THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :
Pierre-Jean BARJHOUX

le vendredi 31 janvier 2020

Titre :

Towards efficient solutions for large scale structural optimization problems with categorical and continuous mixed design variables

Vers une résolution efficace de problèmes d'optimisation de structures de grande taille avec variables mixtes catégorielles et continues

---

École doctorale et discipline ou spécialité :

ED AA : Mathématiques et Applications et Génie mécanique, mécanique des matériaux

Unité de recherche :

Institut Clément Ader

Directeur(s) de Thèse :

M. Joseph MORLIER (directeur de thèse)
M. Youssef DIOUANE (co-directeur de thèse)

Jury :

Mme Sonia CAFIERI Professeure ENAC - Présidente
M. Youssef DIOUANE Professeur Associé ISAE-SUPAERO - Co-directeur de thèse
M. Pierre DUYSINX Professeur Ordinaire Université de Liège - Rapporteur
M. Michael KOKKOLARAS Professeur McGill University - Rapporteur
M. Marco MONTEMURRO Professeur Associé Arts et Métiers ParisTech
M. Joseph MORLIER Professeur ISAE-SUPAERO - Directeur de thèse

Remerciements

Je souhaite tout d'abord remercier M. Pierre Duysinx et M. Michael Kokkolaras d'avoir accepté d'évaluer ce travail de thèse en tant que rapporteurs, ansi que Mme Sonia Cafieri et M. Marco Montemurro pour leur rôle d'examinateurs dans le jury de thèse.

Je souhaite également exprimer toute ma reconnaissance envers Anne Gazaix à qui je dois l'opportunité d'avoir pu réaliser cette thèse, pour son accueil au sein du projet MDA-MDO à l'IRT Saint Exupéry. Je la remercie également très chaleureusement pour sa bienveillance et sa disponibilité tout au long de cette aventure. J'adresse également mes remerciements à mes encadrants de thèse qui ont su me guider tout au long de la thèse. Je remercie ainsi mon directeur de thèse Joseph Morlier ainsi que mon co-directeur Youssef Diouane pour tous leurs conseils, leur soutien et leur disponibilité au quotidien, notamment lors des phases de rédaction. Je remercie également très chaleureusement mon encadrant industriel Stéphane Grihon, pour le partage de sa vision scientifique, et son engagement tout au long de la thèse. Je remercie également Dimitri Bettebghor pour ses conseils et son soutien. Merci à tous pour la qualité de nos échanges et ces discussions techniques passionnantes, merci pour votre soutien technique et humain.

Tout au long de ces 3 années, j'ai partagé mon temps entre l'IRT Saint Exupéry et l'ISAE-SUPAERO. Je remercie ainsi tous mes collègues du projet MDA-MDO de l'IRT Saint Exupery. Je remercie en particulier François Gallard pour tous ses conseils techniques et humains, pour son aide autour de GEMS. Je pense également à Romain, mon voisin de bureau, pour sa bonne humeur quotidienne, Vincent et Benoît pour nos bavardages en pause, Nicolas, Vincent, Matthias et Jean-Christophe pour toutes nos conversations du midi où nous avons pu refaire le monde. Je remercie également plus généralement l'IRT Saint Exupéry en tant que structure, qui m'a offert un cadre privilégié pour la réalisation de cette thèse, au croisement du monde académique et de l'industrie. Mes remerciements vont donc aux équipes de la direction scientifique qui veillent sur le bon déroulement des thèses, aux équipes des ressources humaines pour leur accompagnement et leur bienveillance, depuis le recrutement et jusqu'à l'après-thèse. Je pense également à tous mes collègues à l'ISAE-SUPAERO : Eyal, Alexandre, Narjes, Rémi, Franco, Eric, Karabas, Gaspard que je remercie pour leur accueil chaleureux. Je remercie également plus généralement les équipes du département DISC pour leur accueil, ainsi que Mme Maryse Herbillon de l'école doctorale ED-AA pour son accompagnement lors des étapes administratives.

Je remercie également toute l'équipe montagne Toulousaine pour son enthousiasme notamment lors des sorties ski les week-end, quelles que soient les conditions météo tant que les bars sont ouverts : merci à vous Jonathan, Lise, Alexis, Adrià, Florence et Roberto. Ces sorties ont fortement contribué à me préserver un équilibre tout au long de ces trois années. Je pense bien sûr également à Adrien et Marina (et maintenant Luc!) toujours prêts à passer de bons moments, et que je salue notamment pour leurs performances dans la production et "le partage" des confettis. L'équipe Parisienne n'est pas en reste non plus, merci à vous Rémi, Anne-Lise, Pascal, Cécile, Aurélien, re-Anne-Lise, Benjamin, Guigui, Céline, Yann, JeJe, et Jonathan pour ces retrouvailles dans toute la France toujours aussi réussies.

Je souhaite également remercier toute ma famille, en particulier mes parents et ma soeur qui m'ont soutenu durant toute ma scolarité et encouragé quels que soient mes choix - même lors de la reprise d'études pour un doctorat après 5 ans d'activité en entreprise. Je profite également de l'occasion pour remercier ma belle famille pour son accueil en Normandie, sans oublier l'aide précieuse en bricolage et travaux divers à l'appartement! Et enfin, last but not least, je te remercie infiniment Audrey pour tout ton soutien, ta gentillesse et ta patience tout au long de ces trois longues années. Je te remercie de m'avoir accompagné lors de cette aventure, dont la réussite te revient autant qu'à moi.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Résumé

Dans le domaine de la conception avion, l'optimisation de structure fait intervenir deux types de variables. On trouve des variables continues de forme, décrivant des épaisseurs de panneau et celles des profils de section de raidisseur. Egalement, certaines variables d'optimisation sont catégorielles. Ces variables sont associées à des choix technologiques, comme par exemple un choix de matériau ou de type de raidisseur. Différents algorithmes permettent de traiter ce type de problème, mais avec une efficacité limitée en comparaison aux algorithmes basés sur le gradient. Le plus souvent, les codes d'optimisation de structure traitent les variables continues et ignorent les variables catégorielles. Cela est notamment lié à la difficulté de gérer les deux types de variables simultanément tout en préservant l'efficacité de l'algorithme.

Une méthodologie industrielle existante s'appuie sur une décomposition du problème. Elle consiste à réaliser dans un premier temps des optimisations locales à efforts internes figés, élément par élément. Puis, ces choix optimaux locaux figés, une mise à jour les efforts internes est réalisée. L'algorithme, capable de traiter des problèmes de taille industrielle, s'appuie cependant sur des choix locaux et l'optimum trouvé est parfois sous-optimal. Enfin, les contraintes globales comme celles de déplacement ne sont pas visibles et ne peuvent donc pas être prises en compte à travers cette approche. L'objectif principal de ces travaux consiste donc à mettre en place une méthodologie permettant de traiter les variables continues et catégorielles, tout en respectant les contraintes industrielles en

termes de performance et d'efficacité.

Pour atteindre cet objectif, un problème jouet (P) illustrant l'aspect combinatoire de ce problème est mis en place. Ce problème utilise un modèle structure constitué de n éléments structuraux de type barre travaillant en tension et compression. Les variables catégorielles sont notées $\boldsymbol{c}$, les variables continues $\boldsymbol{a}$. Le problème d'optimisation inclut des contraintes $\boldsymbol{s}$ sur les contraintes structurales, appelées également contraintes stress, ainsi que des contraintes $\boldsymbol{\delta}$ inter-éléments sur les déplacements, appelées contraintes de déplacement. Les contraintes stress incluent des contraintes de flambage locales et globales sur chaque élément, en plus des contraintes en traction et compression. L'objectif de ce problème est la masse de la structure, notée w.

$$\begin{aligned} \underset{\boldsymbol{c}\in\Gamma^n, \boldsymbol{a}\in\mathbb{R}^n}{\text{minimiser}} \quad & w(\boldsymbol{a},\boldsymbol{c}) \\ \text{soumis à} \quad & \boldsymbol{s}(\boldsymbol{a},\boldsymbol{c}) \leq \mathbf{0}_{n,m} \\ & \boldsymbol{\delta}(\boldsymbol{a},\boldsymbol{c}) \leq \mathbf{0}_d \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \end{aligned} \tag{P}$$

Il permet de pouvoir faire varier facilement le nombre d'éléments structuraux, et donc le nombre de variables continues et catégorielles dont dépend directement la complexité du problème. Il est ainsi possible avec ce problème d'évaluer les performances des algorithmes proposés en construisant des instances différentes de ce problème, en augmentant progressivement le nombre d'éléments structuraux. Les problèmes de petite taille, pour lesquels il est possible de calculer la solution exacte, servent à évaluer la qualité des solutions des algorithmes proposés. En augmentant le nombre d'éléments, il est également possible d'évaluer l'évolution du coût de calcul, puisqu'il s'agit d'une des principales difficultés de cette étude.

Dans les trois prochains chapitres sont proposés trois nouveaux algorithmes permettant de résoudre le problème d'optimisation mixte décrit ci-dessus. Chacun d'entre eux inclut une description théorique de la méthodologie, la présentation détaillée de la résolution d'un problème treillis 3 barres, et des applications numériques évaluant la qualité des solutions et les performances en terme de coût de calcul. Dans le Chapitre 2 est présenté un algorithme basé sur une approche de type séparation et évaluation (Branch & Bound). Le calcul des minorants s'effectuant habituellement grâce à l'évaluation d'une version continûment relâchée du problème d'origine, une relaxation spécifique à ce problème d'optimisation a donc été proposée. Une méthode de séparation est également présentée. Dans le Chapitre 3, la méthodologie proposée consiste à décomposer le problème en deux sous-problèmes, un problème supérieur et inférieur. Le problème inférieur correspond au problème d'origine avec les variables catégorielles figées. Une approximation du résultat de cette optimisation est ensuite optimisée au sein du problème supérieur, par rapport aux variables catégorielles uniquement. Dans le Chapitre 4 est présenté un algorithme s'appuyant sur les deux précédents algorithmes. Une formulation continue du problème bi-niveau du chapitre précédent est proposée. Cette formulation permet l'utilisation de gradients des solutions du problème inférieur par rapport aux variables catégorielles, formulées à présent de façon continue. Ces gradients sont efficacement calculés par méthode de pénalisation, et utilisés pour construire des hyperplans du problème inférieur, exploités pour réaliser des coupes dans l'espace catégoriel. Pour finir, le Chapitre 5 présente les conclusions et perspectives de ces travaux.

On résume ci-après les objectifs de ces travaux, ainsi que des caractéristiques clé du problème traité :

- L'objectif principal de ces travaux consiste à construire un algorithme qui permette la résolution de problèmes d'optimisation de structures de grande taille à variables mixtes catégorielles et continues,
- Les variables catégorielles sont des variables discrètes à valeurs non ordonnées et non continûment relâchables,
- La valeur d'une variable catégorielle, également appelée catalogue, correspond à un choix de matériaux et de type de raidisseur pour un élément structural,
- Les variables continues sont des variables dites de dimensionnement, à savoir les aires de section des éléments structuraux,
- Le problème d'optimisation est un problème d'optimisation combinatoire, avec un nombre de configurations possibles pour décrire la structure complète qui est égal au nombre de catalogues à la puissance du nombre d'éléments structuraux,
- Le problème d'optimisation inclut les contraintes sur les contraintes structurales, appelées également contraintes stress, ainsi que des contraintes inter-éléments sur les déplacements, appelées contraintes de déplacement.

1.1 Industrial context

Overall aircraft performance is often measured in terms of fuel consumption or range. Therefore, in aeronautics industry, optimizing these indicators is a major concern for engineering departments. These indicators depend directly on the overall weight of the structure, that encompasses primary and secondary structure weight. The primary structure is composed of every load carrying structural elements like spars, ribs, frames, load bearing skin, stiffeners, etc. The secondary structure includes structural elements of an aircraft that carry only air and inertial loads generated in the secondary structure. The aircraft primary structure can mainly be described as an assembly of stiffened panels. This kind of modular structure is widely used in aeronautics. A stiffened panel, also called super-stiffener, is an abstraction of the structure made of a stiffener and the two halves of surrounding panels. Some aircraft manufacturers use this notion as an elementary structural component for stability analyses (buckling). For example, fuselage stiffened panels are made of load carrying skin reinforced by orbital frames and longitudinal stiffeners, as seen in Figure 1.1. In the present work, the structure will always refer to the primary structure, seen as an assembly of stiffened panels.

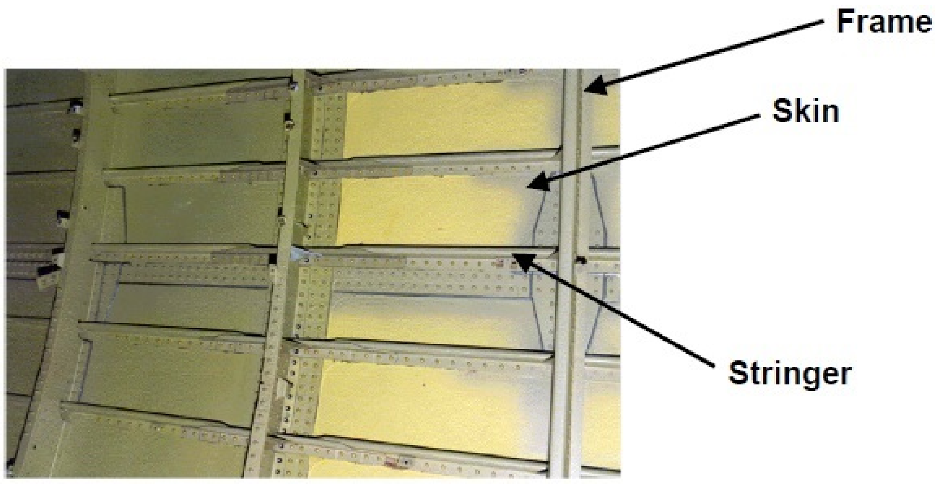


Figure 1.1: Stiffened panel example (Grihon et al. 2009)

The structure is also defined by variations in materials and thicknesses. Each structural element is dimensioned by critical load cases, which may result in different materials and stiffening principles optimal choices, and a specific thickness distribution over the major aircraft components (e.g., fuselage, wings). In his concern to minimize the weight of the structure, the designer will hence need to find the best choice of material, stiffener, and size of the elements. The objective of this work is to enhance the resolution of the mathematical problem through numerical optimization and heuristic strategies.

Material definitions
With a view to minimize the weight of the structure, the structural designer has to choose the best material for each structural element. Aircraft structures have been largely made of different grades of aluminium (from Duralumin to current Lithium-based grades) since the 30's thanks to its excellent mechanical properties and specific weight with respect to other classical materials. Its weight, compared to other metals like steel that is referred to as heavy, was indeed one of the main reason of the massive use of aluminium by aircraft manufacturers. More specifically, the strength to weight ratio was a real advantage when compared to other available materials in the past. It is not the strongest of the pure materials, however, its strength is improved when combined to other elements. Additionally, it offers an interesting compromise between its performance characteristics, fabrication costs, established manufacturing methods and practice. This is why aluminium alloys are, still today, used in the industry. In addition to aluminium grades, for some highly-loaded massive components (e.g., door lintel, pylon, landing gear) and local structural reinforcements, grades of titanium are often used in today civil aircrafts thanks to their excellent corrosion properties. Steel grades can also be used in some specific parts (typically in the landing gear). Moreover some impact contraints (e.g., HTP tips, inlet) or complex geometries impose the use of metal alloys.

Since the 70s, the use of composite materials in loaded aeronautics structures is growing continually. In aircraft industry, recent airplanes are made of more than 50% of composite, for example the Airbus A350 in Figure 1.2. Composites have been the most important

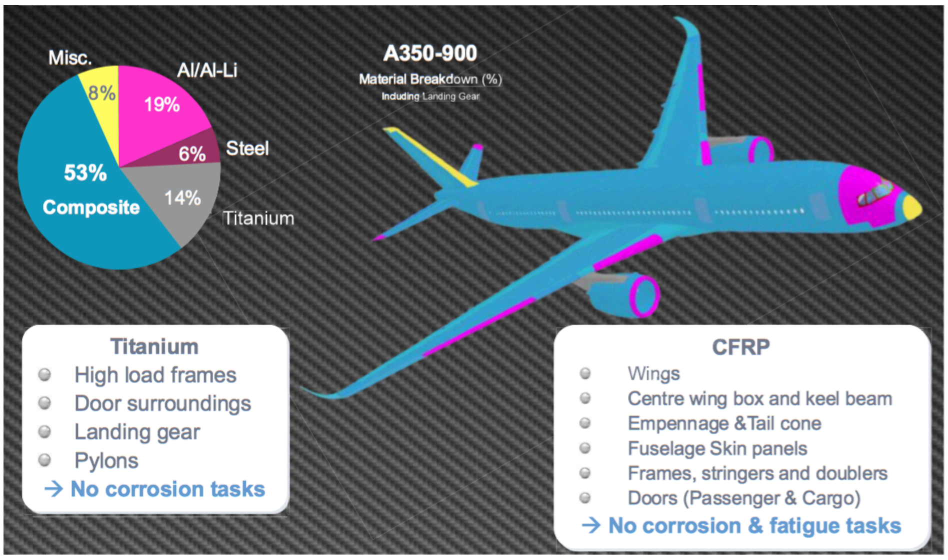


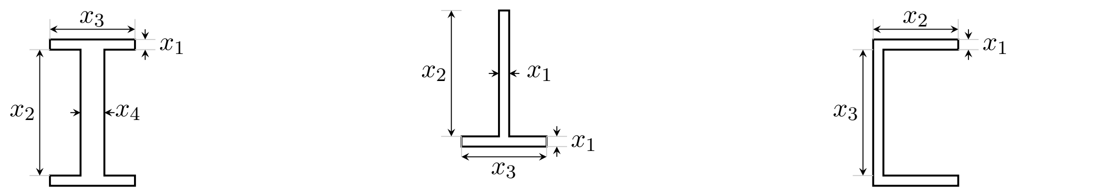
Figure 1.2: Composite application on Airbus A350 (image courtesy of Airbus).

materials introduced in the aerospace field since the use of aluminium alloys. Their benefit from a very low density when compared to metal alloys. A composite material is in facts a composition material, made of two (or more) materials. Each of them plays a different role: one material acts as a matrix and the other as a reinforcement material. The two components have significantly different physical properties, and remain separate in the composite material. The resulting material has properties different from the individual materials involved.

There are several different types of composites, according to the nature of the matrix involved: organic matrix composites (e.g., silicium, carbon) that are the most used in the industry, ceramic matrix composites relevant for high temperature environments, metal matrix composites (e.g.; GLARE), carbon fiber reinforced polymer, . . . In the industrial target problem, only laminates organic matrix composites are considered. Laminates composites are composites made of multiple layers (also named plies) of unidirectional fibers. Indeed, the fibers oriented on the x axis (for example) will never contribute to the composite resistance along the y axis. This means that a composite material subjected to exterior stresses in multiple directions has to be composed of a superposition of plies with different orientations in order to withstand the stresses. Thus, the designers need to choose the best combination of layers to optimize the composite material characteristics with respect to the external stresses. The characterization of a composite material in terms of layer fiber orientation and number of layers is called the stacking sequence. This optimization of the stacking sequence will allow to build structures with a high ratio of performance over weight.

Stiffening principle definitions

A stringer or stiffener, in aircraft construction, is a thin bar on which the skin of the aircraft structure is fastened. Stiffeners have a major role in an aircraft structure, since they are responsible for transfering the loads acting on the skin onto the frames. The stiffeners allow to cancel the overall buckling modes of the stiffened structure. Buckling is thus reduced



(a) Example of an "I"-stiffener, described by 4 geometrical variables.

(b) Example of a "T"-stiffener, described by 3 geometrical variables.

(c) Example of a "C" profile, described by 3 geometrical variables.

Figure 1.3: Examples of commonly used stiffeners in aircraft structural design. The internal geometrical variables are latent variables, scaled by the area of the cross-section.

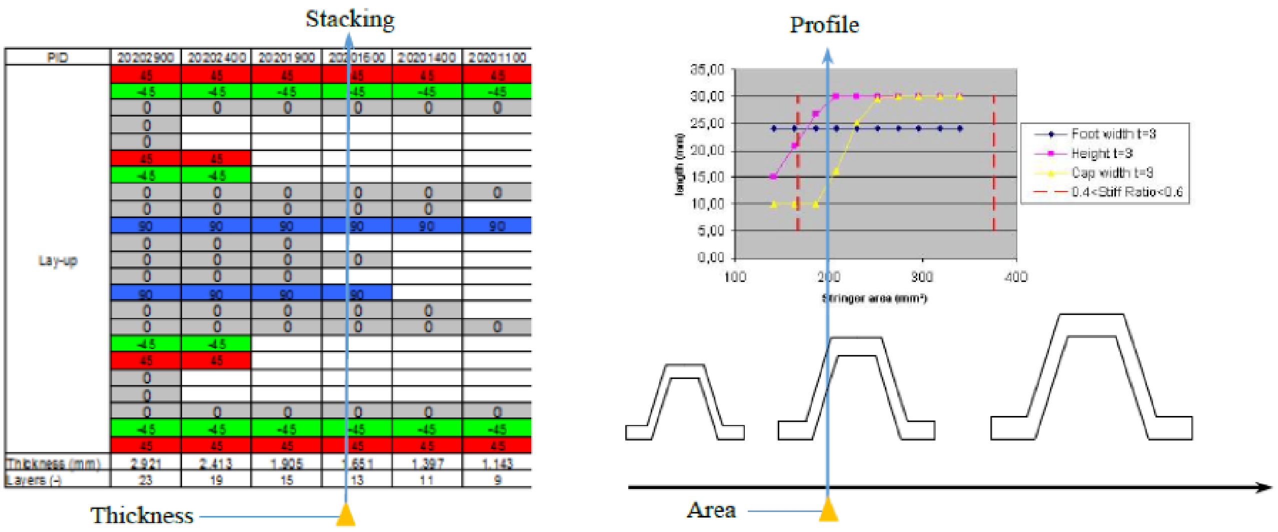
to the instability of the stringer itself of the skin. For example, the distance between two stiffeners is directly involved in the computation of the skin buckling modes. This is called local buckling. The use of stiffeners also depicts a mesh that, in some cases, can contain a damage in a specific zone or transfer loads in other parts of the panel. The material of panel stringers can be alluminium or composite. Examples of stiffeners geometries are given in Figure 1.3.

Sizing definition

The sizing of a structure consists of finding the best (according to a specific objective, that is mainly the weight minimization) geometrical dimensions that define the structural elements subjected to external stresses. In current aircraft or aerostructures manufacturers design office, common practice is to set materials, build process and overall stiffening principle prior to structural sizing, as these choice highly impacts both manufacturing costs and maintenance costs and therefore the overall aircraft cost. The sizing is obviously constrained by physical limits, that can be maximum limits on stresses due to material properties, limits induced by buckling of stiffeners, by manufacturing processes, costs, reliability, . . . They can be applied to each structural element and involve local physical behaviors (e.g., buckling limit stress of a structural element), or applied on physical quantities depending on the overall structure (e.g., maximum displacement of a wing tip). These so-called limits are chosen by the designer when the design problem is formulated. They are the translation of the design requirements specification into physical constraints onto the design problem. The choice of these criteria is crucial, because they have a strong influence on the performance of the structure (in most cases, the weight). This sizing problem is thus subjected to constraints that limit the designs to feasible ones.

1.2 Problem statement

In the field of structural design, weight minimization of the structure is a major concern for engineers. In the aircraft industry for example, structural optimization problems can combine changes in choices of materials, cross-section types, or sizes of elements based on manufacturer catalogs. As a consequence, the number of design variables grows significantly



PID	20202900	20202400	20201900	20201600	20201400	20201100
Lay-up	45	45	45	45	45	45
	-45	-45	-45	-45	-45	-45
	0	0	0	0	0	0
	0					
	0					
	45	45				
	-45	-45				
	0	0	0	0	0	0
	0	0	0	0	0	
	90	90	90	90	90	90
	0	0	0			
	0	0	0	0		
	0	0	0			
	90	90	90	90		
	0	0	0	0	0	
	0	0	0	0	0	0
	-45	-45				
	45	45				
	0					
	0					
	0	0	0	0	0	0
	-45	-45	-45	-45	-45	-45
	45	45	45	45	45	45
Thickness (mm)	2.921	2.413	1.905	1.651	1.397	1.143
Layers (-)	23	19	15	13	11	9

Figure 1.4: Catalogs principle, where the links between the thickness and stacking sequences, area and detailed variables as proposed in (Grihon 2018).

and prevents practical resolution of the associated optimization problems. In this work, it is proposed to solve large scale structural weight minimization problems with both categorical and continuous variables, subject to stress and displacements constraints. The topology of the structure is fixed. Continuous variables describe the size of aircraft structural parts: in case of thin-sheet stiffened sizing, they describe panel thicknesses and stiffening cross-sections (in the sense of the areas). The categorical variables take values belonging to an unordered and unrelaxable set. Typically, in the context of structural optimization, the choices of materials or cross-section types are depicted by categorical variables. In the literature, a catalog for an element can be considered as a guide to a given composite panel (Adams et al. 2004; Carpentier et al. 2006a; Carpentier et al. 2006b). Composite guide approach has been extended to stiffening profiles as illustrated in the Figure 1.4. Indeed, as described in (Grihon 2018), a choice of thickness maps to a choice of a stacking, and a choice of area corresponds to a choice of scaled profile (including stacking). Furthermore, a change of catalog choice leads to a change in stacking sequence (symmetric, balanced, oriented, iso). In case of a metal, the catalogs reduce to a list of profiles and the thicknesses do not drive choices of stacking sequences.

Most existing algorithms used to handle such classes of problems (Fister et al. 2013) are known to scale badly as the number of the categorical design variables increases. To illustrate the curse of dimensionality encountered for such problems, an order of magnitude of the targeted industrial problem's dimensions can be as follows. Consider an engine pylon structural model as addressed in (Gazaix et al. 2019) with 100 elements, each one having a hundred possible choices of material and stiffening principles. In this case, the categorical design space counts 100^{100} possible configurations to describe the whole structure. Thus, the high combinatorial dimension of the categorical design space enforces the need for a methodology to solve such problems efficiently.

The optimization problem, as formulated at Airbus, is defined as follows. Let be $\boldsymbol{t}$ the thicknesses of the stiffened panels skin and $\boldsymbol{a}$ the areas of the profiles, n being the number of elements in the structural model. The set Γ is an enumerated set which allows to list choices, for example the material or the stiffener type of an element. With p is the number of catalogs, we can note Γ the categorical design space

$$\Gamma = \{1, \ldots, p\}$$

that contains the p available combinations of materials and stiffeners.

All these variables are vectors for which each component is associated to an element. In our industrial case, a finite element model involves about $n = 100$ elements leading to 200 continuous variables, and 10 up to 100 categorical choices per element. The categorical variable can take a value among (number of categorical choices per element at the power of the number of elements) values of the set. This high combinatory dimension demonstrates the need for a methodology to solve efficiently such problems. Within the framework of stiffened panels, as key concept of aeronautics structures, targeted mixed categorical continuous optimization problem to solve is:

$$\begin{aligned}
\underset{\boldsymbol{c}\in\Gamma^n, (\boldsymbol{a},\boldsymbol{t})\in\mathbb{R}^{n\times n}}{\text{minimize}} \quad & w(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c}) \\
\text{subject to} \quad & \boldsymbol{s}(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c}) \leq \mathbf{0}_{n,m} \\
& \boldsymbol{\delta}(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c}) \leq \mathbf{0}_d \\
& \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \\
& \underline{\boldsymbol{t}} \leq \boldsymbol{t} \leq \bar{\boldsymbol{t}}
\end{aligned} \tag{1.1}$$

where $(\underline{\boldsymbol{a}}, \underline{\boldsymbol{t}}) \in \mathbb{R}^{n\times n}$ and $(\bar{\boldsymbol{a}}, \bar{\boldsymbol{t}}) \in \mathbb{R}^{n\times n}$ are the lower and upper bounds on areas and thicknesses, respectively. The constraints $\boldsymbol{\delta}$ on displacements $\boldsymbol{u}$ ensure that on d given nodes of the truss the displacements will not exceed predefined upper bounds $\bar{\boldsymbol{u}} \in \mathbb{R}^d$. With $\boldsymbol{P}$ a projector that selects the elements on which the displacement constraint will apply, the definition of $\boldsymbol{\delta}$ function is given as follows:

$$\begin{aligned}
\boldsymbol{\delta}\colon \mathbb{R}^{n\times n} \times \Gamma^n &\to \mathbb{R}^d \\
(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c}) &\mapsto \boldsymbol{P}\boldsymbol{u}(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c}) - \bar{\boldsymbol{u}}.
\end{aligned}$$

The structural constraints function $\boldsymbol{s}$

$$\boldsymbol{s}\colon \mathbb{R}^{n\times n} \times \Gamma^n \to \mathcal{M}(\mathbb{R}^{n,m})$$

is of the form

$$
\begin{array}{cc}
 & \begin{array}{ccc} \text{Constraint type 1} & \dots & \text{Constraint type } m \end{array} \\
\begin{array}{c} elt_1 \\ elt_2 \\ \vdots \\ elt_n \end{array} & \begin{pmatrix} \boldsymbol{s}_{11}(\boldsymbol{a}_1, \boldsymbol{t}_1, \boldsymbol{c}_1, \boldsymbol{\Phi}_1(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c})) & \dots & \boldsymbol{s}_{1m}(\boldsymbol{a}_1, \boldsymbol{t}_1, \boldsymbol{c}_1, \boldsymbol{\Phi}_1(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c})) \\ \boldsymbol{s}_{21}(\boldsymbol{a}_2, \boldsymbol{t}_2, \boldsymbol{c}_2, \boldsymbol{\Phi}_2(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c})) & \dots & \boldsymbol{s}_{2m}(\boldsymbol{a}_2, \boldsymbol{t}_2, \boldsymbol{c}_2, \boldsymbol{\Phi}_2(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c})) \\ \vdots & \vdots & \ddots \\ \boldsymbol{s}_{n1}(\boldsymbol{a}_n, \boldsymbol{t}, \boldsymbol{c}_n, \boldsymbol{\Phi}_n(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c})) & \dots & \boldsymbol{s}_{nm}(\boldsymbol{a}_n, \boldsymbol{t}_n, \boldsymbol{c}_n, \boldsymbol{\Phi}_n(\boldsymbol{a}, \boldsymbol{t}, \boldsymbol{c})) \end{pmatrix}
\end{array}
\tag{1.2}
$$

and ensures, element per element, that the structural stress does not exceed a limit stress value. The internal loads of the structure, computed by a finite element method (FEM), are noted $\boldsymbol{\Phi}$.

1.3 Motivations

The need for a methodology that supports trade-off studies in early design stages both in terms of choice of stiffening principle and material is not new in the aircraft industry. Methodologies have emerged in the industry to tackle the curse of dimensionality when dealing with categorical variables in structural optimization (Collier et al. 2002; Grihon 2018). For instance at Airbus, an existing tool named PRE-sizing Solution for Trade-Offs (PRESTO), as been developed (Grihon 2018). It solves a problem that is close to (1.1), but different for two reasons. First, the sizing variables $\boldsymbol{a}$ and $\boldsymbol{t}$ are not continuous but belong to the discrete real subspaces $Y_{\boldsymbol{a}}$ and $Y_{\boldsymbol{t}}$, respectively. Second, there is no constraint on displacements in the problem solved by PRESTO.

The full discrete PRESTO approach uses a bi-step strategy involving massively parallel element-wise optimizations. The algorithm consists of the following steps, after an initialization of all the design variables. First, a FEM analysis is performed in order to get the internal loads $\boldsymbol{\Phi}_i$ in the structure, given the areas, thicknesses, and categorical design variables fixed. Second, the internal loads are fixed to their computed value in the stress constraints of the optimization problem. The stress constraints functions on a structural element i are thus independent from the definition of the other elements. Thus, the following optimization problem is solved for each element independently ($\forall i \in \{1, \dots, n\}$):

$$
\begin{array}{ll}
\underset{\boldsymbol{c}_i \in \Gamma^n, \boldsymbol{a}_i \in Y_{\boldsymbol{a}}, \boldsymbol{t}_i \in Y_{\boldsymbol{t}}}{\text{minimize}} & w(\boldsymbol{a}_i, \boldsymbol{t}_i, \boldsymbol{c}_i) \\
\text{subject to} & \boldsymbol{s}_{ij}(\boldsymbol{a}_i, \boldsymbol{t}_i, \boldsymbol{c}_i, \boldsymbol{\Phi}_i) \le 0 \quad \forall j \in \{1, \dots, n\} \\
 & \underline{\boldsymbol{a}}_i \le \boldsymbol{a}_i \le \bar{\boldsymbol{a}}_i \\
 & \underline{\boldsymbol{t}}_i \le \boldsymbol{t}_i \le \bar{\boldsymbol{t}}_i.
\end{array}
$$

These optimizations are performed in parallel, and return a new candidate solution. The FEM is then initialized with the returned areas, thicknesses and catalogs. The process is repeated until a tolerance on the optimal weight or a maximal number of iterations is reached.

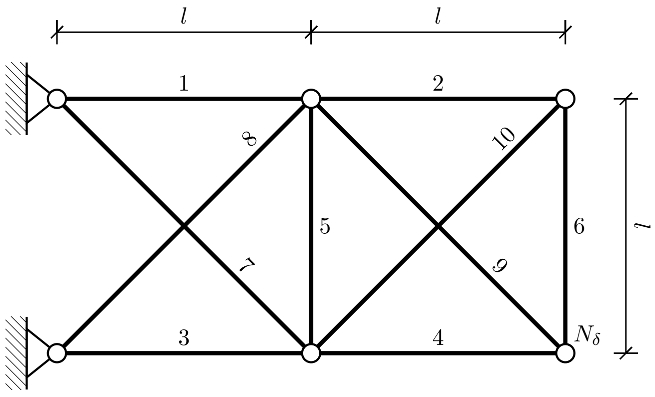


Figure 1.5: A 10-bar truss structure.

This approach industrially used at Airbus simplifies the impact of each categorical choice on the overall optimal internal loads distribution by deporting the optimization at element (subsystem) level. Although the industrial approach is highly scalable, it can not handle system-level behavior (optimum internal load distribution) nor system-level constraints (e.g., flutter, modal or displacement constraints) in a same optimization process. The absence of such constraints in the problem formulation is not representative of aircraft structure design problems, in a multidisciplinary context for instance. Furthermore, there is no formal proof that at the end of the algorithm, the optimum is the exact one.

The objective of this work is thus to build an algorithm that solves a large scale mixed categorical-continuous structural optimization problem that is subject to stress constraints and constraints on displacements. The main difficulty is to introduce the constraint on displacements, while keeping the computational cost as low as possible in case of large scale problems. The best compromise between computational cost and the quality of the optimum has to be found.

1.4 A toy problem

Instead of tackling the targeted industrial case that deals with aeronautics box-section structures, a toy problem has consisted in the implementation of a test-case that can be handled within short execution-times. The structural model is a truss model where members support tension and compression forces. This makes easier the exploration of new algorithms and allows for identifying interesting leads. A well-known truss used in structural optimization is the 10-bar truss illustrated Figure 1.5. The design variables definition is similar to the one implemented in the existing industrial approach. However in the toy problem, the structural elements are bars instead of stiffened panels. Therefore, the categorical design variables $\boldsymbol{c}$ denote choices of a material (among metal alloys) and stiffener and the continuous design variables are reduced to the areas $\boldsymbol{a}$. The mixed categorical continuous optimization problem becomes:

$$\begin{aligned} \underset{\boldsymbol{c}\in\Gamma^n,\boldsymbol{a}\in\mathbb{R}^n}{\text{minimize}} \quad & w(\boldsymbol{a},\boldsymbol{c}) \\ \text{subject to} \quad & \boldsymbol{s}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_{n,m} \\ & \boldsymbol{\delta}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_d \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \end{aligned} \tag{P}$$

where $\underline{\boldsymbol{a}} \in \mathbb{R}^n$ and $\bar{\boldsymbol{a}} \in \mathbb{R}^n$ are the lower and upper bounds on areas, respectively. The objective function (i.e., the global weight of the structure) is defined as follows:

$$\begin{aligned} w\colon \mathbb{R}^n \times \Gamma^n &\to \mathbb{R} \\ w(\boldsymbol{a},\boldsymbol{c}) &\mapsto \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{\rho}_i(\boldsymbol{c}_i)\boldsymbol{L}_i, \end{aligned}$$

where $\boldsymbol{\rho}_i(\boldsymbol{c}_i)$ corresponds to the material density of element i driven by the choice $\boldsymbol{c}_i$.

The constraints $\boldsymbol{\delta}$ on displacements $\boldsymbol{u}$ ensures that on d given nodes of the truss the displacements will not exceed predefined upper bounds $\bar{\boldsymbol{u}} \in \mathbb{R}^d$. In the case of a truss model, the constraint on displacements $\boldsymbol{\delta}$ does not depend from thicknesses anymore, and becomes:

$$\begin{aligned} \boldsymbol{\delta}\colon \mathbb{R}^n \times \Gamma^n &\to \mathbb{R}^d \\ (\boldsymbol{a},\boldsymbol{c}) &\mapsto \boldsymbol{P}\boldsymbol{u}(\boldsymbol{a},\boldsymbol{c}) - \bar{\boldsymbol{u}}. \end{aligned}$$

In this toy problem, four different structural constraints per element will be considered, i.e., $m = 4$ in the generic structural constraints expressions given by (1.2). The constraint function $\boldsymbol{s}$ is then defined as:

$$\boldsymbol{s}\colon \mathbb{R}^n \times \Gamma^n \to \mathcal{M}(\mathbb{R}^{n,4})$$

and is of the form

$$\begin{array}{c|cccc} & \text{Allowable Tension} & \text{Allowable Compression} & \text{Euler Buckling} & \text{Local Buckling} \\ \hline elt_1 & \boldsymbol{s}_{11}(\boldsymbol{a}_1,\boldsymbol{c}_1,\boldsymbol{\Phi}_1(\boldsymbol{a},\boldsymbol{c})) & \boldsymbol{s}_{12}(\boldsymbol{a}_1,\boldsymbol{c}_1,\boldsymbol{\Phi}_1(\boldsymbol{a},\boldsymbol{c})) & \boldsymbol{s}_{13}(\boldsymbol{a}_1,\boldsymbol{c}_1,\boldsymbol{\Phi}_1(\boldsymbol{a},\boldsymbol{c})) & \boldsymbol{s}_{14}(\boldsymbol{a}_1,\boldsymbol{c}_1,\boldsymbol{\Phi}_1(\boldsymbol{a},\boldsymbol{c})) \\ elt_2 & \boldsymbol{s}_{21}(\boldsymbol{a}_2,\boldsymbol{c}_2,\boldsymbol{\Phi}_2(\boldsymbol{a},\boldsymbol{c}))) & \boldsymbol{s}_{22}(\boldsymbol{a}_2,\boldsymbol{c}_2,\boldsymbol{\Phi}_2(\boldsymbol{a},\boldsymbol{c})) & \boldsymbol{s}_{23}(\boldsymbol{a}_2,\boldsymbol{c}_2,\boldsymbol{\Phi}_2(\boldsymbol{a},\boldsymbol{c})) & \boldsymbol{s}_{24}(\boldsymbol{a}_2,\boldsymbol{c}_2,\boldsymbol{\Phi}_2(\boldsymbol{a},\boldsymbol{c})) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ elt_n & \boldsymbol{s}_{n1}(\boldsymbol{a}_n,\boldsymbol{c}_n,\boldsymbol{\Phi}_n(\boldsymbol{a},\boldsymbol{c}))) & \boldsymbol{s}_{n2}(\boldsymbol{a}_n,\boldsymbol{c}_n,\boldsymbol{\Phi}_n(\boldsymbol{a},\boldsymbol{c})) & \boldsymbol{s}_{n3}(\boldsymbol{a}_n,\boldsymbol{c}_n,\boldsymbol{\Phi}_n(\boldsymbol{a},\boldsymbol{c})) & \boldsymbol{s}_{n4}(\boldsymbol{a}_n,\boldsymbol{c}_n,\boldsymbol{\Phi}_n(\boldsymbol{a},\boldsymbol{c})) \end{array} \tag{1.3}$$

First, one has two constraints in tension and compression, respectively, given by

$$\begin{aligned} s_{i1}(\boldsymbol{a},\boldsymbol{c}) &:= \frac{\boldsymbol{\Phi}_i(\boldsymbol{a},\boldsymbol{c})}{\boldsymbol{a}_i} - \sigma^t(\boldsymbol{c}_i) \\ s_{i2}(\boldsymbol{a},\boldsymbol{c}) &:= \frac{\boldsymbol{\Phi}_i(\boldsymbol{a},\boldsymbol{c})}{\boldsymbol{a}_i} - \sigma^c(\boldsymbol{c}_i) \end{aligned}$$

with $\sigma^t(\boldsymbol{c}_i) \in \mathbb{R}$ the stress limit in tension and $\sigma^c(\boldsymbol{c}_i) \in \mathbb{R}$ the stress limit in compression. The two other constraints are the Euler and local buckling constraints, respectively, given by

$$\begin{aligned} s_{i3}(\boldsymbol{a},\boldsymbol{c}) &:= \frac{\boldsymbol{\Phi}_i(\boldsymbol{a},\boldsymbol{c})}{\boldsymbol{a}_i} - \frac{\pi^2 E(\boldsymbol{c}_i) I(\boldsymbol{a}_i,\boldsymbol{c}_i)}{\boldsymbol{a}_i \boldsymbol{L}_i^2} \\ s_{i4}(\boldsymbol{a},\boldsymbol{c}) &:= \frac{\boldsymbol{\Phi}_i(\boldsymbol{a},\boldsymbol{c})}{\boldsymbol{a}_i} - \frac{4\pi^2 E(\boldsymbol{c}_i)\mathcal{K}^2(\boldsymbol{c}_i)}{12(1-\nu^2(\boldsymbol{c}_i))} \end{aligned}$$

with $E(\boldsymbol{c}_i)$, $I(\boldsymbol{a}_i,\boldsymbol{c}_i)$, $\boldsymbol{L}_i$, $\nu(\boldsymbol{c}_i)$ respectively the Young modulus, the area moment of inertia, the length and the poisson coefficient of element i. The ratio between cross-section internal sizes, depending on the stiffener profile, is given by $\mathcal{K}(\boldsymbol{c}_i)$. It is worth to note that the choice of stiffener's profile affects the buckling constraints through the terms $I(\boldsymbol{a}_i,\boldsymbol{c}_i)$ and $\mathcal{K}(\boldsymbol{c}_i)$ in the buckling constraints definitions. Indeed, the computation of $I(\boldsymbol{a}_i,\boldsymbol{c}_i)$ and $\mathcal{K}(\boldsymbol{c}_i)$ involves a detailed description of the cross-section geometry. For each section i, the vector of detailed variables $\boldsymbol{x}^{(i)}$ of the i^{th} bar varies with respect to the detailed variables of the reference profile, noted $\boldsymbol{x}_0(\boldsymbol{c}_i)$ such that:

$$\boldsymbol{x}^{(i)} := \boldsymbol{\tau}_i \boldsymbol{x}_0(\boldsymbol{c}_i) \quad \forall i \in \{1,\ldots,n\} \tag{1.4}$$

The corresponding cross-section can be written as:

$$\boldsymbol{a}_i = \boldsymbol{\tau}_i^2 \boldsymbol{a}_0(\boldsymbol{c}_i) \quad \forall i \in \{1,\ldots,n\} \tag{1.5}$$

From equations (1.4) and (1.5) can be deduced an expression of $\boldsymbol{\tau}_i$, the scaling ratio of the profiles:

$$\boldsymbol{\tau}_i = \sqrt{\frac{\boldsymbol{a}_i}{\boldsymbol{a}_0(\boldsymbol{c}_i)}} \quad \forall i \in \{1,\ldots,n\}$$

Thus, the simple mathematical function detailed below can be used to compute the detailed variables with respect to areas $\boldsymbol{a}_i$ coming from the optimizer, and the reference profile described by $\boldsymbol{a}_0$ and $\boldsymbol{x}_0(\boldsymbol{c}_i)$:

$$\boldsymbol{x}^{(i)}(\boldsymbol{a}_i) = \sqrt{\frac{\boldsymbol{a}_i}{\boldsymbol{a}_0(\boldsymbol{c}_i)}} \boldsymbol{x}_0(\boldsymbol{c}_i).$$

The detailed geometric variables $\boldsymbol{x}^{(i)}$ are thus indirectly optimized as latent variables that depend on $\boldsymbol{a}_i$. This description of the internal stiffener geometry is inspired from existing approaches like for example the PRESTO methodology in (Grihon 2012) or in (Gao et al. 2018).

Internal forces $\boldsymbol{\Phi}$ and displacements $\boldsymbol{u}$ will be computed using the direct stiffness method, introduced in (Turner 1959; Turner et al. 1964). Structural elements are considered as truss elements with pin-jointed connections. This means that the bars will only carry axial forces. At each node, displacements are allowed along the global axes. Each element i is defined by the elementary stiffness matrix $\boldsymbol{K}_i^e(\boldsymbol{a}_i,\boldsymbol{c}_i) \in \mathbb{R}^{q,q}$, with q the number of free nodes multiplied

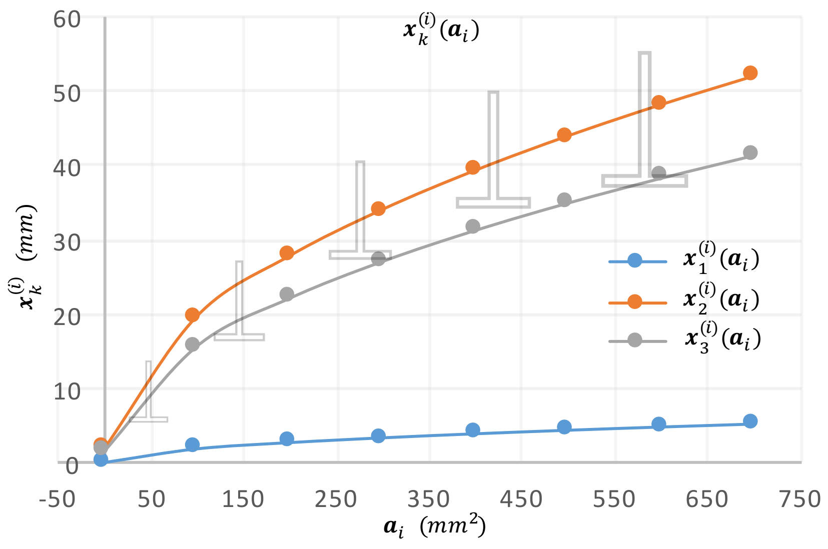


Figure 1.6: Scaling of a bar section. Example with "T"-stiffener.

by the number of physical space dimensions. The global stiffness of the whole truss is given by the matrix $\boldsymbol{K}(\boldsymbol{a}, \boldsymbol{c}) \in \mathbb{R}^{q,q}$ in global coordinates. Such matrix can be computed as the sum of each elementary stiffness matrix expressed after its transformation with the i^{th} element rotation matrix $\boldsymbol{T}_i$, i.e., (Turner 1959; Turner et al. 1964):

$$\boldsymbol{K}(\boldsymbol{a}, \boldsymbol{c}) = \sum_{i=1}^{n} [\boldsymbol{T}_i^t \boldsymbol{K}_i^e(\boldsymbol{a}_i, \boldsymbol{c}_i) \boldsymbol{T}_i].$$

Given a vector $\boldsymbol{f} \in \mathbb{R}^q$ of external loads applied on each of the free nodes in the global coordinates, the vector of displacements $\boldsymbol{u} \in \mathbb{R}^q$ can be obtained by solving the following equation:

$$\boldsymbol{K}(\boldsymbol{a}, \boldsymbol{c})\boldsymbol{u}(\boldsymbol{a}, \boldsymbol{c}) = \boldsymbol{f}.$$

The vector of internal forces $\boldsymbol{\Phi} \in \mathbb{R}^n$ is then given by:
$\forall i \in \{1, \ldots, n\}$,

$$\boldsymbol{\Phi}_i(\boldsymbol{a}, \boldsymbol{c}) = \boldsymbol{K}_i^e(\boldsymbol{a}_i, \boldsymbol{c}_i)\boldsymbol{T}_i \boldsymbol{u}_i(\boldsymbol{a}, \boldsymbol{c}),$$

where $\boldsymbol{\Phi}_i$ is the axial internal force through element i and $\boldsymbol{u}_i$ its displacement vector.

