HEC Management School
of the
University of Liège

Optimization Methods for the design and production of Naval Structures

A dissertation submitted to the faculty of HEC-Ulg
in fulfillment of
the requirements for the degree of
Doctor in Management Sciences

by
Maud Bay

Jury:

Professor Yves Crama, Advisor
Professor Pierre Duysinx
Professor Pierre Hansen
Professor Philippe Rigo
Professor Michaël Schyns
Professor Philippe Toint

Contents

Introduction **6**

I Ship Structural Design **9**

1 Introduction to structural design **10**
- 1.1 Ship design and production 10
  - 1.1.1 Historical perspective 10
  - 1.1.2 Design process 11
  - 1.1.3 Ship structure model 12
  - 1.1.4 Mechanical behaviors 16
- 1.2 Mathematical models for structural design 18
  - 1.2.1 Three-dimensional model 19
    - 1.2.1.1 Variables 19
    - 1.2.1.2 Objective 19
    - 1.2.1.3 Constraints 21
    - 1.2.1.4 Behavior models 25
    - 1.2.1.5 Load cases 25
  - 1.2.2 Two-dimensional model 26
    - 1.2.2.1 Variables 26
    - 1.2.2.2 Objective 26
    - 1.2.2.3 Constraints 27
- 1.3 Ship structural optimization 29
- 1.4 Multicriterion optimization 31

2 Local search algorithms for time expensive problems **34**
- 2.1 Introduction 34
- 2.2 Literature review 35
- 2.3 Model 38
- 2.4 Algorithms 40
  - 2.4.1 Dive and Fix heuristic 41
    - 2.4.1.1 Convergence 43
    - 2.4.1.2 Solution of continuous relaxations 44
  - 2.4.2 Branch and Fix heuristic 46
- 2.5 Computational experiments 48
  - 2.5.1 Industrial test cases 48
  - 2.5.2 Results 49
  - 2.5.3 Comparison with branch-and-bound 51
  - 2.5.4 Parameters of the heuristics 52
  - 2.5.5 Numerical tolerance issues 54
- 2.6 Multicriterion optimization 54
- 2.7 Discussion and conclusions 58

3 Decomposition and Direct Search methods for explicit problems with fast function evaluations 60
3.1 Introduction 60
3.2 Literature review 61
3.3 Model 64
3.4 Decomposition method 66
3.5 Subproblems and direct search 69
3.5.1 Definition of the mesh and search directions 70
3.5.1.1 Using the structure of the objective function . 72
3.5.1.2 Using the constraints values and derivatives . 76
3.5.1.3 Definition of the step size parameter 76
3.5.2 Algorithm 77
3.6 Computational experiments 79
3.6.1 Industrial test cases 79
3.6.2 Preliminary results 81
3.6.3 Convergence of the algorithm 83
3.7 Discussion and conclusions 84

II Assembly Area Scheduling 86
Introduction 87

4 Models for space and time allocation problems 89
4.1 Problem statement 89
4.2 Mathematical model 89
4.2.1 Relation to the 3-Dimensional Bin Packing problem . . 91

5 Methods for space and time allocation problems 93
5.1 A three steps algorithm 93
5.1.1 Finding feasible solutions 93
5.1.1.1 General approach 93
5.1.1.2 Guided local search 94
5.1.1.3 Fast local search 95
5.1.2 Selecting the blocks 97
5.2 Computational experiments 100
5.2.1 Data and test instances 101
5.2.2 The λ parameter 102
5.2.3 Performance of *GLS* as a function of its running time . 104
5.2.4 Optimization procedures 104
5.3 Industrial issues 108
5.3.1 Additional constraints 110
5.3.2 Robustness 111
5.3.3 Workforce capacity and requirement 111

5.3.4 Subcontracting blocks 112
5.4 Conclusion 113

III Conclusions 115

IV Bibliography 118

V Appendix 128

A Three-dimensional model 129
A.0.1 Structural Constraints 129
A.0.2 Geometrical Constraints: 132

B Differential equations of stiffened panels 135
B.0.3 Differential equations of stiffened panels 135

C Two dimensional model 139
C.1 Mathematical formulation 139
C.1.1 Variables 139
C.1.2 Objective function 139
C.1.3 Constraints 140
C.1.3.1 Equality constraints 140
C.1.3.2 Technological constraints 140
C.1.3.3 Geometrical constraints 140
C.1.3.4 Global constraints 141
C.2 Shear stress computation method 146

This thesis could not have been done and released without the support of persons who helped me in many ways for many years. First, I owe a tremendous debt of gratitude to my supervisor, Professor Yves Crama, who guided me in the wide area of operations research and learned me the importance of exact and clear formulations. His patience and insightful comments were and still are unvaluable and deeply appreciated.

I thank Professor Philippe Rigo for giving me the opportunity to work on a practical engineering problem, making the bridge between management and engineering through optimization. The collaboration with the colleagues of ANAST department was fruitful and their friendship very important to me.

I would like to thank the members of the jury, Professor Pierre Hansen for his warm welcome in a snowy Montreal, for his sound advices on nonlinear and constrained model and solver, and particularly to open the door on direct search methods. I am very grateful also to Professor Philippe Toint for his expertise and guidance, for sharing his knowledge and algorithmic work. Thanks also for the Han-sur-Lesse Mathematical Programming Conference he organizes yearly, it is a great opportunity to meet other doctoral students in our field of research.
+

I really appreciated the useful suggestions of Professor Pierre Duysinx, his deep knowledge of structural optimization in aerospace design helped me to better understand the great similitudes between naval and aerospace structural designs. I would like to express my gratitude to Professor Michaël Schyns for his well-appreciated support and his unlimited involvement in guiding students to fulfill their doctoral program.

I thank Thomas Richir for our collaboration on multiobjective optimization, Yves Langer for our joined work on STA problem and Pierre Schaus for his support on constraint programming in the same chapter.

