbent to (WP).

Given the incumbent $(\hat{x}, \hat{u}_1, \dots, \hat{u}_M)$ and the integer $m > 0$.
Let `done := false` and let $p := m$.

while *not*(`done`) **do**

/* Generate the left branch node. */

Add the constraint $\Delta(x, \hat{x}) \leq p$ to (WP).

Attempt to solve problem (WP) within node limits.

if (P-1) is proven infeasible **then**

`done := true`.

else if optimal solution found **then**

Reverse the constraint $\Delta(x, \hat{x}) \leq p$ to $\Delta(x, \hat{x}) \geq p + 1$.

Let $(x^*, u_1^*, \dots, u_M^*)$ denote the optimal solution to (WP).

if $\rho^T x^* < \rho^T \hat{x}$ **then**

/* Update the incumbent. */

$(\hat{x}, \hat{u}_1, \dots, \hat{u}_M) \leftarrow (x^*, u_1^*, \dots, u_M^*)$.

Let $p := m$.

else

`done := true`.

end

else if feasible point found **then**

Let $(x^*, u_1^*, \dots, u_M^*)$ denote the found feasible point to (WP).

Delete the last local branching constraint $\Delta(x, \hat{x}) \leq p$.

if $\rho^T x^* < \rho^T \hat{x}$ **then**

/* Update the incumbent. */

$(\hat{x}, \hat{u}_1, \dots, \hat{u}_M) \leftarrow (x^*, u_1^*, \dots, u_M^*)$.

Let $p := m$.

else

Let $p := p - \lceil p/2 \rceil$.

end

else if no feasible point found **then**

Delete the last local branching constraint $\Delta(x, \hat{x}) \leq p$.

Let $p := p - \lceil p/2 \rceil$.

end

end

6 Design and Implementation

The branch-and-cut method for solving the discrete topology design problem and related finite element routines are implemented in C++ using Visual Studio 2005, Version 8.0. For the considered problem classes and the chosen methods the vast majority of the computation time is spent on solving the relaxations and solving linear programs for generating Benders' cuts. The

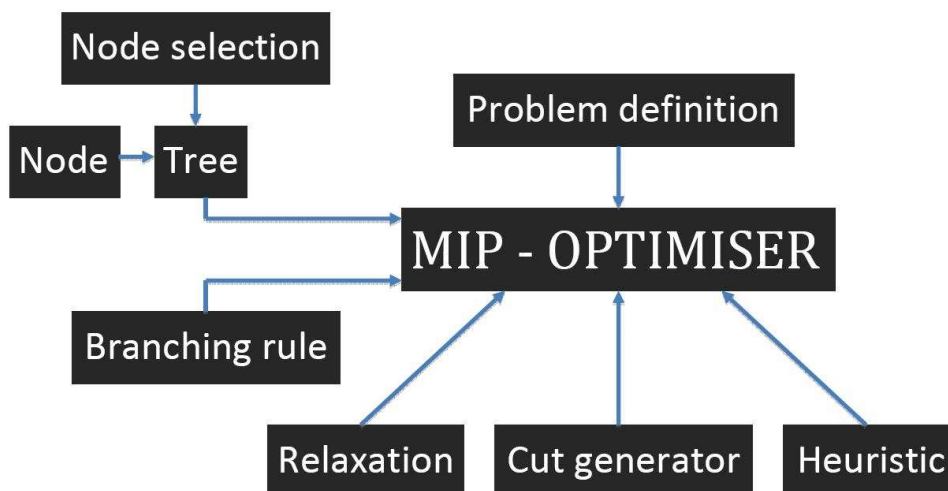


Figure 1: Design chart of the solver in PLATO-N project

routines for solving these problems are state-of-the art routines implemented in C++ using object oriented design.

The design is realized in a way that can be easily modified and extended. The system in detail is presented in the Figure 1. The common objects of parts in the framework is unique. They are refereed by a reference or a pointer. The pool of cut is also design as an array of pointer so the basic operations (the ones which are often called) such as adding, removing, replacing, ... can be done easily and quickly. The number of nodes in the search tree is possibly (and often) large. We just storage the information about fixed variables at a node.

The KKT point of the all quadratic programs (W-Q2P) and (C-Q2P) are found by using PENNLP solver, a smooth nonlinear optimization solver [16]. PENNLP is also used to solve the convex problems (RW) and (RC). The linear programs which appear as part of the Benders' cut generation or finding displacements for a candidate design are solved by Simplex solvers in CLP [14].

We try to generate Combinatorial Benders' cuts at every node whose depth, i.e., number of fixed variables, in search tree is smaller than quarter of number of design variable. A Benders' cut is generated from the integer part of designed variables obtained as part of optimal solution to the relaxations after rounding all variables in $[0.25, 0.75]$. At each time, we try to generate at most ten cuts (strategy of generating cut is still an open question). A cut is said to be efficient if the number of variables constructing this cut is less or equal to the square root of the number of design variables. The cuts are stored in a global cut pool, described by the system of linear inequality $Ax \leq b$. An efficient cut is included in the cut pool only if it is not already present and it is not redundant by previous cuts .