In Table 1.1 are depicted the dependencies between the design variables, the latent variables and the main functions of interest of the optimization problem. Thus, it is worth to note that the objective depends only on the choice of materials and areas, while the constraints depend on both (excepted the density) the material properties and the stiffener definition.

	$\boldsymbol{c}$					$\boldsymbol{a}$		
	Stiffener	Material				Sizing variables		
	−	ρ	E	σ_{allow}	ν	$\boldsymbol{a}$	Φ	$\mathcal{K}$
weight		●				●		
stress constraints (compr/tension)				●			●	
stress constraints (local/euler)	●		●		●	●	●	●
displacements constraints			●			●	●	
all constraints	●		●	●	●	●	●	●
internal forces $\boldsymbol{\Phi}$			●			●		
detailed geometrical parameters $\mathcal{K}$						●		

Table 1.1: Dependencies between the main functions of the optimization problem and the design variables $\boldsymbol{c}$ and $\boldsymbol{a}$.

1.5 Research background

Optimization problems can involve different kinds of design variables, as illustrated Figure 1.7. First, there are continuous design variables. When the objective and constraints functions are continuous and derivable, it is possible to leverage the efficiency of gradient-based algorithms. There are many existing gradient descent methods used to solve engineering problems, e.g., sequential linear programming (Marcotte and Dussault 1989; Etman et al. 1996), sequential quadratic programming (Boggs and Tolle 1995; Wright and Nocedal 2006), method of moving asymptotes (Svanberg 2002). Second, there are non-continuous design variables. This group of design variables collects two different kinds of variables. First, there are discrete design variables. For example, these discrete design variables can take integer, or binary values. It is worth to note that these values are scalar values, and not vector of values. In addition, these variables are relaxable. This means that the optimizer can rely on evaluations of the objective and constraints at intermediate values. If the functions are derivable, the optimizer can even benefit from their gradient. The discrete variables can also be ordered, meaning that there exists a natural definition of a neighborhood. The main difference with continuous optimization lies in the fact that the result of a discrete optimization has to be a discrete optimum. Finally, there are categorical design variables. These variables take values in a finite set of instances where there is no definition of intermediate values. Furthermore, in the case of non-ordered categorical values there is no neighborhood definition neither. Most often, the categorical values are either non-numerical ones, or vector of values. In the context of this work, the optimization problem involves both continuous design variables (the areas), and categorical design variables (the materials and cross-section types).

In general, to handle mixed categorical-continuous optimization problems, many optimization algorithms can be used, e.g., metaphor-based metaheuristics and swarm intelligence algorithms (Goldberg 1989; Nouaouria and Boukadoum 2011; Liao et al. 2014). Pattern search strategies have also been proposed to solve mixed variable optimization problems with

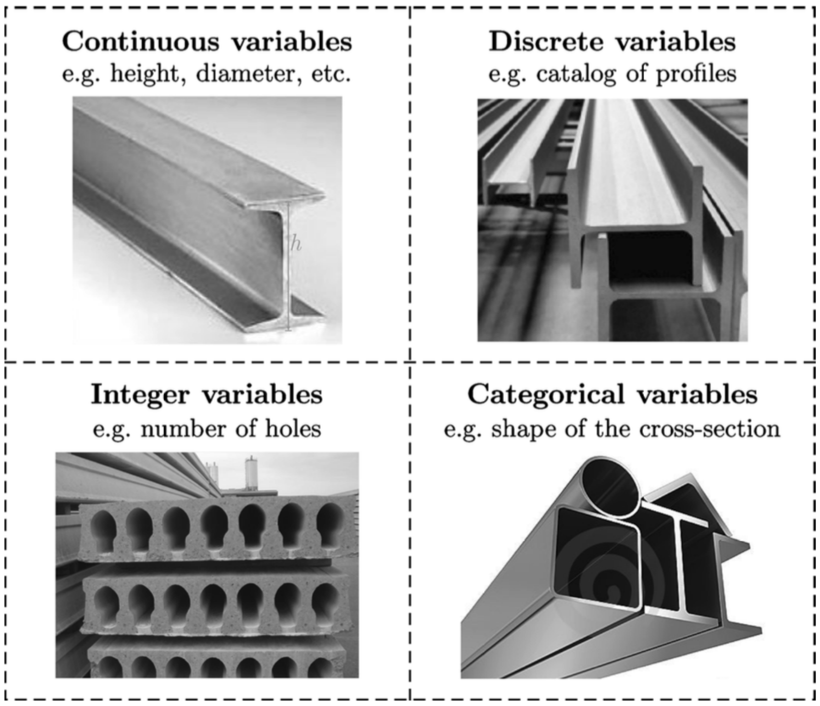


Figure 1.7: Illustration of continuous, discrete, integer and categorical design variables (Herrera et al. 2014).

categorical variables (Audet and Dennis 2001; Abramson et al. 2009; Audet et al. 2018). However, this type of methods is not designed to solve efficiently large scale optimization problems (Sigmund 2011; Stolpe 2011). In these approaches, mixed variables programming (MVP) is combined with mesh adaptive direct search (MADS) and a surrogate-assisted strategy (Audet et al. 2018). The drawback of such approaches is mainly related to the definition of a suitable neighborhood to be able to handle the categorical choices. Other approaches based on the discrete global descent method have been proposed to solve mixed optimization problems (Lindroth and Patriksson 2011).

In the context of structural optimization problems, various surrogate-based optimization strategies have been extended to categorical variables (Filomeno Coelho 2014; Müller et al. 2013; Herrera et al. 2014; Roy et al. 2017; Roy et al. 2019; Garrido-Merchán and Hernández-Lobato 2018; Pelamatti et al. 2019). One of the main challenges of such approaches is related to their efficiency when handling large dimension categorical design space. Furthermore, a definition of a neighborhood is often required during the construction of the surrogate model. As an example, in (Pelamatti et al. 2019), the neighborhood is defined through an appropriate kernel definition. In these approaches, once the surrogate model is built, the optimizer still faces a large scale discrete optimization problem. A recent work (Gao et al. 2018) also suggests reducing the dimension of the problem by finding implicit correlation between the design variables. Existing works propose to solve structural optimization problems (with multi-material and multi-cross-section design variables) using a continuous formulation of the design space which is provided by means of interpolation schemes (Stegmann and Lund 2005;

Krogh et al. 2017). Although such approaches allow to leverage the efficiency of gradient-based optimization algorithms, there is no guarantee that the optimization will retrieve integer values corresponding to a given material, for instance.

Other existing approaches rely on the structure of the mathematical mixed variable problem to decompose the intial problem into several more tractable subproblems. For instance, by the use of Benders decomposition (Benders 1962; Geoffrion 1972) or by outer approximation schemes (Duran and Grossmann 1986; Hijazi et al. 2014). For structural optimization, decomposition schemes have been mostly applied to continuous optimization problems, e.g., StiffOpt (Samuelides et al. 2009), Quasi Separable Decomposition (QSD) (Haftka et al. 2006; Schutte et al. 2004). The QSD has then been applied to structural optimization of large scale composite structures (Bettebghor et al. 2011; Bettebghor et al. 2018). In this context, the composite stacking sequences were formulated as continuous variables by using lamination parameters. In (Allaire and Delgado 2015), both the composite fiber, lay-up sequence and the ply topology are optimized into a bi-level scheme. The main difficulty of existing decomposition schemes is related to the fact that they are not able to handle large scale mixed optimization problems with categorical variables. In the industry, methodologies have emerged to tackle the curse of dimensionality when dealing with categorical variables in structural optimization. For instance, (Grihon 2018) uses a bi-step strategy involving massively parallel element-wise optimizations. This approach industrially used at Airbus simplifies the impact of each categorical choice on the overall optimal internal loads distribution by deporting the optimization at element (subsystem) level. Although the proposed approach is highly scalable, it can not handle system-level behavior (optimum internal load distribution) nor system-level constraints (e.g., flutter, modal or displacement constraints). The absence of such constraints in the problem formulation is not representative of aircraft structure design problems, in a multidisciplinary context for instance.

To summarize, in one hand the algorithms that tackle mixed categorical continuous problems in a generic way are often metaheuristics, which are known to fail at solving large scale instances efficiently (curse of dimensionality). Another class of existing algorithms (non metaheuristics) in the literature are either (a) solving a different problem (compared with the problem tackled in this work), (b) based on the definition of a neighborhood, or (c) rely on a straightforward continuous relaxation of the variables. All these approaches are not possible to use in the case of categorical design variables.

1.6 Outline

In Chapter 1, the industrial context and the motivations of the thesis are given. The main goal of the research includes the implementation of an algorithm that can solve efficiently a large scale mixed categorical structural optimization problem. This optimization problem involves continuous variables that are the areas of the structural elements, and the non-relaxable non-ordined design variables, also called categorical variables. This problem is highly sensitive to the number of structural members as well as the values that can take the categorical variables.

The formulation of the industrial optimization problem is presented. Then, a toy problem that illustrates the industrial problem combinatorial complexity is defined. Several instances of this problem will be used to assess the proposed algorithms performances. In the three next chapters are proposed three new algorithms that can solve the mixed categorical structural optimization problem. All of them include a theoretical description of the methodology, a step by step solution on a 3-bar truss problem, numerical applications with evaluations of the computational cost scaling, and a comparison to a state of the art algorithm. The numerical implementation of the proposed methodologies relies on the Generic Engine for MDO Scenarios (GEMS) (Gallard et al. 2018). Details on the implementations have been published into a conference proceeding (Gallard et al. 2019).

In Chapter 2 is presented a Branch & Bound based algorithm. This single-tree approach allows to find the optimum without proceeding to a complete enumeration of the solutions. It splits the main problem into several smaller problems for which a lower bound of the solution can be efficiently computed. Usually, these lower bounds are obtained thanks to a continuous relaxation of the optimization problem. However in the case of the targeted problem, it is not possible to turn the categorical space into a continuous one. A specific continuous formulation of these smaller problems is proposed in order to make possible the evaluation of the lower bounds. A specific branch rule is also presented. The algorithm is then tested over several truss optimization problems instances.

In Chapter 3, the proposed methodology consists of using a bi-level decomposition involving two problems, named master and slave. For given categorical choices, the slave addresses the continuous variables of our optimization problem. The master consists of minimizing a first order like approximation of the slave problem with respect to the categorical design variables. The algorithm is then tested over several truss optimization problems instances.

Chapter 4 introduces a new multi-tree based framework. A continuous formulation of the optimization problem is proposed. A bi-level decomposition is then applied to this continuous formulation. Thanks to the continuous formulation of the problem, the gradients of the slave problem are used to build supporting hyperplanes of the slave problem. They are efficiently computed thanks to a post-optimal sensitivity analysis. The resulting algorithm is then tested over several truss optimization problems instances.

Finally, the Chapter 5 presents the conclusions and perspectives of the thesis. The similarities, differences, advantages and drawbacks of the proposed approaches are discussed. The further perspectives and development are suggested.

1.7 Contributions

The main objective of this thesis is building an algorithm that can solve large scale mixed categorical-continuous optimization problems. To that end, it was first necessary to lay the theoretical bases of a new algorithm that can handle the target problem. In this work are presented three theoretical contributions, that correspond to three new algorithms described in three different chapters, respectively. The work presented in the Chapter 2 has been published into a conference proceeding (Barjhoux et al. 2018b). The work presented in the Chapter 3 has been published into a conference proceeding (Barjhoux et al. 2018a) and into a journal article (Barjhoux et al. 2020). Finally, an article focused on the work presented in the Chapter 4 is being drafted. Second, it was necessary to implement these algorithms numerically. These developments are themselves contributions. Implemented in a generic way, components of these implementations can be re-used to solve other optimization problems. The numerical implementations of the proposed methodologies rely on the Generic Engine for MDO Scenarios (GEMS) (Gallard et al. 2018). The tool offers an efficient way to test multi-level formulations, with built-in classes that facilitate optimization problems manipulations. For example, the resulting generic algorithms implementations allow to change the truss geometries of the structural optimizations with minor changes in the launch scripts. This was a major advantage since the scalability of the algorithm had to be rated. Details on the implementations have been published into a conference proceeding (Gallard et al. 2019).

In this Chapter, the following items have been discussed:

- The main objective of this work is to build an algorithm that solves a large scale mixed categorical-continuous structural optimization problem,
- The categorical variables are non-ordered and non-relaxable discrete variables,
- The value of a categorical design variable corresponds to a choice of both material and stiffening principle for a structural element,
- The continuous design variables are sizing variables, that are areas (cross-sections) of the structural elements,
- The optimization problem is a combinatorial optimization problem, with a number of possible configurations to describe the whole structure that is equal to the number of catalogs at the power of the number of structural elements,
- The optimization problem includes stress constraints and inter-element constraints, e.g., constraints on displacements.

Chapter 2

Hybrid Branch & Bound : a global approach

Contents

Résumé

Dans le contexte de l'optimisation de structure dans l'industrie, les variables de conception peuvent décrire des tailles d'éléments, par exemple issues de listes mises à disposition par les fabricants. Ces variables sont discrètes ordonnées et relâchables. Pour résoudre ce type de problème et éviter d'attaquer de front la combinatoire complète, on résout souvent une suite de sous-problèmes plus simples. C'est le cas des algorithmes basés sur des approches

de séparation et évaluation (Branch & Bound). Historiquement, ces approches s'appuient sur la résolution d'une suite de sous-problèmes relâchés continûment (Land and Doig 1960; Dakin 1965), c'est-à-dire où la variable discrète est considérée comme étant continue. Les optima correspondant sont, sous réserve d'hypothèse de convexité des sous-problèmes, des minorants du problème d'origine. Une comparaison de ces minorants à la meilleure solution connue (majorant) permet d'éliminer des sous-espaces de solution. L'efficacité de la méthode dépend fortement de l'ordre dans lequel ces sous-problèmes sont résolus.

Dans ces travaux de thèse, le problème considéré ne rentre pas directement dans le cadre de cette approche, du fait du caractère catégoriel (non-ordonné et non-relâchable) de certaines des variables d'optimisation. En effet, sans possibilité de relâcher les variables catégorielles dans le problème d'origine, le calcul du minorant nécessite une adaptation de la formulation des sous-problèmes. Ainsi, la relaxation suivante (rP($\mathcal{I}$)) du problème (P) est proposée:

$$\begin{aligned}
\underset{\boldsymbol{E}\in\widetilde{\Gamma}^n,\boldsymbol{a}\in\mathbb{R}^n}{\text{minimiser}} \quad & \widetilde{w}(\boldsymbol{a},\boldsymbol{E}) \\
\text{soumis à} \quad & \widetilde{\boldsymbol{s}}(\boldsymbol{a},\boldsymbol{E}) \leq \boldsymbol{0}_{m,n} \\
& \widetilde{\boldsymbol{\delta}}(\boldsymbol{a},\boldsymbol{E}) \leq \boldsymbol{0}_d \\
& \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}}, \\
& \boldsymbol{E}_k = E(\mathcal{I}_k)\ \forall k \in \{1,\dots,|\mathcal{I}|\}.
\end{aligned} \tag{rP($\mathcal{I}$)}$$

avec $\widetilde{w}$, $\widetilde{\boldsymbol{s}}$, et $\widetilde{\boldsymbol{\delta}}$ des sous-estimateurs des fonctions masse, contraintes stress et déplacements, respectivement. Le vecteur $\mathcal{I}$ est composé des $|\mathcal{I}|$ valeurs de variables catégorielles fixées au noeud courant de l'arbre des solutions. Les variables catégorielles ont été remplacées par les propriétés continues que sont les modules de Young $\boldsymbol{E}$. Ce sous-problème est donc à présent un problème complètement continu, et peut se résoudre efficacement à l'aide d'un algorithme basé sur le gradient. L'algorithme résultant, appelé Branch & Bound hybride (h-B&B), repose donc sur la méthode Branch & Bound avec une modification de la formulation des sous-problèmes, rendue compatible avec les variables catégorielles de ce problème.

Une analyse des performances de l'algorithme proposé permet les observations suivantes. D'après la Table 2.3, on note que les optima obtenus sur des instances de treillis 10 barres sont identiques à ceux obtenus par énumération. Cela permet de valider la qualité des solutions de l'algorithme proposé. En revanche, on observe sur la Figure 2.13 que l'effort de calcul évolue de façon exponentielle en fonction du nombre d'éléments structuraux. Cette évolution du coût de calcul rend l'approche incompatible avec le besoin de pouvoir traiter des problèmes de taille industrielle. On notera toutefois que les coûts de calculs mentionnés correspondent à des résultats obtenus après convergence des optimisations jusqu'à l'optimalité, garantie par la théorie propre aux approches de type Branch & Bound. On observe sur la Figure 2.11, que la solution optimale est obtenue après la résolution de 18 sous-problèmes, et que les 87 résolutions suivantes servent uniquement à prouver que cet optimum est le meilleur.

2.1 Introduction

In the context of structural optimization in the industry, the design variables can describe the structural element sizes, that could be selected from a list of commercially available discrete values. The presence of such design variables, called (ordered and relaxable) discrete variables, makes the optimization problem belong to the class of mixed optimization problems. Due to the hard nature of combinatorial problems, it is often easier to tackle the problem by breaking up, iteratively, the admissible set of solutions into several subsets and thus define sub-problems. This kind of problem can be handled, for example, by solvers relying on the Branch & Bound theory. However, this kind of problem is different from the target mixed categorical optimization problem, where categorical variables are non-ordered and non-relaxable.

Historically, the notion of Branch & Bound as a proof of concept for integer programming is first given in (Markowitz and Manne 1957). In this article, the Branch & Bound method is described as a general optimization scheme, not an algorithm. The methodology is more presented as a general approach that could be subjected to variations, than an automatic algorithm. The Branch & Bound automatic method, in the context of linear programming, is then provided a few years later in (Land and Doig 1960), and improved in (Dakin 1965). More than twenty years after, the Branch & Bound method is successfully applied to convex non-linear integer programming problems (Gupta and Ravindran 1985). To understand the origins of the Branch & Bound theory, one can refer to (Cook 2012).

In the context of structural optimization, various methods based on the Branch & Bound have been presented to solve convex non-linear mixed optimization problems. In (Schmit and Fleury 1980), dual methods are used to solve the mixed problem. The efficiency of this approach relies on a separable approximation of the original problem. In (Bremicker et al. 1990), a sequential linear discrete programming method is used. Dakin's method is applied in (Gupta and Ravindran 1983) to non-linear mixed-integer problems, using a generalized reduced gradient method to solve the nonlinear continuous sub-problem at each node. The same approach is used in (Sandgren 1990b), with equality constraints having binary variables. This method has been applied to solve various design problems, as in (Sandgren 1990a) where discrete (0-1) variables represent a choice between a number of different options such as materials or cross-section types. In this last work, a test case involves the design of a planar truss considering material and cross-section geometric choices among I, C and O types. The major drawback of this approach is the explosion of the number of design variables and constraints used to reformulate the problem. In order to reduce the computational cost of the non-linear Branch & Bound, heuristics have been implemented. In (Hager and Balling 1988), the design of a 16 members steel frame is achieved, with choices among 194 standard sections. In this approach, the discrete optimum is computed from the continuous optimum. Once the continuous-optimum solution is obtained, the problem is converted into a linear problem. It is then solved using the Branch & Bound method in the neighborhood of the continuous optimum. Improvements of this method have then been proposed in (Tseng et al. 1995). The Branch & Bound approach has also been applied to solve topology optimization of truss structures. For example in (Sheu and Schmit Jr 1972), the structure to optimize is a truss with redundant members (that is called a ground structure). In most of the aforementioned proposed Branch & Bound based methods, the structural optimization problem is not formulated with categorical non-ordered non-relaxable categorical design variables. In this work, a Branch & Bound algorithm is proposed to solve a mixed categorical-continuous optimization that does not fall in the scope of the methodologies presented in the literature. This is mostly due to the fact that the categorical design variables can not be be continuously relaxed in the problem (P).

This chapter is organized as follows. First, the general theory of Branch & Bound in the frame of mixed integer (relaxable) programming will be detailed and illustrated with an analytic example. The proposed formulation relying this generic Branch & Bound theory will then be presented, including adaptations of the major branch and bound steps to handle the categorical design variables involved in (P). After a description of the implementation, the accuracy of the optimum and the scalability of the proposed approach are compared with a state-of-the-art algorithm. Finally, possible future concluding remarks will be given in the last section.

2.2 Theory of discrete Branch & Bound algorithms

For the purpose of this section, a generic problem (MINLP) is defined as follows :

$$\begin{aligned} \underset{\boldsymbol{x}\in X,\ \boldsymbol{y}\in Y}{\text{minimize}} \quad & f(\boldsymbol{x},\boldsymbol{y}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq 0 \end{aligned} \tag{MINLP}$$

In this problem, a function $f\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is minimized w.r.t. continuous design variables $\boldsymbol{x} \in X \subset \mathbb{R}^n$ and n discrete design variables $\boldsymbol{y}$, while satisfying m constraints $\boldsymbol{g}\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$. As a remark, this problem is thus different from (P) since the discrete design space is relaxable and ordered, and the functions are continuously defined. Each discrete design variable can take a value among p possible discrete values, so that the set Y counts a combination of p^n discrete values :

$$Y = \{0, \ldots, p\}^n$$

The simplest way to solve (MINLP) that we can imagine could consist in enumerating the p^n instances of (MINLP) knowing $\boldsymbol{y}$ fixed. Each of these optimizations would thus be solved w.r.t. the continuous design variables $\boldsymbol{x}$ only. It is clear that such approach, although it ensures the solution to be the optimal one, would be impractical in terms of computation time.

In order to solve efficiently large scale instances of such problems, a mathematical program that would allow to find the optimum without proceeding to a complete enumeration of the solutions is needed. The goal of the Branch & Bound theory is to avoid this complete enumeration of the solutions: it restricts progressively the set of solutions. This is why this kind of methodology is also called implicit enumeration. The underlying idea is to solve the original problem by solving a succession of smaller and easier subproblems. In this Section, the basics of the Branch & Bound theory are provided. The process of spawning subproblems is named the *branch* step, presented in section 2.2.1. It is followed by a *bound* step, responsible of the evaluation of the subproblems, and detailed in section 2.2.2. According to the results of the bound step, the nodes will be discarded or not, reducing the enumeration of the subproblems. The Branch & Bound algorithms alternatively proceed to a branch step and a bound step, as described in section 2.2.

2.2.1 The branch step

In this Section is presented the step responsible of the restriction of the set of solutions. The principle of a solution tree is first presented, followed by an exemple of a problem decomposition.

2.2.1.1 Tree structure

According to the Branch & Bound theory, the iterative process of solving the problem (MINLP) consists of searching in a state space. A state space is made of a set of states and operators. The states include the original problem to be solved and all the subproblems that can be generated. Each state is illustrated by a *node*. The operators are branching rules that map one node to another. The set of states forms a *graph*, where two states are connected by a *branch* if a branching rule transforms the first state into the second one. The branching rules are thus used to decompose a problem into subproblems. In general, it is done by reducing the design space. Although the choice of a branching rule is problem dependent, a good rule of thumb is to decompose a problem into child subproblems so that their feasible subspace forms a partition of the parent feasible space. This means that, by design, a solution to a child subproblem cannot be solution to any other sibling subproblem. This generic branching rule is at the origin of most of the resulting branch and bound algorithms. As a consequence, all the branches in the graph link different nodes, since all child subproblems are uniquely defined. The resulting graph can thus be seen as a *tree*, depicted in Figure 2.1. The nodes are organized in *levels* in the tree. There is only one node at the level 0, also called *root* node. The root node contains the entire tree, by construction of the subproblems and their design space (as subspace of the parent node).

2.2.1.2 Problem decomposition

The decomposition of the main problem into subproblems (nodes) is done iteratively, by restricting the feasible set of the parent problem. Let name $(\underline{\boldsymbol{y}}_b, \bar{\boldsymbol{y}}_b)$ a design variable to branch on with $b \in \{1, \ldots, n\}$, also named branching variable. Each node of the tree is uniquely defined by a set of lower and upper bounds $(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})$ on the integer design variable $\boldsymbol{y}$, with $1 \leq \underline{\boldsymbol{y}}_b$ and $\bar{\boldsymbol{y}}_b \leq p$, with $b \in \{1, \ldots, n\}$. These bounds ensure that the generated subspace

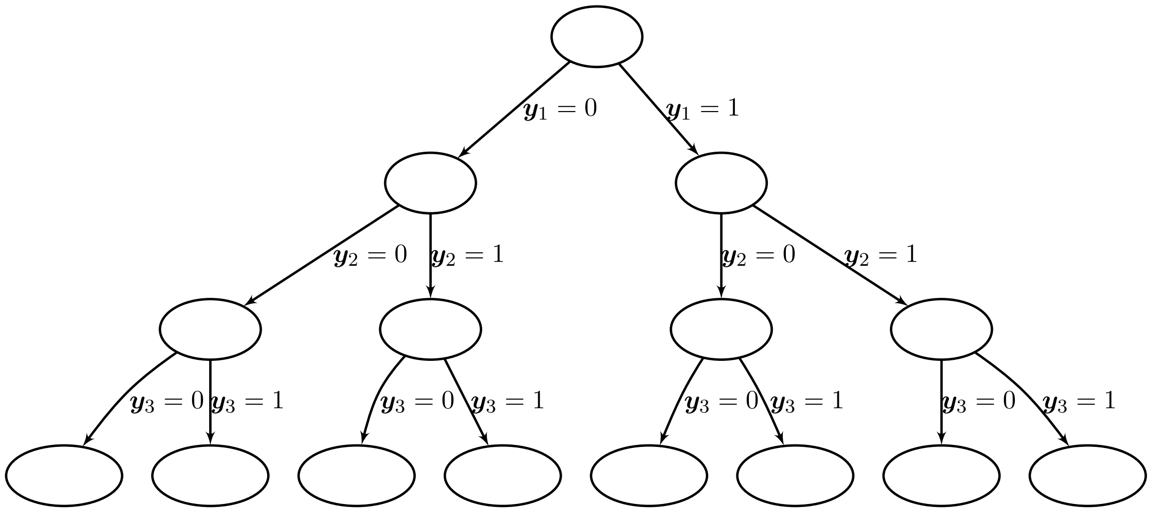


Figure 2.1: A tree structure corresponding to problem (MINLP) with three binary design variables, such that $n = 3$ and $p = 1$.

and thus the subproblem are restrictions of the ascending ones. The resulting subproblem associated to the node defined by $(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})$ is given by :

$$\begin{aligned} \underset{\boldsymbol{x}\in X,\ y\in Y}{\text{minimize}} \quad & f(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \\ & \underline{\boldsymbol{y}} \leq \boldsymbol{y} \leq \bar{\boldsymbol{y}} \end{aligned} \tag{MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)}$$

The restriction is brought by the addition of bound constraints over the integer design variables. It can be noted that the root node (MINLP) is also defined by (MINLP$(-\infty, +\infty)$). The selection of the design variable to branch on, and the definition of the bounds $(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})$ are part of the *branch strategy*. It is detailed in the Section 2.2.4.2.

2.2.2 The bound step

In practice, the entire tree (as depicted in Figure 2.1) is never completely explored (otherwise the algorithm would be inefficient), thanks to the properties of the bounding principle presented hereafter. More specifically, the efficiency of the algorithm depends on the number of nodes, or subspaces, for which we know that they do not contain the optimal solution. The bound step is a major step of the Branch & Bound theory, because it allows to identify what subspace, and thus what part of the tree is useless to be explored.

To each subspace, or subproblem (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)), say each node, is associated another problem called evaluation problem. This evaluation problem allows to compute a lower bound of the solution of the problem (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)). One of the most employed techniques to compute the lower bound consists of solving a *relaxation* of this problem. A relaxation problem of an original problem (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) has a design space that contains the original one, and/or when the objective and constraint functions are proved to be underestimate functions of those involved in the original problem (in case of a minimization problem, with lower inequality constraints). The *continuous relaxation* is probably the most common relaxation technique when it is applied to discrete optimization problems. It consists of removing the integrity constraints. In other words, the integer design variables are set as continuous design variables. The continuous relaxation of the problem (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) is written as follows :

$$\begin{aligned} \underset{\boldsymbol{x}\in X,\ \widetilde{\boldsymbol{y}}\in\widetilde{Y}}{\text{minimize}} \quad & f(\boldsymbol{x}, \widetilde{\boldsymbol{y}}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}, \widetilde{\boldsymbol{y}}) \leq 0 \\ & \underline{\boldsymbol{y}} \leq \widetilde{\boldsymbol{y}} \leq \bar{\boldsymbol{y}} \end{aligned} \qquad (\text{NLP}(\underline{\boldsymbol{y}},\ \bar{\boldsymbol{y}}))$$

where $\widetilde{Y}$ is the continuous relaxation of the integer design space Y, is defined by :

$$\widetilde{Y} := [1, p]^n.$$

The continuous design space $\widetilde{Y}$ is such that it contains the integer space Y :

$$Y \subset \widetilde{Y}.$$

This ensures that the optimum of (NLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) is a lower bound of (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)). Let the solution of the problem (NLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) be $(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})})$, such that :

$$(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}) := \underset{\boldsymbol{x}\in X,\ \widetilde{\boldsymbol{y}}\in\widetilde{Y}}{\text{argmin}} \{f(\boldsymbol{x}, \boldsymbol{y}) \text{ s.t. } \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq 0;\ \underline{\boldsymbol{y}} \leq \widetilde{\boldsymbol{y}} \leq \bar{\boldsymbol{y}}\}.$$

The optimal value of the objective $f(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})})$ is a lower bound of the feasible space of the problem (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)). If $\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}$ is an *integral* optimal solution of the problem (NLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)), $(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})})$ is also the optimal solution of (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)), the restriction of (MINLP) over the bounds $(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})$. In other words, the optimal objective value $f(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})})$ with $\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}$ integral, yields an upper bound of the optimal solution of the generic problem (MINLP).

2.2.3 The generic algorithmic process

As stated before, the Branch & Bound theory lies on the exploration of a tree that depicts a breakdown of the integer design space. This tree is explored following a process involving the aforementioned branch and bound tasks, such that the set of feasible solutions of (MINLP) is not to be entirely explored. In this section is given the generic Branch & Bound process.

Let be Q, named *queue*, the set of *active* nodes in the tree. A node is considered as active when it has not been explored, in the sense that it has neither been branched, nor it has been pruned during the process. Let define U an upper bound on the optimal solution of (MINLP). An initialization of the process consists of setting up an upper bound U of the optimal solution

Algorithm 1 An example of generic Branch & Bound framework

```
 1: initialize $U := \infty$ or best known solution of (MINLP)
 2: initialize $Q := \{(\text{NLP}(-\infty, \infty))\}$
 3: while $Q \neq \{\emptyset\}$ do
 4:     Pick a node $(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))$ in $Q$    ▷ Search strategy
 5:     $Q \leftarrow Q \setminus \{(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))\}$
 6:     Select a variable $\boldsymbol{y}_i^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}$    ▷ Branch strategy
 7:     Set $\bar{\boldsymbol{y}}_i^- := \left\lfloor \boldsymbol{y}_i^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})} \right\rfloor$, $\underline{\boldsymbol{y}}^- := \underline{\boldsymbol{y}}$ and $\underline{\boldsymbol{y}}_i^+ := \left\lceil \boldsymbol{y}_i^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})} \right\rceil$, $\bar{\boldsymbol{y}}^+ := \bar{\boldsymbol{y}}$
 8:     $children \leftarrow \{(\text{NLP}(\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-)), (\text{NLP}(\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+))\}$
 9:     for $(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))$ among $children$ do
10:         Solve $(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))$ and let its solution be $(\boldsymbol{x}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})})$    ▷ Bound evaluation
11:         if $(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))$ is infeasible then
12:             Prune node (infeasible case)
13:         else if $f(\boldsymbol{x}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}) > U$ then
14:             Prune node (dominated)
15:         else if $\boldsymbol{y}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}$ is integral then
16:             Update best known solution :
17:             $U \leftarrow f(\boldsymbol{x}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})})$
18:             $(\boldsymbol{x}^*, \boldsymbol{y}^*) \leftarrow (\boldsymbol{x}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})})$
19:         else
20:             $Q \leftarrow Q \cup \{(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))\}$
21:         end if
22:     end for
23: end while
24: return $\boldsymbol{x}^*, \boldsymbol{y}^*, f^* = U$.
```

of (MINLP). Either a value is already known by experience, or the upper bound is set to $+\infty$. Also, the queue is set such that it contains only the root node $(\text{NLP}(-\infty, \infty))$.

The solution tree can now be progressively built. This is first done by picking a node $(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))$ in Q following a search strategy, as long as the queue is not empty. At the first iteration of the algorithm, the picked node is the root node. The algorithm proceeds then to the branching on the picked node according to a branch strategy, for example the one defined in the Section 2.2.4.2. For each of the child node $\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})$, the following steps rules are applied.

The subproblem $(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))$ is solved. The solution is noted $(\boldsymbol{x}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})})$. There are four possible cases :

- (*infeasible node*) If $(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))$ is infeasible, then any descending subproblem in the tree is also infeasible, by design of the tree. The node can be prune, meaning that the entire associated subspace can be discarded.

- (*dominated node*) If the optimal value $f(\boldsymbol{x}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})}, \boldsymbol{y}^{(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})})$ of $(\text{NLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}}))$ is dominated by the best known solution U so far, then the node can be pruned. Indeed, if the lower bound $f(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})},\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})})$ of (MINLP) is greater than the best known solution, this means that the subspace does not worth to be explored. Every feasible solution of this node or its descendant will be greater than the best known solution.

- (*integer feasible node*) If the solution $(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})},\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})})$ is such that $\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}$ is integral, and if $f(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})},\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}) < U$, then the node is a new incubent. The best known solution can be updated such that $U := f(\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})},\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})})$, $(\boldsymbol{x}^*,\boldsymbol{y}^*) := (\boldsymbol{x}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})},\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})})$. If $f(\boldsymbol{y}^{(\underline{\boldsymbol{y}},\bar{\boldsymbol{y}})}) > U$, the integer feasible solution is worst than the best known solution. The node can thus be discarded.
- (*active node*) If the current child does not meet any of the aforementioned conditions, the node is still active. This means that the design sub-space covered by the child node contains possibly the optimal solution. The node is thus added into the queue, and Q is updated accordingly.

This ends the current iteration. The next one starts by picking another node in Q, until Q is empty. The result obtained at the end of the procedure is given by the current upper bound. It is proved to be a global optimum. The described procedure is detailed in Algorithm 1.

The major advantage of this generic approach lies in the fact that it can be applied to every combinatorial optimization problem instances (MINLP). The algorithms built on the basis of this Branch & Bound general scheme are defined by the strategies involved in the described steps, that are :

- the bound evaluation (Section 2.2.2), thanks to the definition of the evaluation problem NLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$). It is a key tool in the Branch & Bound theory, since the number of pruned node is strongly related to the quality of the lower bound.
- the search strategy (Section 2.2.4.1), that drives the choice of the new node to explore. The efficiency of the resulting algorithm is also strongly dependent of this heuristics, since it will influence the order in which the nodes are explored in the tree.
- the branch strategy (Section 2.2.4.2), that is to say the process of spawning subproblems. One example is given in the Algorithm 1.

The performances of such schemes are highly depending on the way these steps are performed.

2.2.4 On the exploration strategies

During the Algorithm 1, a node among active nodes in the tree has to be selected. In this Section are detailed some usual strategies that define how a new node to be branched on is selected. In Section 2.2.1, the concept of a solution tree has been presented. Each node represents a subproblem of (P) with bounds on some design variables. In this Section, it is then proposed to detail how these bounds are chosen during the process: this is part of the branch strategy.

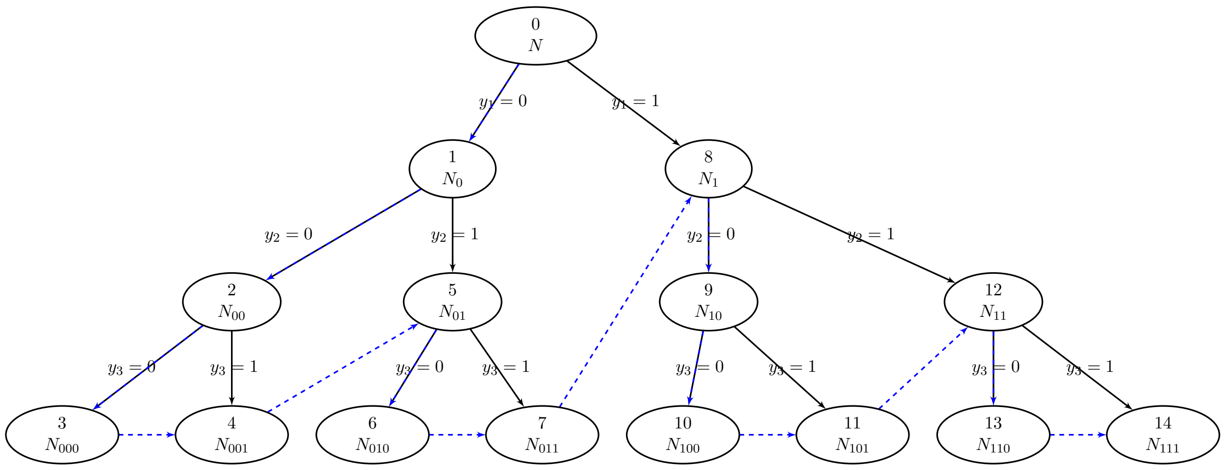


Figure 2.2: History of the tree structure corresponding to problem (MINLP) with three design variables, when the complete tree is explored following the Depth First Strategy.

2.2.4.1 The search strategies

The *search strategy* aims to select a node to branch on. There is no unconditional best strategy to define the order in which the nodes are picked in the queue Q.

- In the *depth first search* strategy (Dakin 1965), the node in Q with the largest level in the tree is chosen for exploration. One of the advantages of this strategy is that it allows to rapidly reach the bottom of the tree (where all the solutions are integral), and thus may lead to an update of the upper bound. The Figure 2.3 depicts the exploration sequence of the entire tree shown in Figure 2.1. The dashed blue arrows follows the order of the nodes for the given depth first strategy.
- In contrast to depth first strategy, the *breadth first search* strategy consists in exploring the tree horizontally. The exploration sequence is illustrated in Figure 2.3. The number of nodes at each level of the search tree grows exponentially with the level in the tree. It infeasible to do breadth first search for larger problems.
- Finally, the *best first search* (Land and Doig 1960) strategy consists in systematically picking the node in Q with the lowest lower bound. This means that the node in Q with the highest potential optimum is chosen.

2.2.4.2 The branch strategies

The branch strategy is a rule that defines a way of branching a node. As explained in Section 2.2.1, the subproblems are built by restricting the feasible space of the parent problem. This is done by adding bounds on the branching variable. The branch strategy encompasses the

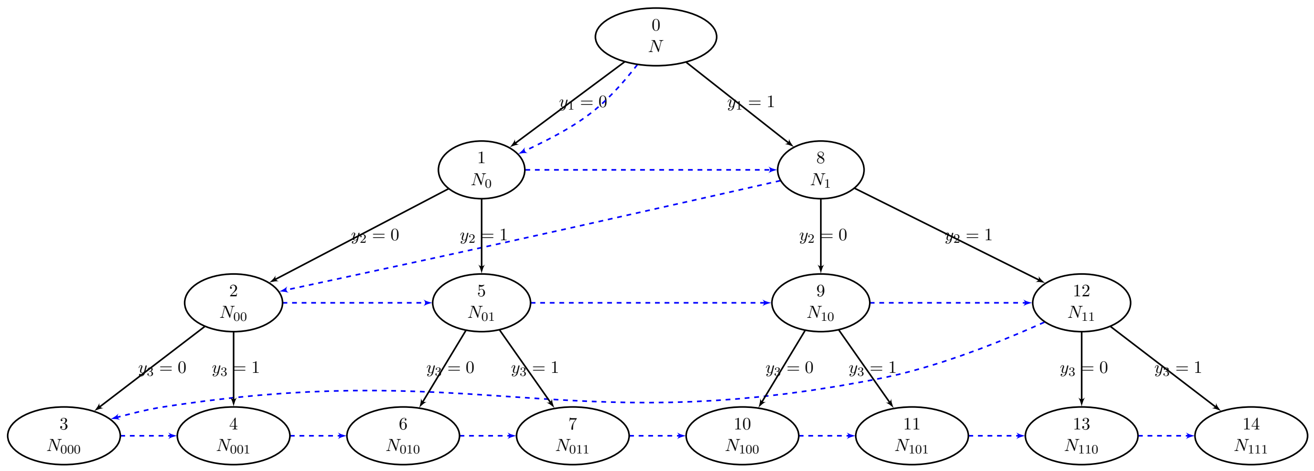


Figure 2.3: History of the tree structure corresponding to problem (MINLP) with three design variables, when the complete tree is explored following the Breadth First Strategy.

choice of the branching variable and the branching rule that defines the bounds of the subproblems. It defines the process of spawning subproblems, through the definition of subspaces. Therefore, the choice of a branch strategy could have a significant impact on the efficiency of the algorithm, since it affects the exploration of the tree. An example of a branch procedure is given hereafter. The steps are also included in the Algorithm 1.

The choice of the branching variable is a key component of the Branch & Bound algorithms. The sequence of branching variables has to be chosen such that the number of nodes to be explored is as low as possible. Therefore, this directly affects the efficiency of the algorithm. Usually, the branching variable is chosen so that it maximizes the increase in the lower bound at the child node. Indeed, it offers the best chances to compute a new lower bound that is higher than the upper bound, so that the new node can be pruned. Several strategies have been presented in the literature, like the maximum fractional branching (Ostrovsky et al. 1990), the strong branching (Applegate et al. 1995), or pseudocost branching (Benichou et al. 1971). It is also possible to branch over a randomly or predefined sequence of variables, if the problem structure is known beforehand.

Once the branching variable has been selected, the branching rule dictates the decomposition of the problem into subproblems. Usually, the decomposition is performed as follows. Let name $(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}})$ the optimal solution of (NLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)), with $\widetilde{\boldsymbol{y}}_b$ a fractional value. Let be $\widetilde{\boldsymbol{y}}_b$ the branching variable. The branching of (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) is defined such that it outputs two child nodes. These nodes get two new sets of bounds, defined as follows :

$$(\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-) := \begin{cases} (\underline{\boldsymbol{y}}_k, \lfloor \widetilde{\boldsymbol{y}}_k \rfloor) & \text{if } k = b, \\ (\underline{\boldsymbol{y}}_k, \bar{\boldsymbol{y}}_k) & \text{otherwise.} \end{cases}$$

and

$$(\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+) := \begin{cases} (\lceil \tilde{\boldsymbol{y}}_k \rceil, \bar{\boldsymbol{y}}_k) & \text{if } k = b, \\ (\underline{\boldsymbol{y}}_k, \bar{\boldsymbol{y}}_k) & \text{otherwise.} \end{cases}$$

The resulting nodes are noted (MINLP($\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-$)) and (MINLP($\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+$)).

2.2.5 An example

In order to illustrate the Algorithm 1, it is proposed to describe the steps of the algorithm applied to the following pure discrete problem, also known as an instance of the knapsack problem (KP) :

$$\begin{aligned} \underset{\boldsymbol{y} \in \{0,1\}^6}{\text{minimize}} \quad & J(\boldsymbol{y}) = -20\boldsymbol{y}_1 - 16\boldsymbol{y}_2 - 11\boldsymbol{y}_3 - 9\boldsymbol{y}_4 - 7\boldsymbol{y}_5 - \boldsymbol{y}_6 \\ \text{subject to} \quad & 9\boldsymbol{y}_1 + 8\boldsymbol{y}_2 + 6\boldsymbol{y}_3 + 5\boldsymbol{y}_4 + 4\boldsymbol{y}_5 + \boldsymbol{y}_6 - 12 \leq 0 \end{aligned} \tag{KP}$$

An illustration of the tree at the end of the process is given in Figure 2.4. In this example, it is proposed to branch on the design variables by following this order : $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\boldsymbol{y}_3$, $\boldsymbol{y}_4$, $\boldsymbol{y}_5$, $\boldsymbol{y}_6$. The restriction (KP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) of the problem (KP) is defined by

$$\begin{aligned} \underset{\boldsymbol{y} \in \{0,1\}^6}{\text{minimize}} \quad & J(\boldsymbol{y}) = -20\boldsymbol{y}_1 - 16\boldsymbol{y}_2 - 11\boldsymbol{y}_3 - 9\boldsymbol{y}_4 - 7\boldsymbol{y}_5 - \boldsymbol{y}_6 \\ \text{subject to} \quad & 9\boldsymbol{y}_1 + 8\boldsymbol{y}_2 + 6\boldsymbol{y}_3 + 5\boldsymbol{y}_4 + 4\boldsymbol{y}_5 + \boldsymbol{y}_6 - 12 \leq 0 \\ & \underline{\boldsymbol{y}} \leq \boldsymbol{y} \leq \bar{\boldsymbol{y}} \end{aligned} \tag{KP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)}$$

At each level of the tree, the nodes are branched in two nodes, (KP($\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-$)) and (KP($\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+$)). Since the design variables are binary variables, the set of bounds ($\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-$) and ($\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+$) are such that in practice it is equivalent, in both corresponding problems, to fix the branched design variable either to 0 or 1.