I would like to express my great pleasure to work with motivated and motivating colleagues in the QuantOM research group and ANAST department : Yasemin,Thierry, Cédric (at improbable hours), Guillaume, Sabine, Géraldine, Véronique, Jean-David, Fred, Renaud, Paco... I am also grateful to enjoy various types of supports from Pounette and David, Sylvie, Christine, Anne-Marie...

I dedicate this work my family, especially my husband and my three daugthers. Without their support and love this would not have been possible.

Introduction

The objective of this work is to study combinatorial optimization methods for discrete sizing in structural design and for managerial issues in production planning. These issues arise in various manufacturing industries; the thesis is realized in the context of shipbuilding industry with a make-to-order production of large and unique marine structures. The dissertation is structured into two parts. The first part is devoted to the optimization of the preliminary structural design of large passenger ships, and management issues arising in manufacturing facilities are the subject of the second part.

Part I of the thesis addresses the design of ship metallic structures. The structural design of the hull is studied in a preliminary phase of the project, even before the client order is placed. The structure has to satisfy robustness requirements at a competitive fabrication cost. The material and manufacturing cost is lowered by choosing standard parts from catalogs and by building subassemblies. The robusteness of the entire assembly (the ship hull) is checked with structural analysis methods, available in LBR-5 software developed in the Naval Architecture and Transport System Analysis sector (ANAST) of the University of Liege

The contribution of Part I is to propose new models for the ship design problem by adding the imposition of choosing parts in standard catalogs, and propose original algorithms to solve the large combinatorial optimization problems defined, in complement to the continuous optimization algorithm already available in the software.

In Chapter 1 we give an introduction to ship design and to the two models available in LBR-5 for the structural analysis. The mathematical formulations include our adaptations to consider discrete variables. The first model for structural analysis is a three dimensional model (3D model) relying on systems of differential equations to calculate the values of the stress, deformations and displacements at various points of the structure. A large number of implicit functions with expensive computation time is the main feature of this model. A second model of structural analysis considers the ship structure as a beam. The problem reduces to a two-dimensional model for which explicit formulas are available, and the structural analysis is very fast. Despite being less detailed, the beam model is sufficient to provide results compliant with the requirements of international certification organizations for sea vessels.

In Chapter 2 we propose heuristics for discrete optimization problems as formulated in Chapter 1. Two original algorithms are presented to solve the discrete optimization problem based on the 3D model. These methods are inspired from branching strategies commonly used in combinatorial optimization. They are tested on real world applications provided by the French shipyard *STX*, and are shown to converge towards discrete optimal solutions within a reasonable budget of computation time. [1]

In Chapter 3 we consider discrete optimization for the beam ship model. We present a new approach that combines the methods of decomposition and direct search. We benefit from the explicit formulation of the functions to exploit their mathematical structure, and decompose the model in sub-problems of smaller size. The original concept is to solve a continuous relaxation of the complete problem and consider the discreteness in the sub-problems only. To solve the sub-problems we propose an adaptation of Direct Search methods to discrete optimization. The approach has been implemented in LBR-5 and gives encouraging results on a real ship structure.

The second Part of the thesis is devoted to the problem of space and time allocation for the shipyard's production facilities. When a ship is ordered, the shipyard has to set up a production plan to estimate the project's requirements in raw material, human resources and production resources in the facility. Production consists of several steps: one of them is the assembly of large blocks composed of steel plates, stiffeners and equipment. These blocks are produced in the shipyard's large assembly halls and the *STA* problem consists in maximizing the number of blocks produced in the halls over a given time period. Blocks that cannot be produced in the halls have to be sub-contracted at higher cost.

We develop new methods to improve the resource management for the production in assembly halls. The *STA* problem statement and mathematical model are proposed in Chapter 5. We discuss its relation to the 3-Dimensional Bin Packing problem, a classical combinatorial optimization problem which is known to be hard to solve. Chapter 6 proposes heuristic approaches to address the problem. Our solution approach is actually largely

[1]The work presented in this chapter has been submitted for publication in Engineering and Optimization and is currently under revision.

inspired from previous work on bin packing. The heuristics have been applied on real test instances and we test its performance by comparing it with a Constraint Programming approach. We also discuss additional industrial issues that have been taken into account in the software delivered to the shipyard. Although the methods are developed for shipyards, they are generic enough to be applied to other types of production facilities.[2]

The conclusions are finally presented in the last part of the dissertation.

[2]This part has been published in Annals of Operational Research, Volume 179 (2010), Number 1, Pages 57-76,with the title : Space and time allocation in a shipyard assembly hall.

Part I

Ship Structural Design

Chapter 1

Introduction to structural design

1.1 Ship design and production

1.1.1 Historical perspective

As many industries in the last decades, the global market context has led shipyards to merge and increase their productivity. In Europe, French and Scandinavian shipyards have merged and are working in collaboration with other European shipyards in order to face the growing Asian competitors. Countries such as Japan, South Korea and China not only benefit from low production costs but also develop efficient production methods to reach high efficiency; this strategy results in mass production of standard works such as tankers and container ships. In the meantime, the European shipyards comforted their technological knowledge to focus on complex works such as large cruise ships and special offshore vessels and structures.

In 1912 the famous Titanic was the greatest ocean liner ever built, with a length of 269 meters, a gross tonnage of 46,300 and accommodation for 2,300 passengers and crew members. In 1940, the RMS Queen Elizabeth, was the largest cruise ship for almost the next fifty six years, with a length of 314 meters and a gross tonnage of 83,673[1]. In 2004 the Queen Mary II was built in Saint Nazaire (France) and held the record with almost 150,000 gross tons. The symbolic value of 200,000 tons has been exceeded with the venue of the sister cruise ships MS Oasis of the Seas and MS Allure of the seas. They are 360m long with 225,000 gross tons: about half again as large as the next largest cruise ship in operation. These ships are floating cities, providing leisure, catering and luxurious accommodation for more than six thousands passengers.