We limit the number of iteration for PENNLP to 100 at each time launching. If PENNLP reach this limit, we increase the limit to 1000 and restart PENNLP.

A point (x, u_1, \dots, u_M) is considered numerically feasible with respect to the general constraints if the constraint violation is not larger than the prescribed feasibility tolerance $\delta > 0$. Particularly, (x, u_k) is considered feasible to the equilibrium constraint for load k-th if

$$\|K(x)u_k - f_k\|_2 / \|f_k\|_2 \leq \delta$$

The point (x, u) is considered integer feasible if x also satisfies

$$x_j \in [0, \delta] \quad \text{or} \quad x_j \in [1 - \delta, 1] \quad \forall j$$

In the implementation $\delta = 10^{-7}$. An integer feasible point is set to be optimal if either the relative or absolute optimality gap are smaller than the optimality tolerance 5^{-3} .

7 Numerical experiments

We present some preliminary numerical experiments with the outlined branch-and-cut method applied to instances of the multiple load problem ($\widetilde{\text{WC}}$) and ($\widetilde{\text{WP}}$).

The first instance is the modified version of Zhou and Rozvany example [26]. The design domain and the two external loads are illustrated in Figure 2(a). The construction material is isotropic with Young's modulus equal to

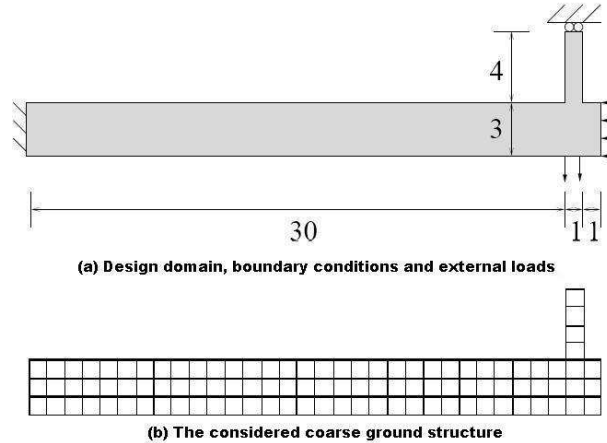


Figure 2: The modified Zhou and Rozvay example

one ($E := 1$) and the Poisson's ratio equal to zero ($\mu := 0$). The design

variable x_j represents the thickness of the j^{th} element and must only attain the values unity or zero. We have two external loads. The vertical load applied at the lower side of the design domain gives two nonzero entries, both with value $-1/2$. The horizontal load is modeled with four nonzero entries. The node forces at the corners are both set to -1 while the two inner nodes have forces equal to -2 . The finite elements used in the computations are bilinear iso parametric element in plane stress. The stiffness matrices are computed using 2nd order Gauss quadrature rule, see [11]. This is the same type of elements which are used in [22]. The design domain is partitioned into square elements of equal size, Figure 2(b).

Figure 2(c) illustrate the global optimal design obtained. The parameter computations are summarized in the Table 1.

n	d	M	$\sum_k \bar{\gamma}_k$	Rel.	Opt.Val.	Itn.	Cuts
100	270	2	565	68.72526	72	1498	38

Table 1: Numerical result of modified Rhou and Rozvany example

The second example is about beam design optimization. We take the same values for Young's modulus and Poisson's ratio. We also have two external with the value -1 . The design domain and the two external loads are plotted in Figure 3(a). Figure 3(b) illustrates the partitioned domain into square element size.

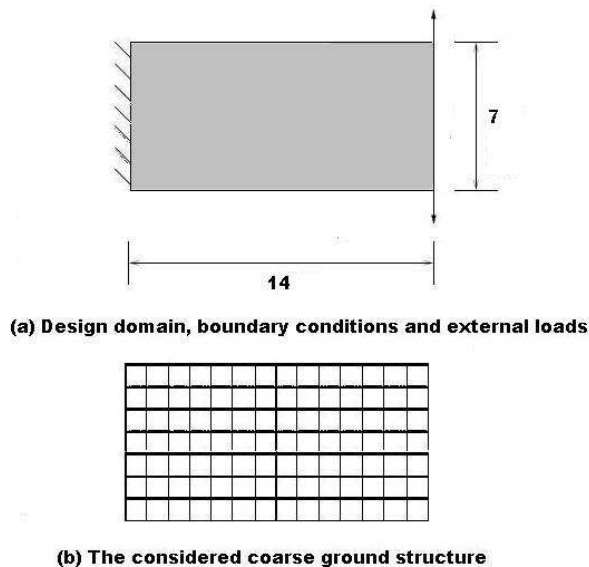


Figure 3: The beam example

Figure 3(c) illustrate the global optimal design obtained. The parameter computations are summarized in the Table 2.

n	d	M	$\sum_k \bar{\gamma}_k$	Rel.	Opt.Val.	Itn.	Cuts
98	224	2	93	74.53905	80	19986	45

Table 2: Numerical result of beam example

8 Summary

We have presented some of the most important aspects of a non-linear branch and cut method for solving structural topology optimization problems that are considered in the PLATO-N project.