The lower bounds are computed by continuous relaxation of the integer variables. For example, the non-linear restricted problem (NL-KP($\underline{\boldsymbol{y}}^-$, $\bar{\boldsymbol{y}}^-$)) of the problem KP is defined by

$$\begin{aligned} \underset{\boldsymbol{y} \in [0,1]^6}{\text{minimize}} \quad & J(\boldsymbol{y}) = -20\boldsymbol{y}_1 - 16\boldsymbol{y}_2 - 11\boldsymbol{y}_3 - 9\boldsymbol{y}_4 - 7\boldsymbol{y}_5 - \boldsymbol{y}_6 \\ \text{subject to} \quad & 9\boldsymbol{y}_1 + 8\boldsymbol{y}_2 + 6\boldsymbol{y}_3 + 5\boldsymbol{y}_4 + 4\boldsymbol{y}_5 + \boldsymbol{y}_6 - 12 \leq 0 \\ & \underline{\boldsymbol{y}} \leq \boldsymbol{y} \leq \bar{\boldsymbol{y}} \end{aligned} \tag{NL-KP($\underline{\boldsymbol{y}}^-$, $\bar{\boldsymbol{y}}^-$)}$$

The process of the Algorithm 1 is now applied to the problem (KP). The discrete variables are branched in the following order: $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_6$. The tree is explored following a best first search strategy. First, the upper bound is initialized. It will be assumed that no upper bound of the optimum is known, thus $U := +\infty$. It can be noted that in this case, it would be easy to have an initial guess of an upper bound by finding a feasible solution, for example

$\boldsymbol{y}_i = 0\ \forall i \in \{1, \dots, 6\}$. The queue is initialized with the root node : $Q := \{\text{NL-KP}(-\infty, \infty)\}$.

At the first level, we pick the only node from Q, and branch on the variable $\boldsymbol{y}_1$. Q is now empty. The problem (KP) is split into two nodes $(\text{KP}(\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-))$ and $(\text{KP}(\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+))$ with $\forall k \in [\![1, 6]\!]$

$$(\underline{\boldsymbol{y}}_k^-, \bar{\boldsymbol{y}}_k^-) := \begin{cases} (0,0) & \text{if } k = 1, \\ (-\infty, \infty) & \text{otherwise.} \end{cases}$$

and

$$(\underline{\boldsymbol{y}}_k^+, \bar{\boldsymbol{y}}_k^+) := \begin{cases} (1,1) & \text{if } k = 1, \\ (-\infty, \infty) & \text{otherwise.} \end{cases}$$

Since $\underline{\boldsymbol{y}}_1^- = \bar{\boldsymbol{y}}_1^- = 0$ and $\underline{\boldsymbol{y}}_1^+ = \bar{\boldsymbol{y}}_1^+ = 1$, in practice the branched variable is fixed in both problems such that $\boldsymbol{y}_1 = 0$ and $\boldsymbol{y}_1 = 1$, respectively. The results of the corresponding non-linear problems $\text{NL-KP}(\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-)$, $\text{NL-KP}(\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+)$ are such that $J(\boldsymbol{y}^{(\underline{y}^-, \bar{y}^-)}) = -23.33$ and $J(\boldsymbol{y}^{(\underline{y}^+, \bar{y}^+)}) = -26$. These values are the lower bounds of the problems $(\text{KP}(\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-))$. These solutions are lower than the best known solution U, meaning that none of the two nodes can be pruned. The nodes are added to the queue : $Q \leftarrow Q \cup \{\text{NL-KP}(\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-), \text{NL-KP}(\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+)\}$. This is the end of the first branch and bound iteration.

The new iteration starts with the selection of a node in Q. According to the best first strategy, the new node to be selected is the one with the lowest lower bound : -26 obtained when $\boldsymbol{y}_1 = 1$. The second variable $\boldsymbol{y}_2$ is branched, so that selected node is split into two child nodes $(\text{NL-KP}(\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-))$ and $(\text{NL-KP}(\underline{\boldsymbol{y}}^+, \bar{\boldsymbol{y}}^+))$ with $\forall k \in [\![1, 6]\!]$

$$(\underline{\boldsymbol{y}}_k^-, \bar{\boldsymbol{y}}_k^-) := \begin{cases} (1,1) & \text{if } k = 1, \\ (0,0) & \text{if } k = 2, \\ (-\infty, \infty) & \text{otherwise.} \end{cases}$$

and

$$(\underline{\boldsymbol{y}}_k^+, \bar{\boldsymbol{y}}_k^+) := \begin{cases} (1,1) & \text{if } k = 1, \\ (1,1) & \text{if } k = 2, \\ (-\infty, \infty) & \text{otherwise.} \end{cases}$$

The fixed design variables are such that $\boldsymbol{y}_1 = 1$, $\boldsymbol{y}_2 = 0$ in the first node, and $\boldsymbol{y}_1 = 1$, $\boldsymbol{y}_2 = 1$ in the second one. The result of the first node is $J(\boldsymbol{y}^{(\underline{y}^-, \bar{y}^-)}) = -25.5$ and the second one has no feasible solution. The infeasible node is thus pruned, meaning that Q is updated so that $Q \leftarrow Q \cup \{\text{NL-KP}(\underline{\boldsymbol{y}}^-, \bar{\boldsymbol{y}}^-)\}$.

The same process is repeated 4 times. At this moment, the new child nodes are nodes 11 and 12 in the Figure 2.4). At these nodes, all the design variables are fixed. The associated lower bounds are -20 and -21, meaning that at the end of this branch and bound iteration the upper bound is updated such that $U = -21$. The process continues by the selection of the node 1 that was still active in Q, until the solution is found at node 21. The optimum is $J^* = -23$ with $\boldsymbol{y}_1^* = \boldsymbol{y}_3^* = \boldsymbol{y}_4^* = 0$ and $\boldsymbol{y}_2^* = \boldsymbol{y}_5^* = \boldsymbol{y}_6^* = 0$. The algorithm required the

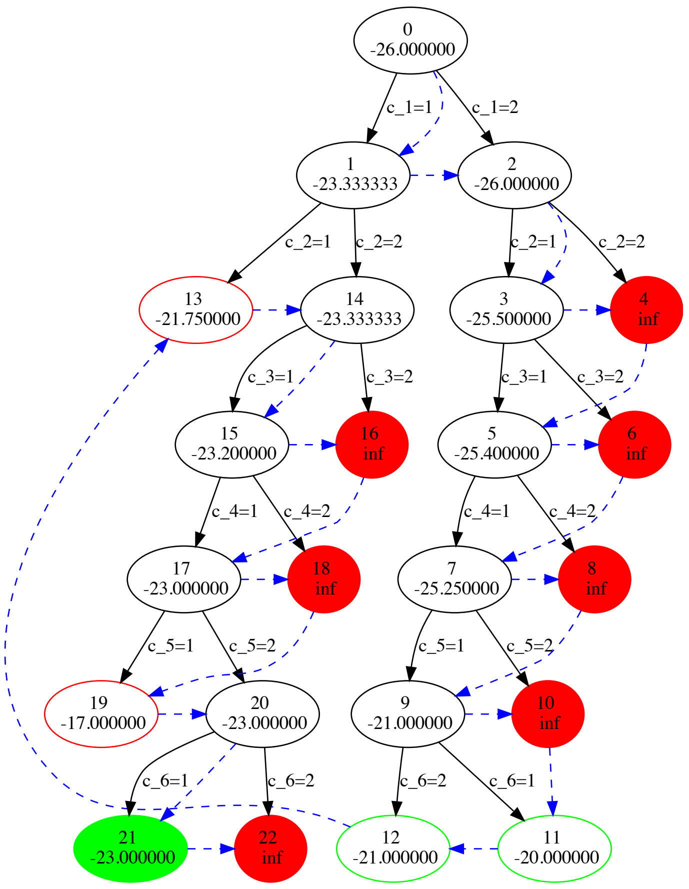


Figure 2.4: Tree solution of Problem (KP), using a depth first search strategy. The variables are branched in the following order : x_1, x_2, x_3, x_4, x_5, x_6. The lower bounds are computed by solving continuous relaxations of Problem (KP).

exploration of 23 nodes, meaning that 23 problems (NL-KP($\underline{\boldsymbol{y}}^-$, $\bar{\boldsymbol{y}}^-$)) have been solved. The total number of nodes in the tree is equal to 127, while an enumeration of 65 optimizations would have been required to solve the problem by enumeration.

2.3 A B&B formulation for solving a mixed categorical-continuous structural optimization problem

2.3.1 Problem statement

The mixed categorical-continuous optimization problem (P) is recalled here after :

$$\begin{aligned} \underset{\boldsymbol{c}\in\Gamma^n,\boldsymbol{a}\in\mathbb{R}^n}{\text{minimize}} \quad & w(\boldsymbol{a},\boldsymbol{c}) \\ \text{subject to} \quad & \boldsymbol{s}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_{n,m} \\ & \boldsymbol{\delta}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_d \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \end{aligned} \tag{P}$$

As presented in Section 2.2.2, one of the most common approaches to compute lower bounds consists of solving a continuous relaxation (NLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) of the generated sub-problems (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) in the tree. However, the categorical nominal nature of $\boldsymbol{c}$ is not compatible with a continuous relaxation. In this section, a branch and bound algorithm that allows to solve the problem (P) is presented. It is derived from the Branch & Bound theory and adapted to handle the categorical design variables of the problem (P). In particular, a branch strategy is presented in Section 2.3.2. A problem dependent relaxation that allows to compute the lower bound of the subproblem (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) is also presented in Section 2.3.3.

2.3.2 A branch strategy

As explained in the Section 2.2.4.2, the branch strategy encompasses the choice of a variable to be branch on, and the definition of the restriction of the parent node design space. The design variables that are fixed, level by level in the tree, are chosen on a criteria based on the engineering judgement. The firsts variables to be fixed are indeed those that correspond to the (supposed) highest loaded elements. That way, it maximizes the chances of producing nodes that are dominated by the current upper bound U. Unlike the choice of a branching variable, the definition of the design space restriction is not case dependent, but driven by the categorical nature of the discrete design variables. As explained in the Section 2.2.4.2, the common branching process over a node (MINLP($\underline{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$)) produces two subproblems (MINLP($\underline{\boldsymbol{y}}^-$, $\bar{\boldsymbol{y}}^-$)) and (MINLP($\underline{\boldsymbol{y}}^+$, $\bar{\boldsymbol{y}}^+$)). However, the problem (P) is different from (MINLP) because of the non-ordered and non-relaxable nature of the categorical design variable. In this adapted branch strategy, it is proposed to fix the lower and upper bounds of the branched variable to the same value. In other terms, in each new child node the branched design variable is fixed to a single value in Γ. This multiway branching produces, at each branch step, as many child nodes than the number of available catalogs in Γ. This multiplies the number of nodes at each branch step but these can be evaluated in parallel.

The subproblems are mathematically defined as follows. Suppose that the categorical design variable to branch on is $\boldsymbol{c}_b$, with $b \in \{1, \ldots n\}$. Each child node is uniquely defined by an ordered list of values taken by all previously branched variables, in addition to the new one. This list is noted $\mathcal{I}$, and is built iteratively during the Branch & Bound process. Each time a new node is built, the new list $\mathcal{I}$ relies on the parent node list updated with the value $j \in \Gamma$ of the newly fixed branched variable $\boldsymbol{c}_b = j$:

$$\mathcal{I} \leftarrow \mathcal{I} \cup \{j\} \text{ with } j \in \Gamma.$$

Let name (P($\mathcal{I}$)) the restriction of (P) on I and defined by

$$\begin{aligned} \underset{\boldsymbol{c}\in\Gamma^n,\boldsymbol{a}\in\mathbb{R}^n}{\text{minimize}} \quad & w(\boldsymbol{a},\boldsymbol{c}) \\ \text{subject to} \quad & \boldsymbol{s}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_{n,m} \\ & \boldsymbol{\delta}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_d \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \\ & \boldsymbol{c}_k = \mathcal{I}_k \quad \forall k \in \{1,\ldots,|\mathcal{I}|\} \end{aligned} \tag{P($\mathcal{I}$)}$$

This means that for all k structural elements for which $\boldsymbol{c}_k$ is fixed, with $k \in \{1,\ldots,|\mathcal{I}|\}$, all material and shape properties are fixed according to the value of $\boldsymbol{c}_k$. The design space of the problem (P($\mathcal{I}$)) counts thus $n - |\mathcal{I}|$ categorical (free) design variables and n continuous design variables (the areas). The number of structural elements with a fixed categorical design variable increases with the depth ($|\mathcal{I}|$) of the node in the tree. The maximum depth of the tree is equal to the number of categorical design variables, so that $|\mathcal{I}| \leq n$. The maximum number of nodes is defined by the number of nodes in the full tree, where no node has been pruned. Noted N_{max}, it is given by

$$N_{max} := 1 + 3 + 9 + \cdots + p^n = \sum_{i=0}^{n} p^i = \frac{p^{n+1}-1}{p-1}.$$

At the bottom of this tree (where $|I| = n$), the sub-problems (P($\mathcal{I}$)) consists of optimizations where all categorical components of $\boldsymbol{c}$ are fixed. These sub-problems correspond to a sizing optimization, that is to say an optimization with respect to the areas only. All the structural elements are defined in terms of cross-section profile and materials. The maximum number of leaves in the tree is given by the total number of nodes at the bottom of this tree. Noted N_{enum}, it corresponds to the number of sizing optimizations that would be required to solve (P) by enumeration:

$$N_{enum} := p^n.$$

2.3.3 The bound step

As explained in Section 2.2.2, the bound step aims to compute a lower bound of a restricted problem (P($\mathcal{I}$)) optimum. In general, the lower bound is obtained by solving a continuous relaxation of the problem (P($\mathcal{I}$)). It consists of turning the remaining free discrete design variables into continuous design variables. In the case of the subproblem ($\mathrm{P}(\mathcal{I})$), the categorical design space Γ is unordered and non-relaxable. Therefore, the relaxation of ($\mathrm{P}(\mathcal{I})$) is not as straightforward as the continuous relaxation of ($\mathrm{MINLP}(\underline{\boldsymbol{y}}, \bar{\boldsymbol{y}})$), where the functions are defined over a continuous space. A specific relaxation problem is thus needed in order to be able to provide a lower bound of each subproblem in the tree.

First, the categorical nature of Γ is discussed. To each categorical variable corresponds a choice of stiffener profile and material. The choice of a material involves latent physical material properties. Each of these latent variables is continuous and ordered : the density, Young modulus, Poisson modulus, stress limits, etc. Actually, the non-ordered nature of the material choices comes from the fact that there are several material properties. There is no straightforward norm definition to sort a list of materials, that would have a physical meaning. However, it is interesting to remark that the computation of the internal forces depends on (only) one specific property, that is the Young modulus (c.f. Table 1.1).

Relying on this, a specific relaxation problem is proposed. It leverages the continuous definition of the Young modulus property. In order to address the remaining categorical part of the design variables, the proposed relaxation problem involves an under-estimation of the objective and constraint functions. Combined with the continuous relaxation of the Young modulus, this ensures to provide a lower bound of ($\mathrm{P}(\mathcal{I})$). Here is defined the design space $\widetilde{\Gamma}$:

$$\widetilde{\Gamma} = [E_{min}, E_{max}],$$

that is a continuous design set bounded by the minimum and maximum Young modulus values (E_{min} and E_{max}, repectively) of all the materials available through all choices in Γ :

$$\begin{cases} E_{min} = \min\{E(c)\ \forall c \in \Gamma\} \quad (E_{min} \in \mathbb{R}) \\ E_{max} = \max\{E(c)\ \forall c \in \Gamma\} \quad (E_{max} \in \mathbb{R}). \end{cases}$$

In other words, $\widetilde{\Gamma}$ is a continuous relaxation of Γ over only one latent variable, that is the Young modulus. No other description of any material nor stiffener property is included into $\widetilde{\Gamma}^n$, meaning that the other missing components will need to be introduced through new definitions of the functions involved in the optimization problem. First, the function $\widetilde{w}$ that computes the weight depends only on the areas $\boldsymbol{a}$ and is given by :

$$\widetilde{w}(\boldsymbol{a}) = \begin{cases} \rho_{min} \sum_{k=1}^{n} \boldsymbol{a}_k \boldsymbol{L}_k, & \text{if } |\mathcal{I}| = 0 \\ \sum_{k=1}^{|\mathcal{I}|} \boldsymbol{a}_k \rho(\mathcal{I}_k) \boldsymbol{L}_k + \rho_{min} \sum_{k=|\mathcal{I}|+1}^{n} \boldsymbol{a}_k \boldsymbol{L}_k, & \text{if } 1 \leq |\mathcal{I}| < n \\ \sum_{k=1}^{n} \boldsymbol{a}_k \rho(I_k) \boldsymbol{L}_k, & \text{otherwise.} \end{cases}$$

with $\mathcal{I}_k$ the $k^{(th)}$ element of the list $\mathcal{I}$. For the k structural elements for which the categorical variable is fixed to $\mathcal{I}_k$, the density is set to $\rho(\mathcal{I}_k)$. The density of the other elements is fixed to the minimum density ρ_{min}, in order to ensure that the weight computed by $\widetilde{w}(\boldsymbol{a})$ is always lower than the weight computed by $w(\boldsymbol{a})$.

In the stress constraints definition, similarly, the material properties and quadratic moments involved in the limit stress definitions are fixed to the most appropriate value (their maximum value). In facts, the stress limits on undetermined structural elements are defined so that original constraints s_{kj} are always stronger than their relaxation $\widetilde{s}_{kj}$ for $j \in \{1, \ldots, 4\}$. The relaxed stress constraints definitions are given hereafter, with $k \in \{1, \ldots, n\}$:

$$\widetilde{s}_{k1}(\boldsymbol{a}, \boldsymbol{E}) := \begin{cases} \dfrac{\boldsymbol{\Phi}_k(\boldsymbol{a}, \boldsymbol{E})}{\boldsymbol{a}_k} - \sigma^t(\mathcal{I}_k), & \text{if } k \leq |\mathcal{I}| \\ \dfrac{\boldsymbol{\Phi}_k(\boldsymbol{a}, \boldsymbol{E})}{\boldsymbol{a}_k} - \sigma^t_{max}, & \text{otherwise} \end{cases}$$

$$\widetilde{s}_{k2}(\boldsymbol{a}, \boldsymbol{E}) := \begin{cases} \dfrac{\boldsymbol{\Phi}_k(\boldsymbol{a}, \boldsymbol{E})}{\boldsymbol{a}_k} - \sigma^c(\mathcal{I}_k), & \text{if } k \leq |\mathcal{I}| \\ \dfrac{\boldsymbol{\Phi}_k(\boldsymbol{a}, \boldsymbol{E})}{\boldsymbol{a}_k} - \sigma^c_{max}, & \text{otherwise} \end{cases}$$

$$\widetilde{s}_{k3}(\boldsymbol{a}, \boldsymbol{E}) := \begin{cases} \dfrac{\boldsymbol{\Phi}_k(\boldsymbol{a}, \boldsymbol{E})}{\boldsymbol{a}_k} - \dfrac{\pi^2 E(\mathcal{I}_k) I(\boldsymbol{a}_k, \mathcal{I}_k)}{\boldsymbol{a}_k \boldsymbol{L}_k^2}, & \text{if } k \leq |\mathcal{I}| \\ \dfrac{\boldsymbol{\Phi}_k(\boldsymbol{a}, \boldsymbol{E})}{\boldsymbol{a}_k} - \dfrac{\pi^2 E_{max} I_{max}(\boldsymbol{a}_k)}{\boldsymbol{a}_k \boldsymbol{L}_k^2}, & \text{otherwise} \end{cases}$$

$$\widetilde{s}_{k4}(\boldsymbol{a}, \boldsymbol{E}) := \begin{cases} \dfrac{\boldsymbol{\Phi}_k(\boldsymbol{a}, \boldsymbol{E})}{\boldsymbol{a}_k} - \dfrac{4\pi^2 E(\mathcal{I}_k) \mathcal{K}^2(\mathcal{I}_k)}{12(1 - \nu^2(\mathcal{I}_k))}, & \text{if } k \leq |\mathcal{I}| \\ \dfrac{\boldsymbol{\Phi}_k(\boldsymbol{a}, \boldsymbol{E})}{\boldsymbol{a}_k} - \dfrac{4\pi^2 E_{max} \mathcal{K}^2_{max}}{12(1 - \nu^2_{max})}, & \text{otherwise.} \end{cases}$$

In the case of the constraint on displacements, the replacement of the categorical design variable by the Young Modulus is the only change in the function $\boldsymbol{\delta}$. The constraints on displacements are given by :

$$\boldsymbol{\delta}(\boldsymbol{a}, \boldsymbol{E}) := \boldsymbol{P}\boldsymbol{u}(\boldsymbol{a}, \boldsymbol{E}) - \bar{\boldsymbol{u}}.$$

The relaxed optimization problem (rP($\mathcal{I}$)) of the problem (P($\mathcal{I}$)) is thus given by

$$\begin{array}{ll} \underset{\boldsymbol{E} \in \widetilde{\Gamma}^n, \boldsymbol{a} \in \mathbb{R}^n}{\text{minimize}} & \widetilde{w}(\boldsymbol{a}, \boldsymbol{E}) \\ \text{subject to} & \widetilde{\boldsymbol{s}}(\boldsymbol{a}, \boldsymbol{E}) \leq \boldsymbol{0}_{m,n} \\ & \widetilde{\boldsymbol{\delta}}(\boldsymbol{a}, \boldsymbol{E}) \leq \boldsymbol{0}_d \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}}, \\ & \boldsymbol{E}_k = E(\mathcal{I}_k) \ \forall k \in \{1, \ldots, |\mathcal{I}|\}. \end{array} \tag{rP($\mathcal{I}$)}$$

with $|\mathcal{I}|$ the number of elements in $\mathcal{I}$. This optimization problem is a full continuous optimization problem that yields a lower bound of the corresponding subproblem (P($\mathcal{I}$)). This problem can be efficiently solved by a gradient based algorithm.

Despite this advantage, the following remark concerning this relaxation formulation has to be noted. Indeed, a solution of the problem (rP($\mathcal{I}$)) corresponding to a discrete Young modulus selection (not an intermediate one) is not necessary a solution of (P($\mathcal{I}$)). Indeed, this is due to the inconsistency between the material design variables that are the Young moduli, and the definition of the limit stresses and the densities of the free structural elements. For example, let be $(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})})$ the solution of the problem (rP($\mathcal{I}$)). Let suppose that $(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})})$ is such that the Young modulus value of the k^{th} element is equal to $\boldsymbol{E}_k(j)$ with $j \in \Gamma$. There is no guarantee that the same catalog j verifies $\rho(j) = \rho_{max}$, $\sigma^t(j) = \sigma^t_{max}$, $\sigma^c(j) = \sigma^c_{max}$, $\nu(j) = \nu_{max}$, $I(\boldsymbol{a}_k^{(\mathcal{I})}, j) = I_{max}(\boldsymbol{a}_k^{(\mathcal{I})})$, and $\mathcal{K}(j) = \mathcal{K}_{max}$. As a consequence, a solution $(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})})$ such that the Young moduli $\boldsymbol{E}^{(\mathcal{I})}$ of all structural elements correspond to existing values $\boldsymbol{E}(\boldsymbol{c})$ with $\boldsymbol{c} \in \Gamma^n$, is rarely a solution of (P($\mathcal{I}$)). In other words, the chances to update the upper bound U when a solution corresponds to discrete values of Young moduli is found are reduced, when compared to the usual branch and bound algorithms that involve trivial continuous relaxations (as presented in Section 2.2.5), without changing the definition of the problem functions. However, when the categorical design space is entirely fixed (i.e., at the bottom of the tree when $|\mathcal{I}| = n$), the (fixed) categorical design variables noted $\boldsymbol{c}^{(\mathcal{I})}$ and defined by

$$\boldsymbol{c}_k^{(\mathcal{I})} := \mathcal{I}_k \quad (\forall k \in \{1, \ldots, n\})$$

are such that $(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{c}^{(\mathcal{I})})$ is the optimum solution of (P($\mathcal{I}$)). Indeed, in this case all the categorical design variables are fixed, and the problems (P($\mathcal{I}$)) and (rP($\mathcal{I}$)) are equivalent.

2.3.4 The search strategy

In practice, the definition of the search strategy is strongly dependent to the structure of the problem. In the case of the mixed categorical-continuous structural optimization problem (P), it can be noted that during the bound step all the areas are involved in each evaluation problem (rP($\mathcal{I}$)). In practice, it has been observed that the evaluation problems almost always admit a feasible solution, thanks to the areas. Indeed, the optimal areas tend to balance the effects of a potential under-optimal categorical fixed choice, so that the optimal solution remains feasible. This means that in practice, the pruning rule based on the evaluation problem solution feasibility will not be useful in the case of (P). Of course, under-optimal fixed choices are responsible of large optimal areas values, such that the resulting optimal weight is increased as well. Hence, the remaining helpful pruning rule consists of updating the upper bound U as often as possible. By doing so, the chances to get dominated nodes are improved. The proposed search strategy is thus build according to these remarks. It consists of a depth first search strategy combined with a best first search strategy as described hereafter. The node picked in the queue is indeed chosen according to two criteria. The node is chosen so that it has the highest level value $|\mathcal{I}|$ in the tree. However, due to the multiway-branching strategy, there are at least p nodes that are on the same level. Thus, the picked node is chosen so that its lower bound is the lowest, among all the nodes with the same highest level value.

2.3.5 The algorithmic process

The Branch & Bound algorithm proposed process adapted to the problem (P) is described in the Algorithm 2. It follows the main steps of the generic branch and bound algorithm example given in Algorithm 1. However, the branch strategy has been adapted, as described in Section 2.3.2, as well as the bound step given in Section 2.3.3 and the search strategy in Section 2.3.4.

Let be Q the set of *active* nodes in the tree. Let define U an upper bound of the optimal solution of (P). An initialization of the process consists of setting up an upper bound U of the optimal solution of (P). Either a value is already known by experience by a designer, or the upper bound is set to $+\infty$. Also, the queue is initialized such that it contains only the root node (rP($\{\emptyset\}$)).

The solution tree can now be progressively built. This is first done by picking up a node (rP($\mathcal{I}$)) in Q following the search strategy detailed in the Section 2.3.4, as long as the queue is not empty. At the first iteration of the algorithm, the picked node is the root node where no categorical design variable has been fixed. The algorithm proceeds then to the branching on the picked node according to the branch strategy defined in the Section 2.3.2. It outputs p child nodes that are (rP($\mathcal{I} \cup \{1\}$)), $\ldots$, (rP($\mathcal{I} \cup \{p\}$)). For each of the p child nodes, the following steps rules are applied.

The subproblem (rP($\mathcal{I}$)) is solved. The solution is noted $(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})})$. There are four possible cases :

- (*infeasible node*) If (rP($\mathcal{I}$)) is infeasible, then any descending subproblem in the tree is also infeasible, by design of the tree. The node can be prune, meaning that the entire associated subspace can be discarded.
- (*dominated node*) If (rP($\mathcal{I}$)) is feasible and the optimal value $\widetilde{w}(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})})$ of (rP($\mathcal{I}$)) is dominated by the best known solution U so far, then the node can be pruned. Indeed, if the lower bound $\widetilde{w}(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})})$ of (P) is greater than the best known solution, this means that the subspace does not worth to be explored. Every feasible solution of this node or its descendant will be greater than the best known solution.
- (*fixed node*) If the node is such that all the categorical choices are fixed (if $|\mathcal{I}| = n$), and if $\widetilde{w}(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})}) < U$, then the node is a new incubent. The best known weight can be updated such that $U := \widetilde{w}(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})})$, obtained with the following optimal areas and choices $(\boldsymbol{a}^*, \boldsymbol{c}^*) := (\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{c}^{(\mathcal{I})})$ with $\boldsymbol{c}_k^{(\mathcal{I})} := \mathcal{I}_k \quad (\forall k \in \{1, \ldots, |\mathcal{I}|\})$.
- (*active node*) If the current child does not meet any of the aforementioned conditions, the node is still active. This means that the design sub-space covered by the child node contains possibly the optimal solution. The node is thus added into the queue, and Q is updated accordingly.

This ends the current iteration. The next one starts by picking another node in Q, until Q is empty. The result obtained at the end of the procedure is given by the current upper bound. It is proved to be a global optimum, as soon as the relaxation problems are solved to global optimality. The described procedure is detailed in Algorithm 2.

Algorithm 2 An example of generic Branch & Bound framework

```
 1: initialize $U := \infty$ or best known solution of (P)
 2: initialize $\mathcal{I} := \{\emptyset\},\ Q := \{\mathrm{rP}(\mathcal{I})\}$
 3: while $Q \neq \{\emptyset\}$ do
 4:     Pick a node $(\mathrm{rP}(\mathcal{I}))$ in $Q$                              ▷ Search strategy
 5:     $Q \leftarrow Q \setminus \{\mathrm{rP}(\mathcal{I})\}$
 6:     Select a variable $\boldsymbol{c}_b$, with $b \in \{|\mathcal{I}|, \dots, n\}$           ▷ Branch strategy
 7:     children $\leftarrow \{\mathrm{rP}(\mathcal{I} \cup \{1\}), \dots, \mathrm{rP}(\mathcal{I} \cup \{p\})\}$
 8:     for every $(\mathrm{rP}(\mathcal{I}))$ among children do
 9:         Solve $(\mathrm{rP}(\mathcal{I}))$ and let its solution be $(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})})$      ▷ Bound evaluation
10:         if $(\mathrm{rP}(\mathcal{I}))$ is infeasible then
11:             Prune node (infeasible case)
12:         else if $\widetilde{w}(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{E}^{(\mathcal{I})}) > U$ then
13:             Prune node (dominated)
14:         else if $|\mathcal{I}| = n$ then
15:             Update best known solution :
16:             $\boldsymbol{c}_k^{(\mathcal{I})} := \mathcal{I}_k \quad (\forall k \in \{1, \dots, |\mathcal{I}|\})$
17:             $U \leftarrow w(\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{c}^{(\mathcal{I})})$
18:             $(\boldsymbol{x}^*, \boldsymbol{c}^*) \leftarrow (\boldsymbol{a}^{(\mathcal{I})}, \boldsymbol{c}^{(\mathcal{I})})$
19:         else
20:             $Q \leftarrow Q \cup \{\mathrm{rP}(\mathcal{I})\}$
21:         end if
22:     end for
23: end while
24: return $\boldsymbol{x}^*, \boldsymbol{y}^*, w^* \leftarrow U$.
```

2.4 Implementation details and comparison solvers

Algorithm 2 has been implemented using the Generic Engine for MDO Scenarios (GEMS) (Gallard et al. 2018) in Python. The tool offers an efficient way to implement and test multi-level formulations, with built-in classes that facilitate optimization problems manipulations (Gallard et al. 2019). Adapted post processing have also been implemented, including graphical representations of the solution tree step by step during the optimization. The continuous optimization problems are solved with the Method of Moving Asymptotes (MMA) (Svanberg 2002). In what comes next, the resulting implementation of Algorithm 2 will be called the hybrid branch and bound, in reference to the fact that it can handle both categorical design variables and continuous design variables. It will be noted h-B&B.

Two solvers will be compared to h-B&B. First, a baseline solver where we proceed an exhaustive enumeration of continuous optimizations w.r.t. $\boldsymbol{a}$ taken at every available choice

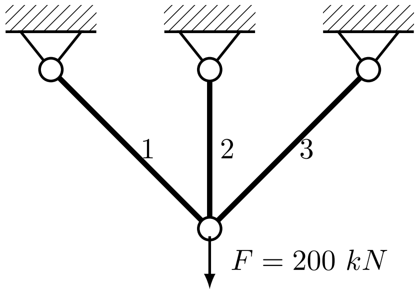


Figure 2.5: A 3-bar truss structure where a downward load $F = 200\ kN$ is applied on the free node.

in Γ^n, the obtained solution by this solver will be denoted as Baseline. Second, a Genetic algorithm (Deb and Goyal 1998) using the implementation given by Distributed Evolutionary Algorithms in Python (DEAP) (Fortin et al. 2012). This solver will be referred by Genetic in our comparison tests. Due to the stochastic nature of Genetic, the obtained results (for this solver) will be displayed as the average of ten runs.

In all what comes next, the computation effort of a given solver will be measured by counting the number of structural analyses (noted #FEM) including those required by the computation of the gradients (when needed). The obtained optimal weights (by each solver) will be noted w^*, the latter will allow us to evaluate the quality of the optima found by each solvers. We note also that in our setting, the Baseline solution can be seen as the best known categorical choices for the regarded problem. Thus, in this context, it is important to evaluate how far the categorical choices (obtained by the tested solvers) from the Baseline optimal choices are. This information will be given using the Hamming distance (noted d_h) where we will count the number of structural elements that has an optimal choice different to the Baseline categorical choices.

2.5 Numerical results

In the present section, the proposed methodology will be applied to two different test cases, that are the well-known 10-bar truss structure (Haftka and Gürdal 1992), and a 2D cantilever structure (Shahabsafa et al. 2018). In order to evaluate the scalability of the methodology with respect to the number of structural elements, the 2D cantilever structure is made scalable by varying the number of blocks.

2.5.1 A step by step example: a 3-bar truss structure

To illustrate how the h-B&B method works, we will now describe in detail its application to a simple 3-bar truss structure depicted in Figure 2.5. For this problem, each element can take a value among three possible choices that respectively point to materials AL2024, AL2139, TA6V and the same “I”-profile (see Figure 1.3). The materials properties are listed in Table

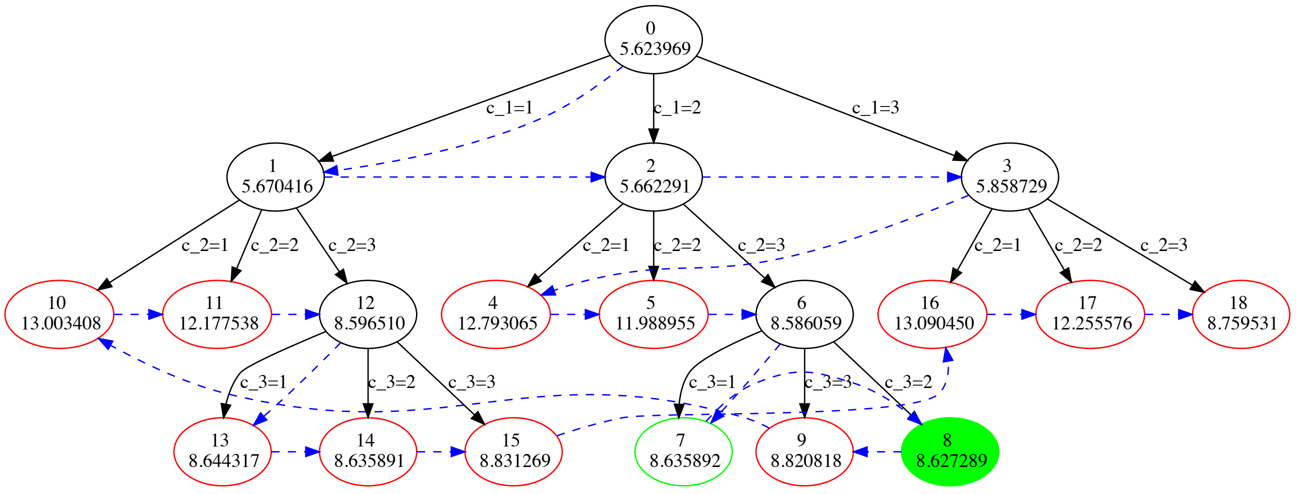


Figure 2.6: A solution tree of the 3-bar truss categorical-continuous optimization problem with 3 catalogs. The categorical variables are branched following the order $[\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3]$. The blue arrows follow the evaluation order of the nodes.

2.1. For this simple case, one has $n = 3$, $p = 3$, and $\Gamma = \{1, 2, 3\}$. For all elements, the lower

	AL2139	AL2024	TA6V
Density (kg/mm^3)	$2.8\ 10^{-6}$	$2.77\ 10^{-6}$	$4.43\ 10^{-6}$
Young modulus (MPa)	$7.1\ 10^4$	$7.4\ 10^4$	$11.0\ 10^4$
Poisson coefficient $(-)$	0.3	0.33	0.33
Tension allow. (MPa)	$1.5\ 10^2$	$1.6\ 10^2$	$11.0\ 10^2$
Compression allow. (MPa)	$2.0\ 10^2$	$2.1\ 10^2$	$8.6\ 10^2$

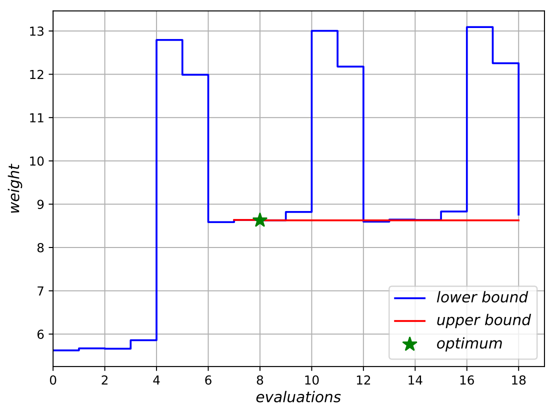
Table 2.1: Numerical details on materials attributes.

and upper bounds on areas are respectively fixed to 100 mm^2 and 2000 mm^2. The initial areas are fixed to the upper bound values, as detailed in Table 2.2. A maximum downward displacement equal to 1 mm is allowed on the only free node of the structure.

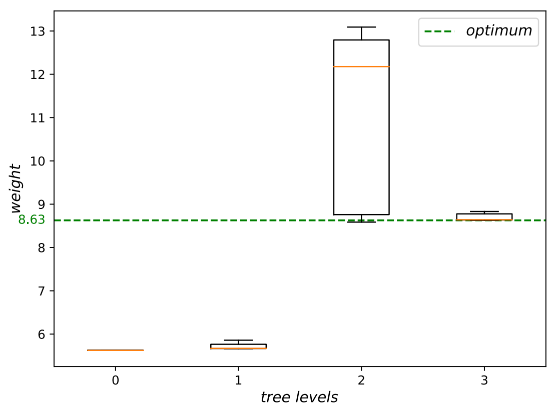
Each node of the solution tree corresponds to a relaxation problem of the 3-bar instance of the restriction problem $(\mathrm{P}(\mathcal{I}))$. Each relaxation problem is given by:

$$\begin{aligned} \underset{\boldsymbol{E}\in\widetilde{\Gamma}^3, \boldsymbol{a}\in\mathbb{R}^3}{\text{minimize}} \quad & \widetilde{w}(\boldsymbol{a}, \boldsymbol{E}) \\ \text{subject to} \quad & \widetilde{\boldsymbol{s}}(\boldsymbol{a}, \boldsymbol{E}) \leq \boldsymbol{0}_{m,n} \\ & \widetilde{\boldsymbol{\delta}}(\boldsymbol{a}, \boldsymbol{E}) \leq \boldsymbol{0}_d \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}}, \\ & \boldsymbol{E}_k = E(\mathcal{I}_k)\ \forall k \in \{1, \dots, |\mathcal{I}|\}. \end{aligned} \tag{r-3P($\mathcal{I}$)}$$

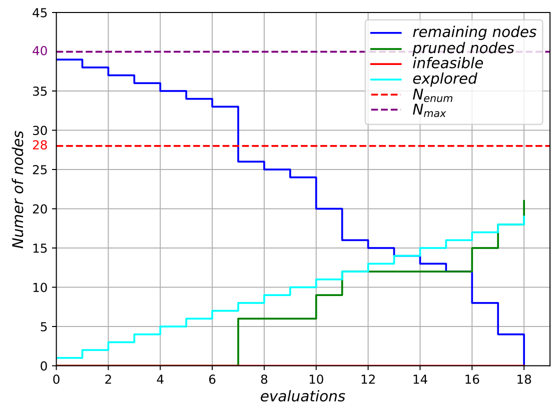
with $\mathcal{I}$ the set of values of the fixed categorical design variables.



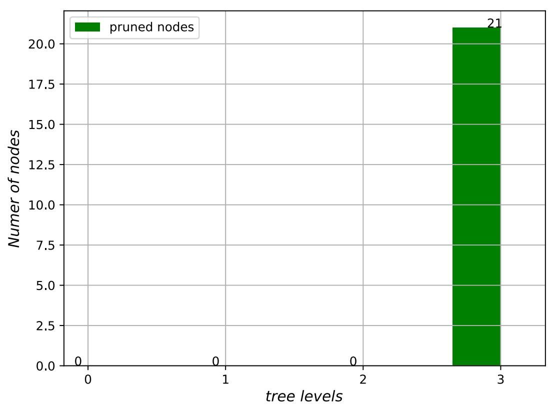
(a) Lower and upper bounds w.r.t. the number of problem evaluations



(b) Statistics on the lower bounds computed at each tree level.



(c) Evolution of the number of nodes in the solution tree.



(d) Number of pruned nodes in the tree that have not been branched.

Figure 2.7: The convergence history of the branch and bound Algorithm 2, on a 3-bar truss instance problem with 3 catalogs.

The ordered list of branching variables is set to : $[\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3]$, that are the categorical design variables describing material and shape of the structural elements 1, 2 and 3 respectively. The queue Q is initialized with the node (r-3P($\mathcal{I}$)), with $\mathcal{I} = \{\emptyset\}$. We assume that there is no prior knowledge of a feasible solution to the problem. The upper bound is thus fixed to infinity. The solution tree is now progressively built. It is possible to follow the tree generation in Figure 2.6 thanks to the node creation order, given by the first number in each circle. The optimization history is depicted on Figure 2.7.

The first iteration consists first of taking the only node in Q (Q is now empty). The first branching variable is $\boldsymbol{c}_1$. The categorical variable is fixed successively to 1, 2 and 3, such that $\mathcal{I} = \{1\}$, $\mathcal{I} = \{2\}$, $\mathcal{I} = \{3\}$, respectively. This corresponds to the nodes 1, 2 and 3 (respectively) in the Figure 2.6. The lower bounds of these three new nodes are obtained by solving the relaxation problem instances (rP($\mathcal{I}$)). Their values are 5.67 kg, 5.66 kg, 5.86 kg,

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$2000\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$2000\ mm^2$

Table 2.2: Bounds on areas, and initial areas values of the 3-bar truss optimization case.

respectively. There is a feasible solution for each of these evaluation problems, and each of these lower bound values is lower than the current upper bound (initialized to infinity). The three nodes are thus flagged as active nodes and added in the queue.

The second iteration begins, we proceed to the next branching and bounding steps. Following the depth first and best first search strategy detailed in section 2.3.4, the best of the three nodes of the highest level ($|\mathcal{I}| = 1$) in the tree is taken from the queue. Since the best node is the one that returned the best (lowest) lower bound, the node 2 ($I = [2]$) is the node that will be explored. It is removed from Q. The second branching variable is $\boldsymbol{c}_2$. Again, the newly fixed categorical variable $\boldsymbol{c}_2$ is fixed successively to 1, 2 and 3, while $\boldsymbol{c}_1 = 2$: this corresponds to the nodes 4, 5 and 6, where $\mathcal{I} = \{2, 1\}$, $\mathcal{I} = \{2, 2\}$, $\mathcal{I} = \{2, 3\}$ (respectively). The corresponding computed lower bound values are 12.79 kg, 11.99 kg, 8.59 kg. There is a feasible solution for each of these evaluation problems, and the lower bound value is lower than the current upper bound (initialized to infinity). The three nodes are thus flagged as active nodes and added in the queue, that counts the nodes 1,3,4,5,6.

The third iteration begins, we proceed to the next branching and bounding steps. The best lower bound of the level 2 in the tree is returned by the node 6 ($I = [2, 3]$) that is the next node to explore. The third branching variable is $\boldsymbol{c}_3$. The newly fixed categorical variable $\boldsymbol{c}_3$ is fixed successively to 1, 2 and 3, while $\boldsymbol{c}_1 = 2$ and $\boldsymbol{c}_2 = 3$: This corresponds to the nodes 7, 8 and 9, where $\mathcal{I} = \{2, 3, 1\}$, $\mathcal{I} = \{2, 3, 2\}$, $\mathcal{I} = \{2, 3, 3\}$ (respectively). Since this is the last level in the tree, at these nodes all the categorical design variables are fixed. This means that in the restriction problem and the evaluation problem, the areas are the only remaining design variables. The solutions of these node are thus upper bounds of the optimal solution of (P). The corresponding computed upper bound values are 8.635 kg, 8.627 kg, 8.82 kg. The best known solution is thus returned by the node 9, with $\mathcal{I} = \{2, 3, 2\}$. The (current) optimal categorical choices are thus given by $\boldsymbol{c}^* = \boldsymbol{c}^{\mathcal{I}} = [2, 3, 2]$, and the upper bound is updated such that $U = 8.627\ kg$. None of the newly created nodes 7, 8 and 9 are stored in Q, since it is not possible to split their design space ($|\mathcal{I}| = 3$).

A first solution has thus be found, as expected since this is the first time that all the categorical variables are fixed, and provided that the upper bound was initialized to ∞. However, there are still active nodes in the queue, meaning that potential better solutions are still to be found. Among these active nodes, the nodes 4 and 5 are the next candidates. However, since they are dominated by the current best solution, they are discarded. A new node is picked, the node 1, and new child nodes are evaluated. The same process is repeated over all the remaining nodes in Q until Q is empty.

At the end of the process, the optimum found is $\boldsymbol{c}^* = [2, 3, 2]$, with an optimal weight that is $w^* = 8.627\ kg$. In the Figure 2.7a, is depicted the upper bound evolution during the process. It can be seen that the upper bound is unchanged after the 9^{th} problem evaluation (root node 0 included). In the Figure 2.7c, one can see gaps in the evolution of the number of remaining nodes to be explored, and the number of pruned nodes. This corresponds to the removal of the dominated nodes once the upper bound has been updated. Since 7 nodes have been discarded at a level 2 in the tree, the exploration of $3 \times 7 = 21$ nodes was spared. This is displayed in the Figure 2.7d, where we can see the number of pruned nodes that were not needed to be explored. In the Figure 2.7c, it can also be seen that 19 nodes (including the root node) were explored, meaning that 19 optimizations were needed to end the algorithm. This has to be compared to the 28 optimizations that would be required to solve the problem (P) by enumeration of sizing optimizations. Finally, in the Figure 2.7b one can see the value of the lower bounds obtained per level in the tree. These values have to be compared to the optimal weight 8.63 kg. Indeed, it indicates that the lower bounds computed at the level 2 are higher than the optimum. In our case, the level 2 corresponds to restrictions where the design variable $\boldsymbol{c}_2$ is fixed. This provides an interesting information, because it may help at refining the branch strategy. Indeed, it could mean that branching first on the variable $\boldsymbol{c}_2$ (instead of $\boldsymbol{c}_1$) would accelerate the convergence of the branch and bound algorithm.