In the tanker class, as a comparison, *Mont* is considered as the largest

[1]The gross tonnage (GT) is calculated based on "the molded volume of all enclosed spaces of the ship".

ship ever built with 450 meters long and 261,000 gross tons. Such dimensions make it incapable to navigate the English Channel, the Suez nor the Panama Canals. In the context of marine freight transportation has originated the increase of cargo sizes and now motivates the expansion projects of Panama and Suez canals. .

Technology and management methods in shipyards have evolved with the sizing of the vessels. Research has focused on technology and mechanical performance of ships, and their ability to resist various solicitations due to navigation, sea conditions and loading. As ship structures became more complex and large, not only the ship size, power and on board equipments have evolved but also security requirements. A kind of insurance against operational risks appeared in the second half of the eighteenth century and the insurance societies needed a way of assessing the quality of ships as a probability of structural failure. In the early 1760, the Lloyd's Register of Shipping was the first form of ship classification society, Bureau Veritas – the second most important society nowadays – was created about seventy years later and American Bureau of Shipping more than one century after Lloyd's. Today more than fifty classification societies are active worldwide. They establish and maintain standards for the construction and classification of ships, supervise their construction according to these standards and carry out periodic surveys of ship in service to ensure the compliance with standards. The model presented in section 1.2.2 is based on standards established by the International Association of Classification Societies (IACS), founded in 1968. Today IACS is constituted of ten classification societies among which Lloyd's register (London), American Bureau of Shipping (ABS), and Bureau Veritas (BV).

As technology evolved so did management methods to improve the efficiency of the shipbuilding process. Current hulls are assemblies of large blocks pre-assembled in specialized workshops, where automation is extensively used before the final assembly is realized in dry docks. Production management methods include the use of optimization algorithms and simulation software to improve the efficiency of shipyards. Practical considerations about production methods are considered at the early phase of a ship conception. Twelve months were necessary to achieve the construction of the Titanic hull in 1912; almost one century later, the keel of "Oasis of the Seas", roughly five times bigger than Titanic, was built in 21 months only.

1.1.2 Design process

When a ship-owner comes to one or more shipyards and asks for an offer, he gives only a few specifications such as the ship's overall size, its loading capacity and speed requirements. Then the shipyard realizes a preliminary design of the ship to produce the offer: it is the earliest phase of the project, occurring before the order is placed by the customer. During this very first phase, the shipyards have to plan and harness the usage of their production resources as well as the material needed to build the ship in order to minimize the production cost while complying with the customer's specifications and the international security standards.

The design a ship structure consists of drawing the steel structure of the ship and sizing all its main components (plates, piping, etc.) given some overall size specifications. It is clear that decisions taken during this preliminary phase will greatly influence the final structure and will drastically limit the subsequent choices of the production techniques and of the main constitutive elements of the structure.

In consequence a special attention is paid to the production cost and to compliance with robustness and security standards in this early study. At the end of the design study, the dimensions of all the plates and constitutive elements of the structure (the scantling) are decided, as well as its overall weight and production cost.

This is why the preliminary structural design always poses difficult problems to designers: they have to make the most adequate choices within a very short period of time. This kind of preliminary study daily happens in numerous industries working with large projects where the product is unique and has to be custom-designed at the very beginning of the project (or even before the client's order). This is the case in such industries as the naval, spatial or civil sectors.

1.1.3 Ship structure model

In the following sections we present two mathematical models for ship structural design at preliminary design stage. These models have been developed (considering all variables as continuous), at the ANAST department and are implemented in LBR5 software (Rigo 2002).

The variables are the dimensions and positions of the constitutive elements of the ship. The objective function considered is usually to be the weight of the hull, its cost, its moment of inertia or a combination of these three elements. The set of variables and the formulation of structural constraints are specific to each model.

The basis for the two models used in this thesis is the representation of a ship as prismatic structures. Only midship sections are considered, excluding the bow and the stern parts. These front and back ends have particular shapes and require adapted structural analysis methods. Similar considerations lead to ignoring superstructures, as they do not participate significantly in the global hull response within the mechanical conditions studied.

The general modeling approach and resolution methods have been developed at ANAST department at the University of Liège in accordance with the major reference book *Ship Structural design, a rationally-based, computer-aided optimization approach*, authored by O. Hughes with a first edition in 1983.

A three-dimensional model is first considered. This very accurate model allows computation of the values of the forces in many points of the ship structure. Due to its mathematical form this model leads to expensive computation times. Mainly due to this drawback, and to set up a method closer to those used by classification societies, a second model is considered. This is a two-dimensional model that approximate a midship section with a beam. We present a mathematical description of these models in sections 1.2.1 and 1.2.2; the reader will find detailed formulations in Appendices A and C.

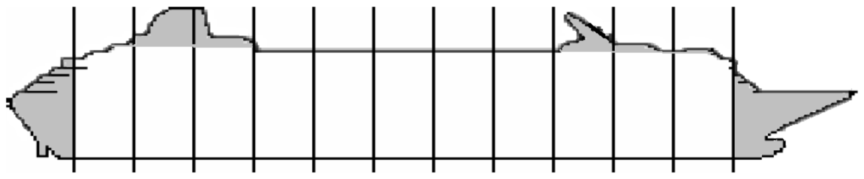


Figure 1.1: Decomposition of a ship into transversal sections

The overall hull structure is subdivided into sub-modules - or blocks - linked with a number of elements, sharing a given geometry or specified connections. This leads to virtually decomposing ships into prismatic transversal cross-sections as represented on Figure 1.1. Due to left-right symmetry and the symmetry of internal and external forces, only a half cross section has to be modeled. Figure 1.2 shows the cross section of a ship as an assembly of stiffened panels. Generally speaking, the number, arrangement, width and length of the main elements of the cross-section are not determined by structural considerations but rather by more general requirements such as customer specifications (cargo capacity...) and characteristics of the production equipment of the shipyards.