Preliminary results show that the method gives a global optimal solution to all test problems.

References

- [1] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33:42–54, 2005.
- [2] W. Achtziger, M.P. Bendsøe, A. Ben-Tal, and J. Zowe. Equivalent displacement based formulations for maximum strength truss topology design. *Impact of Computing in Science and Engineering*, 4:315–345, 1992.
- [3] W. Achtziger and M.Stolpe. Truss topology optimization with discrete design variables - guaranteed global optimality and benchmark examples. *Structural and Multidisciplinary Optimization*, 34(1):1–20, 2007.
- [4] W. Achtziger and M.Stolpe. Global optimization of truss topology with discrete bar areas-part i: theory of relaxed problems. *Computational Optimization and Applications*, 40:247–280, 2008.
- [5] M.P. Bendsøe and O. Sigmund. *Topology Optimization — Theory, Methods and Applications*. Springer, 2003.
- [6] M. Benichou, J.M.Gauthier, P.Girodet, and G.Hengtges. Experiments in mixed-integer programming. *Math Programming*, 1:76 – 94, 1971.
- [7] R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice – closing the gap. In M.J.D. Powell and S. Scholtes, editors, *System Modelling and Optimization: Methods, Theory, and Applications*, pages 19 – 49. Kluwer Academic Publishers, 2000.

- [8] S. Bollapragada, O. Ghattas, and J.N. Hooker. Optimal design of truss structures by logical-based branch and cut. *Operations Research*, 49(1):42–51, 2001.
- [9] N.Nguyen Canh and M. Stolpe. Mathematical modelling of multiple load structural topology design problems. Technical Report PLATO-N Public Report PU-R-3-2008, EU project FP-6 STREP 30717 PLATO-N (Aeronautics), Report available on www.plato-n.org, April 2008.
- [10] G. Codato and M. Fischetti. Combinatorial benders’ cuts for mixed integer linear programming. *Operations Research*, 54(4):756–766, 2006.
- [11] R.D. Cook, D.S. Malkus, and M.E. Plesha. *Concepts and applications of finite element analysis*. John Wiley & Sons, 3rd edition, 1989.
- [12] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102:71 – 90, 2005.
- [13] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23 – 47, 2003.
- [14] J. Forrest. *CLP User Guide*. <https://projects.coin-or.org/Clp>, 2004.
- [15] J. Hooker. *Logic-based methods for optimization*. John Wiley & Sons, 2000.
- [16] M. Kočvara and M. Stingl. *PENNLN User’s guide*. www.penopt.com, 2005.
- [17] M.Stolpe, R.Stainko, and M. Kočvara. Lower bounding problems for stress constrained discrete structural topology optimization problems. Technical Report PLATO-N Public Report PU-R-8-2007, EU project FP-6 STREP 30717 PLATO-N (Aeronautics), Report available on www.plato-n.org, September 2007.
- [18] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1998.
- [19] P.M. Pardalos and M.G.C. Resende, editors. *Handbook of Applied Optimization*. Oxford University Press, 2000.
- [20] M. Parker and J. Ryan. Finding the minimum weight iis cover of an feasible systems of linear inequalities. *Annals of Mathematics ans Artificial Intelligence*, 17:107–126, 1996.
- [21] M.H. Ramussen and M.Stolpe. Global optimization of large-scale topology design problems using branch-and-bound methods. *Computers and Structures*, ,in press, 2008.

- [22] O. Sigmund. A 99 line topology optimization code written in matlab. *Structural and Multidisciplinary Optimization*, 21(2):120–127, 2001.
- [23] M. Stolpe. Global optimization of minimum weight truss topology problems with stress, displacement, and local buckling constraints using branch-and-bound. *International Journal for Numerical Methods in Engineering*, 61(58):1270–1309, 2004.
- [24] M. Stolpe. On the reformulation of topology optimization problems as linear or convex quadratic mixed 0-1 programs. *Optimization and Engineering*, 8:163–192, 2007.
- [25] M. Stolpe and K. Svanberg. Modeling topology optimization problems as linear mixed 0–1 programs. *International Journal for Numerical Methods in Engineering*, 57(5):723–739, 2003.
- [26] M. Zhou and G. Rozvany. On the validity of eso type methods in topology optimization. *Structural and Multidisciplinary Optimization*, 1(21):80–83, 2001.