For the purpose of this example, the same optimization is thus performed with the only chance in the order of the branching variables : $[\boldsymbol{c}_2, \boldsymbol{c}_1, \boldsymbol{c}_3]$. The computation cost is indeed reduced since the number of explored nodes falls to 13. The solution tree is smaller, as seen in Figure 2.8. The efficiency of this new ordering of the branching variables can be observed in the Figure 2.9. In Figure 2.9b, it can be remarked that the lower bounds at the first level of the tree depicted in the Figure 2.8 are such that they are above the optimal weight. Thus, once the node 7 reached, the upper bound is updated and the 2 nodes at the first level of the tree are pruned. This is at the origin of the step visible in Figure 2.9c, on the green curve for example. On the Figure 2.9d, one can see that since 2 nodes have been branched and pruned at the first level of the tree, 2×3 nodes are discarded at the second level of the tree, and $2 \times 3 \times 3$ at the bottom of the tree (among the 21 pruned nodes mentionned in the plot).

2.5.2 A 10-bar truss structure

The 10-bar truss problem is illustrated Figure 3.3. A downward load $F = 200\ kN$ is applied vertically on the free node N_δ. A constraint on displacements is applied on the same node. Five cases with different bounds values $\bar{\boldsymbol{u}}$ on displacements are considered. For each of these cases, the displacements constraint is applied on node N_δ. Catalogs 1 and 2 point to materials AL2139 and TA6V, respectively. Materials properties are listed in Table 2.1. In this case, $n = 10$ and $p = 2$, $\Gamma = \{1, 2\}$. The bounds on areas, and initial areas are fixed as detailed in Table 2.4.

The results of the proposed methodology (h-B&B) are thus compared to the global optima obtained with `Baseline` (obtained by enumeration), and with the results returned by Genetic algorithm as shown in Table 2.3. In all these cases, the optima obtained with h-B&B and `Baseline` (obtained by enumeration) are identical. This means that the h-B&B, in these

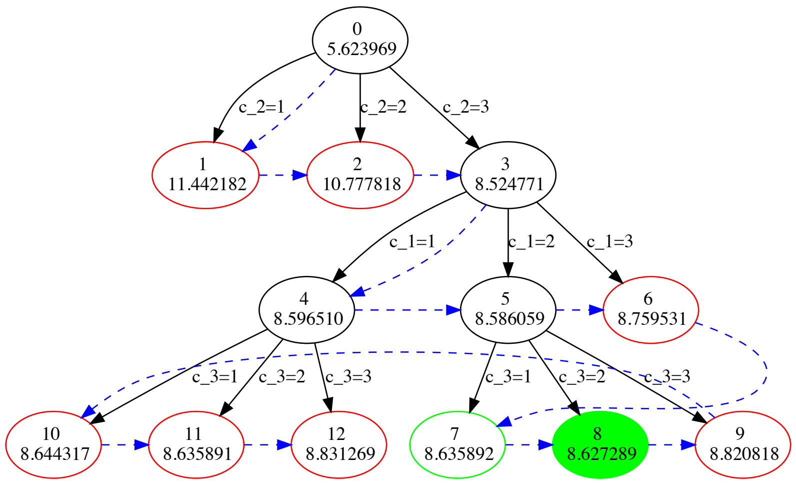


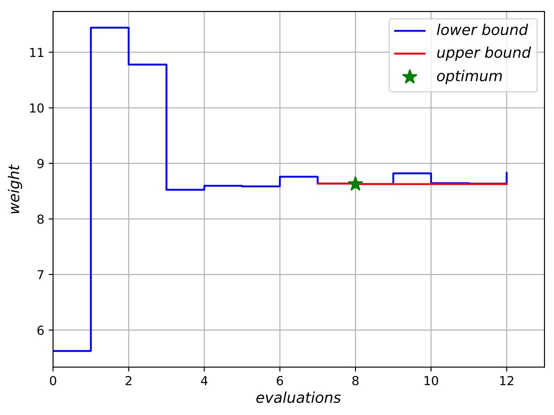
Figure 2.8: A solution tree of the 3-bar truss categorical-continuous optimization problem with 3 catalogs. The categorical variables are branched following the order $[\boldsymbol{c}_2, \boldsymbol{c}_1, \boldsymbol{c}_3]$. The blue arrows follow the order of the evaluations.

Table 2.3: Results of 10-bar truss mixed optimization with 5 different values of constraint on displacements. Comparison between the the Baseline solutions obtained by enumeration of the 2^{10} continuous optimizations, h-B&B, and the Genetic algorithm. The catalog 1 corresponds to material AL2139 and catalog 2 to TA6V.

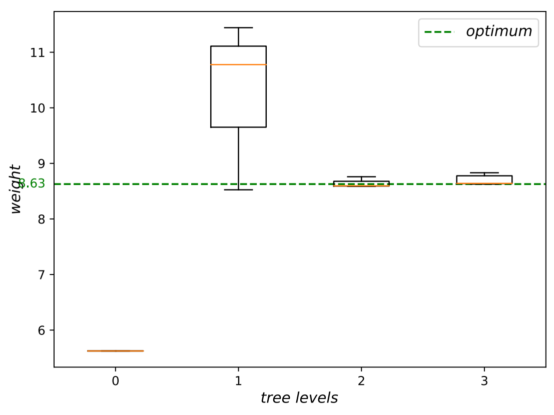
$\bar{\boldsymbol{u}}$ (mm)	Baseline		h-B&B		Genetic	
	c^*	$w^*(kg)$	d_h	$w^*(kg)$	d_h	$w^*(kg)$
-22	[2,2,1,1,1,2,2,1,2,1]	12.988	0	12.988	0	13.283
-20	[2,1,1,1,1,1,2,1,1,1]	13.996	0	13.996	0	14.423
-19	[2,1,1,1,1,1,2,1,1,1]	14.570	0	14.570	0	14.802
-18	[1,1,1,1,1,1,1,1,1,1]	15.175	0	15.175	2	15.642
-17	[1,1,1,1,1,1,1,1,1,1]	15.912	0	15.912	3	16.258

cases, provides the global solution. On the other hand, the weights returned by the Genetic algorithm are greater than the optimal weight found by the h-B&B approach.

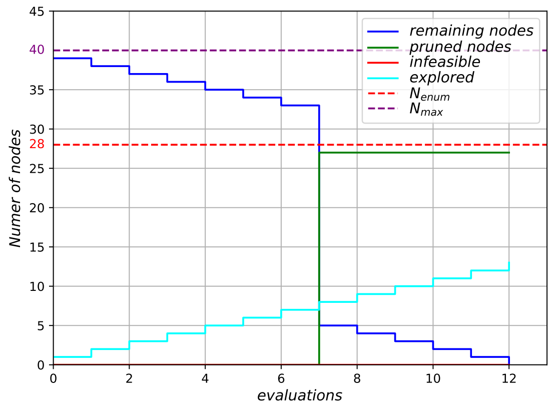
As an example, it is proposed to analyse the convergence history of the case with a bound on displacements $\bar{\boldsymbol{u}} = -22\ mm$, displayed in the Figure 2.11. The maximum number of nodes in the tree is equal to 2047 (N_{max}), and the number of nodes that correspond to optimizations with all categorical variables fixed (N_{enum}) is 1024. In Figure 2.11a, one can see that the optimum is found after 21 nodes evaluations, meaning that all the next 84 optimizations until the h-B&B is converged only serve to prove that there is no better solution. In Figure 2.11c, one can see that 10 nodes have been discarded at the level 4 in the tree. Nodes pruned at this low level value in the tree highly contribute to the efficiency of the algorithm, since it means that a subspace of $5 \times (2^{10-4-1} - 1) = 310$ solutions in Γ^{10} are automatically discarded.



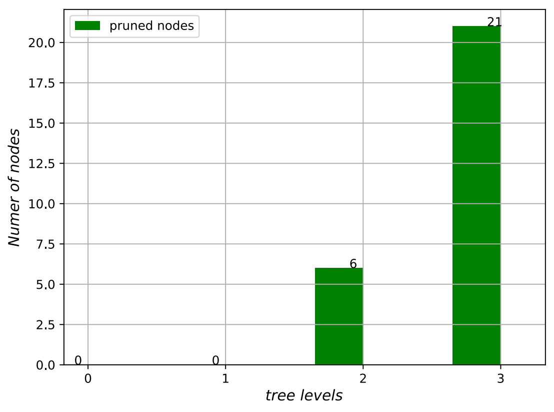
(a) Lower and upper bounds w.r.t. the number of problem evaluations



(b) Statistics on the lower bounds computed at each tree level.



(c) Evolution of the number of nodes in the solution tree.



(d) Number of pruned nodes in the tree that have not been branched.

Figure 2.9: The convergence history of the branch and bound Algorithm 2, on a 3-bar truss instance problem with 3 catalogs.

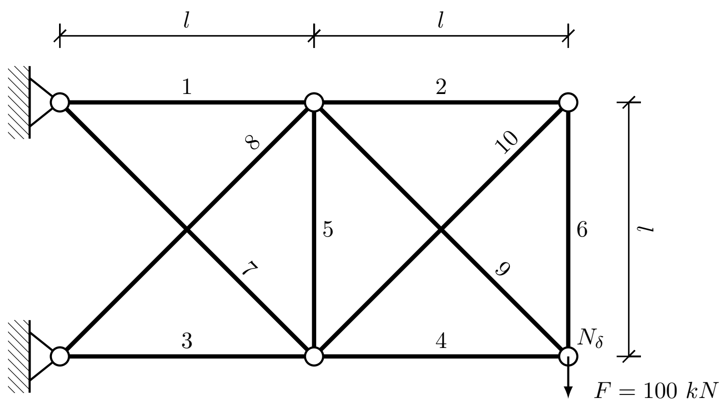


Figure 2.10: 10-bar truss, seen as a scalable 2D cantilever problem with 2 blocks.

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$1300\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$1300\ mm^2$

Table 2.4: Bounds on areas, and initial areas values of the 10-bar truss optimization case.

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$2000\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$2000\ mm^2$

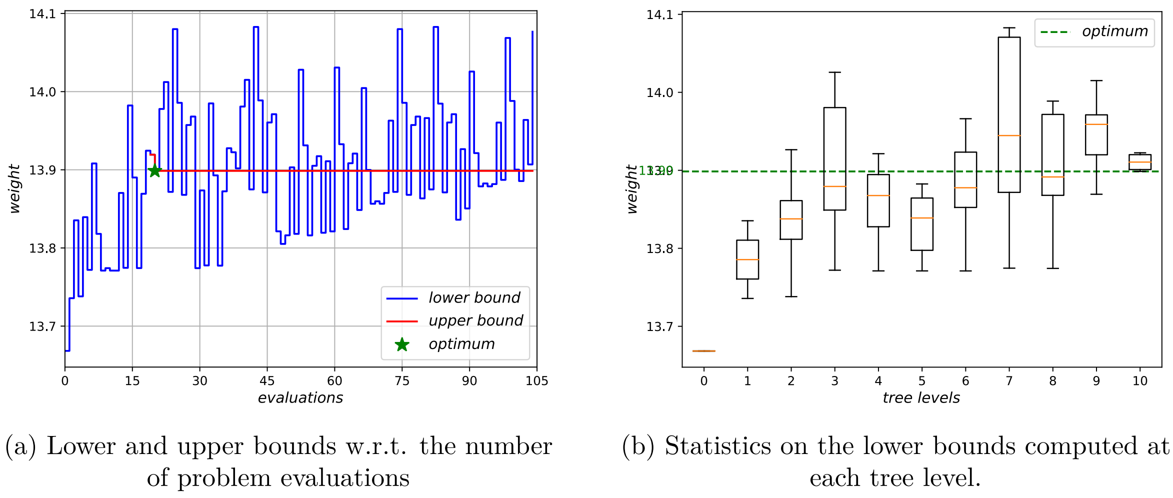
Table 2.5: Bounds on areas, and initial areas values of the 2D cantilever truss optimization case.

2.5.3 A scalable 2D cantilever problem

The objective of this test case is to describe the evolution of the computation cost with respect to the number of structural elements. This case can be seen as a generalization of the well-known 10-bar truss structure (Haftka and Gürdal 1992). It has been used in the literature to demonstrate the scalability of algorithms, for example in (Shahabsafa et al. 2018). The structure is made scalable by varying the number of blocks. Each block is composed of 4 nodes that are linked by 5 bars. An example of scalable 2D cantilever structure with 3 blocks is given in Figure 3.4. In Table 3.7 are presented the results obtained with structures composed of 1 to 10 blocks. In all cases, a downward load $F = 30\ kN$ is applied on the node N_δ. The bounds on areas, and initial areas are fixed as detailed in Table 2.5.

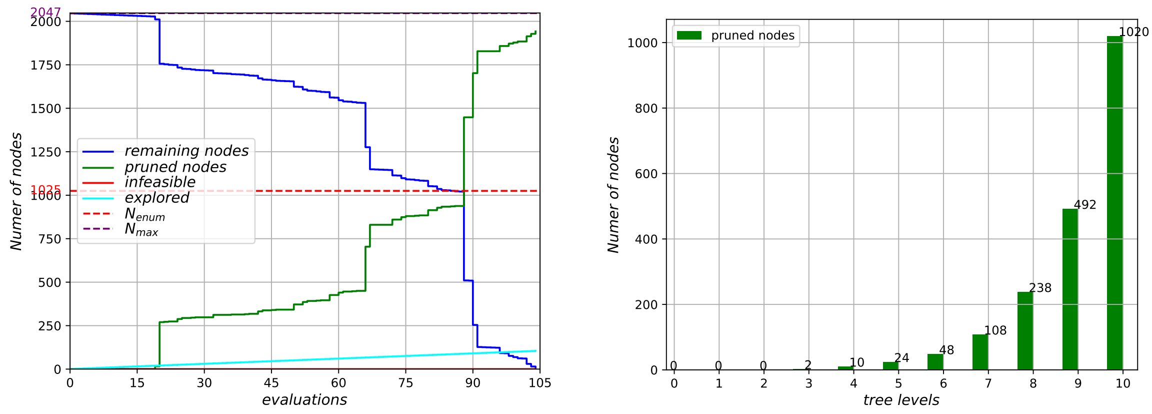
For each of the 10 cases, the results obtained by the h-B&B are compared to those obtained with reference solutions (Baseline) when available, and with Genetic). When optimizations last more than 24 hours, the solver Baseline or h-B&B is stopped and the current solution (if exists) is marked by (*). First, for cases with 5 to 15 bars where a reference solution is available, it can be observed the global solution is found by the h-B&B. For cases with 15 to 50 bars, the solutions obtained by h-B&B are compared to those obtained by Genetic solver only. In these cases, all the solutions returned by h-B&B are better than those obtained by Genetic solver. In particular, for the cases 35 to 50, even given the fact that the h-B&B was stopped before the end of the process, the returned optimal weights are better than those obtained by Genetic.

The trends in terms of computational cost with respect to the number of elements are graphically represented in Figure 2.13. In terms of computational cost, one can remark that the h-B&B requires less computational effort to converge to a solution for cases with 5 to 25 bars than Genetic. However, the computational cost of h-B&B for cases with more than 25 elements is dominates the computational cost of Genetic solver.



(a) Lower and upper bounds w.r.t. the number of problem evaluations

(b) Statistics on the lower bounds computed at each tree level.



(c) Evolution of the number of nodes in the solution tree.

(d) Number of pruned nodes in the tree that have not been branched.

Figure 2.11: The convergence history of the branch and bound Algorithm 2, on a 10-bar truss instance problem with 2 catalogs.

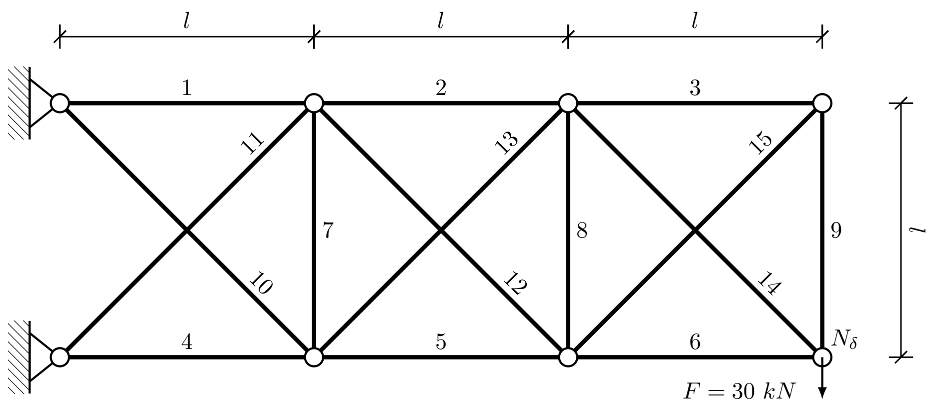


Figure 2.12: An example of 2D cantilever problem with 3 blocks.

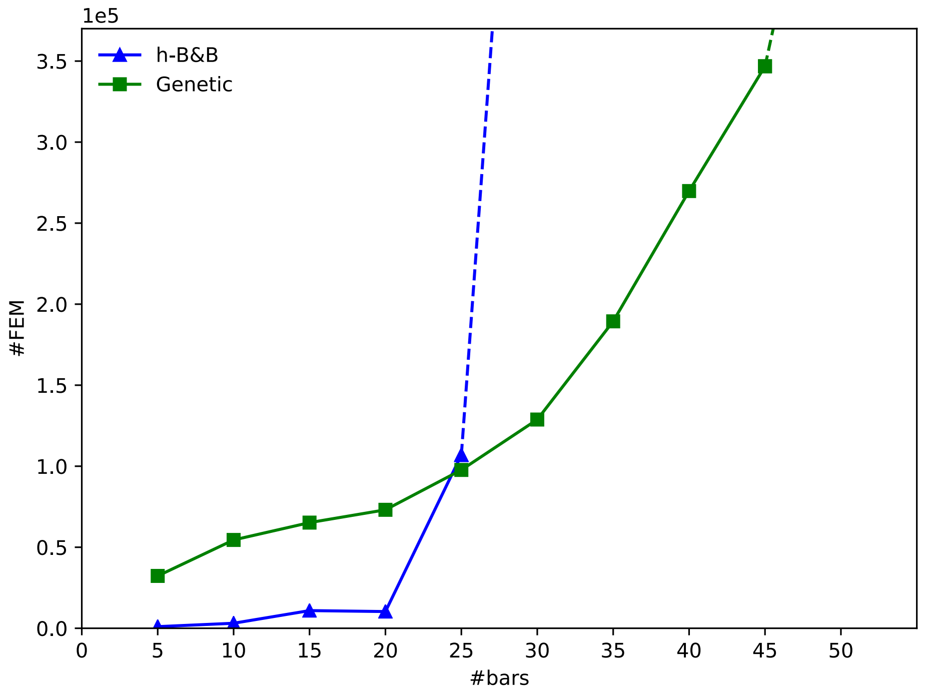


Figure 2.13: Scalability of the h-B&B w.r.t. the number of elements. The computational cost's scaling of h-B&B with respect to the number of bars is exponential like the Genetic solver. For cases lower than 25 elements, the computational cost of h-B&B is lower than Genetic solver. However, the computational cost's scaling of the h-B&B prevents from obtaining a solution for cases greater than 25 elements.

2.6 Conclusion

The proposed methodology allows to find the optimum of the mixed categorical-continuous structural optimization problem (P) in a finite number of steps. The methodology relies on a general Branch & Bound framework, where specific steps of branching and bounding have been proposed in order to tackle the non-relaxable and non-ordered nature of the categorical design variables. The branching step consists of fixing one of the remaining free categorical design variables to each available choice of material and stiffening principle. This multiway branching outputs as many nodes than there are available choices in the categorical design space. Then the output nodes need to be evaluated (the bound step), in the sense that a lower bound of their solution is computed. The categorical nature of the remaining free design variables prevents from using the common evaluation strategy, which is the continuous relaxation. This is why the formulation of a relaxation problem dedicated to the problem (P) has been proposed. The proposed relaxed problem is a full continuous optimization problem, where the objective and constraints functions underestimate the functions of the original optimization problem (P). This ensures that the evaluation of a node is a lower bound of the original problem. Furthermore, the relaxed problem can be efficiently solved by a gradient-

#bars	Baseline	h-B&B				Genetic			
	$w^*(kg)$	d_h	$w^*(kg)$	#iter	#FEM	d_h	$w^*(kg)$	#iter	#FEM
5	2.56	0	2.56	10	1004	0	2.57	32	32300
10	6.06	0	6.06	26	3097	1	6.14	54	54500
15	10.23	0	10.23	95	10907	2	10.27	65	65200
20	†	†	15.33	135	10315	†	15.59	73	73100
25	†	†	21.36	1199	610347	†	22.06	98	97700
30	†	†	28,30	4432	723388	†	28.84	129	128800
35	†	†	$36,17^{(*)}$	$5793^{(*)}$	$1096968^{(*)}$	†	37.00	189	189400
40	†	†	$44,97^{(*)}$	$5570^{(*)}$	$939726^{(*)}$	†	45.64	270	269800
45	†	†	$54,70^{(*)}$	$4181^{(*)}$	$818455^{(*)}$	†	55.98	347	346800
50	†	†	$65,35^{(*)}$	$4316^{(*)}$	$717627^{(*)}$	†	67.48	561	561200

Table 2.6: A comparison of the obtained solutions for 10 instances of the scalable 2D cantilever problem are compared, with a varying number of bars (from 5 to 50 bars). We note that when optimizations last more than 24 hours, the solver (Baseline, h-B&B) is stopped and the current solution (if exists) is marked by $(*)$. When reference solutions (Baseline) are not available, optimal weights are noted by †, as well as the distances d_h to these solutions.

based algorithm. The solutions of the proposed Branch & Bound methodology show that, in low-dimension cases where reference solutions exist, the optimal solution is found. The obtained solutions are also compared to those obtained by a state-of-the-art genetic solver. The comparison show that the optima found by the proposed methodology are equivalent of better than those obtained by the genetic algorithm. In terms of computational cost, the genetic algorithm requires more evaluations than the proposed algorithm in the presented low dimension test cases. However, for structures with more than 25 structural elements, the computational cost of the methodology is very high when compared to the genetic approach. It is shown that the exponential trend of the computational cost prevents from using the proposed methodology to solve large scale industrial optimizations. Finally, it has been remarked that the firsts optimal solutions obtained during the optimizations by the Branch & Bound approach are often either close to the optimal solution, or equal to the optimal solution. In the aforementioned test cases, most of the computational effort of the Branch & Bound serves at proving that one of the firsts solution is indeed the optimal one.

In this Chapter, the following items have been discussed:

- The proposed algorithm relies on the well-known Branch & Bound framework,
- In order to handle the categorical (non-ordered non-relaxable) design variables, a multiway branching has been proposed,
- In addition, a relaxed problem definition has been proposed, that allows to compute the optimal weight lower bound of the original problem,
- The relaxed problem is a pure continuous optimization problem, that can be efficiently solved by gradient-based optimization algorithms,
- Under the hypothesis that the objective and constraint functions of these relaxed problems are convex, the optima obtained by the proposed Branch & Bound algorithm the exact ones,
- The obtained numerical solutions are equal to the available reference solutions obtained by enumeration,
- The computational cost of the proposed methodology is lower than the reference algorithm (genetic algorithm) for low dimension problems,
- The computational cost of the algorithm prevents from solving large scale optimization problems.

Chapter 3

A bi-level framework using a first order-like approximation

Contents

Résumé

Dans le chapitre précédent a été présentée une méthode capable de résoudre le problème d'optimisation de structure à variables catégorielles et continues. Bien que cette méthode présente l'avantage de fournir des solutions optimales, le coût de calcul semble prohibitif pour des applications industrielles. Il est donc proposé dans ce chapitre un algorithme s'appuyant sur une approximation du problème d'origine de façon à limiter le temps de calcul.

Dans ce chapitre, la méthodologie proposée est basée sur une décomposition multi-niveau. Cette décomposition fait intervenir deux problèmes optimisation, l'un au niveau inférieur et l'autre au niveau supérieur. Dans le problème du niveau inférieur, les variables catégorielles sont fixées et l'optimisation est réalisée par rapport aux variables continues

uniquement. Ce problème peut ainsi être résolu efficacement en s'appuyant sur des algorithmes basés sur le gradient. Le problème d'optimisation du niveau supérieur consiste à minimiser le résultat de l'optimisation au niveau inférieur par rapport aux variables catégorielles. Ainsi, la complexité du problème d'origine est concentrée au niveau supérieur. Afin de casser cette complexité, il est proposé de minimiser une approximation au premier ordre du résultat de l'optimisation du niveau inférieur, et non directement le résultat du niveau inférieur.

Bien que cet algorithme ne soit couvert par aucune preuve d'optimalité, les expérimentations numériques montrent que les solutions obtenues sont identiques à celles de référence, lorsqu'elles sont disponibles, comme le montre la Table 3.7. De plus, on observe sur la Figure 3.5 que l'évolution du coût de calcul en fonction du nombre d'éléments structuraux est quasiment linéaire par rapport à l'algorithme h-B&B ainsi que l'algorithme Genetic. Un problème treillis 120 barres avec 4 catalogues est résolu grâce à cette formulation.

3.1 Motivations

The main motivation of the work presented in this Chapter relies on the conclusions on the previously described h-B&B algorithm. Indeed in Chapter 2 has been presented an algorithm derived from the Branch & Bound theory, that can solve the mixed categorical continuous structural optimization problem (P). The numerical tests revealed that the methodology was competitive for low scale problems when compared to the genetic algorithm. Since the algorithm relies on the Branch & Bound theory, it converges in a finite number of steps to the optimum of the problem (P). However, the scaling of the computational cost prevents from using such algorithm to solve large scale problems instances. This is why a new methodology with a reduced computational cost is now presented.

In this Chapter, the proposed methodology is based on a multilevel decomposition. The problem is indeed formulated using a bi-level decomposition involving master and slave problems. Like the Branch & Bound, it consists of breaking up of the original problem into smaller sub-problems. The continuous design variables are handled by the slave problem, where the categorical variables are driven by the master. The latter consists of solving a first order-like approximation of the slave problem with respect to the categorical design variables. The aim of multilevel optimization is twofold. First, it allows to overcome the complexity of the original problem by solving smaller subproblems. Second, depending on the decomposition, these subproblems can be solved in parallel. When implemented in the mixed optimization context of (P), this helps to drastically reduce the combinatorial explosion raised by the categorical variables. Furthermore, the master problem is driven by a first order like approximation in order to reduce the computational cost.

This Chapter is organized as follows. In Section 3.2, we describe the problem formulation, the physical model involved, and the links with the design variables. In Section 3.3, the bi-level decomposition and the approximation at the upper level are presented. Finally, the accuracy of the optimum and the scalability of the proposed approach are compared with state-of-the-art algorithms in Section 3.5. Concluding remarks will be given in the last Section.

3.2 Problem definition

The problem tackled in this Chapter is formulated as follows:

$$\begin{aligned} \underset{\boldsymbol{c}\in\Gamma^n,\boldsymbol{a}\in\mathbb{R}^n}{\text{minimize}} \quad & w(\boldsymbol{a},\boldsymbol{c}) \\ \text{subject to} \quad & \boldsymbol{s}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_{n,m} \\ & \boldsymbol{\delta}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_d \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \end{aligned} \tag{P}$$

where $\underline{\boldsymbol{a}} \in \mathbb{R}^n$ and $\bar{\boldsymbol{a}} \in \mathbb{R}^n$ are the lower and upper bounds on areas, respectively. The objective and constraints are continuously derivable with respect to the continuous design variables $\boldsymbol{a}$.

3.3 Methodology

3.3.1 Decomposition

For a given $\boldsymbol{c}$, let $\Omega(c)$ be the set of feasible constraints given by

$$\begin{aligned} \Omega(\boldsymbol{c}) := \{ \ \boldsymbol{a} \in \mathbb{R}^n; & \\ & \boldsymbol{s}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_{m,n} \ ; \\ & \boldsymbol{\delta}(\boldsymbol{a},\boldsymbol{c}) \leq \boldsymbol{0}_d \ ; \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \ \}. \end{aligned}$$

An efficient way to solve pure continuous optimization problems is by taking advantage of gradient based algorithms. In the problem introduced in Section 3.2, it can be seen that by fixing (temporarily) the categorical variables in (P), the problem becomes a continuous optimization problem, parameterized with $\boldsymbol{c}$. This means that given $\boldsymbol{c}$, the weight w can be minimized with respect to the continuous design variables, that are the areas $\boldsymbol{a}$ subject to $\Omega(\boldsymbol{c})$. This leads to the following slave problem, that reduces to a structural sizing optimization problem (sP):

$$\Psi(\boldsymbol{c}) \ := \ \min_{\boldsymbol{a}\in\Omega(\boldsymbol{c})} w(\boldsymbol{a},\boldsymbol{c}). \tag{sP}$$

The structure of the problem is such that this remaining optimization problem becomes more tractable. In fact, the decomposition leverages the use of the gradients (with respect to $\boldsymbol{a}$)

of the objective and constraints to solve (sP). This is the main motivation in handling the continuous variables separately from the categorical ones. In our approach, the categorical variables will be handled by a master problem (mP) of the form

$$\min_{\boldsymbol{c}\in\Gamma^n} \ \Psi(\boldsymbol{c}), \tag{mP}$$

where $\Psi(\boldsymbol{c})$ is being the result of the slave Problem (sP).

The slave Problem (sP) takes these complicating variables as parameters while optimizing with respect to continuous design variables. This follows the generalized Benders decomposition in (Geoffrion 1972), initially designed to handle linear optimization problems in (Benders 1962). For given choices of materials and cross-sectional types for all elements, the continuous optimization will be performed using a gradient based method. The obtained result from this optimization can be seen as a function $\Psi(\boldsymbol{c})$ which is parameterized by the categorical choices. Namely, $\Psi(\boldsymbol{c})$ corresponds to the optimal weight of the slave problem knowing the categorical variables $\boldsymbol{c}$. This function is then taken as the objective of the master optimization Problem (sP). Although the slave problem can be easy to handle using gradient-based algorithm, the difficult part remains in the master problem. In fact, the (mP) problem is a large-scale categorical optimization problem, that usual metaheuritic algorithms fail to solve efficiently. In this work, we propose to consider, at the master level, the minimization of an approximated model $\hat{\Psi}(\boldsymbol{c})$ instead of $\Psi(\boldsymbol{c})$, so that the combinatorial can be reduced drastically.

For that sake, we use the following iterative scheme: given an iteration (k), the master problem (mP) of the bi-level formulation reduces to the following :

$$\boldsymbol{c}^{(k+1)} := \operatorname*{argmin}_{\boldsymbol{c}\in\Gamma^n} \ \hat{\Psi}_k(\boldsymbol{c}) \tag{amP}$$

where $\hat{\Psi}_k$ is a given approximation function of Ψ at the iteration (k) that depends locally on the previous iteration. Such problem will be called the approximation master mixed Problem (mP) at iteration (k). At each iteration (k) of the algorithm, instead of Ψ, an approximation function $\hat{\Psi}_k$ is minimized with respect to the global variable $\boldsymbol{c}$.

Let call $\boldsymbol{a}^{(k)}$ the optimal areas obtained by solving Problem (sP) for given choices $\boldsymbol{c}^{(k)}$:

$$\boldsymbol{a}^{(k)} := \operatorname*{argmin}_{\boldsymbol{a}\in\Omega(\boldsymbol{c}^{(k)})} \ w(\boldsymbol{a},\boldsymbol{c}^{(k)}).$$

Let also $w^{(k)}$ be the optimal weight returned by the evaluation of the weight function taken at $\boldsymbol{a}^{(k)},\boldsymbol{c}^{(k)}$, i.e.,

$$w^{(k)} := w(\boldsymbol{a}^{(k)},\boldsymbol{c}^{(k)}).$$

The termination criterion is based on the stationarity of the optimal weights, i.e., $|w^{(k+1)} - w^{(k)}| \leq \epsilon$ for a given small $\epsilon > 0$. However, there will be no guarantee that a weight decreases lower than ϵ during the optimization process. A possible way to ensure such decrease would be proving that the proposed algorithm converges to a fixed point w^* independently of the starting point. Proving this issue in the general case would require imposing a decrease on the weight sequence $\{w^{(k)}\}$ over the iterations. In Section 3.3.4, an heuristic is proposed to handle that by further exploration of the design space.

The generic process of the proposed iterative scheme is given in Algorithm 3.

Algorithm 3 A generic Bi-level framework

```
1: initialize $\boldsymbol{c}^{(0)}$and set $k = 0$
2: while a termination criteria is not reached do
3:     Step 1 $\boldsymbol{c}^{(k+1)} \leftarrow \text{argmin} \;\; \hat{\Psi}_k(\boldsymbol{c})$ s.t $\boldsymbol{c} \in \Gamma^n$
4:     Step 2 $\boldsymbol{a}^{(k+1)} \leftarrow \text{argmin } w(\boldsymbol{a}, \boldsymbol{c}^{(k+1)})$ s.t. $\boldsymbol{a} \in \Omega(\boldsymbol{c}^{(k+1)})$
5:     Increment $k$
6: end while
7: return $\boldsymbol{a}^{(k+1)}$, $\boldsymbol{c}^{(k+1)}$, and $w^{(k+1)} \leftarrow w(\boldsymbol{a}^{(k+1)}, \boldsymbol{c}^{(k+1)})$.
```

3.3.2 On the approximation $\hat{\Psi}_k$

The problem (amP3) involves a number of categorical choices combinations that increases exponentially (p^n) with the number of catalogs and structural elements. In this subsection, we target to propose an approximation $\hat{\Psi}_k$ to the function Ψ so that one can reduce the resulting problem complexity. In fact, we propose to solve an approximation $\hat{\Psi}_k$ by using a first order approximation of Ψ. The expression of the approximation $\hat{\Psi}_k$ around the categorical variable $\boldsymbol{c}^{(k)}$ is given by:

$$\hat{\Psi}_k(\boldsymbol{c}) = \Psi(\boldsymbol{c}^{(k)}) + \sum_{i=1}^{n} \left.\frac{\Delta\Psi}{\Delta\boldsymbol{c}_i}\right|_{\boldsymbol{c}^{(k)}} (\boldsymbol{c}_i - \boldsymbol{c}_i^{(k)}), \tag{3.1}$$

where the scalar value $\Delta\boldsymbol{c}_i$ denotes the perturbation of the i^{th} component of $\boldsymbol{c}$ starting from $\boldsymbol{c}^{(k)}$, $\left.\frac{\Delta\Psi}{\Delta\boldsymbol{c}_i}\right|_{\boldsymbol{c}^{(k)}} \in \mathbb{R}$ model the rate of the Ψ function taken at $\boldsymbol{c}^{(k)}$ after a perturbation $\Delta\boldsymbol{c}_i$. The term $\left.\frac{\Delta\Psi}{\Delta\boldsymbol{c}_i}\right|_{\boldsymbol{c}^{(k)}}$ is computed as follows :

$$\left.\frac{\Delta\Psi}{\Delta\boldsymbol{c}_i}\right|_{\boldsymbol{c}^{(k)}} = \frac{\Psi(\boldsymbol{c}^{(k)} + \Delta\boldsymbol{c}_i\boldsymbol{e}_i) - \Psi(\boldsymbol{c}^{(k)})}{\Delta\boldsymbol{c}_i}, \tag{3.2}$$

with $\boldsymbol{e}_i$ a vector of size n where the i^{th} component is equal to 1 and 0 everywhere else.

We note that the term $\Delta\boldsymbol{c}_i$ has no physical meaning and, due to the categorical nature of the set Γ, there is no straightforward neighborhood definition. This prevents from choosing $\boldsymbol{c}_i$ so that $\boldsymbol{c}_i - \boldsymbol{c}^{(k)}$ is close to the perturbation $\Delta\boldsymbol{c}_i$ used to compute the rate $\left.\frac{\Delta\Psi}{\Delta\boldsymbol{c}_i}\right|_{\boldsymbol{c}^{(k)}}$. To overcome this issue, we propose to set the perturbation $\Delta\boldsymbol{c}_i$ equal to $\boldsymbol{c}_i - \boldsymbol{c}_i^{(k)}$. In this case, combining (3.1) and (3.2), one gets

$$\hat{\Psi}_k(\boldsymbol{c}) = \Psi(\boldsymbol{c}^{(k)}) + \sum_{i=1}^{n} \Big(\Psi(\boldsymbol{c}^{(k)} + \Delta \boldsymbol{c}_i \boldsymbol{e}_i) - \Psi(\boldsymbol{c}^{(k)}) \Big). \tag{3.3}$$

Equation (3.3) verifies, in the trivial case where the structure is composed of one element ($n = 1$), that the approximation $\hat{\Psi}_k(\boldsymbol{c})$ is equal to $\Psi(\boldsymbol{c})$, for every $\boldsymbol{c}^{(k)} + \Delta \boldsymbol{c}_i \boldsymbol{e}_i$ in Γ. Knowing $\Delta \boldsymbol{c}_i = \boldsymbol{c}_i - \boldsymbol{c}_i^{(k)}$, the term $\boldsymbol{c}^{(k)} + \Delta \boldsymbol{c}_i \boldsymbol{e}_i$ is equal to $\boldsymbol{c}^{(k)}$ except that the i^{th} component which is equal to $\boldsymbol{c}_i$.

Physically, the approximation (3.1) suggests that the effects of the couplings between the categorical variables on the optimized weight solutions of (sP), can be neglected. The block diagonally dominance property of the stress constraints jacobian with respect to the material properties and quadratic moments makes this approximation relevant. This property has been largely used in the literature in the frame of structural sizing problems, as in (Haftka and Gürdal 1992; Haftka and Watson 2005; Haftka et al. 2006; Bettebghor 2011; Grihon 2018; Bettebghor et al. 2018) to cite a few. However, a break down of the categorical variables as proposed in the Quasi Separable Decomposition scheme (Haftka and Watson 2005) in the frame of sizing variables, is not investigated here. The categorical variables are taken as design variables in the master problem (mP) only. The same remark applies for the areas, that are optimized in the slave problem only. In the proposed approach, the quasi-separable property is leveraged through the result of the areas optimizations in (sP). It is worth to note that the couplings between the elements are still partially impacting the optimizations thanks to the optimized areas, solutions of (sP). Elements are not optimized independently, since at each change of categorical variable, a sizing of the whole structure is performed.

3.3.3 On the minimization of $\hat{\Psi}_k$

In the previous section, it was proposed to substitute the expression of Ψ with the approximation $\hat{\Psi}_k$ when solving the master level problem (amP). Since $\Psi(\boldsymbol{c}^{(k)})$ is constant, the minimization of (3.3) according to `Step 1` of the Algorithm 3 is equivalent to :

$$\forall i \in \{1, ..., n\}, \ \min_{\boldsymbol{c}_i \in \Gamma} \ \Psi \left(\boldsymbol{c}^{(k)} + (\boldsymbol{c}_i - \boldsymbol{c}_i^{(k)}) \boldsymbol{e}_i \right).$$

Indeed, the approximated master problem (amP) can be written as a number of n independent categorical optimizations (sP). This reduces drastically the combinatorial explosion, i.e., instead of minimizing over Γ^n we get n minimizations but only over the space Γ. This is a crucial point of the proposed methodology. In fact, at each iteration (k) of the algorithm, the first order like assumption of the model $\hat{\Psi}_k$ makes the combinatorial of problem (mP) drop from p^n to $n \times (p-1)$ combinations of choices.

Namely, in `Step 1` of the Algorithm 3, the current categorical choices $\boldsymbol{c}^{(k+1)} := \left[\boldsymbol{c}_1^{(k+1)}, \ldots, \boldsymbol{c}_n^{(k+1)} \right]$

is given by solving the following approximated master problem:

$$\boldsymbol{c}_1^{(k+1)} := \underset{\boldsymbol{c}_1 \in \Gamma}{\operatorname{argmin}} \;\; \Psi\left(\left[\boldsymbol{c}_1,\ \boldsymbol{c}_2^{(k)},\ \ldots,\ \boldsymbol{c}_n^{(k)}\right]\right),$$

for $i \in \{2, \ldots, n-1\}$:

$$\boldsymbol{c}_i^{(k+1)} := \underset{\boldsymbol{c}_i \in \Gamma}{\operatorname{argmin}} \;\; \Psi\left(\left[\ldots,\ \boldsymbol{c}_{i-1}^{(k)},\ \boldsymbol{c}_i,\ \boldsymbol{c}_{i+1}^{(k)},\ \ldots\right]\right), \tag{amP2}$$

and

$$\boldsymbol{c}_n^{(k+1)} := \underset{\boldsymbol{c}_n \in \Gamma}{\operatorname{argmin}} \;\; \Psi\left(\left[\boldsymbol{c}_1^{(k)},\ \ldots,\ \boldsymbol{c}_{n-1}^{(k)},\ \boldsymbol{c}_n\right]\right).$$

Each of these n optimizations is solved by enumeration of all the remaining values that can take $\boldsymbol{c}_i$ over Γ. This means that at each iteration (k) of the algorithm, Ψ is evaluated $n \times (p-1)$ times to build a new solution $\boldsymbol{c}^{(k+1)}$. All these evaluations can be performed in parallel. The results of all the optimal weights computed during this enumeration process will be stored in a matrix $W^{(k)}$, i.e.,

$$(\forall i \in \{1, \ldots, n\}) \;\; \boldsymbol{c}_i^{(k+1)} = \underset{j \in \Gamma}{\operatorname{argmin}} \;\; W_{ij}^{(k)}. \tag{amP3}$$

A detailed view of the resulting Bi-level process is given in Fig. 3.1.

As described in Algorithm 3, each iteration (k) counts two main steps. `Step 1` consists to build a new solution $\boldsymbol{c}^{(k+1)}$. Physically, for every change of material and cross-section type of one element in the structure, a sizing optimization is performed. The new categorical choice of the current element is chosen such that the corresponding optimal weight (with respect to the continuous variables) is the lowest one. This process is repeated for each of the n elements, in parallel. In `Step 2`, the materials and cross-section types of every elements are updated with the new categorical variables. A sizing optimization is then performed, leading to a new optimal weight.

However, in practice the first order approximation (3.3) can be responsible of convergence issues during the optimization : an increase in the optimized weight could be observed from an iteration to the next one. For cases where this situation occurs, it is proposed to apply a strategy that iteratively builds a new solution based on information stored in $W^{(k)}$. The proposed strategy is detailed in the next section.

3.3.4 A strategy to ensure weight decrease

As only a first order like approximation of Ψ is used, the coupling between the structural elements through the categorical variables are neglected. This may be responsible of convergence issues as the obtained optimal weight at the current iteration $w^{(k+1)}$ may be greater than the previous optimal weight $w^{(k)}$. In the context of continuous optimization, adaptive strategies based on step-size control are typically used (e.g., line search, trust-region methods) (Ivanov and Zadiraka 1975). However, in the context of categorical optimization, the use of such strategy is not straightforward anymore due to the lack of a neighborhood definition.

In order to fix this convergence issue observed on some cases, the following iterative strategy is proposed. At a given iteration (k), the proposed process is triggered if the new optimal weight $w^{(k+1)}$ candidate is higher than the previous one $w^{(k)}$. It can be seen as an additional third step that would be introduced in Algorithm 3.

For each iteration (k) such that $w^{(k+1)}$ candidate is higher than the previous one $w^{(k)}$, we activate the following iterative strategy starting form $\boldsymbol{c}^{(k+1)}$. Let (t_k) denotes the outer iteration number of the proposed strategy related to the k^{th} iteration of Algorithm 3. At each iteration (t_k) of the strategy, a new categorical solution $\boldsymbol{c}^{(t_k)}$ will be built. The first step of the process consists to retrieve the element number i_{t_k} and corresponding choice j_{t_k} of the t^{th} best weight in $W^{(k)}$, i.e.,

$$i_{t_k}, j_{t_k} := \underset{(i,j)\in[\![1,n]\!]\times[\![1,p]\!]\setminus\mathcal{F}_{t_k}}{\operatorname{argmin}} W^{(k)},$$

where $\mathcal{F}_{t_k}$ denotes the set of indices i, j of all iterations anterior to t_k. The new candidate solution $\boldsymbol{c}^{(t_k)}$ is given by

$$\boldsymbol{c}^{(t_k)} := [\boldsymbol{c}_1^{(t_k-1)},\ \ldots,\ \boldsymbol{c}_{i_{t_k}-1}^{(t_k-1)},\ j_{t_k},\ \boldsymbol{c}_{i_{t_k}+1}^{(t_k-1)},\ \ldots,\ \boldsymbol{c}_n^{(t_k-1)}].$$

Once $\Psi(\boldsymbol{c}^{(t_k)})$ is evaluated, the corresponding optimal weight w is compared to $w^{(k)}$ on the following way: if the new optimal weight is found lower than $w^{(k)}$, then the candidate solution $\boldsymbol{c}^{(t_k)}$ becomes the new $\boldsymbol{c}^{(k+1)}$. The process stops and goes back to the main algorithm with the new optimal solution of iteration $(k+1)$. Algorithm 4 gives a detailed description of the proposed strategy.

Note that another possible decrease strategy would consist to repeat `Step 1` by changing only one structural element. In other words, from an iteration (k) to $(k+1)$, one would change only one categorical variable. In this case, equation (3.3) would lead to $\Psi(\boldsymbol{c}) = \hat{\Psi}(\boldsymbol{c})$, meaning that the weight decrease would be ensured by minimization of $\hat{\Psi}$. However, in practice the weight decrease has shown to be lower compared to the weight decrease obtained by Algorithm 4.

3.4 Implementation details and comparison solvers

Algorithm 3 has been implemented using the Generic Engine for MDO Scenarios (GEMS) (Gallard et al. 2018) in Python. The tool offers an efficient way to implement and test multi-level formulations, with built-in classes that facilitate optimization problems manipulations (Gallard et al. 2019). The continuous optimization problems (i.e., evaluations of Ψ) are solved with the Method of Moving Asymptotes (MMA) (Svanberg 2002). In what comes next, the resulting implementation of Algorithm 3 will be called Bi-level.