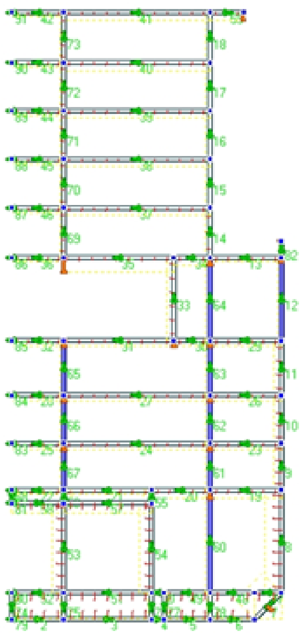


Figure 1.2: Cross section of a structure

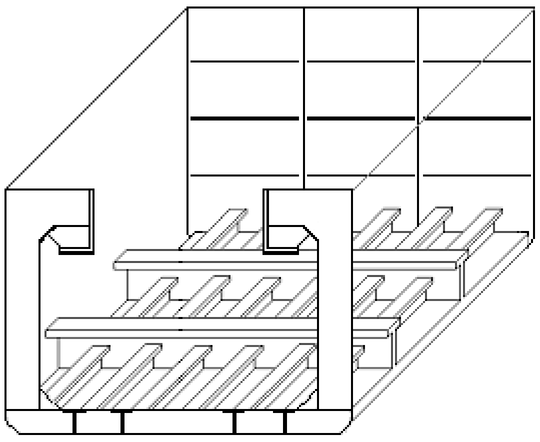


Figure 1.3: Submodel: a bloc unit

Stiffened panels of ship hulls may have three kinds of members, identified according to their orientation and size. Frames are big transverse members.

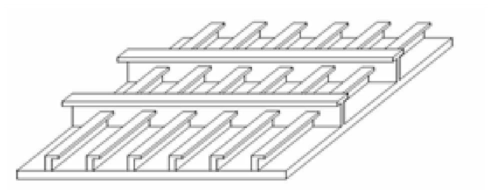


Figure 1.4: A stiffened panel

Longitudinal stiffening (parallel to the length of the ship) include stiffeners and girders, the latest being the biggest of these two kinds. In practice the girders are not present in our applications and we will leave them aside in the following to lighten model formulations, without loss of generality. Although not included in the text, they are available in the structural analysis model and software as well as in the optimization algorithms.

The design variables of a structure are the spacing and scantlings of the structural members: plate thickness, member dimensions and spacing. We note z_1 the direction along the length of the ship and z_2 the direction of a panel in the transverse plan. Each panel is characterized with eleven dimensions (see Fig. 1.5):

- $\delta^{\,p}$ is the plate thickness,
- $h^p_{z_1}$ is the web length of the stiffeners
- $d^p_{z_1}$ is the web thickness of the stiffeners
- $w^p_{z_1}$ is the flange length of the stiffeners
- $\mathrm{t}^p_{z_1}$ is the flange thickness of the stiffeners
- $\Delta\,^p_{z_1}$is the spacing between two stiffeners fitted along the z_1 direction,
- $h^p_{z_2}$,is the web length of the transverse frames
- $d^p_{z_2}$,is the web thickness of the transverse frames
- $w^p_{z_2}$ is the flange length of the transverse frames
- $\mathrm{t}^p_{z_2}$ is the flange thickness of the frames
- $\Delta\,^p_{z_2}$ is the spacing between two transverse frames fitted along the z_2 direction,

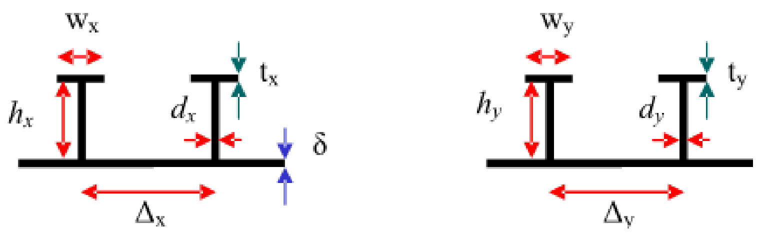


Figure 1.5: dimensions labeling along x and y directions

1.1.4 Mechanical behaviors

We will now proceed to the description of the physical constraints that apply to ships. During their working life time ships encounter various weather and sea conditions. These conditions are modeled as sea states that induce external forces and moments on the overall structure. Various loading conditions also occur: the ship may be empty in harbor, with a partial or complete load of fuel, ballast, cargo. These loadings states are combined with sea conditions to generate various load cases. A structure has to pass robustness tests based on standard load cases to be certified.

To study ship mechanical responses to various loading conditions, ship structures are idealized as a hollow thin-wall box beam, referred to as the *hull girder*. We study elastic beams floating on the water surface and subject to a range of fluctuating and steady loads. These loads result in a bending moment and a shear force accompanied with hogging and sagging effects on the overall structure.

When modeling the hull girder under steady water conditions, two main opposite forces apply on the structure: a downward force due to the own weight (considered as a force) and the upward force due to the water, as shown in Figure 1.6. An overall static equilibrium of the hull girder implies that the net resultant is null and the longitudinal center of buoyancy coincides with the center of gravity.

In operations, various load cases induced by varying amount and dispositions of cargo, consumables or ballast at any intermediate stage of the voyage, (departure, mid-voyage, arrival) must be considered. The load scenarios addressed by the certification rules cover operations such as seagoing conditions, loading and unloading, ballast water exchange situations, special operations in harbor, etc.