Algorithm 4 A proposed weight decrease strategy

```
1: function DECREASE_STRATEGY($w^{(k)}, \boldsymbol{c}^{(k+1)}, W^{(k)}$)
2:     $\boldsymbol{c}^{(0)} \leftarrow \boldsymbol{c}^{(k+1)}$ and set $t_k = 0$
3:     while $\mathcal{F}_{t_k} \neq [\![1, n]\!] \times [\![1, p]\!]$ do
4:         $i_{t_k}, j_{t_k} \leftarrow \underset{(i,j)\in[\![1,n]\!]\times[\![1,p]\!]\setminus\mathcal{F}_{t_k}}{\operatorname{argmin}} W^{(k)}$
5:         $\boldsymbol{c}^{(t_k)} \leftarrow [\boldsymbol{c}_1^{(t_k-1)}, \ldots, \boldsymbol{c}_{i_{t_k}-1}^{(t_k-1)}, j_{t_k}, \boldsymbol{c}_{i_{t_k}+1}^{(t_k-1)}, \ldots, \boldsymbol{c}_n^{(t_k-1)}]$
6:         $\boldsymbol{a}^{(t_k+1)} \leftarrow \operatorname{argmin} w(\boldsymbol{a}, \boldsymbol{c}^{(t_k+1)})$ s.t. $\boldsymbol{a} \in \Omega(\boldsymbol{c}^{(t_k+1)})$
7:         $w^{(t_k+1)} \leftarrow w(\boldsymbol{a}^{(t_k+1)}, \boldsymbol{c}^{(t_k+1)})$
8:         $\mathcal{F}_{t_k+1} \leftarrow \mathcal{F}_{t_k} \cup \{(i_{t_k}, j_{t_k})\}$
9:         if $w^{(t_k)} < w^{(k)}$ then
10:            break
11:        end if
12:        increment $t_k$
13:    end while
14:    $\boldsymbol{c}^{(k+1)} \leftarrow \boldsymbol{c}^{(t_k)}$
15:    $\boldsymbol{a}^{(k+1)} \leftarrow \boldsymbol{a}^{(t_k)}$
16:    $w^{(k+1)} \leftarrow w^{(t_k)}$
17: return $w^{(k+1)}, \boldsymbol{a}^{(k+1)}, \boldsymbol{c}^{(k+1)}$
18: end function
```

Three solvers will be compared to Bi-level. First, a baseline solver where we proceed an exhaustive enumeration of continuous optimizations w.r.t. $\boldsymbol{a}$ (Problem sP) taken at every available choice in Γ^n, the obtained solution by this solver will be denoted as Baseline. Second, a hybrid branch and bound (Algorithm 2, noted h-B&B), where one uses a relaxation procedure combined with a branch and bound algorithm. Similarly to Bi-level,h-B&B uses the MMA method to solve the slave problem. Under the assumption that these problems are convex with respect to the sizing variables $\boldsymbol{a}$, Baseline and h-B&B return the global optimum of the overall problem. The third solver used in the comparison is a Genetic algorithm (Deb and Goyal 1998) using the implementation given by Distributed Evolutionary Algorithms in Python (DEAP) (Fortin et al. 2012). The latter solver will be referred by Genetic in our comparison tests. Due to the stochastic nature of Genetic, the obtained results (for this solver) will be displayed as the average of ten runs.

In all what comes next, the computation effort of a given solver will be measured by counting the number of structural analyses (noted #FEM) including those required by the computation of the gradients (when needed). The obtained optimal weights (by each solver) will be noted w^*, the latter will allow us to evaluate the quality of the optima found by each solvers. We note also that in our setting, the Baseline solution can be seen as the best known categorical choices for the regarded problem. Thus, in this context, it is important to evaluate how far the categorical choices (obtained by the tested solvers) from the Baseline optimal choices are. This information will be given using the Hamming distance (noted d_h) where we will count the number of structural elements that has an optimal choice different to the Baseline categorical choices.

3.5 Numerical results

In the present section, our proposed methodology will be applied to three different test cases: (i) the well-known 10-bar truss structure (Haftka and Gürdal 1992), (ii) a 2D cantilever structure (Shahabsafa et al. 2018), and (iii) a 120-bar dome truss structure (Saka and Ulker 1992). In order to evaluate the scalability of the methodology with respect to large number of structural elements, the 2D cantilever structure is made scalable by varying the number of blocks. The third test case is included to evaluate the capability of handling more complex structures.

3.5.1 A step by step example: a 3-bar truss structure

To illustrate how the Bi-level method works, we will now describe in detail its application to a simple 3-bar truss structure. For this problem, each element can take a value among three possible choices that respectively point to materials AL2139, AL2024, TA6V and the same "I"-profile (see Fig. 1.3). The materials properties are listed in Table 3.1. For this simple

	AL2139	AL2024	TA6V
Density (kg/mm^3)	$2.8\ 10^{-6}$	$2.77\ 10^{-6}$	$4.43\ 10^{-6}$
Young modulus (MPa)	$7.1\ 10^4$	$7.4\ 10^4$	$11.0\ 10^4$
Poisson coefficient ($-$)	0.3	0.33	0.33
Tension allow. (MPa)	$1.5\ 10^2$	$1.6\ 10^2$	$11.0\ 10^2$
Compression allow. (MPa)	$2.0\ 10^2$	$2.1\ 10^2$	$8.6\ 10^2$

Table 3.1: Numerical details on materials attributes.

case, one has $n = 3$, $p = 3$, and $\Gamma = \{1, 2, 3\}$. For all elements, the lower and upper bounds on areas are respectively fixed to 100 mm^2 and 2000 mm^2. The initial areas are fixed to the upper bound values, as detailed in Table 3.2. A maximum downward displacement equal to 1 mm is allowed on the only free node of the structure.

During the application of the Bi-level method, the initialization of $\boldsymbol{c}$ is such that :

$$\boldsymbol{c}^{(0)} = [1, 2, 3]\,, \quad w^{(0)} = 13.82\ kg.$$

The first iteration (i.e. $k = 1$) of the Bi-level method can be described as follows: first, the

$\underline{\boldsymbol{a}}$	100 mm^2
$\bar{\boldsymbol{a}}$	2000 mm^2
$\boldsymbol{a}_{\text{ini}}$	2000 mm^2

Table 3.2: Bounds on areas, and initial areas values of the 3-bar truss optimization case.

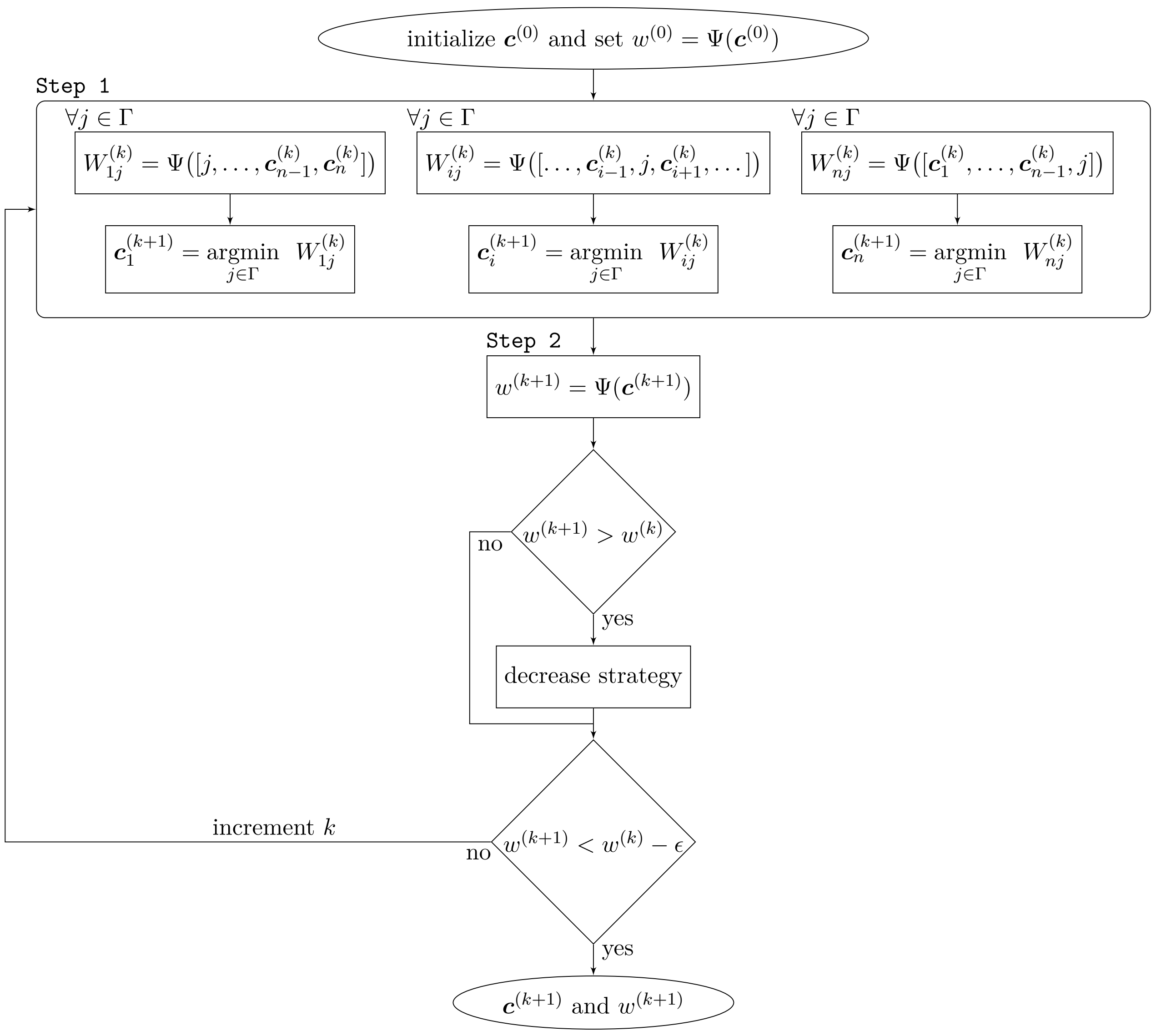

initialize $\boldsymbol{c}^{(0)}$ and set $w^{(0)} = \Psi(\boldsymbol{c}^{(0)})$
Step 1
$\forall j \in \Gamma$
$W_{1j}^{(k)} = \Psi([j, \ldots, \boldsymbol{c}_{n-1}^{(k)}, \boldsymbol{c}_n^{(k)}])$
$\boldsymbol{c}_1^{(k+1)} = \operatorname{argmin}_{j\in\Gamma} \; W_{1j}^{(k)}$
$\forall j \in \Gamma$
$W_{ij}^{(k)} = \Psi([\ldots, \boldsymbol{c}_{i-1}^{(k)}, j, \boldsymbol{c}_{i+1}^{(k)}, \ldots])$
$\boldsymbol{c}_i^{(k+1)} = \operatorname{argmin}_{j\in\Gamma} \; W_{ij}^{(k)}$
$\forall j \in \Gamma$
$W_{nj}^{(k)} = \Psi([\boldsymbol{c}_1^{(k)}, \ldots, \boldsymbol{c}_{n-1}^{(k)}, j])$
$\boldsymbol{c}_n^{(k+1)} = \operatorname{argmin}_{j\in\Gamma} \; W_{nj}^{(k)}$
Step 2
$w^{(k+1)} = \Psi(\boldsymbol{c}^{(k+1)})$
$w^{(k+1)} > w^{(k)}$
no
yes
decrease strategy
increment k
no
$w^{(k+1)} < w^{(k)} - \epsilon$
yes
$\boldsymbol{c}^{(k+1)}$ and $w^{(k+1)}$

Figure 3.1: Illustration of the proposed methodology.

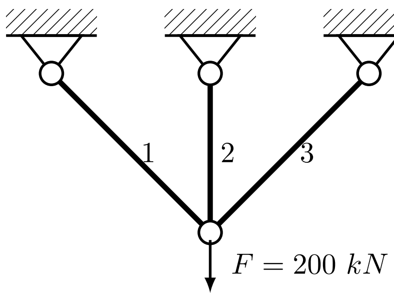


Figure 3.2: A 3-bar truss structure where a downward load $F = 200\ kN$ is applied on the free node.

	$j=1$	$j=2$	$j=3$
$i=1$	$\boldsymbol{c}^{(0)}$ $W_{11}^{(0)} = w^{(0)}$	$[②, 2, 3]$ $W_{12}^{(0)} = 13.62$	$[③, 2, 3]$ $W_{13}^{(0)} = 13.92$
$i=2$	$[1, ①, 3]$ $W_{21}^{(0)} = 14.85$	$\boldsymbol{c}^{(0)}$ $W_{22}^{(0)} = w^{(0)}$	$[1, ③, 3]$ $W_{23}^{(0)} = 8.83$
$i=3$	$[1, 2, ①]$ $W_{31}^{(0)} = 13.74$	$[1, 2, ②]$ $W_{32}^{(0)} = 13.53$	$\boldsymbol{c}^{(0)}$ $W_{33}^{(0)} = w^{(0)}$

Table 3.3: Enumeration of the $n \times (p-1) = 6$ perturbed categorical variables (circled components) at first iteration, with the corresponding optimal weight in (kg).

optimization problems (given by (amP2)) are solved by enumeration of the evaluation of Ψ for all values in Γ. Both categorical choices and corresponding optimal weights are given in Table 3.3.

As described in problems (amP2), the vector of categorical variables at the first iteration is composed of the values in Γ corresponding to the best weights given in Table 3.3, element per element. For example, $\boldsymbol{c}_1^{(1)} = \underset{j \in \Gamma}{\operatorname{argmin}} \ W_{1j}^{(0)} = 2$. The new optimal vector of categorical variables $\boldsymbol{c}$ at iteration $k = 1$ is thus $\boldsymbol{c}^{(1)} = [2, 3, 2]$, leading to an optimal weight $w^{(1)} = 8.63$ kg. After this first iteration, the optimal weight drops from 13.82 kg to 8.63 kg.

According to Algorithm 3, the same steps (that are not detailed) are executed in the next iteration $k = 2$. The same categorical vector solution is found, leading to the same optimal weight. Since a stationarity of the optimal weight is reached, the method stops. Using the Baseline method (by enumerating evaluations of (sP) over the space Γ^3), we find that the best solution is indeed equal to $\boldsymbol{c}^{(1)}$.

3.5.2 A 10-bar truss structure

This well-known low-dimensional 10-bar truss problem (Haftka and Gürdal 1992) allows to solve the mixed categorical-continuous optimization problem by enumeration or hybrid branch and bound (h-B&B) (Barjhoux et al. 2018b). As explained in subsection 3.4, these approaches provide global solutions, that are taken as reference solutions to evaluate quality solutions of the Bi-level algorithm.

The 10-bar truss problem is illustrated Fig. 3.3. A downward load $F = 100$ kN is applied vertically on node N_δ. A constraint on displacements is applied on the same node. Five cases with different bounds values $\bar{\boldsymbol{u}}$ on displacements are considered. For each of these cases, the displacements constraint is applied on node N_δ. Catalogs 1 and 2 points to materials AL2139 and TA6V, respectively. Materials properties are listed in Table 3.1. In this case, $n = 10$ and $p = 2$, $\Gamma = \{1, 2\}$. The upper and lower bounds on areas, and initial areas values, are fixed

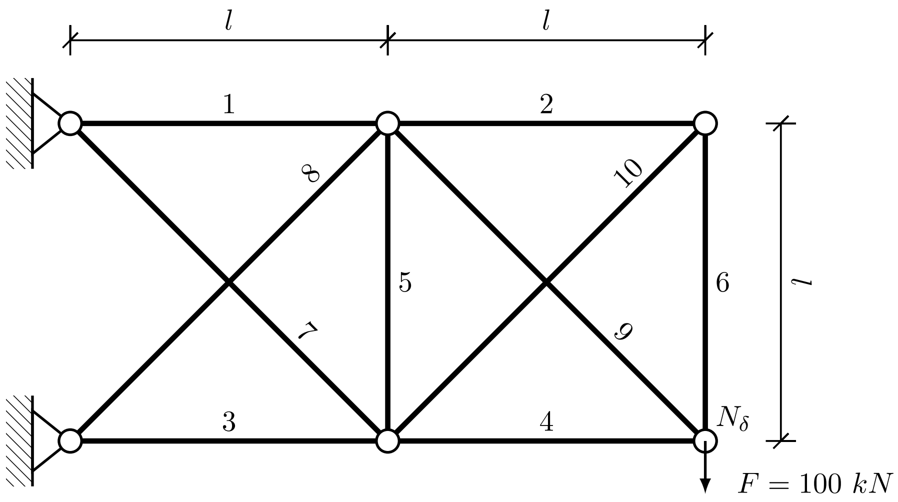


Figure 3.3: 10-bar truss, seen as a scalable 2D cantilever problem with 2 blocks.

Table 3.4: Results of 10-bar truss mixed optimization with 5 different values of constraint on displacements. Comparison between the Bi-level, the Baseline solutions obtained by enumeration of the 2^{10} continuous optimizations, h-B&B, and the Genetic algorithm. The catalog 1 corresponds to material AL2139 and catalog 2 to TA6V.

$\bar{\boldsymbol{u}}$ (mm)	Baseline		h-B&B		Genetic		Bi-level	
	c^*	$w^*(kg)$	d_h	$w^*(kg)$	d_h	$w^*(kg)$	d_h	$w^*(kg)$
-22	[2,2,1,1,1,2,2,1,2,1]	12.988	0	12.988	0	13.283	0	12.988
-20	[2,1,1,1,1,1,2,1,1,1]	13.996	0	13.996	0	14.423	0	13.996
-19	[2,1,1,1,1,1,2,1,1,1]	14.570	0	14.570	0	14.802	0	14.570
-18	[1,1,1,1,1,1,1,1,1,1]	15.175	0	15.175	2	15.642	0	15.174
-17	[1,1,1,1,1,1,1,1,1,1]	15.912	0	15.912	3	16.258	0	15.912

as detailed in Table 3.5.

The results of the proposed methodology (Bi-level) are thus compared to the global optima, as shown in Table 3.4. In all these cases, the optima obtained with Baseline (obtained by enumeration), h-B&B the Bi-level approaches are identical. This means that the Bi-level, in these cases, provides the global solution. On the other hand, the weights returned by the Genetic algorithm are greater than the optimal weight found by the Bi-level approach.

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$1300\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$1300\ mm^2$

Table 3.5: Bounds on areas, and initial areas values of the 10-bar truss optimization case.

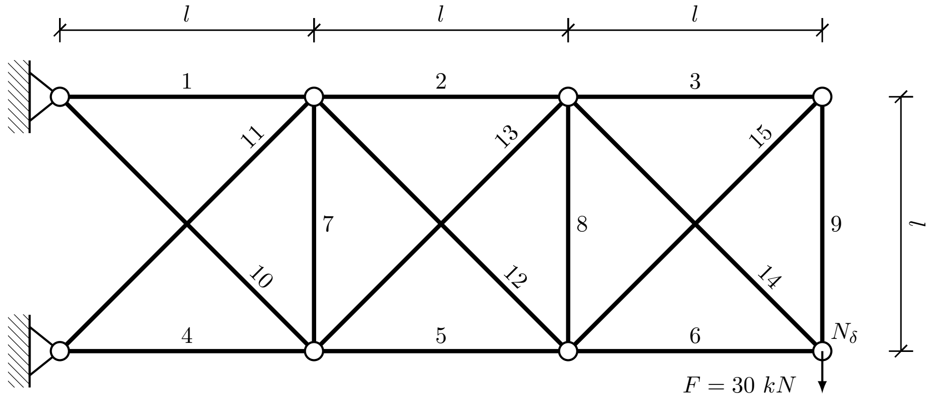


Figure 3.4: An example of 2D cantilever problem with 3 blocks.

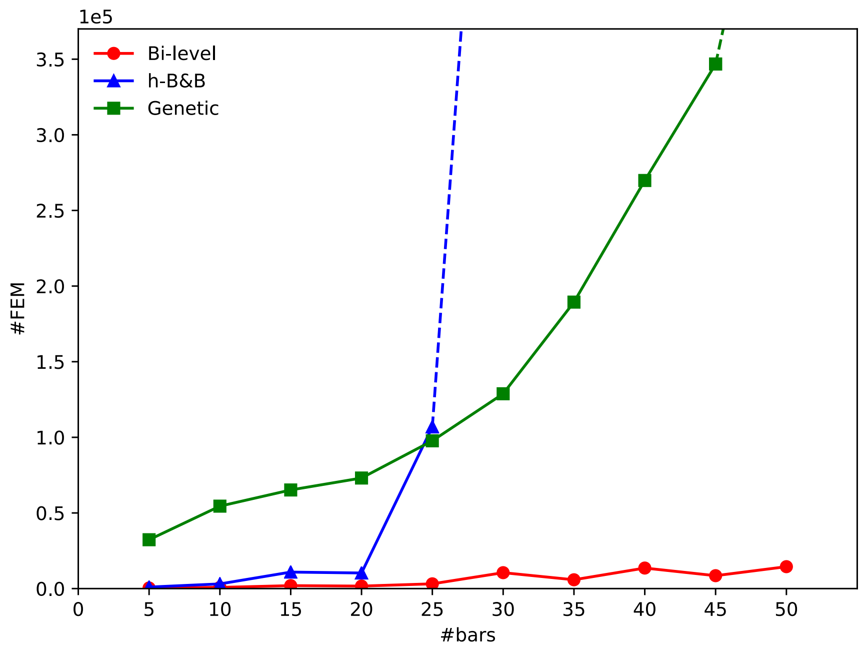


Figure 3.5: Scalability of the Bi-level w.r.t. the number of elements. The exponential computational cost's scaling of Bi-level with respect to the number of bars is quasi-linear, compared to the exponential computational cost of the h-B&B and Genetic solvers. The computation cost's scaling of the h-B&B prevents from obtaining a solution for cases greater than 25 elements.

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$2000\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$2000\ mm^2$

Table 3.6: Bounds on areas, and initial areas values of the 2D cantilever truss optimization case.

3.5.3 A scalable 2D cantilever problem

The objective of this test case is to describe the evolution of the computation cost with respect to the number of structural elements. This case can be seen as a generalization of the well-known 10-bar truss structure (Haftka and Gürdal 1992). It has been used in the literature to demonstrate the scalability of algorithms, for example in (Shahabsafa et al. 2018). The structure is made scalable by varying the number of blocks. Each block is composed of 4 nodes that are linked by 5 bars. An example of scalable 2D cantilever structure with 3 blocks is given in Figure 3.4. The bounds on areas, and initial areas are fixed as detailed in Table 3.6. In Table 3.7 are presented the results obtained with structures composed of 1 to 10 blocks. In all cases, a downward load $F = 30\ kN$ is applied on the node N_δ.

#bars	Baseline	h-B&B				Genetic				Bi-level			
	$w^*(kg)$	d_h	$w^*(kg)$	#iter	#FEM	d_h	$w^*(kg)$	#iter	#FEM	d_h	$w^*(kg)$	#iter	#FEM
5	2.56	0	2.56	10	1004	0	2.57	32	32300	0	2.56	2	400
10	6.06	0	6.06	26	3097	1	6.14	54	54500	0	6.06	2	792
15	10.23	0	10.23	95	10907	2	10.27	65	65200	0	10.23	4	1955
20	†	†	15.33	135	10315	†	15.59	73	73100	†	15.33	2	1659
25	†	†	21.36	1199	610347	†	22.06	98	97700	†	21.36	3	3142
30	†	†	28,30	4432	723388	†	28.84	129	128800	†	28.30	8	10522
35	†	†	$36,17^{(*)}$	$5793^{(*)}$	$1096968^{(*)}$	†	37.00	189	189400	†	36.19	3	5830
40	†	†	$44,97^{(*)}$	$5570^{(*)}$	$939726^{(*)}$	†	45.64	270	269800	†	44.97	7	13577
45	†	†	$54,70^{(*)}$	$4181^{(*)}$	$818455^{(*)}$	†	55.98	347	346800	†	54.71	4	8531
50	†	†	$65,35^{(*)}$	$4316^{(*)}$	$717627^{(*)}$	†	67.48	561	561200	†	65.34	6	14487

Table 3.7: A comparison of the obtained solutions for 10 instances of the scalable 2D cantilever problem are compared, with a varying number of bars (from 5 to 50 bars). We note that when optimizations last more than 24 hours, the solver (Baseline, h-B&B) is stopped and the current solution (if exists) is marked by $(*)$. When reference solutions (Baseline) are not available, optimal weights are noted by †, as well as the distances d_h to these solutions.

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$1300\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$1300\ mm^2$

Table 3.8: Bounds on areas, and initial areas values of the 120-bar truss optimization case.

For each of the 10 cases, the results obtained by the Bi-level are compared to those obtained with reference solutions (Baseline & h-B&B) when available. First, for cases with 5 to 30 elements where a reference solution is available, it can be observed the global solution is found by the Bi-level. For cases with more than 30 elements, the optima found by the Bi-level are slightly better than those obtained by the Genetic algorithm. The h-B&B solutions are noted with (*) since they are intermediate solutions : the solver has been stopped after 24 hours. The Bilevel solutions are very close (difference of 10^{-2} kg) to those obtained by the h-B&B. Furthermore, the number of analyses required by Bi-level is always lower than the number of those needed by the compared approaches. The trends in terms of computational cost with respect to the number of elements are graphically represented in Fig. (3.5). The cost of the Genetic algorithm dominates the cost of h-B&B and Bi-level. The scaling of the Bi-level approach is nearly linear when compared to the h-B&B and Genetic approach. The observed efficiency makes the proposed approach relevant for higher dimensional problems.

3.5.4 120 bars truss

In this example, the structure of a 120-bar dome truss (Saka and Ulker 1992) is considered and described Figure (3.6). In this case, $n = 120$ and $p = 4$, $\Gamma = \{1, 2, 3, 4\}$ There is no grouping of elements, meaning that the design space counts 120 categorical design variables and 120 continuous design variables. For each element, the categorical variable can take a value among 4 catalogs, that point to combinations of I and C-profiles with materials AL2139 and AL2024. Materials properties are listed in Table 3.1. The structure is subjected to an active constraint on displacements : a maximum downward displacement of 10 mm is allowed. A downward load of 60 kN is applied on node 1, while a downward load of 30 kN is applied on nodes 2 to 13 and 10 kN on nodes 14 to 37. The bounds on areas, and initial areas are fixed as detailed in Table 3.8.

The graphical solution obtained by the Bi-level algorithm is displayed on Fig. (3.7). To each categorical choice is associated a color on the structure. The continuous variables are qualitatively illustrated by the size of each truss element. The solution found by the algorithm shows that a number of two catalogs in Γ have been selected. A list of optimal choices and areas for each element is given in Table 3.9. On elements 1 to 12 and 25 to 48, the catalog 4 is the optimal one : AL2024 with C-profile. For the other elements, the optimal choice is the catalog 3 : AL2024 with I-profile. Thus for the entire truss, the stiffest and lightest material has been selected. Euler buckling constraints applied to elements 49 to 96 are active. The optimal choice for these elements is thus confirmed by the fact that the area moment of inertia of the I-profile is higher than the C-profile. The material with the highest Young

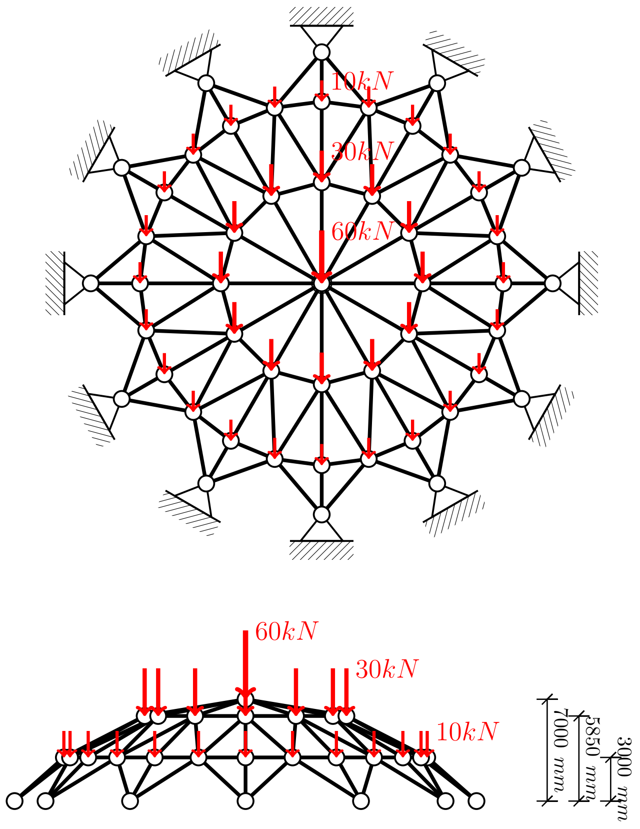


Figure 3.6: Top and side view a 120-bar truss structure. Downward loads with three different magnitudes are applied.

bars	$\boldsymbol{a}$ $[mm^2]$	$\boldsymbol{c}$ $[-]$
1-12	4553,6	4
13-24	1800,1	3
25-48	2313,0	4
49-72	745,6	3
73-96	589,9	3
97-108	1609,0	3
109-120	1109,2	3

Table 3.9: Solution of 120-bar truss mixed categorical-continuous optimization.

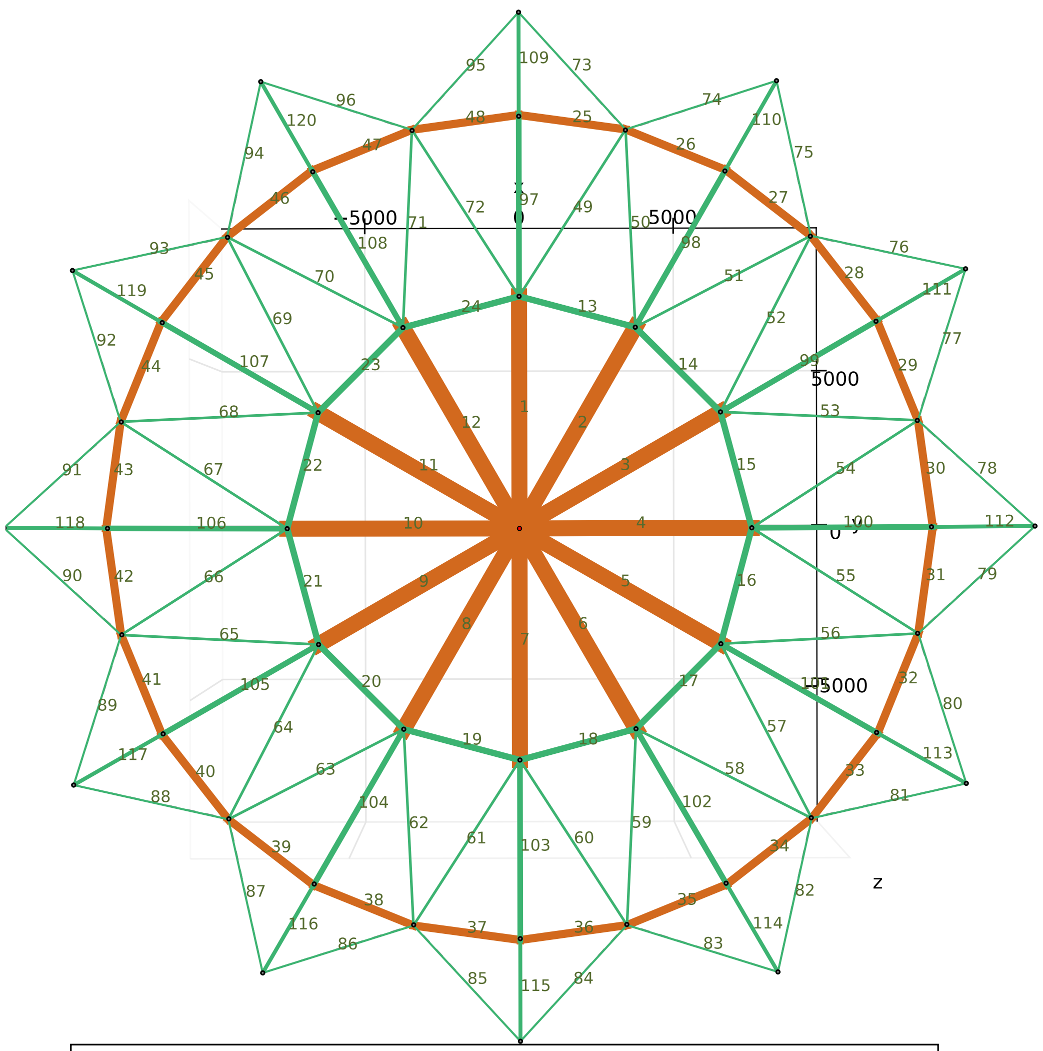
modulus has been selected for the entire truss. This is in accordance with the fact that the other active constraint is the global constraint on displacements. The Genetic algorithm has been applied on this case with settings adapted to the problem scale. However, it was not able to find a feasible region.

3.6 Conclusions

The proposed methodology is an efficient heuristic algorithm to handle large scale instances of the categorical-continuous structural weight minimization problem (P). It consisted of using a bi-level decomposition involving two problems: master and slave. The master problem was driven by a first order like-approximation, this made it possible to reduce drastically the combinatorial exploration cost raised by the categorical design space. Once the categorical decisions are driven by the master problem, the continuous variables are handled by the slave problem using a gradient-based approach. Using the proposed implementation, one was able to find the exact solutions on low-dimensional cases. Furthermore, on larger test cases, the scaling of our method revealed to be quasi-linear with respect to the number of structural elements. Particularly, the proposed approach allowed us to solve problems that are very hard to solve with standard algorithms. The proposed methodology offers thus an interesting compromise between the quality of the results, the computational effort and the ease of implementation. Compared to the Branch & Bound based algorithm presented in Chapter 2, the main drawback of the proposed algorithm is that there is no mathematical proof that the solution found is the exact solution.

In this Chapter, the following items have been discussed:

- The original mixed categorical-continuous optimization problem is reformulated as a bi-level optimization problem,
- Categorical design variables are handled at the master level,
- In the proposed algorithm, the master problem consists of minimizing a first order approximation of the slave problem result,
- The numerical tests show that the same solutions than enumeration and Branch & Bound algorithms are found for the 10-bar truss cases,
- The scaling of the computational cost with respect to the number of structural elements is quasi-linear,
- A 120-bar truss case with 4 catalogs is solved ($4^1 20$ possible configurations),
- The numerical tests show that the algorithm offers an interesting compromise between the output weight value and the computational cost.



🟧 I, AL2139	🟦 C, AL2139	🟩 I, AL2024	🟫 C, AL2024

Figure 3.7: Top view of the 120-bar truss mixed categorical-continuous optimization result.

Chapter 4

An outer approximation framework using post-optimal sensitivities

Contents

Résumé

Deux nouveaux algorithmes ont été présentés dans les chapitres précédents. L'un se base sur une approche arborescente, garantissant l'optimalité mais dont l'effort de calcul est prohibitif pour des applications industrielles. L'autre s'appuie sur une décomposition bi-niveau et une approximation au premier ordre de la masse optimale du niveau inférieur, permettant de limiter le coût de calcul.

Dans ce chapitre, il est proposé un troisième algorithme, tirant parti des conclusions des chapitres précédents, tout en essayant de concilier preuve d'optimalité avec coût de calcul limité. Tout d'abord, une reformulation continue (BP) du problème d'optimisation

d'origine est proposée:

$$\begin{aligned} \underset{\boldsymbol{B}\in\mathcal{C}^{n,p},\boldsymbol{a}\in\mathbb{R}^n}{\text{minimiser}} \quad & \widetilde{w}(\boldsymbol{a},\boldsymbol{B}) \\ \text{soumis à} \quad & \widetilde{\boldsymbol{s}}(\boldsymbol{a},\boldsymbol{B}) \leq \boldsymbol{0}_{n,m} \\ & \widetilde{\boldsymbol{\delta}}(\boldsymbol{a},\boldsymbol{B}) \leq \boldsymbol{0}_{d} \\ & \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \\ & \boldsymbol{B}_{ij}(\boldsymbol{B}_{ij}-1) = 0 \quad \forall(i,j) \in [\![1,n]\!] \times [\![1,p]\!]. \end{aligned} \tag{BP}$$

où $\boldsymbol{B}$ est une matrice de coefficients continus à valeurs comprises entre 0 et 1. Chaque ligne de la matrice décrivant un choix catégoriel, la somme des termes sur chacune de ces lignes doit être égale à un. Naturellement, le résultat de l'optimisation doit être tel que chacun de ces coefficient doit être égal à 0 ou 1, dans la mesure où des valeurs intermédiaires n'ont aucun sens physique. Ceci est garanti par l'algorithme proposé. Les fonctions objectifs $\widetilde{w}$, et contraintes $\widetilde{\boldsymbol{s}}$ et $\widetilde{\boldsymbol{\delta}}$, sont des versions continues des fonctions du problème d'origine. La même décomposition bi-niveau que celle exposée dans le chapitre précédent a été employée, à l'exception que le niveau supérieur traite à présent les variables discrètes $\boldsymbol{B}$ remplaçant $\boldsymbol{c}$. Le niveau supérieur est donc purement discret. On propose ici d'employer l'algorithme Outer Approximation (OA) (Duran and Grossmann 1986; Fletcher and Leyffer 1994) dans sa version purement discrète pour résoudre ce problème supérieur. Dans cet algorithme, les approximations au premier ordre ne sont pas exploitées en tant que direction de descente comme dans l'algorithme du chapitre précédent, mais utilisées pour réaliser des coupes dans l'espace de design. Sous réserve de convexité du problème, la solution retournée par l'algorithme est optimale. Egalement, au lieu de calculer les approximations au premier ordre par énumération, la définition continue du problème d'optimisation permet d'offrir un cadre suffisant couvert par la théorie des calculs de sensibilités par pénalisation détaillée dans (Fiacco 1976).

Les expérimentations numériques montrent que les solutions obtenues sont identiques à celles de référence, lorsqu'elles sont disponibles, comme le montre la Table 4.4. De plus, on observe sur la Figure 4.5 que l'évolution du coût de calcul en fonction du nombre d'éléments structuraux est quasiment linéaire par rapport à l'algorithme h-B&B ainsi que l'algorithme Genetic. On observe également sur la Figure 4.6 que le coût de calcul est quasiment insensible à l'évolution du nombre de catalogues disponibles par variables catégorielles. Ainsi les performances de l'algorithme rendent possible la résolution d'un problème treillis 120 barres avec 90 catalogues disponibles par variable catégorielle, soit une combinatoire totale de 90^{120} configurations possibles de treillis en termes de choix de matériaux et raidisseurs. Ce problème a été résolu en 58 optimisations continues, la solution étant optimale d'après les preuves théoriques fournies. Il est également important de noter que la solution optimale est obtenue dès la 5ème optimisation continue, et donc que les 53 optimisations suivantes ne servent qu'à apporter la preuve que cette solution est optimale.

4.1 Introduction

In this chapter, a new methodology is proposed to solve the mixed categorical-continuous optimization problem (P). This methodology relies on the conclusions of the previous two chapters. In Chapter 2, it has been shown that the hybrid Branch & Bound based approach (`h-B&B`) is costly in terms of the number of calls to the finite elements model. The exploration cost grows exponentially with the number of elements and categorical choices, preventing from using this algorithm to solve large scale problem instances. However, the method ensures, under the hypothesis that the continuously relaxed subproblems are convex, that the solution is the optimum. In Chapter 3, the computational cost of the proposed bi-level methodology (`Bi-level`) is quasi-linear with respect to the number of structural elements. The computational cost is also lower than `h-B&B` and allows to solve medium scale problems. The `Bi-level` offers an interesting compromise between the quality of the solution, the computational cost, provided the simplicity of the methodology. However the `Bi-level` method is defined as an heuristic. There is no proof that the solution is the optimum. Additionally, even with a quasi-linear computational cost and a parallelization of the sub-optimizations at each outer iteration, the method still requires large computing ressources.

The algorithms that solve mixed integer non-linear problems can be sorted into two classes. In the first one, there are the tree-based methods like the branch and bound (Dakin 1965; Gupta and Ravindran 1985) and branch and cut (Stubbs and Mehrotra 1999), which requires to solve a large number of non-linear sub-problems. The second class of algorithms called multi-tree algorithms decomposes the MINLP into an alternating sequence of NLP sub-problems and MILP relaxations. At each outer iteration of a multi-tree method, a MILP relaxation of the original problem is solved by a tree-based approach. The Outer Approximation algorithm (Duran and Grossmann 1986; Fletcher and Leyffer 1994; Bonami et al. 2008; Grossmann 2009), generalized Benders decomposition (Geoffrion 1972), as well as extended cutting-plane method (Westerlund and Pettersson 1995) belong to the multi-tree class. According to (Li and Sun 2006; Kronqvist et al. 2019; Nowak 2005), Outer Approximation is one of the most efficients algorithms that can solve MINLP problems. The Outer Approximation is introduced in (Duran and Grossmann 1986), and was dedicated to solve problems where the involved functions are separable in the continuous and discrete variables, and linear with respect to the (relaxed) integer variables. The method has then been generalized in (Yuan et al. 1988; Fletcher and Leyffer 1994) to a broader class of problems where the functions are not necessarily separable in the continuous and discrete variables, neither are linear with respect to the (relaxed) discrete variables. In what follows, the algorithm named OA will refer to the Generalized Outer Approximation. The OA framework has been investigated in various engineering fields. One can cite control systems (Pecci et al. 2017), chemistry (Gopinath et al. 2016), traffic management (Asadi Bagloee and Sarvi 2018), structural optimization (Allaire and Delgado 2015; Stolpe 2015; Stolpe and Sandal 2018).

In this Chapter, the new methodology based on the OA framework is proposed. It combines the efficiency of the branch and bound in case of linear problems, the relevance of the first order approximation of the slave problem as shown in Chapter 3, and the use of post-optimal sensitivities usually involved in the frame of continuous optimization. The mixed categorical-continuous problem is first re-formulated as a mixed integer-continuous problem with relaxable integer design variables. The functions involved in the problem are formulated as continuously derivable functions. The proposed algorithm can thus leverage the use of their gradients. The new problem is then formulated using a bi-level decomposition involving master and slave problems. The continuous design variables are handled by the slave problem, where the integer variables are driven by the master. The latter consists of solving a mixed integer linear problem, that is built iteratively by collecting linearizations of the slave problem solution with respect to the integer variables. This approach is different from the OA framework presented in (Fletcher and Leyffer 1994) in the sense that this is the solution of the slave problem that is linearized, not its composing objective and constraints functions. An efficient way to compute the gradients of the linearizations is proposed. The resulting methodology falls within the theoretical framework of Outer Approximation. This Chapter is organized as follows. In Section 4.2, the formulation of the mixed integer-continuous optimization problem is presented. In Section 4.3, the bi-level decomposition and the new methodology are presented. Finally, the scalability of the approach is compared with state-of-the-art algorithms in Section 4.4. A large scale problem is also presented.

4.2 Problem statement

As formulated in (P), the problem involves categorical non-ordered and non-relaxable design variables. This formulation prevents from using algorithms that exploit the gradient of the functions with respect to all the design variables. This is why a new formulation of the problem (P), involving continuous functions as objective and constraints, is presented in this Section.

4.2.1 A continuous design space definition

In this problem formulation we use a continuous dummy coding of the categorical variable. Let be $\mathcal{C}^{n,p}$, the set of matrices $\boldsymbol{B}$ of real coefficients, such as :

$$\mathcal{C}^{n,p} = \left\{ X = (x_{ij})_{\substack{1\le i\le n\\ 1\le j\le p}} \Big/ \ x_{ij} \in [0,1], \text{ and } \sum_{j=1}^{p} x_{ij} = 1, \quad \forall i \in [\![1,n]\!] \right\}.$$

Each line of the matrix describes the catalog choices composition of a given element. The sum of the coefficients, for each line, has to be equal to 1. When $\boldsymbol{B}$ takes integer values, it corresponds to given choices of materials and shapes. However, there is no underlying physical meaning for intermediate (real) values of $\boldsymbol{B}$. A numerical example of equivalent values for $\boldsymbol{c}$ and $\boldsymbol{B}$ is described in Figure 4.1.

The definitions of Γ and $\mathcal{C}^{n,p}$ are such that, if the values $\boldsymbol{c} \in \Gamma$ and $\boldsymbol{B} \in \mathcal{C}^{n,p} \cap \{0,1\}$

$$\boldsymbol{c} = \begin{pmatrix} 4 \\ 2 \\ 1 \\ 3 \\ 2 \end{pmatrix}$$

(a) Categorical variables : $c \in \Gamma^5$

$$\boldsymbol{B} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

(b) Dummy coding of categorical variable $\boldsymbol{c}$, with $\boldsymbol{B} \in \mathcal{C}^{5,4}$

Figure 4.1: An example of categorical design variable equivalence between direct coding (a) and dummy coding (b), in the case of a 5 elements structure : $n = 4$. In this case Γ is a set of 4 catalogs : $p = 4$.

describe the same choices of materials and shapes, $\boldsymbol{B}$ and $\boldsymbol{c}$ verify the following relation :

$$\boldsymbol{c} = \boldsymbol{B}\boldsymbol{\gamma} \tag{4.1}$$

with $\boldsymbol{\gamma} \in \mathbb{Z}^p$ a vector of all the catalogs in Γ, such that

$$\boldsymbol{\gamma} = [1, \ldots, p]^\top . \tag{4.2}$$

4.2.2 Objective and constraints definition

This continuous representation $\boldsymbol{B}$ of the categorical design variable will weight the the Young modulus, densities and the functions of the optimization problem. It will thus allow continuous definitions of the objective and the constraints, as presented in this Section.

The Young modulus and densities associated to the i^{th} element, noted $\widetilde{\boldsymbol{E}}_i$, are computed as follows, respectively and $\forall i \in [\![1, n]\!]$:

$$\widetilde{\boldsymbol{E}}_i(\boldsymbol{B}) = \sum_{k=1}^{p} \boldsymbol{B}_{ik} E(\boldsymbol{\gamma}_k)$$
$$\widetilde{\boldsymbol{\rho}}_i(\boldsymbol{B}) = \sum_{k=1}^{p} \boldsymbol{B}_{ik} \rho(\boldsymbol{\gamma}_k)$$

with $\boldsymbol{\gamma} \in \mathbb{Z}^p$ a vector of all the catalogs in Γ (equation 4.2).

In the following reformulations, the functions are continuous with respect to $(\boldsymbol{a}, \boldsymbol{B}) \in \mathbb{R}^n \times \mathcal{C}^{n,p}$. The stress constraints $\widetilde{\boldsymbol{s}}_{ij}$ are defined as a linear combination of $\boldsymbol{s}_{ij}$ where the limit stresses are taken at integer values $\boldsymbol{\gamma} \in \mathbb{Z}^p$, a vector of all the catalogs in Γ (equation 4.2).

Each stress constraint (1.3) is thus rewritten as, $\forall (i,j) \in [\![1,n]\!] \times [\![1,4]\!]$:

$$\begin{aligned} \widetilde{\boldsymbol{s}}_{ij} \colon \mathbb{R}^n \times \mathcal{C}^{n,p} &\to \mathbb{R} \\ \widetilde{\boldsymbol{s}}_{ij}(\boldsymbol{a},\boldsymbol{B}) &\mapsto \sum_{k=1}^{p} \boldsymbol{B}_{ik} \boldsymbol{s}_{ij}(\boldsymbol{a}_i, \boldsymbol{\gamma}_k, \boldsymbol{\Phi}_i(\boldsymbol{a},\boldsymbol{B})) \end{aligned} \tag{4.3}$$

where $\boldsymbol{\Phi}$ denotes the internal forces.