Figures 1.7 and 1.9 represent on the left the intensity of buoyancy force on an idealized beam ship under various conditions and, on the right

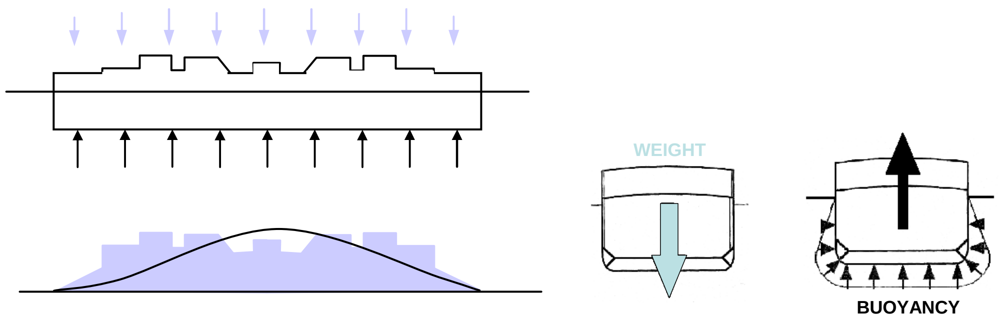


Figure 1.6: weight and buoyancy in a beam ship

a representation of the kind of deformations induced on the ship girder by these forces.

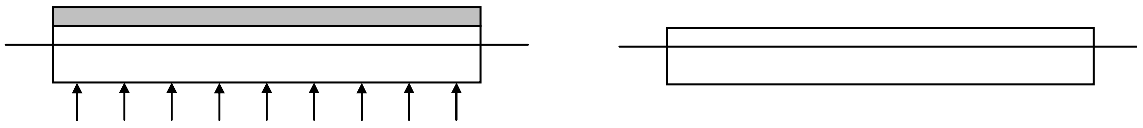


Figure 1.7: Idealized beam ship: buoyancy force and effect in still water (no deformation)

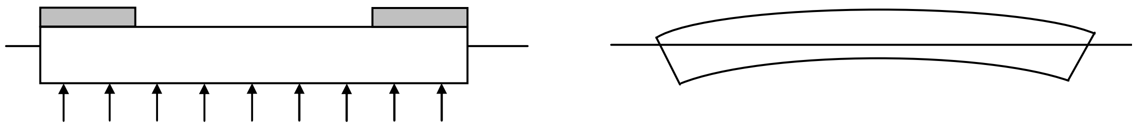


Figure 1.8: buoyancy force and hogging effects due to loading

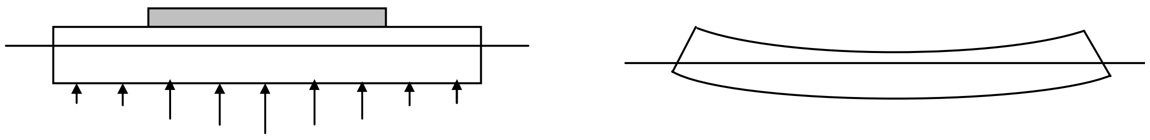


Figure 1.9: buoyancy force and sagging effects due to loading

When operating on the oceans, even in calm conditions, forces due to waves act on the vessel. It has been demonstrated by oceanographers Pirson (1952), Pirson et al. (1955) that the surface of the ocean, despite being highly irregular and totally random, can be locally represented as a superposition of a large number of sinusoidal waves, with different length, height, directions and random phase differences. This finding is a key element that allows us to represent mathematically the surface ocean and allows the use of statistical methods to predict the maximum wave loads encountered by a ship in its lifetime. Figures 1.10 and 1.11 represent the intensity of buoyancy force and the kind of deformation induced by wave effects.

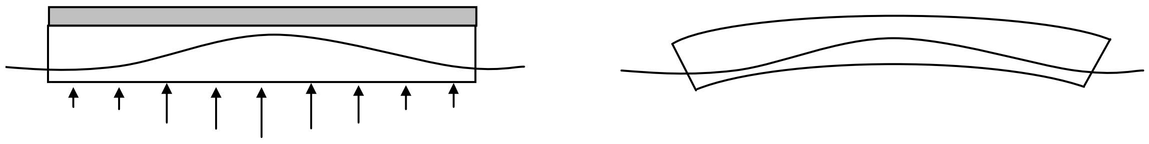


Figure 1.10: buoyancy force and hogging effects due to waves

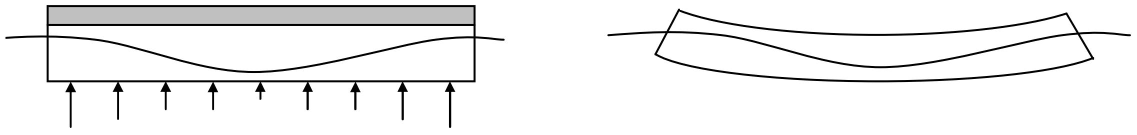


Figure 1.11: buoyancy force and sagging effects due to waves

Figures 1.7 to 1.11 represent on the left the intensity of buoyancy force on an idealized beam ship under various conditions and, on the right a representation of the kind of deformations induced on the ship girder by these forces. Figure 1.7 represents the buoyancy forces distribution for an idealized beam ship with an uniform loading in still water. In the next two figures one can see the hogging effect due to the ship load (Figure 1.8) and the hogging effect to the waves (Figure 1.10). Sagging effects due to the ship loading or due to the wave effects are represented respectively on Figures 1.9 and 1.11.

1.2 Mathematical models for structural design

We consider here the structural design of large passenger ships such as cruise liners or carriers. As stated in the previous sections, preliminary structural design consists in defining the optimal scantling of the constitutive elements of a structure, given the overall dimensions and form of the hull. Ship modeling has been extensively studied at the ANAST department of the University of Liège. Two models have been developed: a three-dimensional (3D) model and a two-dimensional (2D) model.

1.2.1 Three-dimensional model

In the 3D model cylindrical shells are used as the reference panels and stiffened plates are considered as a special case of the more general cylindrical shells (with a very large radius). The linear elastic analysis of stiffened structures implies the resolution of differential equations, a detailed presentation of the differential equations of stiffened panels and of Fourier series expansion used to solve them is presented in Rigo (2005).

1.2.1.1 Variables

The metallic structures are ship sections produced by assembling dozens of stiffened panels. The design variables defined for each panel p are chosen among the thickness of the stiffened plates, the spacing between stiffeners and the dimensions of the stiffeners. It is of common use to not consider the thickness of flanges t_x, t_y as variables and set these dimensions proportional to the flange length. The design variables are presented in section 1.1.3 and classified into nine categories :

δ^p	plate thickness,
h_x^p, d_x^p, w_x^p	stiffeners web length, web thickness, flange length
Δ_x^p	spacing between two stiffeners fitted along the x direction
h_y^p, d_y^p, w_y^p	frames web length, web thickness, flange length
Δ_y^p	spacing between two transverse frames fitted along the y direction

The nature of these variables is intrinsically discrete: the thickness of each panel has standard values, the number of members assembled on each plate must be an integer, and the member shapes are selected from catalogs. For practical reasons, some of these variables may be considered as continuous, but the discrete nature of the remaining ones must be explicitly taken into account in the design phase.

1.2.1.2 Objective

For the conception of a structure, the structure weight, the moment of inertia and the production cost are of critical importance. Minimizing the weight of the steel structure, maximizing its moment of inertia to increase stability and minimizing production cost are conflicting considerations simultaneously considered by the shipyards. In this section we present the mathematical formulation of these quantities, these three quantities can be used either as objective or constraints in an optimization problem, optimization models are discussed in section 1.3.

The weight of the structure is the sum of each panel weight, expressed as follows :

$$Weight = \quad \gamma \, \mathrm{L} \sum_p \mathrm{B^p} \left\{ \delta^p + \frac{h_x^p d_x^p + w_x^p \mathrm{t_x^p}}{\Delta_x^p} + \frac{h_y^p d_y^p + w_y^p \mathrm{t_y^p}}{\Delta_y^p} \right\} \tag{1.1}$$

with

L = length of the panel according to the X coordinate (m)

B breadth of the panel according to the Y coordinate (m)

γ = specific weight (N/m^3)

The cost is either the cost of materials or the complete fabrication cost. The cost of materials is directly derived from the weight function as a sum of each element cost. For each element, the weight is multiplied by the relevant material cost and some corrective factor is applied, according to the element type: plate, longitudinal stiffeners, transverse members,... A complete description of the cost function is described in Rigo (2001a,b) the cost function shows as follows :

$$F_C = \sum_p F_c^p \tag{1.2}$$

$$F_c^p = \mathrm{a}_1 \, \delta + \mathrm{a}_2 \, \delta + \mathrm{a}_3 + \mathrm{b}_1 \frac{d_x^2 \, h_x}{\Delta_x} + \mathrm{b}_1 \frac{d_x w_x \mathrm{t_x}}{\Delta_x} + \mathrm{b}_2 \frac{d_x \, h_x}{\Delta_x} + \mathrm{b}_2 \frac{w_x \mathrm{t_x}}{\Delta_x} + \mathrm{b}_3 \frac{d_x}{\Delta_x} + \mathrm{b}_4 \frac{1}{\Delta_x} \tag{1.3}$$

A total production cost can also be taken as an objective function in the LBR-5 software. This method takes into account detailed production costs of the shipyard: production methods available for each type of element, operational cost for all operations depending on the working area where the operation is performed (for example : welding in a pre-assembly stage is cheaper than welding in the dry dock, when the structure is assembled), and other efficiency parameters specific to the shipyard. This very detailed cost function is available as a code module in LBR5 and will not be considered in this work as its usage would not imply any modification in the methods presented here.

The moment of inertia represents the propensity of the structure to resist external forces without moving, unlike the weight and the cost, this objective is to be maximized. The moment of inertia is the weighted sum of each panel weight (including stiffening elements) multiplied with the distance of the element to the center of inertia of the whole structure.

$$I_{xx} = \sum_p m^p r^p \tag{1.4}$$

with
$m_p = \gamma \text{ L } \text{B}^{\text{p}} \left\{ \delta^p + \frac{h_x^p d_x^p + w_x^p \text{t}_\text{x}^\text{p}}{\Delta_x^p} + \frac{h_y^p d_y^p + w_y^p \text{t}_\text{y}^\text{p}}{\Delta_y^p} \right\}$ the panel weight and
r_p the distance from the axis of the panel p to the neutral axis of the structure.

Multiple objectives are also considered, as a combination of the three objectives above. Minimum weight, minimum cost and maximum moment of inertia are conflicting objectives and the subsequent multicriterion optimization problem is presented in section 1.4.

1.2.1.3 Constraints

The ship hull is submitted to technological, geometric and structural constraints. Considered mathematically, these are bound, linear and nonlinear constraints. It worth noting here that that the objective functions are often considered as constraints, only one of them being considered as the objective. We now proceed to the description of the other type of constraints.

Technological constraints define the admissible interval of the continuous variables and provide us with a minimum and maximum admissible value of the discrete variables. These constraints arise from section availability and fabrication requirements. As an example, plate thickness is limited below to prevent corrosion problems and limited above due to manufacturing or handling capabilities. Geometric constraints restrict the proportion between one variable and another, in order to guarantee a functional, feasible and reliable structure. Structural constraints model the mechanical behavior of the ship in response to external forces, these constraints have nonlinear behaviors. The structural constraints formulation depends on the modeling choices made by the designer. This topic is detailed in section 1.2.1.3.