The constraints on displacements remain unchanged, at the exception of the categorical variable $\boldsymbol{c} \in \Gamma$ replaced by $\boldsymbol{B} \in \widetilde{\mathcal{C}}^{n,p}$:

$$\begin{aligned} \widetilde{\boldsymbol{\delta}} \colon \mathbb{R}^n \times \mathcal{C}^{n,p} &\to \mathbb{R}^d \\ (\boldsymbol{a},\boldsymbol{B}) &\mapsto \boldsymbol{P}\boldsymbol{u}(\boldsymbol{a},\boldsymbol{B}) - \bar{\boldsymbol{u}}. \end{aligned}$$

Finally, the weight function can be written as follows :

$$\begin{aligned} \widetilde{w} \colon \mathbb{R}^n \times \mathcal{C}^{n,p} &\to \mathbb{R} \\ \widetilde{w}(\boldsymbol{a},\boldsymbol{B}) &\mapsto \sum_{i=1}^{n} \widetilde{\boldsymbol{\rho}}_i(\boldsymbol{B}) \boldsymbol{a}_i \boldsymbol{L}_i \end{aligned}$$

The previous definitions of the continuous functions involved in (P) are such that for any $\boldsymbol{c} \in \Gamma^n$ and $\boldsymbol{B} \in \mathcal{C}^{n,p} \cap \{0,1\}$ that verify the equation (4.1), we have :

$$\begin{aligned} \widetilde{\boldsymbol{s}}(\boldsymbol{a},\boldsymbol{B}) &= \boldsymbol{s}(\boldsymbol{a},\boldsymbol{c}) \\ \widetilde{\boldsymbol{\delta}}(\boldsymbol{a},\boldsymbol{B}) &= \boldsymbol{\delta}(\boldsymbol{a},\boldsymbol{c}) \\ \widetilde{w}(\boldsymbol{a},\boldsymbol{B}) &= w(\boldsymbol{a},\boldsymbol{c}). \end{aligned}$$

In other terms, if $\boldsymbol{c} \in \Gamma$ and $\boldsymbol{B} \in \mathcal{C}^{n,p}$ describe the same categorical choices (in this case $\boldsymbol{B}$ takes only integer values), then the output values of stress constraints, displacements constraints and weight function are strictly equivalent whether their expression depends on $(\boldsymbol{a},\boldsymbol{c})$ or $(\boldsymbol{a},\boldsymbol{B})$. Another important remark can be formulated regarding the computational cost induced by the new formulation. Indeed, the computational cost induced by both formulations is equivalent in terms of the number stiffness matrices inversions. This remark applies to the functions as well as their gradients.

However, it is worth to note that an evaluation of $\widetilde{\boldsymbol{s}}$, $\widetilde{\boldsymbol{\delta}}$, or $\widetilde{w}$ at non integer values of $\boldsymbol{B} \in \mathcal{C}^{n,p}$ has no physical meaning. The important advantage of the definitions of functions $\widetilde{\boldsymbol{s}}$, $\widetilde{\boldsymbol{\delta}}$, and $\widetilde{w}$ lies in the fact that they are continuous and derivable with respect to each of their design variables $\boldsymbol{a}$ and $\boldsymbol{B} \in \mathcal{C}^{n,p}$. The functions that were depending on the categorical design variables $\boldsymbol{c}$ have now new definitions, making them continuously derivable, and strictly equivalent to their original definition when evaluated at integer values.

4.2.2.1 Optimization problem statement

Since the objective and constraints have been re-formulated in a continuous way by relying on the continuous design variables $\boldsymbol{B}$, the new mixed categorical continuous optimization Problem (BP) can be defined as follows :

$$
\begin{aligned}
\min_{\boldsymbol{B}\in\mathcal{C}^{n,p},\boldsymbol{a}\in\mathbb{R}^n} \quad & \widetilde{w}(\boldsymbol{a},\boldsymbol{B}) \\
\text{subject to} \quad & \widetilde{\boldsymbol{s}}(\boldsymbol{a},\boldsymbol{B}) \leq \mathbf{0}_{n,m} \\
& \widetilde{\boldsymbol{\delta}}(\boldsymbol{a},\boldsymbol{B}) \leq \mathbf{0}_d \\
& \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \\
& \boldsymbol{B}_{ij}(\boldsymbol{B}_{ij}-1) = 0 \;\; \forall (i,j) \in [\![1,n]\!] \times [\![1,p]\!].
\end{aligned}
\tag{BP}
$$

where, for sake of clarity of the notations, $\boldsymbol{B}$ and $\widetilde{\boldsymbol{s}}$ are vectors instead of matrices. They are obtained by applying the same transformation, written here when applied to a matrix $\boldsymbol{A} \in \mathcal{M}_{n,p}(\mathbb{R})$:

$$
\mathrm{vec}(\boldsymbol{A}) = [\boldsymbol{A}_{11}, \dots, \boldsymbol{A}_{1p}, \boldsymbol{A}_{21}, \dots, \boldsymbol{A}_{2p}, \dots, \boldsymbol{A}_{n1}, \dots, \boldsymbol{A}_{np}]^\top.
$$

The problems (BP) and (P) are strictly equivalent. However, unlike in the problem (P), all the functions can be evaluated at intermediate values of $\boldsymbol{B}$. The optimization algorithm employed to solve the problem (BP) can also benefit from their derivatives. Since integrity constraints force the optimal solution $\boldsymbol{B}^*$ to be a vector of 0-1 integers while $\boldsymbol{a}^*$ remains continuous, the algorithm will have to deal with mixed integer-continuous design variables. This makes the problem (BP) belong to the class of mixed integer-continuous relaxable problems. It is worth to note that there is still no ordering of the integer values, and no physical definition of a neighborhood in terms of integer variables. The main difference with (BP) lies in the fact that the functions are defined at intermediate binary values, and are derivable.

4.3 Methodology

4.3.1 Decomposition

For a given $\boldsymbol{B}$, let $\widetilde{\Omega}(\boldsymbol{B})$ be the set of feasible constraints of the problem (BP) given by

$$
\begin{aligned}
\widetilde{\Omega}(\boldsymbol{B}) := \{ \; & \boldsymbol{a} \in \mathbb{R}^n; \\
& \widetilde{\boldsymbol{s}}(\boldsymbol{a},\boldsymbol{B}) \leq \mathbf{0}_{m\times n} \; ; \\
& \widetilde{\boldsymbol{\delta}}(\boldsymbol{a},\boldsymbol{B}) \leq \mathbf{0}_d \; ; \\
& \underline{\boldsymbol{a}} \leq \boldsymbol{a} \leq \bar{\boldsymbol{a}} \}.
\end{aligned}
\tag{4.4}
$$

An efficient way to solve pure continuous optimization problems is by taking advantage of gradient based algorithms. In the problem introduced in Section 4.2, it can be seen that by fixing (temporarily) design variables $\boldsymbol{B}$ in (BP) at integer values, the optimization problem becomes a continuous one parameterized with $\boldsymbol{B}$, and where integrity constraints can be removed. This means that at a given $\boldsymbol{B}$, the weight $\widetilde{w}$ can be minimized with respect to the remaining continuous design variables that are the areas $\boldsymbol{a} \in \widetilde{\Omega}(\boldsymbol{B})$. This leads to the following slave binary problem (sBP), that reduces to a structural sizing optimization problem:

$$\widetilde{\Psi}(\boldsymbol{B}) \ := \min_{\boldsymbol{a}\in\widetilde{\Omega}(\boldsymbol{B})} \ \widetilde{w}(\boldsymbol{a}, \boldsymbol{B}). \tag{sBP}$$

The structure of the problem is such that this remaining optimization problem becomes more tractable. In fact, the decomposition leverages the use of the gradients (with respect to $\boldsymbol{a}$) of the objective and constraints to solve the problem (sBP). This is the main motivation in handling the continuous variables separately from the integer ones. In this approach, the integer (binary) variables will be handled by a master problem (mBP) of the form

$$\begin{aligned} \min_{\boldsymbol{B}\in\mathcal{C}^{n,p}} \quad & \widetilde{\Psi}(\boldsymbol{B}) \\ & \boldsymbol{B} \circ (\boldsymbol{B} - \mathbf{1}) = \mathbf{0}_{np} \end{aligned} \tag{mBP}$$

with $\circ$ the entry-wise multiplication, or Hadamard product, of two vectors. $\widetilde{\Psi}(\boldsymbol{B})$ is the result of the slave Problem (sBP). The slave problem (sBP) takes these complicating variables $\boldsymbol{B}$ as parameters while optimizing with respect to continuous design variables. This means that during the slave optimization, the choices of materials and cross-section types for all elements remain fixed. This slave problem will be solved using a gradient based method. The obtained solution can be seen as a function $\widetilde{\Psi}(\boldsymbol{B})$ which is parameterized by the categorical choices through the continuous coding $\boldsymbol{B}$. Namely, $\widetilde{\Psi}(\boldsymbol{B})$ corresponds to the optimal weight of the slave problem knowing the variables $\boldsymbol{B}$. This function is then taken as the objective of the master optimization problem (sBP). Although the slave problem can be easy to handle using gradient-based algorithms, the difficult part remains in the master problem. In fact, the problem (mBP) is still a large-scale pure integer non-linear optimization problem, that usual combinatorial optimization solvers fail to solve efficiently. However, unlike the problem presented in Chapter 3, the integer variable $\boldsymbol{B}$ is relaxable and the functions are defined at intermediate non 0-1 values of $\boldsymbol{B}$. Moreover, all the functions of the optimization problem are continuously differentiable. This is a basic requirement to compute the sensitivity of the slave problem solution parametererized in $\boldsymbol{B}$.

4.3.2 On the minimization of $\widetilde{\Psi}$

It is proposed to solve the master problem (mBP) by means of outer approximation cuts, by taking advantage of the gradient of $\widetilde{\Psi}$. The resulting generic framework is an iterative process, where the master problem is replaced by an approximated problem. The generic framework is presented in Section 4.3.2.1. An efficient procedure to compute the sensitivity of the slave problem (gradient of $\widetilde{\Psi}$) is presented in Section 4.3.2.2. Finally, the resulting outer approximation framework using post-optimal sensitivities is described in Section 4.3.2.3.

4.3.2.1 Framework

In this work, it is proposed to consider, at the master level, the minimization of an approximated problem $\mathcal{P}$ instead of (mBP), so that the complexity of the master problem is significantly reduced. For that, the following iterative scheme is implemented. Given an iteration (k), the master problem (mBP) of the bi-level formulation is reduced to a problem $\mathcal{P}^{(k)}$ easier to solve. The definition of $\mathcal{P}^{(k)}$ is recursive and depends on its own formulation at the previous iteration. It also involves the value of the function $\widetilde{\Psi}$ and its derivative taken at $\boldsymbol{B}^{(k)}$. At each iteration (k) of the algorithm, instead of (mBP), the problem $\mathcal{P}^{(k)}$ is solved and outputs a new candidate $\boldsymbol{B}^{(k+1)}$.

The slave optimization problem, or primal, is defined by fixing the binary variables $\boldsymbol{B}^{(k)}$ in the problem (mBP). Since the problem (mBP) is a full integer optimization problem, solving (mBP) at a fixed $\boldsymbol{B}^{(k)}$ reduces to an evaluation of the involved objective $\widetilde{\Psi}$ and constraints functions. Among these involved functions, the equality constraints are constraints that force the optimal solution to be an integer one. Since the primal optimization problem is defined by fixing the variables to binary values, the evaluation of this function is not needed. The optimal solution integrity will indeed be guaranteed by the algorithm. In other terms, the primal problem reduces to an evaluation of the objective $\widetilde{\Psi}(\boldsymbol{B}^{(k)})$. This is also the optimal weight solution of (sP), where the categorical choices are fixed to $\boldsymbol{B}^{(k)}$. The optimal objective value of the primal problem is an upper bound of the solution to (BP), assuming that there is at least one feasible solution depending on the fixed point $\boldsymbol{B}^{(k)}$.

The generic process of the proposed iterative scheme is given in Algorithm 5.

Algorithm 5 A generic bi-level framework with post-optimal sensitivities

```
1: initialize $\boldsymbol{B}^{(0)}$, set $k = 0$
2: while a termination criteria is not reached do
3:     Step 1  $\boldsymbol{a}^{(k)} \leftarrow \text{argmin } \widetilde{w}(\boldsymbol{a}, \boldsymbol{B}^{(k)})$ s.t. $\boldsymbol{a} \in \widetilde{\Omega}(\boldsymbol{B}^{(k)})$
4:     Step 2  $\boldsymbol{B}^{(k+1)} \leftarrow \text{solve } \mathcal{P}^{(k)}\left(\mathcal{P}^{(k-1)}, \widetilde{\boldsymbol{\Psi}}(\boldsymbol{B}^{(k)}), \frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\Big|_{\boldsymbol{B}^{(k)}}\right)$
5:     Increment $k$
6: end while
7: return $\boldsymbol{a}^{(j)}, \boldsymbol{B}^{(j)}, \widetilde{w}^{(j)}$ such that $j = \text{argmin } \widetilde{w}^{(l)}$ s.t. $l \leq k$.
```

4.3.2.2 On the computation of $\dfrac{d\widetilde{\Psi}}{d\boldsymbol{B}}$ at $\boldsymbol{B}^{(k)}$

In optimization, many efficient algorithms rely on the gradient of the functions involved in the optimization problem. This is the case of the proposed algorithm, that requires the gradient of $\widetilde{\Psi}$ with respect to the parameters $\boldsymbol{B}$. This gradient gives information on the behavior of the optimal weight, solution of (sBP), after a small perturbation of $\boldsymbol{B}^{(k)}$. The gradient of $\widetilde{\Psi}$

is also called *post-optimal sensitivity* (Fiacco 1976). It can be noted that the perturbation has no physical meaning since $\boldsymbol{B}$ describes categorical choices.

The efficient computation of the gradient of $\widetilde{\Psi}$ at a given $\boldsymbol{B}^{(k)}$ is a key feature of the proposed methodology. Indeed, if computed by finite differences, the gradient computation cost of the output of an optimization grows with the number of parameters $\boldsymbol{B}$. In this case, there are roughly as many optimization problems instances to solve than the number of parameters. The computational burden prevents from using this approach to compute the gradient of an optimization problem output with respect to a large number of parameters. It is then proposed to take advantage of the post-optimal sensitivity theory using penalty methods detailed in (Fiacco 1976) in order to compute these sensitivities efficiently. In the following Sections, post-optimal sensitivities refer to the sensitivities of $\widetilde{\Psi}$ obtained by penalty methods, not by finite differences.

- **Computation of the Lagrange multipliers**

  First, let be $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{\xi}_{\underline{\boldsymbol{a}}}$, $\boldsymbol{\xi}_{\bar{\boldsymbol{a}}}$ the Lagrange multipliers (column vectors) associated to the constraints $\boldsymbol{s}$, $\boldsymbol{\delta}$, the lower and upper bounds on the areas $\boldsymbol{a}$, respectively. The Lagrangian of the problem (sBP) is defined as follows :

$$\mathcal{L}(\boldsymbol{a},\boldsymbol{B}) := \widetilde{w}(\boldsymbol{a},\boldsymbol{B}) + \boldsymbol{\lambda}^\top \boldsymbol{s}(\boldsymbol{a},\boldsymbol{B}) + \boldsymbol{\mu}^\top \boldsymbol{\delta}(\boldsymbol{a},\boldsymbol{B}) + \boldsymbol{\xi}_{\underline{\boldsymbol{a}}}^\top (\underline{\boldsymbol{a}} - \boldsymbol{a}) + \boldsymbol{\xi}_{\bar{\boldsymbol{a}}}^\top (\boldsymbol{a} - \bar{\boldsymbol{a}})$$

  Let be $\boldsymbol{a}^*$ a local minimizer of problem (sBP($\boldsymbol{B}^{(k)}$)). Let be $\mathcal{A}_{\widetilde{\boldsymbol{s}}}$, $\mathcal{A}_{\widetilde{\boldsymbol{\delta}}}$, $\mathcal{A}_{\underline{\boldsymbol{a}}}$, $\mathcal{A}_{\bar{\boldsymbol{a}}}$ the sets of active constraints $\widetilde{\boldsymbol{s}}$, $\widetilde{\boldsymbol{\delta}}$, and bounds constraints on $\boldsymbol{a}$, respectively, such that

$$\begin{aligned}
\mathcal{A}_{\widetilde{\boldsymbol{s}}} &:= \left\{\forall i \mid \widetilde{\boldsymbol{s}}_i \left(\boldsymbol{a}^*(\boldsymbol{B}^{(k)})\right) = 0\right\}, \\
\mathcal{A}_{\widetilde{\boldsymbol{\delta}}} &:= \left\{\forall i \mid \widetilde{\boldsymbol{\delta}}_i \left(\boldsymbol{a}^*(\boldsymbol{B}^{(k)})\right) = 0\right\}, \\
\mathcal{A}_{\underline{\boldsymbol{a}}} &:= \left\{\forall i \mid \boldsymbol{a}_i^*(\boldsymbol{B}^{(k)}) = \underline{\boldsymbol{a}}_i\right\}, \\
\mathcal{A}_{\bar{\boldsymbol{a}}} &:= \left\{\forall i \mid \boldsymbol{a}_i^*(\boldsymbol{B}^{(k)}) = \bar{\boldsymbol{a}}_i\right\}.
\end{aligned} \tag{4.5}$$

  The active components of the constraints $\boldsymbol{s}$, $\boldsymbol{\delta}$, are noted, respectively, by $\boldsymbol{s}_{\mathcal{A}_{\boldsymbol{s}}}$ and $\boldsymbol{\delta}_{\mathcal{A}_{\boldsymbol{\delta}}}$. The components of $\boldsymbol{a}$ and $\underline{\boldsymbol{a}}$ that belong to $\mathcal{A}_{\underline{\boldsymbol{a}}}$ are noted by $\boldsymbol{a}_{\mathcal{A}_{\underline{\boldsymbol{a}}}}$ and $\underline{\boldsymbol{a}}_{\mathcal{A}_{\underline{\boldsymbol{a}}}}$, respectively. The components of $\boldsymbol{a}$ and $\bar{\boldsymbol{a}}$ that belong to $\mathcal{A}_{\bar{\boldsymbol{a}}}$ are noted by $\boldsymbol{a}_{\mathcal{A}_{\bar{\boldsymbol{a}}}}$ and $\bar{\boldsymbol{a}}_{\mathcal{A}_{\bar{\boldsymbol{a}}}}$, respectively. The components of the optimal Lagrange multipliers $\boldsymbol{\lambda}^*$, $\boldsymbol{\mu}^*$, $\boldsymbol{\xi}_{\underline{\boldsymbol{a}}}^*$, $\boldsymbol{\xi}_{\bar{\boldsymbol{a}}}^*$ of the active constraints of $\widetilde{\boldsymbol{s}}$, $\widetilde{\boldsymbol{\delta}}$, and bounds constraints on $\boldsymbol{a}$, are noted $\boldsymbol{\lambda}_{\mathcal{A}_{\boldsymbol{g}}}^*$, $\boldsymbol{\mu}_{\mathcal{A}_{\boldsymbol{\delta}}}^*$, $\boldsymbol{\xi}_{\mathcal{A}_{\underline{\boldsymbol{a}}}}^*$, $\boldsymbol{\xi}_{\mathcal{A}_{\bar{\boldsymbol{a}}}}^*$, respectively. Let be $\boldsymbol{I}_{\mathcal{A}_{\underline{\boldsymbol{a}}}} \in \mathcal{M}_{|\mathcal{A}_{\underline{\boldsymbol{a}}}|,n}(\mathbb{R})$ and $\boldsymbol{I}_{\mathcal{A}_{\bar{\boldsymbol{a}}}} \in \mathcal{M}_{|\mathcal{A}_{\bar{\boldsymbol{a}}}|,n}(\mathbb{R})$ the gradients of the lower and upper bound constraints, respectively. They are identity matrices where the lines corresponding to non active constraints indices are removed, $\forall j \in [\![1,n]\!]$:

$$\left(\boldsymbol{I}_{\mathcal{A}_{\underline{\boldsymbol{a}}}}\right)_{ij} = \delta_{ij} \quad \forall i \in \mathcal{A}_{\underline{\boldsymbol{a}}} \tag{4.6}$$

$$(\boldsymbol{I}_{\mathcal{A}_{\bar{\boldsymbol{a}}}})_{ij} = \delta_{ij} \quad \forall i \in \mathcal{A}_{\bar{\boldsymbol{a}}} \tag{4.7}$$

  Proposition 4.3.1

Consider $\boldsymbol{a}^*$ a local optimum of (sBP($\boldsymbol{B}^{(k)}$)). If the objective and constraints functions of (sBP($\boldsymbol{B}^{(k)}$)) are continuously differentiable, and the constraints are linearly independent, then ($\boldsymbol{a}^*$, $\boldsymbol{\lambda}^*$, $\boldsymbol{\mu}^*$, $\boldsymbol{\xi}^*_{\underline{a}}$, $\boldsymbol{\xi}^*_{\bar{a}}$) is a stationary point of the Lagrangian :

$$\left.\frac{\partial w}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} + \boldsymbol{\lambda}^*_{\mathcal{A}_{\widetilde{s}}}(\boldsymbol{B}^{(k)})^\top \left.\frac{\partial \widetilde{\boldsymbol{s}}_{\mathcal{A}_s}}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} + \boldsymbol{\mu}^*_{\mathcal{A}_{\widetilde{\delta}}}(\boldsymbol{B}^{(k)})^\top \left.\frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\delta}}}}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} - \boldsymbol{\xi}^*_{\mathcal{A}_{\underline{a}}}(\boldsymbol{B}^{(k)})^\top \boldsymbol{I}_{\mathcal{A}_{\underline{a}}} + \boldsymbol{\xi}^*_{\mathcal{A}_{\bar{a}}}(\boldsymbol{B}^{(k)})^\top \boldsymbol{I}_{\mathcal{A}_{\bar{a}}} = \boldsymbol{0}_n. \quad (4.8)$$

Proof. This is one of the Karush-Kuhn-Tucker (KKT) optimality conditions, detailed in Section A in the Theorem A.1. The Karush–Kuhn–Tucker conditions are first derivative tests (sometimes called first-order) necessary conditions for a solution in nonlinear programming to be optimal, provided that some regularity conditions are satisfied. □

Once the problem (sBP($\boldsymbol{B}^{(k)}$)) is solved, the Lagrange multipliers corresponding to active constraints are thus obtained by solving the linear system in equation 4.8. According to the KKT conditions, the computed values of the Lagrange multipliers have to be non-negative.

- **Expression of** $\left.\dfrac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}}$

  The expression of the post-optimal sensitivity $\left.\dfrac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}}$ is given in the Proposition 4.3.2.

Proposition 4.3.2
Consider $\boldsymbol{a}^*$ a local optimum of (sBP($\boldsymbol{B}^{(k)}$)). If,

- *the functions $\widetilde{w}$, $\widetilde{\boldsymbol{s}}$, and $\widetilde{\boldsymbol{\delta}}$ are twice continuously differentiable w.r.t. $\boldsymbol{a}$, and $\dfrac{\partial \widetilde{w}}{\partial \boldsymbol{a}}$, $\dfrac{\partial \widetilde{\boldsymbol{s}}}{\partial \boldsymbol{a}}$, $\dfrac{\partial \widetilde{\boldsymbol{\delta}}}{\partial \boldsymbol{a}}$ are once continuously differentiable w.r.t. $\boldsymbol{B}$ in a neighborhood of $(\boldsymbol{a}^*, \boldsymbol{B})$,*
- *the second order sufficient conditions (**SOSC**[1]), detailed in Section A, hold at $\boldsymbol{a}^*$, with $\boldsymbol{\lambda}^*$, $\boldsymbol{\mu}^*$, $\boldsymbol{\xi}^*_{\underline{a}}$ and $\boldsymbol{\xi}^*_{\bar{a}}$ the Lagrange multipliers associated to constraints $\widetilde{\boldsymbol{s}}$, $\widetilde{\boldsymbol{\delta}}$, and the bounds on the areas, respectively,*
- *the constraints are linearly independent at $\boldsymbol{a}^*$,*
- *strict complementary slackness conditions (**SCSC**[2]), detailed in Section A, holds at $\boldsymbol{a}^*$,*

then, $\widetilde{\Psi}$ is continuously differentiable, and its derivative taken in $\boldsymbol{B}^{(k)}$, in a neighborhood of $\boldsymbol{B}$, is given by

$$\left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}} = \left.\frac{\partial \widetilde{w}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} + \boldsymbol{\lambda}^*_{\mathcal{A}_s}(\boldsymbol{B}^{(k)})^\top \left.\frac{\partial \widetilde{\boldsymbol{s}}_{\mathcal{A}_{\widetilde{s}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} + \boldsymbol{\mu}^*_{\mathcal{A}_\delta}(\boldsymbol{B}^{(k)})^\top \left.\frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\delta}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}}. \quad (4.9)$$

[1] Refer to Theorem A.2 in Annex A.
[2] Refer to Equation (**SCSC**) in Annex A.

Proof. The continuous differentiable property of $\widetilde{\Psi}$ comes from the Theorem A.3 applied to (sBP($\boldsymbol{B}^{(k)}$)). The first-order derivative formula of the optimal value function is a result of the Theorem A.4 :

$$\left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}} = \left.\frac{\partial \widetilde{w}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} + \boldsymbol{\lambda}^*_{\mathcal{A}_{\widetilde{\boldsymbol{s}}}}(\boldsymbol{B}^{(k)})^\top \left.\frac{\partial \widetilde{\boldsymbol{s}}_{\mathcal{A}_{\widetilde{\boldsymbol{s}}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} + \boldsymbol{\mu}^*_{\mathcal{A}_{\widetilde{\boldsymbol{\delta}}}}(\boldsymbol{B}^{(k)})^\top \left.\frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\boldsymbol{\delta}}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}}$$
$$+ \boldsymbol{\xi}^*_{\mathcal{A}_{\underline{\boldsymbol{a}}}}(\boldsymbol{B}^{(k)})^\top \frac{\partial}{\partial \boldsymbol{B}}\left(\underline{\boldsymbol{a}}_{\mathcal{A}_{\underline{\boldsymbol{a}}}} - \boldsymbol{a}^*_{\mathcal{A}_{\underline{\boldsymbol{a}}}}(\boldsymbol{B}^{(k)})\right) + \boldsymbol{\xi}^*_{\mathcal{A}_{\bar{\boldsymbol{a}}}}(\boldsymbol{B}^{(k)})^\top \frac{\partial}{\partial \boldsymbol{B}}\left(\boldsymbol{a}^*_{\mathcal{A}_{\bar{\boldsymbol{a}}}}(\boldsymbol{B}^{(k)}) - \bar{\boldsymbol{a}}_{\mathcal{A}_{\bar{\boldsymbol{a}}}}\right).$$

Since the bound constraints on the areas do not depend on $\boldsymbol{B}$, the derivative of $\widetilde{\Psi}$ is given by :

$$\left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}} = \left.\frac{\partial \widetilde{w}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} + \boldsymbol{\lambda}^*_{\mathcal{A}_{\widetilde{\boldsymbol{s}}}}(\boldsymbol{B}^{(k)})^\top \left.\frac{\partial \widetilde{\boldsymbol{s}}_{\mathcal{A}_{\widetilde{\boldsymbol{s}}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}} + \boldsymbol{\mu}^*_{\mathcal{A}_{\widetilde{\boldsymbol{\delta}}}}(\boldsymbol{B}^{(k)})^\top \left.\frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\boldsymbol{\delta}}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(k)}}$$

Both aforementioned theorems are detailed in Appendix A, and have been introduced in (Fiacco 1976). □

The value $\left.\dfrac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}}$ is thus obtained by introducing into equation 4.9 the gradients of the objective and the active constraints w.r.t. the parameters $\boldsymbol{B}$ and taken in $(\boldsymbol{a}^*, \boldsymbol{B}^{(k)})$. The Lagrange multipliers have been computed previously thanks to the equation 4.8. In terms of computational cost, the computation of the gradient of $\widetilde{\Psi}$ does only require one evaluation of $\widetilde{\Psi}(\boldsymbol{B}^{(k)})$. This evaluation would have been required anyway for the definition of $\mathcal{P}^{(k)}$. Under the conditions given by the Theorem A.3, the number of evaluations of $\widetilde{\Psi}$ needed into the expression of the post-optimal sensitivity (4.9) is thus independent from the number of parameters $\boldsymbol{B}$, unlike the finite differences approach (4.10).

Note on post-optimal sensitivities computation by finite differences
The gradient of the function $\widetilde{\Psi}$ taken at $\boldsymbol{B}^{(k)} \in \mathcal{C}^{n,p}$ can also be obtained by finite differences. Let be $\boldsymbol{B}^{(k)} + \boldsymbol{h}_i\boldsymbol{e}_i$ a perturbation of the i^{th} component of $\boldsymbol{B}^{(k)}$, with $\boldsymbol{h}_i$ a scalar being the (small enough) step size of the perturbation, and $\boldsymbol{e}_i$ a vector of the canonical basis. The gradient of $\widetilde{\Psi}$, if computed by finite differences, is given by :

$$\left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}} = \left(\frac{\Delta\Psi_{(1)}}{\boldsymbol{h}_1}, \dots, \frac{\Delta\Psi_{(i)}}{\boldsymbol{h}_i}, \dots, \frac{\Delta\Psi_{(np)}}{\boldsymbol{h}_{np}}\right), \tag{4.10}$$

with

$$\Delta\widetilde{\Psi}_{(i)} = \widetilde{\Psi}(\boldsymbol{B}^{(k)} + \boldsymbol{h}_i\boldsymbol{e}_i) - \widetilde{\Psi}(\boldsymbol{B}^{(k)}).$$

This means that the computation of $\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}$ by finite differences requires $np + 1$ evaluations of $\widetilde{\Psi}$, i.e., at $\boldsymbol{B}^{(k)}$ and at every perturbed value $\boldsymbol{B}^{(k)} + \boldsymbol{h}_i\boldsymbol{e}_i$ ($\forall i \in \{1, \dots, np\}$). The computation cost, in terms of evaluations of $\widetilde{\Psi}$, is linear with respect to the number of parameters. An evaluation of $\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}$ computed by finite differences using (4.10) would indeed cost $np+1$ continuous optimizations. This can be computationally expensive, in particular in the case of a large number of parameters $\boldsymbol{B}^{(k)}$. This often the case in an industrial context where $\widetilde{\Psi}$ is a heavy nested function that wraps an optimization involving multiple calls to physical models. This is why it is preferred to rely on the previous theoretical framework for the computation of the gradient of $\widetilde{\Psi}$.

4.3.2.3 On the construction of $\mathcal{P}^{(k)}$

In this Section, a definition of the dual problem is given. Provided that $\boldsymbol{B}^{(k)}$ is a solution of (mBP), and under the assumption that $\widetilde{Psi}$ is convex, then the linearization of $\widetilde{Psi}$ about $\boldsymbol{B}^{(k)}$ and given by

$$\eta \geq \widetilde{\Psi}(\boldsymbol{B}^{(k)}) + \left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}}^{\top} (\boldsymbol{B} - \boldsymbol{B}^{(k)})$$

is an outer approximation of the feasible set of the original problem (BP). This helps in the definition of a new problem (4.11) that has an identical solution to (BP), under the conditions detailed in the Theorem 4.3.3. In the problem (4.11), the function $\widetilde{\Psi}$ is replaced by an hyperplane that is also its linear support at $\boldsymbol{B}^{(k)}$.

Proposition 4.3.3 (optimum equivalence of problem (4.11) and (sBP))
If the assumptions in Proposition 4.3.2 hold, and if $\widetilde{\Psi}$ is convex, then the problem (mBP) and the mixed integer linear program (MILP) named problem (4.11) defined as follows :

$$\begin{aligned} &\min_{\boldsymbol{B}\in\mathcal{C}^{n,p},\eta\in\mathbb{R}} && \eta \\ &\text{subject to} && \eta \geq \widetilde{\Psi}(\boldsymbol{B}^{(k)}) + \left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}}^{\top} (\boldsymbol{B} - \boldsymbol{B}^{(k)}) \quad \forall \boldsymbol{B}^{(k)} \in \mathcal{C}^{n,p} \cap \{0,1\} \\ & && \boldsymbol{B} \circ (\boldsymbol{B} - \boldsymbol{1}) = \boldsymbol{0}_{np} \end{aligned} \tag{4.11}$$

have the same optimal solution ($\boldsymbol{B}^*$).

Proof. If the assumptions in Proposition 4.3.2 hold, then $\widetilde{\Psi}$ is continuously differentiable (P1). Furthermore, the constraints qualification (**LICQ**) holds (P2).
Finally, is assumed that $\widetilde{\Psi}$ is convex (P3). Thus, since the properties (P1), (P2) and (P3) are verified, and according to the Theorem 1 in (Fletcher and Leyffer 1994), the equivalence between the problems (4.11) and (mBP) is proved. □

The problem (4.11) is not a MINLP problem but a MILP problem, meaning that in theory, efficient MILP solvers can solve (4.11). However, solving problem (4.11) directly is impractical since this would require p^n evaluations of $\widetilde{\Psi}$ corresponding to all integer vectors $\boldsymbol{B}^{(k)}$ in $\mathcal{C}^{n,p}$. This means that it would necessitate evaluations of the sizing problem (sBP) taken at every p^n combinations of materials and cross-section shapes available in Γ^n. In other words, once the problem (4.11) is set up, the solution to the problem (BP) is known. Furthermore, another drawback is that (4.11) contains a large number of constraints, proportional to the number of elements in Γ^n. This is why, instead of considering the problem (4.11), the OA algorithm involves a relaxation of the problem (4.11), relying on the solutions of only k evaluations of $\widetilde{\Psi}$:

$$\begin{aligned} \min_{\eta \in \mathbb{R}, \boldsymbol{B} \in \mathcal{C}^{n,p}} \quad & \eta \\ \text{subject to} \quad & \eta \geq \widetilde{\Psi}(\boldsymbol{B}^{(l)}) + \left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}}^{\top} (\boldsymbol{B} - \boldsymbol{B}^{(l)}) \quad \forall \boldsymbol{B}^{(l)} \in K^{(k)} \\ & \boldsymbol{B} \circ (\boldsymbol{B} - \mathbf{1}) = \mathbf{0}_{np} \end{aligned} \tag{4.12}$$

with $K^{(k)}$ a set of k elements in $\mathcal{C}^{n,p}$, such that :

$$K^{(k)} \subset \mathcal{C}^{n,p} \cap \{0, 1\}.$$

Thanks to the convexity of $\widetilde{\Psi}$ and the definition of $K^{(k)}$, the problem (4.12) yields a lower bound to the solution of the Problem (BP). The problem (4.12) can be geometrically interpreted as an exploration of the effects of the outer approximations (i.e., the linear supports) on the objective $\widetilde{\Psi}$. Furthermore, it underestimates the objective while overestimating the feasible region. Therefore, the problem (4.12) yields a lower bound of the Problem (MINLP). In the case where $K^{(k)}$ is equal to $\mathcal{C}^{n,p}$, the problem (4.12) is strictly equivalent to (MINLP), according to the Proposition 4.3.3.

4.3.2.4 A bi-level framework with post-optimal sensitivities

The proposed algorithm consists of solving an alternating sequence of primal and dual problems, as defined previously. The post-optimal sensitivities of the slave problem (sizing) are involved in the definition of the dual problem. Let be (k) the current outer iteration of the algorithm.

First, the primal problem, reduced to an evaluation of $\widetilde{\Psi}$, is solved at $\boldsymbol{B}^{(k)}$. This means that as a first step, the slave continuous optimization problem (sBP) aiming at minimizing the weight while satisfying stress and displacements constraints is solved. The problem (sBP($\boldsymbol{B}^{(k)}$)) is solved and yields a solution $\boldsymbol{a}^{(k)}$ such that :

$$\boldsymbol{a}^{(k)} := \underset{\boldsymbol{a} \in \widetilde{\Omega}(\boldsymbol{B}^{(k)})}{\operatorname{argmin}} \ \widetilde{w}(\boldsymbol{a}, \boldsymbol{B}^{(k)}). \tag{4.13}$$

The upper bound $U^{(k)}$ to the solution of (BP) is defined by :

$$U^{(k)} := \widetilde{w}(\boldsymbol{a}^{(k)}, \boldsymbol{B}^{(k)}),$$

that is also equal to $\widetilde{\Psi}(\boldsymbol{B}^{(k)})$. The best current solution of the original problem (BP) is thus given by the best upper bound returned during the (k) outer iterations :

$$U := \min\{U^{(1)}, \dots, U^{(k)}\}. \tag{4.14}$$

Second, the relaxed MILP problem (4.12) can be set up. Its definition relies on the linearizations of $\widetilde{\Psi}$ taken at the solutions yielded during the (k) previous iterations. While the linearizations from the previous iterates $(l) < (k)$ remain unchanged, the linearization of $\widetilde{\Psi}$ at the current iteration (k) has to be computed. More precisely, the gradient of $\widetilde{\Psi}$ taken at $\boldsymbol{B}^{(k)}$ has to be evaluated. Following the theoretical steps given in Section 4.3.2.2, the computation of the post-optimal sensitivities is decomposed in 5 main steps :

- Build the set of active constraints $\mathcal{A}_{\widetilde{\boldsymbol{s}}}$, $\mathcal{A}_{\widetilde{\boldsymbol{\delta}}}$, $\mathcal{A}_{\underline{\boldsymbol{a}}}$, $\mathcal{A}_{\bar{\boldsymbol{a}}}$ defined by equations (4.5),
- Compute the gradients of the objective and active constraints w.r.t. $\boldsymbol{a}$, at $(\boldsymbol{a}^*, \boldsymbol{B}^{(k)})$,
- Compute the Lagrange multipliers $\boldsymbol{\lambda}^*_{\mathcal{A}_{\boldsymbol{g}}}$, $\boldsymbol{\mu}^*_{\mathcal{A}_{\boldsymbol{\delta}}}$ by solving the equation (4.8),
- Compute the gradients of the objective and active constraints w.r.t. $\boldsymbol{B}$, at $(\boldsymbol{a}^*, \boldsymbol{B}^{(k)})$,
- Compute the post-optimal sensitivity $\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}$ at $\boldsymbol{B}^{(k)}$ using equation (4.9).

Once the linearization of $\widetilde{\Psi}$ has been computed, it is added as constraint in the problem (4.12). Furthermore, since in practice the problem (BP) does not need to be solved to optimality, it is sufficient to generate the new $(\boldsymbol{B}^{(k+1)})$ by adding a tolerance ϵ on the upper bound U as a constraint in the MILP master problem. The resulting mixed integer linear integer problem (MILP($K^{(k)}$)), is thus given by:

$$\begin{aligned}
\min_{\boldsymbol{B}\in\mathcal{C}^{n,p}} \quad & \eta^{(k)} \\
\text{subject to} \quad & \eta \leq U - \epsilon \\
& \eta \geq \widetilde{\Psi}(\boldsymbol{B}^{(k)}) + \left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(k)}}^{\top} (\boldsymbol{B} - \boldsymbol{B}^{(k)}) \\
& \eta \geq \widetilde{\Psi}(\boldsymbol{B}^{(l)}) + \left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(l)}}^{\top} (\boldsymbol{B} - \boldsymbol{B}^{(l)}) \quad \forall \boldsymbol{B}^{(l)} \in K^{(k-1)} \\
& \boldsymbol{B} \circ (\boldsymbol{B} - \mathbf{1}) = \mathbf{0}_{np}
\end{aligned} \tag{MILP($K^{(k)}$)}$$

with $K^{(k)}$ such that

$$K^{(k)} = K^{(k-1)} \cup \{\boldsymbol{B}^{(k)}\},$$

and $K^{(k-1)}$ the set of the $k-1$ previous $\boldsymbol{B}^{(k-1)}$, such that

$$K^{(k-1)} \subset \mathcal{C}^{n,p} \cap \{0, 1\}.$$

The problem (MILP($K^{(k)}$)) is built iteration per iteration by adding, as constraints, linearizations of the functions $\widetilde{\Psi}$ taken at the current solution $(\boldsymbol{B}^{(k)})$. The optimality of the algorithm relies on the convexity of $\widetilde{\Psi}$, ensuring that the linearizations are underestimators of $\widetilde{\Psi}$. Once built, the problem (MILP($K^{(k)}$)) is solved and provides a lower bound of (BP). While there is no guarantee that the upper bounds $U^{(k)}$ will decrease along the overall iterations, the definition of the problem (MILP($K^{(k)}$)) ensures the increase of the lower bound, iteration per iteration. The process of the proposed iterative scheme is given in Algorithm 6.

Algorithm 6 Bi-level algorithm with outer approximation cuts

```
 1: initialize $\boldsymbol{B}^{(0)}$ and $\epsilon$, set $k = 0, K^{(-1)} = \{\emptyset\}, U^{(-1)} = +\infty$
 2: while problem (MILP($K^{(k)}$)) is feasible do
 3:     Evaluate $\Psi(\boldsymbol{B}^{(k)})$, let be $\boldsymbol{a}^{(k)}$ the solution to the problem (sBP)
 4:     if (sBP) is feasible and $\widetilde{\Psi}(\boldsymbol{B}^{(k)}) < U^{(k)}$ then
 5:         $\boldsymbol{a}^* \leftarrow \boldsymbol{a}^{(k)}$
 6:         $\boldsymbol{B}^* \leftarrow \boldsymbol{B}^{(k)}$
 7:         $U^{(k)} \leftarrow \Psi(\boldsymbol{B}^{(k)})$
 8:     else
 9:         $U^{(k)} \leftarrow U^{(k-1)}$
10:     end if
11:     $K^{(k)} \leftarrow K^{(k-1)} \bigcup \{\boldsymbol{B}^{(k)}\}$
12:     Compute $\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}$ at $\boldsymbol{B}^{(k)}$ using equation (4.9)
13:     Solve (MILP($K^{(k)}$)), let the solution be $\boldsymbol{B}^{(k+1)}$
14:     $k \leftarrow k + 1$
15: end while
16: return $\boldsymbol{a}^*$, $\boldsymbol{B}^*$, and $\widetilde{w}^* \leftarrow U^{(k-1)}$.
```

The proposed algorithm leverages the use of post-optimal sensitivities by using them to define supporting hyperplanes of $\widetilde{\Psi}$. These hyperplanes bound the convex hull of the slave problem (sP) solutions. It is worth to note that the number of contraints involved in the master problem (mBP) reduces to the k linearizations of $\widetilde{\Psi}$ from the (k) outer iterations, in addition to the (k) linear equality constraints involved in the definition of $\mathcal{C}^{(n,p)}$. Indeed, the OA algorithm is used to solve the master problem (mBP), so that all the structural sizing constraints are handled by the slave problem (sBP). Hence, the MILP problem (MILP($K^{(k)}$)) counts only $k+n$ linear constraints (including equality constraints from $\mathcal{C}^{(n,p)}$), compared to the $k \times (n \times m + d + n)$ (constraints $\boldsymbol{s}$, $\boldsymbol{\delta}$ and equality constraints from $\mathcal{C}^{(n,p)}$). In industrial cases where the number of structural elements n can reach 5000 elements (e.g., for a fuselage), and the number of constraints m per structural element is about 10 (after screening). The problem (MILP($K^{(k)}$)) can thus involve several millions of constraints. This could induce high computation time (Benson and Horst 1991; Stolpe and Sandal 2018) when solving the problem (MILP($K^{(k)}$)). In the proposed algorithm, the number of constraints in (MILP($K^{(k)}$)) is limited without dropping the proof of optimality, thanks to the bi-level decomposition scheme.

Furthermore, two interesting properties about the OA algorithm efficiency have been introduced in (Fletcher and Leyffer 1994). These properties also apply to the proposed methodology, that falls in the theoretical frame of the OA algorithm. First, under assumptions given in Theorem 4.3.4, the authors prove that the OA ends in a finite number of steps :

Proposition 4.3.4
If $\widetilde{\Psi}$ is convex, then the Algorithm 6 terminates in a finite number of steps at an optimal

solution of (mBP).

Proof. The objective $\widetilde{\Psi}$ of the problem (mBP) is convex (P1), by assumption. The objective $\widetilde{\Psi}$ of the problem (mBP) is continuously differentiable according to Proposition 4.3.2, and so are the linear equality constraints (P2). The set $\mathcal{C}^{n,p}$ is finite, and the continuous variables a belong to a nonempty compact convex set (P3). The problem (mBP) has no set of constraints that can be linearly dependent (P4). Thus, according to the properties (P1), (P2), (P3) and (P4), the Theorem 2 of (Fletcher and Leyffer 1994) states that the Algorithm 6 terminates in a finite number of steps at an optimal solution of (mBP). □

Second, according to (Grossmann 2009) the OA method generally works efficiently. This means that relatively few overall iterations are required. According to the authors, one reason to this behavior is related to the Property 4.3.5.

Proposition 4.3.5 (Trivial convergence when $\widetilde{\Psi}$ is linear (Grossmann 2009))

If $\widetilde{\Psi}$ is linear, then the OA algorithm trivially converges to the solution of (mBP) in one iteration.

Proof. The constraints of the problem (mBP) are linear. If the objective is, by assumption, also linear, then the problem (MILP($K^{(k)}$)) is identical to the original problem (BP). □

4.4 Numerical results

In the present section, our proposed methodology will be applied to three different test cases: (i) the well-known 10-bar truss structure (Haftka and Gürdal 1992) adapted in (Merval 2008), (ii) a 2D cantilever structure (Shahabsafa et al. 2018), and (iii) a 120-bar dome truss structure (Saka and Ulker 1992). In order to evaluate the scalability of the methodology with respect to large number of structural elements, the 2D cantilever structure is made scalable by varying the number of blocks. The third test case is included to evaluate the capability of handling more complex structures, with large number of categorical values.

4.4.1 Implementation details and comparison solvers

Algorithm 6 was implemented using the Generic Engine for MDO Scenarios (GEMS) (Gallard et al. 2018) in Python. The tool offers an efficient way to implement and test multi-level formulations, with built-in classes that facilitate optimization problems manipulations (Gallard et al. 2019). The continuous non-linear optimization problems (i.e., evaluations of $\widetilde{\Psi}$) are solved with the Method of Moving Asymptotes (MMA) (Svanberg 2002) as implemented in the nonlinear-optimization (NLOPT) package (Johnson 2008). The MMA solver is capable of handling non-linear continuous optimization problems with inequality constraints. It is worth to note that other available solvers could also be relevant to solve the continuous problems. As soon as the same continuous optimization solver is used to benchmark the proposed approaches, the consistency of the result analyses is preserved. The mixed integer linear optimization problems are solved with a branch and cut implemented as the Coin-or Branch and Cut (coin-or/Cbc) in (Forrest et al. 2018). The algorithm is launched through the Google ortools suite for optimization (Perron and Furnon 2019). All the default parameters are kept unchanged except the tolerance on the objective function which is set to 10^{-6} kg. In what comes next, the resulting implementation of Algorithm 6 will be called Bi-level OA.