Discreteness constraints In this work, we consider that $h_{z_1}^p$, $d_{z_1}^p$, $w_{z_1}^p$, Δ_x^p, $h_{z_2}^p$, $d_{z_2}^p$ and $w_{z_2}^p$ are discrete variables with generic notation y. The variable $\Delta_{z_2}^p$ is continuous - i.e. it can take real values - and noted x. (Remember that flange thicknesses $t^p_{z_1}$ and $t^p_{z_2}$ have fixed values and are thus not variables of the mathematical model.)

For each discrete variable y a set of admissible discrete value D is generated. Among the y variables, we distinguish the dimensional variables and the spacing variables in the creation of the admissible sets. For dimensional variables (i.e. the thickness of panels, stiffeners and frames; and the length and width of stiffeners and frames) a minimal value y_{min}, a maximal value y_{max}, and a step s are given and we generate the elements of D with the formula below :

$$y \in D\text{with}D = \{y_{min}, y_{min} + s, y_{min} + 2 * s \ldots_{min} + n * s\} \tag{1.5}$$

The sets of admissible values for spacing variables are defined considering that on each panel, the members (stiffeners and frames) have to be equidistantly placed and the number of members on a plate has to be an integer. For each panel p we define the set D^p of discrete admissible values for $\Delta\ ^p_{z_1}$ as follows:

$$y \in D^p\text{with}D^p = \{y^p_{min}, \ldots, y^p_{max}\} \tag{1.6}$$

$$y^p_{min} = \left\lceil \frac{width^p}{n^p} \right\rceil_{(int)} \tag{1.7}$$

$$y^p_{max} = \left\lfloor \frac{width^p}{n^p} \right\rfloor_{(int)} \tag{1.8}$$

Where $width^p$ is the panel dimension and y^p_{min} and y^p_{max} are respectively the minimal and maximal spacing of the members given by the designer for the panel. We use the symbol $\lceil x \rceil_{(int)}$ to denote the first integer superior or equal to x and$\lfloor x \rfloor_{(int)}$ to denote the first integer inferior or equal to x.

Equality constraints In the ship hull, adjacent panels must sometimes have similar characteristics, as for example identical plate thickness or stiffener spacings. To this end, equality constraints are defined and link variables by pairs. For each pair, a dependent variable and an independent variable are defined. A preprocessing step replaces dependent variables with their formulation in term of independent variables. In consequence, equality constraints and dependent variables are thus not considered in the sequel.
A complete list of equality constraints is given in 1.1.3, for example the formulation of equality between longitudinal stiffener spacing of two stiffened panels (i and j) vertically aligned is expressed as:

$$\Delta^i_y = \Delta^j_y \tag{1.9}$$

Bounds constraints Bounds are defined for each variable of the model. The lower bound values are usually determined by technical limitations (for example a lower bound for a thickness variable to limit the impact of corrosion) and the upper bounds are usually used to handle production requirements. Minimal net thickness of plates depends on the location of the panel in the structure and is determined according to certification rules (e.g. Bureau Veritas rules), a maximum thickness is defined for handling purposes, it is written as:

$$d^p_{min} \leq d^p \leq d^p_{max} \tag{1.10}$$

Geometrical constraints These constraints limit the values of some ratios between the design variables to ensure that the structure is functional, feasible and reliable. Proportions are defined between variables of frames, of stiffeners, and between frames and stiffeners to ensure compatibility between their scantlings (for example in standard cases frames are taller than stiffeners: $h^p_{z1} - h^p_{z2} \leq 0$).

The formulations originate from certification rules and norms. A complete list of constraints available for selection is given in Appendix A.0.2, coherent groups of constraints have been developed in the literature and can be selected to build a model. For example the flange width over web height ratio is such a predefined subset proposed by Owen Hugues (Hughes (1988)) for T shape and L shape stiffeners:

For T shape members (O. Hughes' set)

$$0.625\, w^p_{z1} - h^p_{z1} \leq 0 \tag{1.11}$$
$$h^p_{z1} - 2.5\, w^p_{z1} \leq 0 \tag{1.12}$$

For L shape members (O. Hughes' set)

$$1.25\, w^p_{z1} - h^p_{z1} \leq 0 \tag{1.13}$$
$$h^p_{z1} - 5\, w^p_{z1} \leq 0 \tag{1.14}$$

Structural constraints As external loads and forces are applied to the structure, some resultant effects such as displacements, deformations and stress occur. Given a structural design, compression stress, shear stress, bending moments, torsion moments are computed at a number of points on each element of the structure and must not exceed limit values. The constraints of the model implement mathematical models from the literature, and are the following :

- Absolute displacements
- Minimum plate thickness (buckling : Hughes)
- Relative displacements
- Plate yielding (von-Mises)
- Ultimate strength of stiffened panels (Paik)
- Frame yielding (von-Mises)
- Stiffener/longitudinal yielding (von-Mises) and deflection
- Girder yielding (von-Mises)

Maximum admissible values of the constraints are set such as to avoid the apparition of physical phenomenon such as yielding, buckling, cracks... or ruin.

Structural constraints may be distinguished according to the extent of structure which is involved: local constraints relate to some specific panel, frames and transverse stiffening and overall constraints relate to the entire prismatic structure (box girder).

The generic mathematical expression of these constraints is:

$$u \leq u_{max} \quad : \text{limitation on displacement} \tag{1.15}$$
$$\sigma_a \leq \sigma_{a\,max} \quad : \text{limitation on bending stress} \tag{1.16}$$
$$\tau \leq \tau_{max} \quad : \text{limitation on shear stress} \tag{1.17}$$

A structural model include a subset of these generic constraints, they are applied at various points on each elementary plate (see figure 1.12), which is the unstiffened plate included between two frames and two longitudinals. The computation points are shown on the following figure.

A structural analysis is applied to compute the value of stress and displacement at each point. The analytic methodology used to perform the analysis relies on the approximate resolution of differential equations and is summarized in Appendix B.