Four solvers will be compared to Bi-level OA. First, a baseline solver where we proceed with an exhaustive enumeration of continuous optimizations w.r.t. $\boldsymbol{a}$ (Problem (sBP)) taken at every available choice in $\mathcal{C}^{(n,p)}$, the solution resulting with this solver will be denoted as Baseline. Second, a hybrid branch and bound (Algorithm 2, noted h-B&B), where one uses a relaxation procedure combined with a branch and bound algorithm. Under the assumption that these problems are convex with respect to the sizing variables $\boldsymbol{a}$, Baseline and h-B&B return the global optimum of the overall problem. The third solver used in the comparison is a Genetic algorithm (Deb and Goyal 1998) using the implementation given by Distributed Evolutionary Algorithms in Python (DEAP) (Fortin et al. 2012). The latter solver will be referred by Genetic in our comparison tests. Due to the stochastic nature of Genetic, the obtained results (for this solver) will be displayed as the average of ten runs. Finally, the bi-level Algorithm 3, noted Bi-level uses a first order-like approximation of the slave problem output with respect to the categorical variables. Similarly to Bi-level and h-B&B, Bi-level OA uses the MMA method from NLOPT to solve the slave problem.

In all what comes next, the computation effort of a given solver will be measured by counting the number of structural analyses (noted #FEM) including those required by the computation of the gradients (when needed). The obtained optimal weights (by each solver) will be noted w^*, the latter will allow us to evaluate the quality of the optima found by each solver. We note also that in our setting, the Baseline solution can be seen as the best known categorical choices for the corresponding problem instance. Thus, in this context, it is important to evaluate how far the categorical choices (obtained by the tested solvers) are from the Baseline optimal choices. This information will be given using the Hamming distance (noted d_h) where we will count the number of structural elements that have an optimal choice different to the Baseline categorical choices.

4.4.2 A step by step example: a 3-bar truss structure

To illustrate how the Bi-level method works, we will now describe in detail its application to a simple 3-bar truss structure (Figure 4.2). For this problem, each element can take a value among three possible choices that respectively point to materials AL2139, AL2024, TA6V and the same "I"-profile (see Figure 1.3). The materials properties are listed in Table 4.1. For this simple case, one has $n = 3$, $p = 3$, and $\boldsymbol{B} \in \mathcal{C}^{3,3}$. In its direct coding version, $C^{3,3}$ corresponds to $\Gamma^3 = \{1, 2, 3\}^3$. For all elements, the lower and upper bounds on areas are respectively fixed to 100 mm^2 and 2000 mm^2. The bounds on areas, and initial areas are

	AL2139	AL2024	TA6V
Density (kg/mm^3)	$2.8\ 10^{-6}$	$2.77\ 10^{-6}$	$4.43\ 10^{-6}$
Young modulus (MPa)	$7.1\ 10^4$	$7.4\ 10^4$	$11.0\ 10^4$
Poisson coefficient ($-$)	0.3	0.33	0.33
Tension allow. (MPa)	$1.5\ 10^2$	$1.6\ 10^2$	$11.0\ 10^2$
Compression allow. (MPa)	$2.0\ 10^2$	$2.1\ 10^2$	$8.6\ 10^2$

Table 4.1: Numerical details on materials attributes.

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$2000\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$2000\ mm^2$

Table 4.2: Bounds on areas, and initial areas values of the 3-bar truss optimization case.

fixed as detailed in Table 4.2. A maximum downward displacement equal to $\bar{\boldsymbol{u}} = 1\ mm$ is allowed on the only free node of the structure:

$$\begin{aligned}\widetilde{\delta}\colon \mathbb{R}^3 \times \mathcal{C}^{3,3} &\to \mathbb{R}\\ (\boldsymbol{a}, \boldsymbol{B}) &\mapsto \boldsymbol{P}\boldsymbol{u}(\boldsymbol{a}, \widetilde{\boldsymbol{E}}(\boldsymbol{B})) - \bar{\boldsymbol{u}}.\end{aligned}$$

During application of the Bi-level OA method, the initialization is such that :

$$\boldsymbol{B}^{(0)} = \text{vec}\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix},\ \epsilon = 1.e^{-3}kg,\ k = 0,\ K^{(-1)} = \{\emptyset\},\ U^{(-1)} = +\infty.$$

- *First iteration* ($k = 0$)

  The first iteration $k = 0$ starts by solving primal problem, that reduces to an evaluation

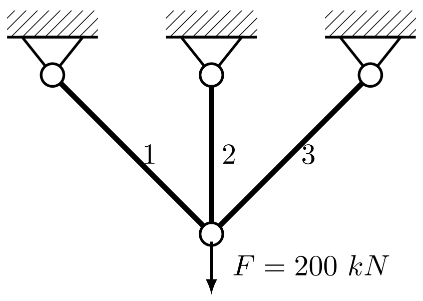


Figure 4.2: A 3-bar truss structure where a downward load $F = 200\ kN$ is applied on the free node.

of $\widetilde{\Psi}$ (by solving (sBP)) at the current guess $\boldsymbol{B}^{(0)}$:

$$U^{(0)} = 13.82\ kg,\ \boldsymbol{a}^{(0)} = [1041.29,\ 2000.0,\ 664.39]\ mm^2$$

Then, the gradient $\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\Big|_{\boldsymbol{B}^{(0)}}$ is computed. To that purpose the active constraints of the problem sBP) at $(\boldsymbol{a}^{(0)}, \boldsymbol{B}^{(0)})$ are identified. At this first iteration, the active constraints are the upper bound constraint on the area of structural element 2, and the constraint on displacements, such that

$$\bar{\boldsymbol{a}}_2 - \boldsymbol{a}_2^{(0)} = 0,\ \delta(\boldsymbol{a}^{(0)}, \boldsymbol{B}^{(0)}) = 0.$$

and then the sets of active constraints indices are

$$\mathcal{A}_{\bar{\boldsymbol{a}}} = \{2\},\ \mathcal{A}_{\widetilde{\delta}} = \{1\},\ \mathcal{A}_{\underline{\boldsymbol{a}}} = \mathcal{A}_{\widetilde{\boldsymbol{s}}} = \{\emptyset\}.$$

The gradients of the weight and active constraints w.r.t. $\boldsymbol{a}$ are computed, respectively:

$$\frac{\partial \widetilde{w}}{\partial \boldsymbol{a}}\Bigg|_{\boldsymbol{a}^*, \boldsymbol{B}^{(0)}} = \begin{pmatrix} 3.96e^{-3} \\ 2.77e^{-3} \\ 6.26e^{-3} \end{pmatrix}^\top,\ \frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\delta}}}}{\partial \boldsymbol{a}}\Bigg|_{\boldsymbol{a}^*, \boldsymbol{B}^{(0)}} = \begin{pmatrix} -1.24e^{-4} \\ -3.70e^{-4} \\ -1.97e^{-4} \end{pmatrix}^\top,\ \boldsymbol{I}_{\mathcal{A}_{\bar{\boldsymbol{a}}}} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}^\top.$$

One can verify that the gradients of the active constraints are linearly independent, and thus (**LICQ**) holds. The stationarity of the Lagragian at $(\boldsymbol{a}^{(0)}, \boldsymbol{B}^{(0)})$, as in Equation (4.8), leads to the following overdetermined linear system (3 equations, 2 unknown Lagrange multipliers):

$$\frac{\partial w}{\partial \boldsymbol{a}}\Bigg|_{\boldsymbol{a}^*, \boldsymbol{B}^{(0)}} + \boldsymbol{\mu}^*_{\mathcal{A}_{\widetilde{\delta}}}(\boldsymbol{B}^{(0)})^\top \frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\delta}}}}{\partial \boldsymbol{a}}\Bigg|_{\boldsymbol{a}^*, \boldsymbol{B}^{(0)}} + \boldsymbol{\xi}^*_{\mathcal{A}_{\bar{\boldsymbol{a}}}}(\boldsymbol{B}^{(0)})^\top \boldsymbol{I}_{\mathcal{A}_{\bar{\boldsymbol{a}}}} = 0.$$

where the gradients are replaced by their value in order to compute the Lagrange multipliers, that are:

$$\boldsymbol{\mu}^*_{\mathcal{A}_\delta}(\boldsymbol{B}^{(0)}) = 31.86\ kg/mm,$$
$$\boldsymbol{\xi}^*_{\mathcal{A}_{\bar{\boldsymbol{a}}}}(\boldsymbol{B}^{(0)}) = 9.02e^{-3}\ kg/mm^2.$$

As a remark, these multipliers illustrate the optimal weight (of the slave problem (sBP)) sensitivity with respect to a perturbation of the constraint on the upper bound or displacements, respectively. The values show that the problem optimum is much more sensitive to the constraint on displacements than the bound constraint.

The gradients of the weight and the displacement constraints w.r.t. $\boldsymbol{B}$ are computed, respectively:

$$\left.\frac{\partial \widetilde{w}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(0)}} = [4.12, 4.08, 6.52, 5.6, 5.54, 8.86, 2.63, 2.6, 4.16]\,,$$

$$\left.\frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\delta}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(0)}} = \left[-0.13, -0.13, -0.2, -0.71, -0.74, -1.1, -8.4e^{-2}, -8.8e^{-2}, -0.13\right].$$

These values are replaced in the following equation (adapted from equation 4.9) in order to compute the gradient of $\widetilde{\Psi}$:

$$\left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(0)}} = \left.\frac{\partial \widetilde{w}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(0)}} + \boldsymbol{\mu}^*_{\mathcal{A}_\delta}(\boldsymbol{B}^{(0)})^\top \left.\frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\delta}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(0)}}. \tag{4.15}$$

Its numeric value is:

$$\left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(0)}} = \left[6.2e^{-3}, -0.21, 0.14, -17.0 - 18.0, -26.2, -5.7e^{-2}, -0.2, -2.5e^{-3}\right].$$

Physically, the values seem to indicate that the optimal weight (of the slave problem (sBP)) is very sensitive to the choices of materials and shapes on the second structural element, when compared to the others. This could be explained by the fact that the load case has a vertical component only, in line with the second element. Furthermore, one can see that excepted the first choice (AL2139) of the first element, all the choices are expected to make the optimal weight decrease. However, generally these values have to be interpreted with caution. Indeed, these sensitivities are only valid in a (close enough) neighborhood of $\boldsymbol{B}^{(0)}$, according to the hypothesis of the proposition 4.3.2. A change in the active constraint set could occur at intermediate values of $\boldsymbol{B}$.

The history of the previous iterations is updated with $\boldsymbol{B}^{(0)}$ such that:

$$K^{(0)} = \{\boldsymbol{B}^{(0)}\}.$$

The MILP problem (MILP($K^{(k)}$)) can now be set up, as follows:

$$\begin{aligned} \min_{\boldsymbol{B}\in\mathcal{C}^{3,3}} \quad & \eta \\ \text{subject to} \quad & \eta \leq U^{(0)} - \epsilon \\ & \eta \geq \widetilde{\Psi}(\boldsymbol{B}^{(0)}) + \left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(0)}}^\top (\boldsymbol{B} - \boldsymbol{B}^{(0)}) \\ & \boldsymbol{B} \circ (\boldsymbol{B} - \mathbf{1}) = \mathbf{0}_9. \end{aligned} \tag{MILP($K^{(0)}$)}$$

The solution of this problem provides the new integer candidate solution $\boldsymbol{B}^{(1)}$:

$$\boldsymbol{B}^{(1)} = \text{vec}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

- *Second iteration* $(k = 1)$

The second iteration $k = 1$ starts by solving primal problem, that reduces to an evaluation of $\widetilde{\Psi}$ (by solving (sBP)) at the current guess $\boldsymbol{B}^{(1)}$:

$$U^{(1)} = 8.63\ kg,\ \boldsymbol{a}^{(1)} = [100,\ 1770.61,\ 100]\ mm^2$$

Then, the gradient $\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\Big|_{\boldsymbol{B}^{(1)}}$ is computed. To that purpose the active constraints of the problem sBP) at $(\boldsymbol{a}^{(1)}, \boldsymbol{B}^{(1)})$ are identified. At this first iteration, the active constraints are the lower bound constraint on the areas of structural elements 1 and 3, and the constraint on displacements, such that

$$\underline{\boldsymbol{a}}_1 - \boldsymbol{a}_1^{(1)} = 0,\ \underline{\boldsymbol{a}}_3 - \boldsymbol{a}_3^{(1)} = 0,\ \delta(\boldsymbol{a}^{(1)}, \boldsymbol{B}^{(1)}) = 0.$$

and then the sets of active constraints indices are

$$\mathcal{A}_{\underline{\boldsymbol{a}}} = \{1, 3\},\ \mathcal{A}_{\widetilde{\delta}} = \{1\},\ \mathcal{A}_{\bar{\boldsymbol{a}}} = \mathcal{A}_{\widetilde{\boldsymbol{s}}} = \{\emptyset\}.$$

The gradients of the weight and active constraints w.r.t. $\boldsymbol{a}$ are computed, respectively:

$$\frac{\partial \widetilde{w}}{\partial \boldsymbol{a}}\Bigg|_{\boldsymbol{a}^*, \boldsymbol{B}^{(1)}} = \begin{pmatrix} 3.9e^{-3} \\ 4.4e^{-3} \\ 3.9e^{-3} \end{pmatrix}^{\top},\ \frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\delta}}}}{\partial \boldsymbol{a}}\Bigg|_{\boldsymbol{a}^*, \boldsymbol{B}^{(1)}} = \begin{pmatrix} -1.3e^{-4} \\ -5.5e^{-4} \\ -1.3e^{-4} \end{pmatrix}^{\top},\ \boldsymbol{I}_{\mathcal{A}_{\underline{\boldsymbol{a}}}} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}^{\top}.$$

One can verify that the gradients of the active constraints are linearly independent, and thus (**LICQ**) holds. The stationarity of the Lagragian at $(\boldsymbol{a}^*, \boldsymbol{B}^{(1)})$, as in Equation (4.8), leads to the following full rank linear system (3 equations, 3 unknown Lagrange multipliers):

$$\frac{\partial w}{\partial \boldsymbol{a}}\Bigg|_{\boldsymbol{a}^*, \boldsymbol{B}^{(1)}} + \boldsymbol{\mu}^*_{\mathcal{A}_{\widetilde{\delta}}}(\boldsymbol{B}^{(1)})^{\top} \frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\delta}}}}{\partial \boldsymbol{a}}\Bigg|_{\boldsymbol{a}^*, \boldsymbol{B}^{(1)}} - \boldsymbol{\xi}^*_{\mathcal{A}_{\underline{\boldsymbol{a}}}}(\boldsymbol{B}^{(1)})^{\top} \boldsymbol{I}_{\mathcal{A}_{\underline{\boldsymbol{a}}}} = 0.$$

where the gradients are replaced by their value in order to compute the Lagrange multipliers, that are:

$$\boldsymbol{\mu}^*_{\mathcal{A}_{\boldsymbol{\delta}}}(\boldsymbol{B}^{(1)}) = 8.05\ kg/mm,$$
$$\boldsymbol{\xi}^*_{\mathcal{A}_{\underline{\boldsymbol{a}}}}(\boldsymbol{B}^{(1)}) = \begin{pmatrix} 2.86e^{-3} \\ 2.86e^{-3} \end{pmatrix}\ kg/mm^2.$$

Again, the values show that the problem optimum is much more sensitive to the constraint on displacements than the bound constraints. The gradients of the weight and the displacement constraints w.r.t. $\boldsymbol{B}$ are computed, respectively:

$$\left.\frac{\partial \widetilde{w}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(1)}} = [0.4, 0.39, 0.63, 4.96, 4.9, 7.8, 0.4, 0.39, 0.63]\,,$$

$$\left.\frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\boldsymbol{\delta}}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(1)}} = \left[-1.2e^{-2}, -1.3e^{-2}, -1.9e^{-2}, -0.63, -0.66, -0.97, -1.2e^{-2}, -1.3e^{-2}, -1.9e^{-2}\right].$$

These values are replaced in the following equation (adapted from equation 4.9) in order to compute the gradient of $\widetilde{\Psi}$:

$$\left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(1)}} = \left.\frac{\partial \widetilde{w}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(1)}} + \boldsymbol{\mu}^*_{\mathcal{A}_{\boldsymbol{\delta}}}(\boldsymbol{B}^{(1)})^\top \left.\frac{\partial \widetilde{\boldsymbol{\delta}}_{\mathcal{A}_{\widetilde{\boldsymbol{\delta}}}}}{\partial \boldsymbol{B}}\right|_{\boldsymbol{a}^*,\boldsymbol{B}^{(1)}}.$$

Its numeric value is:

$$\left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(1)}} = [0.29, 0.29, 0.47, -0.1, -0.37, 0., 0.29, 0.29, 0.47]\,.$$

The history of the previous iterations is updated with $\boldsymbol{B}^{(1)}$ such that:

$$K^{(1)} = K^{(0)} \cup \{\boldsymbol{B}^{(1)}\}.$$

The MILP problem (MILP($K^{(k)}$)) can now be set up, as follows:

$$\begin{aligned}
\min_{\boldsymbol{B}\in\mathcal{C}^{3,3}} \quad & \eta \\
\text{subject to} \quad & \eta \le U^{(1)} - \epsilon \\
& \eta \ge \widetilde{\Psi}(\boldsymbol{B}^{(1)}) + \left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(1)}}^\top (\boldsymbol{B} - \boldsymbol{B}^{(1)}) \\
& \eta \ge \widetilde{\Psi}(\boldsymbol{B}^{(0)}) + \left.\frac{d\widetilde{\Psi}}{d\boldsymbol{B}}\right|_{\boldsymbol{B}^{(0)}}^\top (\boldsymbol{B} - \boldsymbol{B}^{(0)}) \\
& \boldsymbol{B} \circ (\boldsymbol{B} - \mathbf{1}) = \mathbf{0}_9
\end{aligned} \tag{MILP($K^{(1)}$)}$$

The problem (MILP($K^{(1)}$)) does not admit feasible solutions.

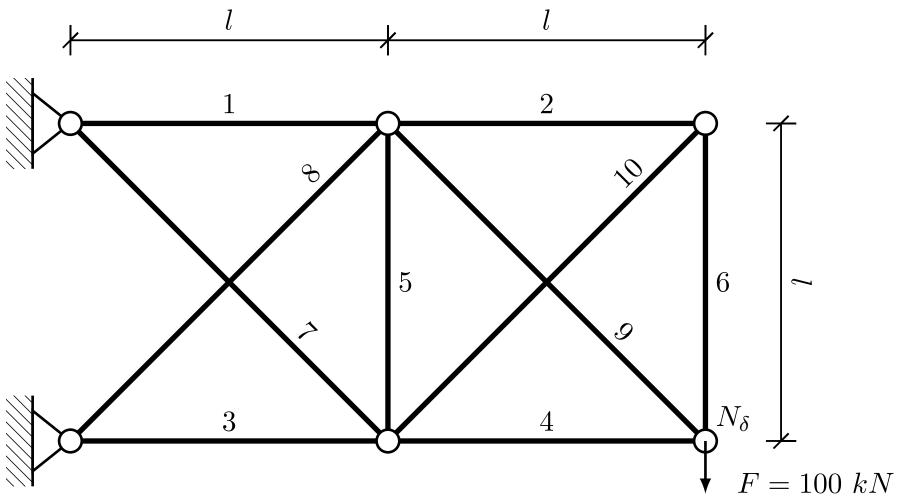


Figure 4.3: 10-bar truss, seen as a scalable 2D cantilever problem with 2 blocks.

The algorithm stops, and the optimum is such that:

$$\begin{aligned} \widetilde{w}^* &= U^{(1)} = 8.63\ kg \\ \boldsymbol{a}^* &= \boldsymbol{a}^{(1)} = [100,\ 1770.61,\ 100]\ mm^2, \\ \boldsymbol{B}^* &= \boldsymbol{B}^{(1)} = \text{vec}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{aligned} \tag{4.16}$$

corresponding to the following optimal catalogs selection, with $\gamma = [1,2,3]^\top \in \Gamma^3$:

$$\boldsymbol{c}^* = \boldsymbol{B}^* \gamma^\top = [2,3,2]^\top, \tag{4.17}$$

in other terms the optimal material for elements 1 and 3 is AL2139 and TA6V.

It can be remarked that after the first outer iteration, that costs 1 NLP and 1 MILP, the optimum is found. During the entire process, 2 NLP have been solved, and 2 MILP. Solving this problem by enumeration (Baseline) would have necessitate 3^3 NLP optimizations, while the Bi-level algorithm required 17 NLP optimizations. The optimum found by Bi-level OA is the same than the one obtained by Baseline and Bi-level algorithms.

4.4.3 A 10-bar truss structure

This well-known low-dimensional 10-bar truss problem (Haftka and Gürdal 1992) is used to solve the mixed categorical-continuous optimization problem by enumeration, Bi-level or hybrid branch and bound (h-B&B) (Barjhoux et al. 2018b). As explained in subsection 4.4.1, these approaches provide global solutions, that are taken as reference solutions to evaluate quality solutions of the Bi-level OA algorithm.

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$1300\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$1300\ mm^2$

Table 4.3: Bounds on areas, and initial areas values of the 10-bar truss optimization case.

Table 4.4: Results of 10-bar truss mixed optimization with 5 different values of constraint on displacements. Comparison between the Bi-level OA, Bi-level, the Baseline solutions obtained by enumeration of the 2^{10} continuous optimizations, h-B&B, and the Genetic algorithm. The catalog 1 corresponds to material AL2139 and catalog 2 to TA6V.

$\bar{\boldsymbol{u}}$ (mm)	Baseline		h-B&B		Genetic		Bi-level		Bi-level OA	
	$\boldsymbol{c}^* = \boldsymbol{B\gamma}$	$w^*(kg)$	d_h	$w^*(kg)$	d_h	$w^*(kg)$	d_h	$w^*(kg)$	d_h	$w^*(kg)$
-22	[2,2,1,1,1,2,2,1,2,1]	12.988	0	12.988	0	13.283	0	12.988	0	12.988
-20	[2,1,1,1,1,1,2,1,1,1]	13.996	0	13.996	0	14.423	0	13.996	0	13.996
-19	[2,1,1,1,1,1,2,1,1,1]	14.570	0	14.570	0	14.802	0	14.570	0	14.570
-18	[1,1,1,1,1,1,1,1,1,1]	15.175	0	15.175	2	15.642	0	15.174	0	15.174
-17	[1,1,1,1,1,1,1,1,1,1]	15.912	0	15.912	3	16.258	0	15.912	0	15.912

The 10-bar truss problem is illustrated Figure 4.3. A downward load $F = 100\ kN$ is applied vertically on node N_δ. A constraint on displacements is applied on the same node. Five cases with different bounds values $\bar{\boldsymbol{u}}$ on displacements are considered. For each of these cases, the displacements constraint is applied on node N_δ. The upper and lower bounds, and initial areas, are fixed as detailed in Table 4.3. Catalogs 1 and 2 point to materials AL2139 and TA6V, respectively. Materials properties are listed in Table 4.1. For this simple case, one has $n = 10$, $p = 2$, and $\boldsymbol{B} \in \mathcal{C}^{10,2}$. In its direct coding version, $C^{10,2}$ corresponds to $\Gamma^{10} = \{1,2\}^{10}$.

The results of the proposed methodology (Bi-level OA) are thus compared to the global optima, as shown in Table 4.4. In all these cases, the optima obtained with `Baseline` (obtained by enumeration), h-B&B, Bi-level and the Bi-level OA approaches are identical. This means that the Bi-level OA, in these cases, provides the global solution. On the other hand, the weights returned by the Genetic algorithm are greater than the optimal weight found by the Bi-level approach. Finally, it is shown that when the displacement constraint becomes more stringent, the material choice goes to the stiffest one despite of its high density. The optimal solutions of cases with displacements lower than $18mm$ and $17mm$ contain indeed only TA6V.

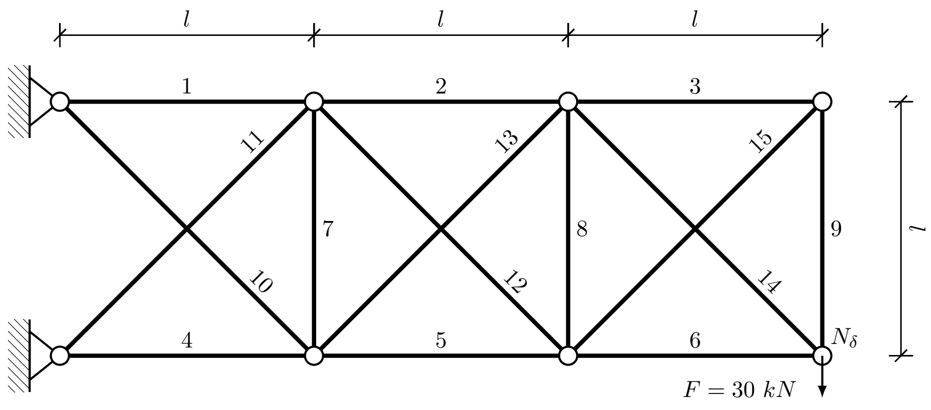


Figure 4.4: An example of 2D cantilever problem with 3 blocks.

4.4.4 On the scalability of the proposed methodology

4.4.4.1 Scalability with respect to the number of elements

The objective of this test case is to describe the evolution of the computational cost with respect to the number of structural elements. This case can be seen as a generalization of the well-known 10-bar truss structure (Haftka and Gürdal 1992). It has been used in the literature to demonstrate the scalability of algorithms, for example in (Shahabsafa et al. 2018). The structure is made scalable by varying the number of blocks. Each block is composed of 4 nodes that are linked by 5 bars. An example of a scalable 2D cantilever structure with 3 blocks is given in Figure 4.4. In Table 4.5 are presented the results obtained with structures composed of 1 to 10 blocks. In all cases, a downward load $F = 30\ kN$ is applied on the node N_δ. The bounds on areas, and initial areas are fixed as detailed in Table 4.6.

#bars	Baseline	h-B&B				Genetic				Bi-level				Bi-level OA			
	$w^*(kg)$	d_h	$w^*(kg)$	#iter	#FEM	d_h	$w^*(kg)$	#iter	#FEM	d_h	$w^*(kg)$	#iter	#FEM	d_h	$w^*(kg)$	#iter	#FEM
5	2.56	0	2.56	10	1004	0	2.57	32	32300	0	2.56	2	400	0	2.56	2	96
10	6.06	0	6.06	26	3097	1	6.14	54	54500	0	6.06	2	792	0	6.06	2	181
15	10.23	0	10.23	95	10907	2	10.27	65	65200	0	10.23	4	1955	0	10.23	6	967
20	†	†	15.33	135	10315	†	15.59	73	73100	†	15.33	2	1659	†	15.33	7	1023
25	†	†	21.36	1199	610347	†	22.06	98	97700	†	21.36	3	3142	†	21.36	13	2312
30	†	†	28,30	4432	723388	†	28.84	129	128800	†	28.30	8	10522	†	28.30	13	2991
35	†	†	$36,17^{(*)}$	$5793^{(*)}$	$1096968^{(*)}$	†	37.00	189	189400	†	36.19	3	5830	†	36.19	6	1496
40	†	†	$44,97^{(*)}$	$5570^{(*)}$	$939726^{(*)}$	†	45.64	270	269800	†	44.97	7	13577	†	44.96	40	11578
45	†	†	$54,70^{(*)}$	$4181^{(*)}$	$818455^{(*)}$	†	55.98	347	346800	†	54.71	4	8531	†	54.67	20	6789
50	†	†	$65,35^{(*)}$	$4316^{(*)}$	$717627^{(*)}$	†	67.48	561	561200	†	65.34	6	14487	†	65.34	42	13290

Table 4.5: A comparison of the obtained solutions for 10 instances of the scalable 2D cantilever problem are compared, with a varying number of bars (from 5 to 50 bars). We note that when optimizations last more than 24 hours, the solver (Baseline, h-B&B) is stopped and the current solution (if exists) is marked by $(*)$. When reference solutions (Baseline) are not available, optimal weights are noted by †, as well as the distances d_h to these solutions.

$\underline{\boldsymbol{a}}$	$100\ mm^2$
$\bar{\boldsymbol{a}}$	$2000\ mm^2$
$\boldsymbol{a}_{\text{ini}}$	$2000\ mm^2$

Table 4.6: Bounds on areas, and initial areas values of the 2D cantilever truss optimization case.

For each of the 10 cases, the results obtained by the Bi-level OA are compared to those obtained with reference solutions (Baseline, h-B&B) & Bi-level when available. First, for cases with 5 to 30 elements where a reference solution is available, it can be observed the global solution is found by the Bi-level OA. For cases with more than 30 elements, the optima found by the Bi-level OA are slightly better than those obtained by the Genetic algorithm. The h-B&B solutions are noted with (*) since they are intermediate solutions : the solver was stopped after 24 hours. The Bilevel OA solutions are very close (difference of 10^{-2} kg) to those obtained by the h-B&B. For cases with 40 and 45 elements, the Bilevel OA solutions are slightly lighter than the Bilevel. Furthermore, the number of analyses required by Bi-level OA is always lower than the number needed by the compared approaches, including Bi-level. The trends in terms of computational cost with respect to the number of elements are graphically represented in Figure 4.5. The cost of the Genetic algorithm dominates the cost of h-B&B and Bi-level. As with the Bilevel, the scaling of the Bi-level OA approach is nearly linear when compared to the h-B&B and Genetic approach. The trends in terms of Bi-level OA computational cost with respect to the number of elements are similar to the Bi-level computation cost. The observed efficiency makes the proposed approach relevant for higher dimensional problems.

4.4.4.2 Scalability with respect to the number of catalogs

The objective of this test case is to describe the evolution of the computational cost with respect to the number of categorical choices. The test case is the same 10-bar truss case presented in Section 4.4.3, with a constraint on displacements such that $\bar{\boldsymbol{u}} = 22mm$. For this simple case, one has $n = 10$, but p is varying from 5 to 90 catalogs. The number of available categorical choices combinations ranges thus from 10^4 to 10^{90}.

In Table 4.7 are presented the results obtained by Bi-level OA and Bi-level. The optimal weight, the number of iterations (#ite), non-linear problems (#NLP) solved, and the number of individual calls to the structural solver (#FEM) are compared. First, in terms of #FEM and #NLP, the computational cost of Bi-level OA reveals to be quasi-independent from the number of categorical values when compared to Bi-level. Furthermore, it is shown that for each case, the optimal weights obtained by both solvers are identical ($< 10^{-3}kg$). Provided that there is an optimality proof of the result returned by Bi-level OA (under the convexity assumption of $\widetilde{\Psi}$), this shows that the Bi-level also returns good quality solutions even in cases with a large categorical design space. Independently from the solver, it can be remarked that the optimal weights are identical from cases 4 to 36, and 45 to 90. This is due to the fact that the categorical values introduced in the categorical design space from case 4 to 36 (or from 45 to 90) do not significantly improve the optimal weight. An improvement is observed from case 45 to case 72. The trends in terms of computational cost with respect to the number of catalogs are depicted in Figure 4.6.

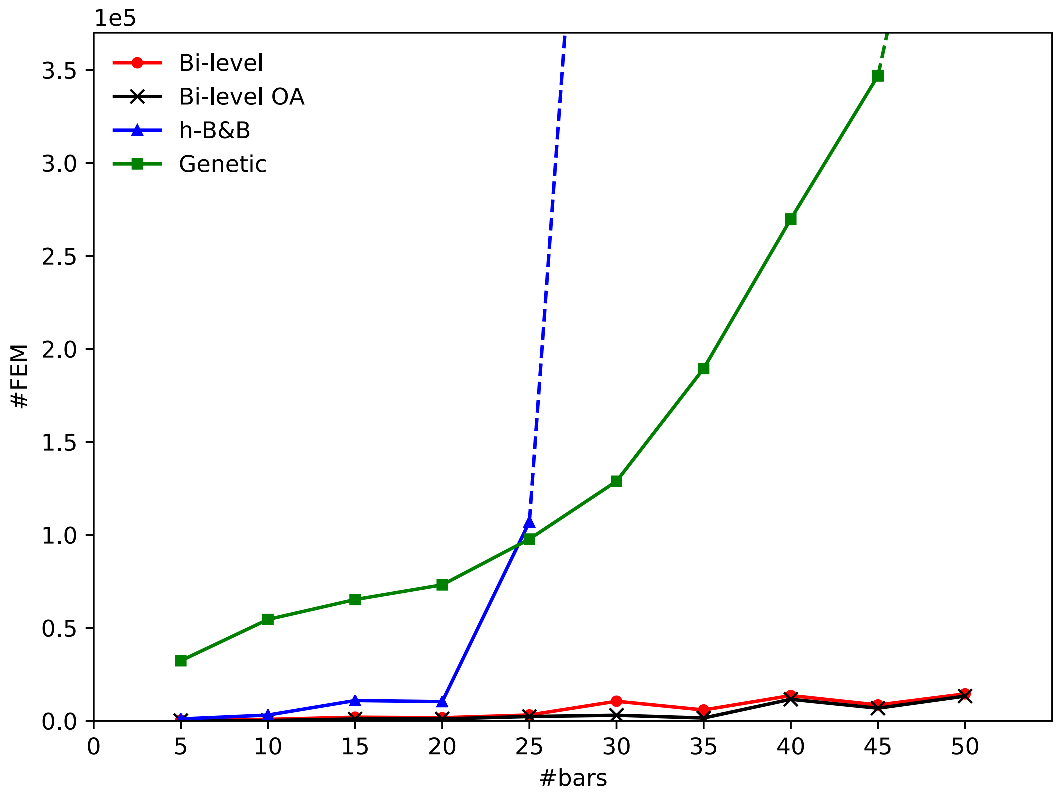


Figure 4.5: Scalability of the Bi-level OA w.r.t. the number of structural elements. The computational cost's scaling of Bi-level OA and Bi-level with respect to the number of bars is quasi-linear, compared to the exponential computational cost of the h-B&B and Genetic solvers. The computational cost's scaling of the h-B&B prevents from obtaining a solution for cases greater than 25 elements.

4.4.5 120-bar truss

In this example, the structure of a 120-bar dome truss (Saka and Ulker 1992) is considered and described in Figure 4.7. In this case, $n = 120$ and $p = 90$, $\Gamma^{120} = \{1, ..., 90\}^{120}$ such that $\mathcal{C}^{120,90}$. There is no grouping of elements, meaning that the design space counts 120 categorical design variables and 120 continuous design variables. For each element, the categorical variable can take a value among 90 catalogs, that point to combinations of materials AL2139, AL2024 and TA6V, and I, T and C-profiles (10 different sizes each). Materials properties are listed in Table 4.1. The structure is subjected to an active constraint on displacements : a maximum downward displacement of 10 mm is allowed. A downward load of 60 kN is applied on node 1, while a downward load of 30 kN is applied on nodes 2 to 13 and 10 kN on nodes 14 to 37. The number of available categorical choices is thus equal to 90^{120}. The bounds on areas, and initial areas are fixed as detailed in Table 4.8.

The optimal weight returned by Bi-level OA is 2501 kg. The convergence history is depicted

#catalogs	Bi-level				Bi-level OA			
	$w^*(kg)$	#iter	#NLP	#FEM	$w^*(kg)$	#iter	#NLP	#FEM
4	12.99	3	98	29245	12.99	84	84	8400
9	12.39	4	334	98871	12.39	89	89	3772
12	12.39	4	445	131358	12.39	61	61	2583
15	12.39	4	573	170407	12.39	45	45	1955
18	12.39	4	708	209201	12.39	65	65	2877
36	12.39	4	1404	416662	12.39	57	57	2232
45	12.15	3	1348	399172	12.15	69	69	2489
72	12.15	3	1864	551042	12.15	64	64	2898
90	12.15	3	2704	799166	12.15	86	86	3952

Table 4.7: A comparison of the obtained solutions for 9 instances of the 10-bar truss problem are compared, with a varying number of catalogs (from 4 to 90 catalogs).

$\underline{\boldsymbol{a}}$	100 mm^2
$\bar{\boldsymbol{a}}$	6000 mm^2
$\boldsymbol{a}_{\text{ini}}$	6000 mm^2

Table 4.8: Bounds on areas, and initial areas values of the 120-bar truss optimization case.

Figure 4.8. The optimization is converged after 58 iterations. This means that it required to solve 58 NLP (primal problems), with a total of 33957 calls to FEM. It can be observed that the optimum is reached at the end of the 5^{th} iteration. In other terms, the remaining 53 iterations serve to prove that the best known solution so far is the optimum. This result can be compared with the theoretical computational cost of the Bi-level in Chapter 3. Solving this 120-bar truss test case with Bi-level would have required to solve a minium of $1+120\times(90-1) = 10680$ NLPs, equivalent to the computational cost of one iteration of Bi-level. It is also worth to note that in an industrial context, it is not always necessary to wait for the end of the optimization. The gap between the upper and lower bound provides indeed information, at each iteration, on what would be the best expected gain in weight. If the expected gain does not worth to wait for the end of the optimization, the optimization can be stopped. If this threshold is known beforehand, it can be assigned to ϵ in Algorithm 6.

4.5 Conclusion

The optimization problem tackled by the proposed algorithm is a large scale mixed categorical-continuous optimization problem (P), where the categorical variables are non-relaxable and

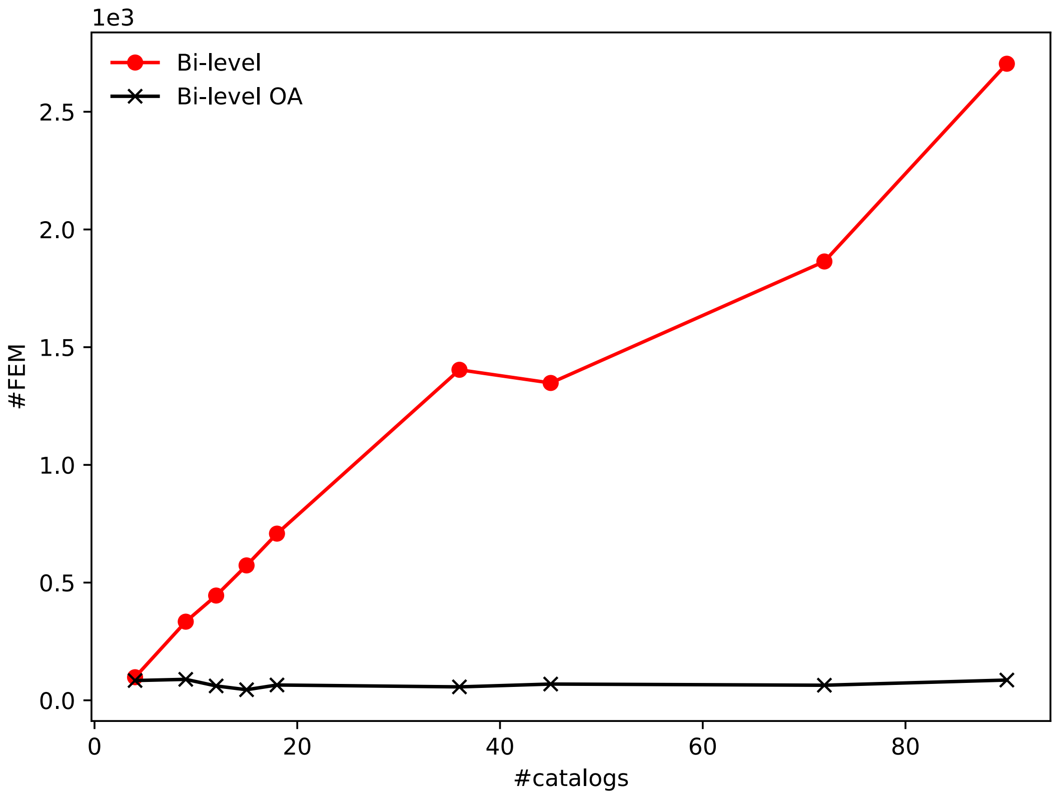


Figure 4.6: Scalability of the Bi-level OA w.r.t. the number of catalogs. The computational cost's scaling of Bi-level OA with respect to the number of catalogs is nearly independent from the number of catalogs, compared to the quasi-linear computational cost of the Bi-level.

non-ordered. It is first formulated as a mixed large scale integer-continuous optimization problem (BP), where the integer variables are relaxable and non-ordered. The objective and constraints functions of (P) are also reformulated such that it is possible to compute their gradients with respect to all the design variables. The proposed algorithm uses a bi-level decomposition of (BP), and solves an iterative sequence of master and slave problems. The discrete decisions are driven by the master problem, that involves linearizations of the slave problem result. However, unlike in Chapter 3, the linearizations are not treated as gradients that provide a search direction. In the proposed algorithm, they serve to bound iteratively the convex hull of the slave problem solution parameterized in the integer variables. They are involved as supporting hyperplanes of the slave problem. They are efficiently computed thanks to a post-optimal sensitivity analysis. The resulting algorithm relies on the theory of the OA algorithm (Fletcher and Leyffer 1994; Bonami et al. 2008; Grossmann 2009). Under the convexity assumption of the slave problem result w.r.t. the integer variables, the solution found is proved to be the optimum. The numerical tests show that Bi-level OA is capable of handling large scale instances of the mixed categorical-continuous problem (P). For example a problem with 120 structural elements is presented, involving 120 categorical variables as well as 120 continuous ones, and a categorical design space with 90 values per variable. The

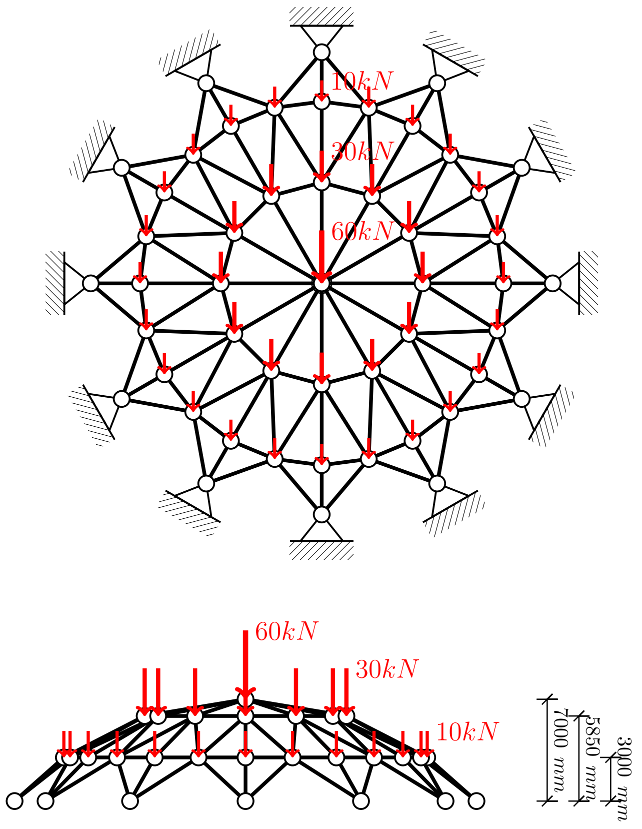


Figure 4.7: Top and side view a 120-bar truss structure. Downward loads with three different magnitudes are applied.

optimal solution is obtained after 58 iterations, where 58 NLP are solved. The scalability of the algorithm, in terms of computational cost, has been tested with respect to the number of elements and number of categorical variables. In the test cases investigated in this work, the computational cost is quasi-linear with respect to the number of structural elements. It is also nearly independent to the available number of values in the categorical design space. Finally, the quality of the optimum has been evaluated, for low dimensional cases where a reference solution exists. It has been shown that the solutions returned by the proposed algorithm are identical to the available reference solutions. Compared to the OA applied to a mono-level optimization problem formulation, the proposed algorithm generates MILP problems where only one constraint (hyperplane of lower level solution) is added iteration per iteration. In an industrial application where the original problem counts thousands of constraints, the MILP problem generated by the OA would rapidly count a huge number of constraints (number of iterations times the number of linearized objectives and constraints functions of the original problem). This is one drawback of the OA algorithm according to (Benson and Horst 1991; Stolpe and Sandal 2018). The proposed methodology alows to overcome this issue, thanks to the bi-level formulation of the origial problem, and the efficiency of the post-optimal sensitivity analysis to set up the linearizations. Despite the aforementioned encouraging results, it is also worth to note that there is no proof of the proposed algorithm complexity. This is one drawback shared by the algorithms that belong to the family of the tree-based solvers. In

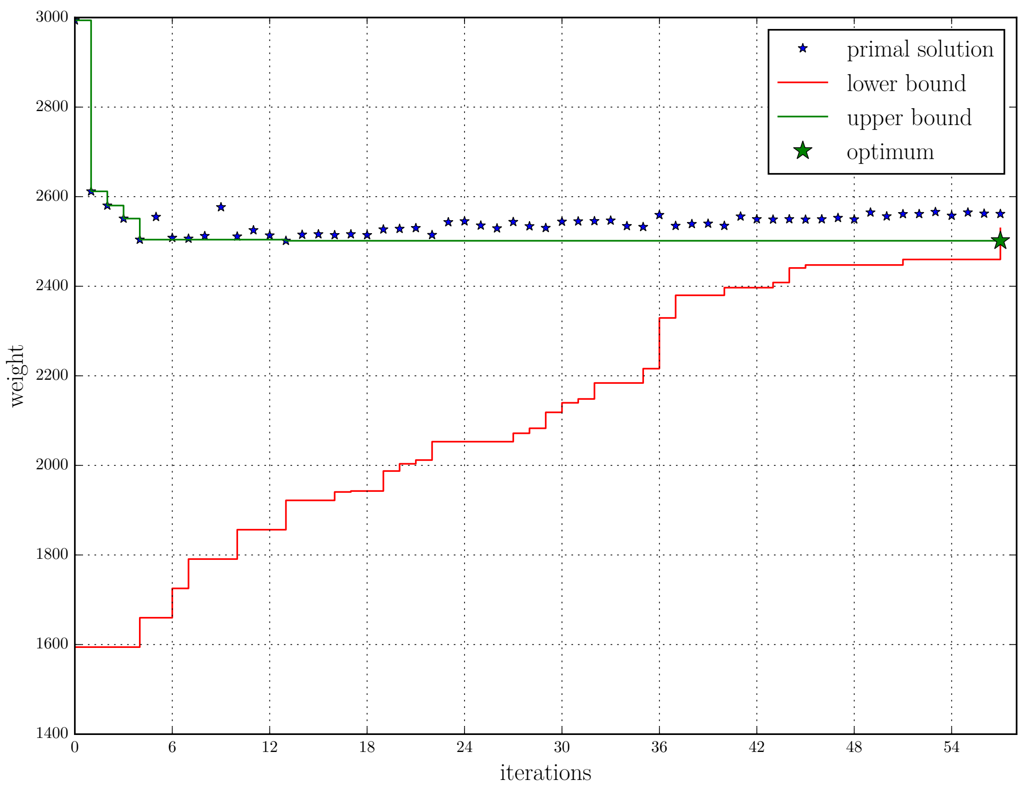


Figure 4.8: History of the convergence of Bi-level OA when solving a 120-bar truss problem instance with 90 possible choices per structural element, so that $\boldsymbol{B} \in \mathcal{C}^{120,90}$.

the other hand, the optimality proof relies on the assumption of the slave problem solution convexity with respect to the design variables. Although no non-convex cases have been encountered so far, a further work could consist to implement a strategy when such cases occur.

In this Chapter, the following items have been discussed:

- The original mixed categorical-continuous optimization problem is reformulated as a mixed integer-continuous optimization problem, with continuously derivable functions,
- The mixed integer-continuous optimization problem is then reformulated as a bi-level optimization problem, where the integer design variables are handled at the master level
- Linearizations of the slave problem results are involved in the master problem as outer approximation cuts,
- Under convexity assumptions, the proposed algorithm returns the exact optimum,
- The numerical tests show that the algorithm computational cost is quasi-linear with respect to the number of structural elements, and nearly independent from the number of categorical values,
- A 120-bar optimization case with 90 available combinations of material and stiffening principle per structural element has been solved, and required to solve 58 non-linear optimizations.

Chapter 5

Conclusions and perspectives

In this work, large scale structural optimization problems involving both non-ordinal categorical and continuous design variables were investigated. The aim was to minimize the weight of a truss structure with respect to the cross-section areas, with optimal materials and stiffening principles selection. The targeted industrial structure counts hundreds if not thousands of structural elements, with dozens of possible choices of materials and cross-section types. In order to handle this kind of problems, three methodologies have been proposed.

In the first Chapter, an algorithm based on the Branch & Bound framework was presented. It has to be noted that categorical optimization is usually not within the scope of Branch & Bound algorithms. The novelty of the proposed methodology mainly lied in the formulation of an original problem specific relaxation. The relaxation consisted of a full continuous problem definition where the involved functions are underestimators of the original problem objective and constraints. This bounding technique also came with a specific multiway branching. Pieced together into the Branch & Bound framework, these strategies allowed to solve the original problem to optimality in a finite number of steps, despite the categorical nature of the discrete variables. The numerical experiments confirmed that the computed optima are the exact ones, for reference (low dimensional) cases. The main drawback of the proposed methodology was its poor scalability with respect to the dimension of the design space. In fact, although the computational cost of the proposed method remained attractive for small test cases, its scalability with respect to the dimension is exponential and hence not compatible with large scale optimization problems.

In the second Chapter, the methodology relies on the same general idea when it comes to solve combinatorial optimization problems. Similarly to the proposed B&B, we proposed to simplify the optimization problems on a sequence of easy-to-solve sub-problems. However, this second methodology did not build a solution tree of non-linear sub-problems (whose exploration has proved to be computationally expensive). No relaxation problem was neither needed. In fact, the problem was reformulated using a bi-level decomposition involving master and slave problems. The continuous design variables were handled by the slave problem, while the categorical variables were driven by the master. Such decomposition allowed to leverage the efficiency of gradient-based optimization to solve the slave (sizing) problems. In the master problem, a first order-like approximation of the slave problem, with respect to the categorical design variables, allowed to overcome the combinatorial explosion of the computational cost required to get the candidate solutions. The numerical results demonstrated the relevance of such approximation. The approach revealed to be very competitive in terms of both results quality and computational cost. The scaling of the computational cost was indeed quasi-linear with respect to the number of structural elements. Particularly, this second approach allowed to solve problems that are very hard to solve with standard algorithms, typically test cases with more than one hundred categorical design variables. However in the frame of industrial design problems, this competitive methodology would still require a large number of sizing optimizations. Furthermore, there was no optimality proof. It was then still needed to improve the computational efficiency, while preserving the result quality as much as possible.

In the third Chapter, the proposed methodology is based on the conclusions of the two aforementioned frameworks. The first conclusion is related with the single tree based approaches (such as B&B) where one can not handle nonlinear problems with a large number of design variables. The second conclusion is that the first order approximation involved in the bi-level approach allows to solve efficiently medium to large scale instances of the original problem. However, this first order approximation was used to build a search direction, although there is no proof that it will lead to a weight decrease. This search direction was also at the origin of most of the computational effort. Thus in this third methodology, first order approximations were involved as linear cuts (instead of search directions). Furthermore, they are efficiently built based on post-optimal sensitivity analyses, that are usually part of continuous multi-level optimization schemes. The master problem turned out to be a mixed integer linear problem, that was efficiently solved by branch and cut algorithms. The resulting algorithm can be seen as a specific instance of the well-known outer approximation algorithm, hence the solutions obtained by our algorithm are provably optimal. The numerical results confirmed the interest of combining bi-level optimization with post-optimal analysis and linearizations of the slave problem. The scaling of the methodology was quasi-linear with respect to the number of structural elements, and quasi-independent from the number of categorical values. This third algorithm outperforms the two aforementioned approaches. The main drawback of this method lies in the fact that there is no exact estimation of the algorithm complexity. It is not possible to get a prior assessment of the computational cost before solving a problem. However, the approach was proved to be competitive in the considered set of large scale problems.

Perspectives

The presented work offers many perspectives, for example in terms of improvements of the proposed methodologies with the aim of solving the same initial problem more efficiently. Indeed, the bi-level methodology with the first-order approximation has shown that the convergence is reached after a very few number of outer iterations. However, the computational cost of each of these iterations can be prohibitive in large scale problems. If it is reduced, this would result in an algorithm that could be competitive when compared to the outer approximation based bi-level approach. To that purpose, it could be interesting to find a way to leverage the post-optimal sensitivities computation efficiency (as in the outer approximation based formulation) in the frame of this algorithm. Also, the outer approximation based bi-level could benefit from multiple improvements. Indeed, it has been shown that the master problem does not always ensure a decrease of the slave problem optimal weight. This is due to the fact that the accuracy of post-optimal sensitivities is impacted (especially in the context of integer variables) by changes in the set of active constraints. This is why the proposed algorithm could be even more efficient if for example quadratic information on the result of the lower problem could be added to the linearization. Another solution could consist of replacing erroneous post-optimal sensitivities components by values obtained by finite differences, as in the bi-level methodology proposed in the second Chapter. These coefficients could be computed in parallel, and may improve the algorithm efficiency.

Also, even if the conclusions of the outer approximation based bi-level framework are promising, the numerical tests involved bar elements (tension-compression) that do not embed the complexity of a full finite element model as used in the industry. First, there is no influence of the cross-section stiffening principles in the bar. This means that the Young modulus was the only categorical-related (relaxable and ordered) physical feature involved by the internal loads computation. In the full FEM industrial model, the methodology will need to deal with the cross-section stiffening principle choices as well. Second, one of the the requirements of the outer approximation bi-level framework optimality proof is the convexity of the result of the slave problem. A special attention shall be paid to the fulfillment of that assumption in the industrial case, if the problem needs to be solved to optimality. The use of convexification (e.g., by means of convex under-estimators) approaches could be part of a further work, with a view to handle non-convex cases.

In addition, the results of the outer approximation based framework are a motivation for extending the scope of the initial problem. One first perspective is to add composite blending constraints and manufacturing constraints to the structural optimization problem. For example, it could consist of defining inter-element constraints that maintain continuity of some ply orientation angles across adjacent structural elements. The proposed framework can indeed natively handle this kind of constraints on the categorical design variables at the upper level of the bi-level decomposition. Furthermore, if formulated as linear functions, these constraints will be treated exactly by the outer approximation framework. In other terms, this will not affect the convergence of the algorithm. In the other hand, it could be interesting to draw a path between the initial optimization problem and topology optimization. Since a categorical design variable could also encode the absence or presence of an element among those of a ground structure, an extension of the proposed methodology to solve a larger structural optimization problem (including multi-material, shape and topology optimization) could be considered. It could also be possible to add the manufacturing cost of a structure in the objective of the optimization problem. Finally, the outer approximation based bi-level derived in this work could be applied to different mixed optimization problems arising in engineering optimization, including other fields than structural optimization, e.g., control systems, chemistry, route planning, aerodynamics, or multidisciplinary design optimization.

Appendix A

Differentiable optimization theory

This Chapter aims to recall bases of continuous optimization on which the presented methodologies will rely on. For this purpose, let define a generic problem (P) that is parameterized in $\boldsymbol{p}$, a real-valued vector :

$$\begin{aligned} \Psi(\boldsymbol{p}) \;:= \underset{\boldsymbol{x}\in\mathcal{S}}{\text{minimize}} \quad & f(\boldsymbol{x},\boldsymbol{p}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x},\boldsymbol{p}) \le 0 \\ & \boldsymbol{h}(\boldsymbol{x},\boldsymbol{p}) = 0 \end{aligned} \tag{P}$$

In this problem, objective and constraints are depending on both $\boldsymbol{x}$ and $\boldsymbol{p}$, but the optimization is performed w.r.t. the design variables $\boldsymbol{x}$, while $\boldsymbol{p}$ remains unchanged during the optimization. Ψ is a nested function, since its evaluation triggers a continuous optimization.

First, define the Lagrange multipliers as $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ associated to inequality and equality constraints $\boldsymbol{g}$ and $\boldsymbol{h}$, respectxively. The Lagrangian of the problem (P($\boldsymbol{p}$)) is defined as follows :

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{p}) := f(\boldsymbol{x},\boldsymbol{p}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x},\boldsymbol{p}) + \boldsymbol{\mu}^\top \boldsymbol{g}(\boldsymbol{x},\boldsymbol{p})$$

A.0.0.1 Optimality conditions

For sake of brevity, in this Section the dependance of the functions w.r.t. $\boldsymbol{p}$ in problem (P) is skipped.

Let introduce the constraints regularity conditions, also called constraint qualifications. Given a point $\boldsymbol{x} \in \mathcal{S}$, let name $\mathcal{A}_{\boldsymbol{g}}(\boldsymbol{x})$, $\mathcal{A}_{\boldsymbol{h}}(\boldsymbol{x})$ the sets of active constraints $\boldsymbol{g}$ and $\boldsymbol{h}$, respectively. The constraints satisfy the Linear Independence Constraints Qualification, named (**LICQ**), if given a point $\boldsymbol{x}$, the active constraints gradients is linearly independent, if there exists $\boldsymbol{\alpha}$ such that :

$$\begin{aligned} \sum_{i\in\mathcal{A}_g} \boldsymbol{\alpha}_i \nabla \boldsymbol{g}_i(\boldsymbol{x}) = 0 &\implies \boldsymbol{\alpha}_i = 0, \forall i \in \mathcal{A}_{\boldsymbol{g}}(\boldsymbol{x}), \\ \sum_{j\in\mathcal{A}_{\boldsymbol{h}}} \boldsymbol{\alpha}_j \nabla \boldsymbol{h}_j(\boldsymbol{x}) = 0 &\implies \boldsymbol{\alpha}_j = 0, \forall j \in \mathcal{A}_{\boldsymbol{h}}(\boldsymbol{x}) \end{aligned} \tag{\textbf{LICQ}}$$

The Karush–Kuhn–Tucker (KKT) conditions are first order necessary conditions that guarantee a solution $\boldsymbol{x}^*$ to be a local optimum of a non-linear problem (P).

Theorem A.1 (KKT necessary optimality conditions (Karush 1939; Kuhn and Tucker 1951))
*Suppose that f, $\boldsymbol{g}$ and $\boldsymbol{h}$ are continuously differentiable at a point $\boldsymbol{x}^*$. If $\boldsymbol{x}^*$ is a local optimum, and $\boldsymbol{g}$ and $\boldsymbol{h}$ verify the constraints qualifications (**LICQ**), then there exists $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ called Lagrange multipliers, such that :*

- *(**Stationarity**) $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a stationary point of the Lagrangian w.r.t. $\boldsymbol{x}$:*

$$\left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*} + \boldsymbol{\lambda}^{*\top} \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*} + \boldsymbol{\mu}^{*\top} \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*} = 0 \tag{A.1}$$

- *(**Primal feasibility**) Constraints are satisfied at $\boldsymbol{x}^*$*

$$\boldsymbol{g}(\boldsymbol{x}^*) \leq 0, \;\; \boldsymbol{h}(\boldsymbol{x}^*) = 0 \tag{A.2}$$

- *(**Dual feasibility**) The Lagrange multipliers related to the inequality constraints are null or positive :*

$$\boldsymbol{\lambda}^* \geq 0 \tag{A.3}$$

- *(**Complementary slackness conditions**) The Lagrange multipliers related to the inequality constraints satisfy :*

$$\boldsymbol{\lambda}^{*\top} \boldsymbol{g}(\boldsymbol{x}^*) = 0 \tag{A.4}$$

In general, these first order necessary conditions are not sufficient for optimality. Other conditions have to be statisfied, such as the Second Order Sufficient Conditions (**SOSC**) given by the Theorem (A.2).

Theorem A.2 (Second order sufficient optimality conditions (McCormick 1967))
*If the following Second Order Sufficient Conditions, named (**SOSC**), hold :*

- *there exist $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ such that KKT conditions (A.1, A.2, A.3, A.4) hold,*
- *$\forall \boldsymbol{z} \neq 0$ such that :*

$$\begin{gathered} \boldsymbol{z}^\top \left.\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*} \boldsymbol{z} > 0 \\ \left[\left.\frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*}, \left.\frac{\partial \boldsymbol{h}_j}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*} \right]^\top \boldsymbol{z} = 0, \;\; s.t. \; \lambda_i > 0 \; (\forall i \in \mathcal{A}_{\boldsymbol{g}}) \end{gathered} \tag{SOSC}$$

then $\boldsymbol{x}^*$ is an isolated local (unique locally) optimum of problem (P).

A.0.0.2 Post-optimal sensitivities of continuous problems

In continuous optimization, many efficient algorithms rely on the gradient of the functions involved in the optimization problem. Suppose that a gradient-based algorithm is used to minimize Ψ with respect to the real-valued vector $\boldsymbol{p} \in \mathbb{R}^n$. The gradient of Ψ with respect to the parameters $\boldsymbol{p}$ at $\boldsymbol{p}' \in \mathbb{R}^n$ has thus to be provided to the optimization algorithm. This gradient gives information on the behavior of the optimal objective value, solution of (P), after a small perturbation of $\boldsymbol{p}'$. This is why the gradient of Ψ is also called *post-optimal sensitivity.*

Let be $\boldsymbol{x}^*(\boldsymbol{p})$ the optimal solution of (P). Let define $\mathcal{A}_{\boldsymbol{g}}$ the set of active inequality constraints, such that

$$\mathcal{A}_{\boldsymbol{g}} := \{\forall i \mid \boldsymbol{g}_i(\boldsymbol{x}^*(\boldsymbol{p}), \boldsymbol{p}) = 0\},$$

$\boldsymbol{g}_{\mathcal{A}_{\boldsymbol{g}}}$ the vector of active inequality constraints and $\boldsymbol{\lambda}_{\mathcal{A}_{\boldsymbol{g}}}$ the corresponding Lagrange multipliers. Let define the strict complementary slackness condition (**SCSC**), whenever

$$\boldsymbol{\lambda}_i = 0 \iff \boldsymbol{g}_i(\boldsymbol{x}^*, \boldsymbol{p}) < 0 \quad \forall i \in \mathcal{A}_{\boldsymbol{g}}, \tag{SCSC}$$

holds.

Theorem A.3 (Basic sensitivity theorem (Fiacco 1976))
Let be $\boldsymbol{x}^*$ a local optimum of (P($\mathbf{p}$)),

- *f, $\boldsymbol{g}$ and $\boldsymbol{h}$ are twice continuously differentiable w.r.t. $\boldsymbol{x}$, and $\nabla_{\boldsymbol{x}} f$, $\nabla_{\boldsymbol{x}}\boldsymbol{g}$, $\nabla_{\boldsymbol{x}}\boldsymbol{h}$ are once continuously differentiable w.r.t. $\mathbf{p}$ in a neighborhood of ($\boldsymbol{x}^*$, $\mathbf{p}$),*
- *second order sufficient conditions (**SOSC**) hold at $\boldsymbol{x}^*$, with $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ the Lagrange multipliers associated to constraints $\boldsymbol{g}$ and $\boldsymbol{h}$, respectively,*
- *linear independence constraint qualification (**LICQ**) holds at $\boldsymbol{x}^*$,*
- *strict complementary slackness condition (**SCSC**) hold,*

then, with $\mathbf{p}'$ in a neighborhood of $\mathbf{p}$,

- *$\boldsymbol{x}^*$ is a local isolated minimizer and the associated Lagrange multipliers are unique,*
- *there exist unique, once continuously differentiable functions $\boldsymbol{x}^*(\mathbf{p}')$, $\boldsymbol{\lambda}^*(\mathbf{p}')$, $\boldsymbol{\mu}^*(\mathbf{p}')$ that satisfy (**SCSC**) for problem (P($\mathbf{p}$)) such that $(\boldsymbol{x}^*(\mathbf{p}),\ \boldsymbol{\lambda}^*(\mathbf{p}),\ \boldsymbol{\mu}^*(\mathbf{p})) = (\boldsymbol{x}^*,\ \boldsymbol{\lambda}^*,\ \boldsymbol{\mu}^*)$. $\boldsymbol{x}^*(\mathbf{p}')$ is also locally unique and the associated Lagrange multipliers are $\boldsymbol{\lambda}^*(\mathbf{p}')$, $\boldsymbol{\mu}^*(\mathbf{p}')$,*
- *the set of active inequalities in (P($\mathbf{p}'$)) remains unchanged w.r.t. the ones in (P($\mathbf{p}$)), (**SCSC**) also hold and satisfy (**LICQ**).*

Based on Theorem A.3, let define the local optimal value functions Ψ^* and $\mathcal{L}^*$ as :

$$\Psi^*(\boldsymbol{p}) = f(\boldsymbol{x}^*(\boldsymbol{p}), \boldsymbol{p})$$

$$\mathcal{L}^*(\boldsymbol{p}) = \mathcal{L}(\boldsymbol{x}^*(\boldsymbol{p}), \boldsymbol{\lambda}^*(\boldsymbol{p}), \boldsymbol{\mu}^*(\boldsymbol{p}), \boldsymbol{p})$$

The expression of the sensitivity of Ψ^* is given in the Theorem A.4.

Theorem A.4 (First-order derivative of the optimal value function (Fiacco 1976))
If assumptions of Theorem A.3 hold for problem P($\boldsymbol{p}$), then for any $\boldsymbol{p}'$ in the neighborhood of $\boldsymbol{p}$:

- *The optimal value function is equal to the Lagrangian optimal value :*

$$\Psi^*(\boldsymbol{p}') = \Psi^*(\boldsymbol{p}') = \mathcal{L}^*(\boldsymbol{p}') \tag{A.5}$$

- *The derivative of Ψ^* w.r.t. $\boldsymbol{p}$ is :*

$$\left.\frac{d\Psi^*}{d\boldsymbol{p}}\right|_{\boldsymbol{p}'} = \left.\frac{\partial f}{\partial \boldsymbol{p}}\right|_{\boldsymbol{x}^*,\boldsymbol{p}'} + \boldsymbol{\lambda}^{*\top}_{\mathcal{A}_g}(\boldsymbol{p}')\left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}\right|_{\boldsymbol{x}^*,\boldsymbol{p}'} + \boldsymbol{\mu}^{*\top}(\boldsymbol{p}')\left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{p}}\right|_{\boldsymbol{x}^*,\boldsymbol{p}'} \tag{A.6}$$

The Theorem A.4 offers thus an analytical expression of the gradient of Ψ, with respect to the parameters $\boldsymbol{p}$ and for any $\boldsymbol{p}'$, that depends on :

- a local isolated minimizer $\boldsymbol{x}^*$ of the Problem (P), obtained after one evaluation of $\Psi(\boldsymbol{p}')$,
- the gradient of the objective function f of Problem (P) with respect to the parameters, and taken at $\boldsymbol{p}$ and $\boldsymbol{x}^*$,
- the gradient of the inequality and equality constraints $\boldsymbol{f}$ and $\boldsymbol{g}$ of Problem (P) with respect to the parameters, and taken at $\boldsymbol{p}$ and $\boldsymbol{x}^*$,
- the Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ associated to the constraints $\boldsymbol{g}$ and $\boldsymbol{h}$.

Numerically, taking advantage of expression (A.6), the computation of the gradient of Ψ does only require one evaluation of Ψ. In addition, the system given by the stationarity condition (A.1) has to be solved to get the values of the Lagrange multipliers. Under the conditions given by the Theorem A.3, the number of evaluations of Ψ needed into the expression of the post-optimal sensitivity (A.1) is thus independent from the number of parameters $\boldsymbol{p}$, unlike the finite differences approach (4.10).

Bibliography

Abramson, Mark A., Charles Audet, James W. Chrissis, and Jennifer G. Walston (2009). "Mesh adaptive direct search algorithms for mixed variable optimization." In: *Optimization Letters* 3.1, pp. 35–47 (cit. on p. 15).

Adams, David B., Layne T. Watson, Zafer Gürdal, and Christine M. Anderson-Cook (2004). "Genetic algorithm optimization and blending of composite laminates by locally reducing laminate thickness." In: *Advances in Engineering Software* 35.1, pp. 35–43 (cit. on p. 7).

Allaire, G and G Delgado (2015). "Stacking sequence and shape optimization of laminated composite plates via a level-set method." In: *arXiv*, pp. 1–33 (cit. on pp. 16, 75).

Applegate, D., R. Bixby, V. Chvatal, and B. Cook (1995). *Finding Cuts in the TSP (A Preliminary Report).* Tech. rep. (cit. on p. 29).

Asadi Bagloee, Saeed and Majid Sarvi (2018). "An outer approximation method for the road network design problem." In: *PLOS ONE* 13.3. Ed. by Yong Deng (cit. on p. 75).

Audet, Charles and J. E. Dennis (2001). "Pattern Search Algorithms for Mixed Variable Programming." In: *SIAM Journal on Optimization* 11.3, pp. 573–594 (cit. on p. 15).

Audet, Charles, Michael Kokkolaras, Sébastien Le Digabel, and Bastien Talgorn (2018). "Order-based error for managing ensembles of surrogates in mesh adaptive direct search." In: *Journal of Global Optimization* 70.3, pp. 645–675 (cit. on p. 15).

Barjhoux, Pierre-jean, Youssef Diouane, Stéphane Grihon, Dimitri Bettebghor, and Joseph Morlier (2018a). "A Bilevel Methodology for solving a Structural Optimization Problem with both Continuous and Categorical Variables." In: *2018 Multidisciplinary Analysis and Optimization Conference.* Atlanta, Georgia: American Institute of Aeronautics and Astronautics, pp. 1–16 (cit. on p. 18).

Barjhoux, Pierre-Jean, Youssef Diouane, Stéphane Grihon, Dimitri Bettebghor, and Joseph Morlier (2018b). "Mixed Variable Structural Optimization: Toward an Efficient Hybrid Algorithm." In: *Advances in Structural and Multidisciplinary Optimization.* Cham: Springer International Publishing, pp. 1880–1896 (cit. on pp. 18, 64, 96).

Barjhoux, Pierre-jean, Youssef Diouane, Stéphane Grihon, Dimitri Bettebghor, and Joseph Morlier (2020). "A bi-level methodology for solving large-scale mixed categorical structural optimization." In: *Structural and Multidisciplinary Optimization* (cit. on p. 18).

Benders, J .F. (1962). "Partitioning procedures for solving mixed-variables programming problems." In: *Numerische Mathematik* 4.1, pp. 238–252 (cit. on pp. 16, 56).

Benichou, M., J. M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent (1971). "Experiments in mixed-integer linear programming." In: *Mathematical Programming* 1.1, pp. 76–94 (cit. on p. 29).

Benson, Harold P. and Reiner Horst (1991). "A branch and bound-outer approximation algorithm for concave minimization over a convex set." In: *Computers & Mathematics with Applications* 21.6-7, pp. 67–76 (cit. on pp. 88, 104).

Bettebghor, Dimitri (2011). "Optimisation biniveau de structures aéronautiques." In: *PhD Thesis* (cit. on p. 58).

Bettebghor, Dimitri, Nathalie Bartoli, Stephane Grihon, Joseph Morlier, and Manuel Samuelides (2011). “Approche en paramètres de stratification pour l’optimisation biniveau de structures composites.” In: *10e colloque national en calcul des structures.* Giens, France (cit. on p. 16).

Bettebghor, Dimitri, Stéphane Grihon, and Joseph Morlier (2018). “Bilevel optimization of large composite structures based on lamination parameters and post-optimal sensitivities. Part 1: Theoretical aspects.” In: (cit. on pp. 16, 58).

Boggs, Paul T. and Jon W. Tolle (1995). “Sequential Quadratic Programming.” In: *Acta Numerica* 4, pp. 1–51 (cit. on p. 14).

Bonami, Pierre, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, and Andreas Wächter (2008). “An algorithmic framework for convex mixed integer nonlinear programs.” In: *Discrete Optimization* 5.2, pp. 186–204 (cit. on pp. 75, 103).

Bremicker, M., P.Y. Papalambros, and H.T. Loh (1990). “Solution of mixed-discrete structural optimization problems with a new sequential linearization algorithm.” In: *Computers & Structures* 37.4, pp. 451–461 (cit. on p. 21).

Carpentier, Alban., Jean-Jacques Barrau, Laurent Michel, and Stéphane Grihon (2006a). “Buckling optimisation of composite panels via lay-up tables.” In: *III European Conference on Computational Mechanics.* Dordrecht: Springer Netherlands, pp. 226–226 (cit. on p. 7).

Carpentier, Alban, Laurent Michel, Stéphane Grihon, and Jean-Jacques Barrau (2006b). “Optimization methodology of composite panels.” In: *12th European Conference on Composite Materials (ECCM 12).* Biarritz, FR, pp. 1–8 (cit. on p. 7).

Collier, Craig, Phil Yarrington, and Barry Van West (2002). “Composite, Grid-Stiffened Panel Design for Post Buckling Using HyperSizer.” In: *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference.* Vol. 1. April. Reston, Virigina: American Institute of Aeronautics and Astronautics, pp. 157–172 (cit. on p. 9).

Cook, William (2012). “Markowitz and Manne + Eastman + Land and Doig = Branch and Bound.” In: I, pp. 227–238 (cit. on p. 21).

Dakin, R.J. (1965). “A tree-search algorithm for mixed integer programming problems.” In: *The Computer Journal* 8.3, pp. 250–255 (cit. on pp. 20, 21, 28, 75).

Deb, Kalyanmoy and Mayank Goyal (1998). “A Flexible Optimization Procedure for Mechanical Component Design Based on Genetic Adaptive Search.” In: *Journal of Mechanical Design* 120.2, p. 162 (cit. on pp. 40, 61, 90).

Duran, Marco A. and Ignacio E. Grossmann (1986). “An outer-approximation algorithm for a class of mixed-integer nonlinear programs.” In: *Mathematical Programming* 36.3, pp. 307–339. arXiv: `arXiv:1011.1669v3` (cit. on pp. 16, 74, 75).

Etman, L. F.P., J. M.T.A. Adriaens, M. T.P. Van Slagmaat, and A. J.G. Schoofs (1996). “Crash worthiness design potimization using multipoint sequential linear programming.” In: *Structural Optimization* (cit. on p. 14).

Fiacco, Anthony V. (1976). “Sensitivity analysis for nonlinear programming using penalty methods.” In: *Mathematical Programming* (cit. on pp. 74, 82, 84, 113, 114).

Filomeno Coelho, Rajan (2014). “Metamodels for mixed variables based on moving least squares: Application to the structural analysis of a rigid frame.” In: *Optimization and Engineering* 15.2, pp. 311–329 (cit. on p. 15).

Fister, Iztok, Xin She Yang, Janez Brest, and Dušan Fister (2013). “A brief review of nature-inspired algorithms for optimization.” In: *Elektrotehniski Vestnik/Electrotechnical Review* 80.3, pp. 116–122. arXiv: `arXiv:1307.4186v1` (cit. on p. 7).

Fletcher, Roger and Sven Leyffer (1994). “Solving mixed integer nonlinear programs by outer approximation.” In: *Mathematical Programming* 66.1-3, pp. 327–349 (cit. on pp. 74–76, 85, 88, 89, 103).

Forrest, John, Ted Ralphs, Stefan Vigerske, LouHafer, Bjarni Kristjansson, jpfasano, Edwin-Straver, Miles Lubin, Haroldo Gambini Santos, rlougee, and Matthew Saltzman (2018). *coin-or/Cbc: Version 2.9.9*. Version releases/2.9.9 (cit. on p. 90).

Fortin, Félix-Antoine, François-Michel De Rainville, M.A. Gardner, Marc Parizeau, and Christian Gagné (2012). “DEAP : Evolutionary Algorithms Made Easy.” In: *Journal of Machine Learning Research*, pp. 2171–2175 (cit. on pp. 40, 61, 90).

Gallard, Francois, Charlie Vanaret, Damien Guenot, Vincent Gachelin, Rémi Lafage, Benoit Pauwels, Pierre-Jean Barjhoux, and Anne Gazaix (2018). “GEMS: A Python Library for Automation of Multidisciplinary Design Optimization Process Generation.” In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics (cit. on pp. 17, 18, 39, 60, 89).

Gallard, Francois, Pierre-Jean Barjhoux, Romain Olivanti, and Anne Gazaix (2019). “GEMS, an Open-Source Generic Engine for MDO Scenarios : Key Features In Application.” In: *AIAA Aviation 2019* (cit. on pp. 17, 18, 39, 60, 89).

Gao, Huanhuan, Piotr Breitkopf, Rajan Filomeno Coelho, and Manyu Xiao (2018). “Categorical structural optimization using discrete manifold learning approach and custom-built evolutionary operators.” In: *Structural and Multidisciplinary Optimization* 58.1, pp. 215–228 (cit. on pp. 12, 15).

Garrido-Merchán, Eduardo C. and Daniel Hernández-Lobato (2018). “Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes.” In: pp. 1–18. arXiv: `1805.03463` (cit. on p. 15).

Gazaix, Anne, Francois Gallard, Vincent Ambert, Damien Guénot, Maxime Hamadi, Patrick Sarouille, Stéphane Grihon, Thierry Druot, Joel Brezillon, Lefebvre Thierry, Nathalie Bartoli, Rémi Lafage, Vincent Gachelin, Justin Plakoo, Nicolas Desfachelles, Selime Gurol, Benoit Pauwels, and Charlie Vanaret (2019). “Application of an Advanced Bi-level MDO Formulation to an Aircraft Engine Pylon Optimization.” In: Dallas, Texas: American Institute of Aeronautics and Astronautics (cit. on p. 7).

Geoffrion, A. M. (1972). “Generalized Benders decomposition.” In: *Journal of Optimization Theory and Applications* 10.4, pp. 237–260 (cit. on pp. 16, 56, 75).

Goldberg, David E (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Vol. Addison-We. Addison-Wesley Longman Publishing Co., Inc., p. 432 (cit. on p. 14).

Gopinath, Smitha, George Jackson, Amparo Galindo, and Claire S. Adjiman (2016). “Outer approximation algorithm with physical domain reduction for computer-aided molecular and separation process design.” In: *AIChE Journal* 62.9, pp. 3484–3504 (cit. on p. 75).

Grihon, Stéphane (2012). “PRESTO: A rapid sizing tool for airframe conceptual design studies.” In: *LMS European Aeronautical Conference*. Toulouse (cit. on p. 12).

Grihon, Stéphane (2018). “Structure Sizing Optimization Capabilities at AIRBUS.” In: *Advances in Structural and Multidisciplinary Optimization*. Ed. by Axel Schumacher, Thomas Vietor, Sierk Fiebig, Kai-Uwe Bletzinger, and Kurt Maute. Cham: Springer International Publishing, pp. 719–737 (cit. on pp. 7, 9, 16, 58).

Grihon, Stéphane, Manuel Samuelides, Antoine Merval, Alain Remouchamps, Michaël Bruyneel, Benoit Colson, and Klaus Hertel (2009). “Fuselage Structure Optimization.” In: pp. 193–245 (cit. on p. 4).

Grossmann, Ignacio E. (2009). “MINLP: Outer Approximation Algorithm.” In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA: Springer US, pp. 2179–2183 (cit. on pp. 75, 89, 103).

Gupta, Omprakash K. and A. Ravindran (1983). “Nonlinear Integer Programming and Discrete Optimization.” In: *Journal of Mechanisms Transmissions and Automation in Design* 105.2, p. 160 (cit. on p. 21).

Gupta, Omprakash K. and A. Ravindran (1985). “Branch and Bound Experiments in Convex Nonlinear Integer Programming.” In: *Management Science* 31.12, pp. 1533–1546 (cit. on pp. 21, 75).

Haftka, Raphael T. and Zafer Gürdal (1992). *Elements of Structural Optimization*. Vol. 11. Solid Mechanics And Its Applications. Dordrecht: Springer Netherlands (cit. on pp. 40, 47, 58, 62, 64, 67, 89, 96, 98).

Haftka, Raphael T and Layne T Watson (2005). “Multidisciplinary Design Optimization with Quasiseparable Subsystems.” In: *Optimization and Engineering* 6, pp. 9–20 (cit. on p. 58).

Haftka, Raphael T, Layne T Watson, R T Haftka, and L T Watson (2006). “Decomposition theory for multidisciplinary design optimization problems with mixed integer quasiseparable subsystems.” In: *Optim Eng* 7.7, pp. 135–149 (cit. on pp. 16, 58).

Hager, Kevin and Richard Balling (1988). “New Approach for Discrete Structural Optimization.” In: *Journal of Structural Engineering* 114.5, pp. 1120–1134 (cit. on p. 21).

Herrera, Manuel, Aurore Guglielmetti, Manyu Xiao, and Rajan Filomeno Coelho (2014). “Metamodel-assisted optimization based on multiple kernel regression for mixed variables.” In: *Structural and Multidisciplinary Optimization* 49.6, pp. 979–991 (cit. on p. 15).

Hijazi, Hassan, Pierre Bonami, and Adam Ouorou (2014). “An Outer-Inner Approximation for Separable Mixed-Integer Nonlinear Programs.” In: *INFORMS Journal on Computing* 26.1, pp. 31–44 (cit. on p. 16).

Ivanov, V. V. and V. K. Zadiraka (1975). “Numerical optimization.” In: *Cybernetics* 9.4, pp. 714–715. arXiv: `NIHMS150003` (cit. on p. 60).

Johnson, Steven G. (2008). “The NLopt nonlinear-optimization package.” In: (cit. on p. 89).

Karush, William (1939). “Minima of functions of several variables with inequalities as side conditions.” MA thesis. Illinois, USA: Department of Mathematics, University of Chicago (cit. on p. 112).

Krogh, Christian, Mathias H. Jungersen, Erik Lund, and Esben Lindgaard (2017). “Gradient-based selection of cross sections: a novel approach for optimal frame structure design.” In: *Structural and Multidisciplinary Optimization* 56.5, pp. 959–972 (cit. on p. 16).

Kronqvist, Jan, David E. Bernal, Andreas Lundell, and Ignacio E. Grossmann (2019). *A review and comparison of solvers for convex MINLP*. Vol. 20. 2. Springer US, pp. 397–455 (cit. on p. 75).

Kuhn, H. W. and A. W. Tucker (1951). “Nonlinear Programming.” In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Calif.: University of California Press, pp. 481–492 (cit. on p. 112).

Land, A. H. and A. G. Doig (1960). “An Automatic Method of Solving Discrete Programming Problems.” In: *Econometrica* 28.3, p. 497 (cit. on pp. 20, 21, 28).

Li, Duan and Xiaoling Sun (2006). *Nonlinear Integer Programming*. Springer (cit. on p. 75).

Liao, Tianjun, Krzysztof Socha, Marco A. Montes de Oca, Thomas Stutzle, and Marco Dorigo (2014). “Ant Colony Optimization for Mixed-Variable Optimization Problems.” In: *IEEE Transactions on Evolutionary Computation* 18.4, pp. 503–518 (cit. on p. 14).

Lindroth, Peter and Michael Patriksson (2011). *Pure Categorical Optimization: a Global Descent Approach*. Tech. rep. Chalmers University of Technology (cit. on p. 15).

Marcotte, Patrice and Jean-Pierre Dussault (1989). “A Sequential Linear Programming Algorithm for Solving Monotone Variational Inequalities.” In: *SIAM Journal on Control and Optimization* 27.6, pp. 1260–1278 (cit. on p. 14).

Markowitz, Harry M. and Alan S Manne (1957). “On the Solution of Discrete Programming Problems.” In: *Econometrica* 25.1, p. 84 (cit. on p. 21).

McCormick, Garth P. (1967). “Second Order Conditions for Constrained Minima.” In: *SIAM Journal on Applied Mathematics* 15.3, pp. 641–652 (cit. on p. 112).

Merval, Antoine (2008). “L’Optimisation Multiniveaux D’une Structure.” In: *PhD Thesis* (cit. on p. 89).

Müller, Juliane, Christine A. Shoemaker, and Robert Piché (2013). “SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems.” In: *Computers & Operations Research* 40.5, pp. 1383–1400 (cit. on p. 15).

Nouaouria, Nabila and Mounir Boukadoum (2011). “A Particle Swarm Optimization approach to mixed attribute data-set classification.” In: *2011 IEEE Symposium on Swarm Intelligence*. IEEE, pp. 1–8 (cit. on p. 14).

Nowak, Ivo (2005). *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*. Vol. 152 (cit. on p. 75).

Ostrovsky, G. M., M. G. Ostrovsky, and G. W. Mikhailow (1990). “Discrete optimization of chemical processes.” In: *Computers and Chemical Engineering* (cit. on p. 29).

Pecci, Filippo, Edo Abraham, and Ivan Stoianov (2017). “Outer approximation methods for the solution of co-design optimisation problems in water distribution networks **This work was supported by the NEC-Imperial SmartWater Systems project. The authors acknowledge the EPSRC Industrial CASE Studentship project EP/I501444/1, from which the case study model BWFLnet was derived.” In: *IFAC-PapersOnLine* 50.1. 20th IFAC World Congress, pp. 5373 –5379 (cit. on p. 75).

Pelamatti, Julien, Loïc Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Yannick Guerin (2019). “Efficient global optimization of constrained mixed variable problems.” In: *Journal of Global Optimization* 73.3, pp. 583–613 (cit. on p. 15).

Perron, Laurent and Vincent Furnon (2019). *OR-Tools*. Version 7.2. Google (cit. on p. 90).

Roy, S., K.T. Moorey, J.T. Hwang, J.S. Gray, W.A. Crossley, and J.R.R.A. Martins (2017). “A mixed integer efficient global optimization algorithm for the simultaneous aircraft allocation-mission-design problem." In: *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2017*, pp. 1–18 (cit. on p. 15).

Roy, Satadru, William A. Crossley, Bret Stanford, Kenneth T. Moore, and Justin S. Gray (2019). "A Mixed Integer Efficient Global Optimization Algorithm with Multiple Infill Strategy - Applied to a Wing Topology Optimization Problem." In: *AIAA Scitech 2019 Forum* (cit. on p. 15).

Saka, M.P. and M. Ulker (1992). "Optimum design of geometrically nonlinear space trusses." In: *Computers & Structures* 42.3, pp. 289 –299 (cit. on pp. 62, 69, 89, 101).

Samuelides, Manuel, Dimitri Bettebghor, Stéphane Grihon, Antoine Merval, and Joseph Morlier (2009). "Modèles réduits en optimisation multiniveau de structures aéronautiques." In: *9e Colloque national en calcul des structures*. CSMA. Giens, France (cit. on p. 16).

Sandgren, E. (1990a). "Nonlinear Integer and Discrete Programming for Topological Decision Making in Engineering Design." In: *Journal of Mechanical Design* 112.1, p. 118 (cit. on p. 21).

Sandgren, E. (1990b). "Nonlinear Integer and Discrete Programming in Mechanical Design Optimization." In: *Journal of Mechanical Design* 112.2, p. 223 (cit. on p. 21).

Schmit, L. A. and C. Fleury (1980). "Discrete-continuous variable structural synthesis using dual methods." In: *AIAA Journal* 18.12, pp. 1515–1524 (cit. on p. 21).

Schutte, Jaco F, Raphael T Haftka, and Layne T Watson (2004). "Decomposition and Two-level Optimization of Structures with Discrete Sizing Variables." In: *Constraints* 0.1, pp. 1–5 (cit. on p. 16).

Shahabsafa, Mohammad, Ali Mohammad-Nezhad, Tamás Terlaky, Luis Zuluaga, Sicheng He, John T. Hwang, and Joaquim R. R. A. Martins (2018). "A novel approach to discrete truss design problems using mixed integer neighborhood search." In: *Structural and Multidisciplinary Optimization* 58.6, pp. 2411–2429 (cit. on pp. 40, 47, 62, 67, 89, 98).

Sheu, C Y and L A Schmit Jr (1972). "Minimum Weight Design of Elastic Redundant Trusses under Multiple Static Loading Conditions." In: 10.2 (cit. on p. 21).

Sigmund, Ole (2011). "On the usefulness of non-gradient approaches in topology optimization." In: *Structural and Multidisciplinary Optimization* 43.5, pp. 589–596 (cit. on p. 15).

Stegmann, J. and E. Lund (2005). "Discrete material optimization of general composite shell structures." In: *International Journal for Numerical Methods in Engineering* 62.14, pp. 2009–2027 (cit. on p. 15).

Stolpe, Mathias (2011). "To bee or not to bee—comments on "Discrete optimum design of truss structures using artificial bee colony algorithm"." In: *Structural and Multidisciplinary Optimization* 44.5, pp. 707–711 (cit. on p. 15).

Stolpe, Mathias (2015). "Truss topology optimization with discrete design variables by outer approximation." In: *Journal of Global Optimization* 61.1, pp. 139–163 (cit. on p. 75).

Stolpe, Mathias and Kasper Sandal (2018). "Structural optimization with several discrete design variables per part by outer approximation." In: *Structural and Multidisciplinary Optimization* 57.5, pp. 2061–2073 (cit. on pp. 75, 88, 104).

Stubbs, Robert A. and Sanjay Mehrotra (1999). "A branch-and-cut method for 0-1 mixed convex programming." In: *Mathematical Programming* 86.3, pp. 515–532 (cit. on p. 75).

Svanberg, Krister (2002). “A class of globally convergent optimization methods based on conservative convex separable approximations.” In: *SIAM Journal on Optimization* 12.2, pp. 555–573 (cit. on pp. 14, 39, 60, 89).

Tseng, C. H., L. W. Wang, and S. F. Ling (1995). “Enhancing Branch-and-Bound Method for Structural Optimization.” In: *Journal of Structural Engineering* 121.5, pp. 831–837 (cit. on p. 21).

Turner, M. J., H. C. Martin, and R. C. Leible (1964). “Further Development and Applications of Stiffness Methods.” In: *Matrix Methods of Structural Analysis.* 1st. Vol. 1. New York: Macmillian, pp. 203–266 (cit. on pp. 12, 13).

Turner, M.J. (1959). “The Direct Stiffness Method of Structural Analysis.” In: *Structural and Materials Panel Paper, AGARD Meeting.* Aachen (cit. on pp. 12, 13).

Westerlund, Tapio and Frank Pettersson (1995). “An extended cutting plane method for solving convex MINLP problems.” In: *Computers & Chemical Engineering* 19.SUPPL. 1, pp. 131–136 (cit. on p. 75).

Wright, Jorge and Stephen J. Nocedal (2006). In: pp. 529–562 (cit. on p. 14).

Yuan, X., S. Zhang, L. Pibouleau, and S. Domenech (1988). “Une méthode d’optimisation non linéaire en variables mixtes pour la conception de procédés.” In: *RAIRO - Operations Research* 22.4, pp. 331–346 (cit. on p. 75).

Résumé — Dans l'industrie aéronautique, les problèmes d'optimisation de structure peuvent impliquer des changements de matériaux, de types de raidisseurs, et de tailles d'éléments. Dans ce travail, il est ainsi proposé de résoudre des problèmes de grande taille (minimisation de masse) par rapport à des variables catégorielles et continues, sujets à des contraintes de stress et de déplacements. Trois algorithmes sont présentés, discutés dans le manuscrit au regard de cas tests de plus en plus complexes. En tout premier lieu, un algorithme basé sur le "branch and bound" a été mis en place. Une formulation d'un problème dédié au calcul de minorants de la masse optimale est proposée. Bien que l'algorithme permette de trouver des solutions optimales, la tendance du coût de calcul en fonction de l'augmentation du nombre d'éléments est exponentielle. Le second algorithme s'appuie sur une formulation bi-niveau du problème d'origine, où le problème supérieur consiste à minimiser une approximation au premier ordre du résultat du niveau inférieur. L'évolution du coût de calcul par rapport à l'augmentation du nombre d'éléments et de valeurs catégorielles est quasiment linéaire. Enfin, un troisième algorithme tire partie d'une reformulation du problème mixte catégoriel continu en un problème bi-niveau mixte avec variables entières continûment relâchables. Les cas tests numériques montrent la résolution d'un problème avec plus d'une centaine d'éléments. Egalement, le coût de calcul est quasi-indépendant du nombre de valeurs de variables catégorielles disponibles par élément.

Mots clés : programmation mixte, variables de design catégorielles, formulations multiniveaux, problèmes de grande dimension, optimisation de structure, multimatériau

Abstract — Nowadays in the aircraft industry, structural optimization problems can be really complex and combine changes in choices of materials, stiffeners, or sizes/types of elements. In this work, it is proposed to solve large scale structural weight minimization problems with both categorical and continuous variables, subject to stress and displacements constraints. Three algorithms have been proposed. As a first attempt, an algorithm based on the branch and bound generic framework has been implemented. A specific formulation to compute lower bounds has been proposed. According to the numerical tests, the algorithm returned the exact optima. However, the exponential scalability of the computational cost with respect to the number of structural elements prevents from an industrial application. The second algorithm relies on a bi-level formulation of the mixed categorical problem. The master full categorical problem consists of minimizing a first order like approximation of the slave problem with respect to the categorical design variables. The method offers a quasi-linear scaling of the computational cost with respect to the number of elements and categorical values. Finally, in the third approach the optimization problem is formulated as a bi-level mixed integer non-linear program with relaxable design variables. Numerical tests include an optimization case with more than one hundred structural elements. Also, the computational cost scaling is quasi-independent from the number of available categorical values per element.

Keywords: mixed integer programming, categorical design variables, multilevel formulation, large scale optimization, structural optimization, multimaterial

Institut Clément Ader, 3 Rue Caroline Aigle
Toulouse, France