The bounding values are calculated using analytical formula and some security coefficients. In this model the structural constraints lead to systems of differential equations (cfr Appendix A.0.2) and to a implicit formulation of the model.

Practically a set of structural constraints for a given instance of ship structural analysis is built by first selecting the relevant type of constraints to be considered among the available set of constraints, then for each type of

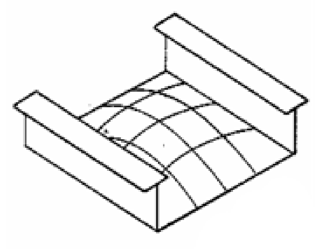


Figure 1.12: Points of computation on an elementary plate

constraint by choosing the method , the code or rules to perform the analysis, and finally to enumerate the points where the constraints will be computed.

1.2.1.4 Behavior models

Structural constraints are closely related to behavior models that have to be carefully selected for the formulation of the constraints appropriate to the structure at hand. In this work, ship structures are modeled as assemblies of panels stiffened along two perpendicular directions (longitudinal stiffeners and transverse frames). The behavior model associated with structures composed of stiffened panels is characterized by a set of well known differential equations (Donnell, Von Karman and Jenkins), that can be solved using for example Taylor series expansions. In this case, the behavior models are so complex that it is not possible to explicitly express the relation between the parameters studied (deflection, stress..) and the design variables.

1.2.1.5 Load cases

A scantling is feasible if the structure is able to withstand some typical loadings and sea conditions without deformation, break or ruin. A load case is composed of a set of forces that are applied at various places on the hull. These forces result from sea conditions that may be encountered by the ship (still water, frontal or side waves, etc.) and various loading scenarios (empty partially loaded, full loaded, ballast, etc.).

To establish compliance tests for ship certification, standard load cases have been defined, their elements are the sea loads (still water, waves with an associated probability of occurrence), the internal loads (liquid, ballast, bulk etc.), the local loads (lateral pressures due to still water and wave for example) and the global loads (due to sea loads, inducing hogging, sagging, horizontal and torsion bending moments and shear forces). “Standard” load cases have been defined using the rules of independent organizations of certification (Bureau Veritas etc.).

1.2.2 Two-dimensional model

The two-dimensional (2D) model is based on beam theory, in this approach the ship hull is considered as a beam with a straight and constant section. The material is homogeneous and in its elastic domain. By comparison with the three dimensional model, the Y dimension is not considered and the variables associated with the Y direction are not part of the model. We consider the weight or the cost as the objective function. The moment of inertia, the section modulus of the hull girder are global constraints for this model along with the gravity center shift.

1.2.2.1 Variables

The variables of a 2D model have to be selected in a subset of the design variables presented in section 1.2.1.1. For each panel component, there are five scantling design variables, that can possibly be selected as optimization variables:

$\delta =$ plate thickness (m),

$(h, d, w)_x =$ dimensions of web and flange of the longitudinals/stiffeners fitted along X

$\Delta_x =$ spacing between two longitudinals/stiffeners fitted along X

1.2.2.2 Objective

The objective functions are similar to those presented in the 3D model, taking off the variables associated with the third dimension. Using the same notation as in the previous section, we have:

The weight

$$F_W = \gamma \, \mathrm{L} \sum_p \mathrm{B}^{\mathrm{p}} \left\{ \delta^p + \frac{h_x^p d_x^p + w_x^p \mathrm{t}_{\mathrm{x}}^{\mathrm{p}}}{\Delta_x^p} \right\} \tag{1.18}$$

The cost

$$F_C = \sum_p F_c^p \tag{1.19}$$

$$F_c^p = \mathrm{a}_1\,\delta + \mathrm{a}_2\,\delta + \mathrm{a}_3 + \mathrm{b}_1 \frac{d_x^2\, h_x}{\Delta_x} + \mathrm{b}_1 \frac{d_x w_x \mathrm{t}_{\mathrm{x}}}{\Delta_x} + \mathrm{b}_2 \frac{d_x\, h_x}{\Delta_x} + \mathrm{b}_2 \frac{w_x \mathrm{t}_{\mathrm{x}}}{\Delta_x} + \mathrm{b}_3 \frac{d_x}{\Delta_x} + \mathrm{b}_4 \frac{1}{\Delta_x} \tag{1.20}$$

1.2.2.3 Constraints

The 2D model figures the same constraints of discreteness, equality, bounds and geometry than the 3D model, where the variables linked to the third dimension are removed, as has been done for the objective functions. In this model, structural constraints rely on a two-dimensional method (beam theory) their nonlinear formulations are provided by the International Association of Classification Societies (IACS) and Bureau Veritas (BV). Six types of constraints are available and listed hereunder. We given now the formulation of the constraints on the moment of inertia, the section modulus of the hull girder and the gravity center shift. These constraints relate to the entire structure and involve all the variables of the model, they are referred to as "global constraint" as opposed to "local constraints" that apply locally on each panel of the structure.

Global constraints

Gravity center shift : a limitation is given on the vertical shift of the gravity center of the structure. The position of the gravity center is given by the sum of the mass of each panel multiplied by the distance of the panel to the neutral axis, divided by the total mass of the panels. The following formula are used respectively to compute the panels mass and the gravity center position.

$$m_p = \gamma\ L\ B^p \left\{ \delta^p + \frac{h_x^p d_x^p + w_x^p \mathrm{t}_\mathrm{x}^\mathrm{p}}{\Delta_x^p} + \frac{h_y^p d_y^p + w_y^p \mathrm{t}_\mathrm{y}^\mathrm{p}}{\Delta_y^p} \right\} \tag{1.21}$$

$$GC = \frac{\sum_p\ m^p r^p}{\sum_p\ m^p} \tag{1.22}$$

Moment of inertia : this moment represents the propensity of the structure to resist to external forces without moving. For a rigid structure consisting of N panels of masses m_p with r_p the distance from their own axis to the neutral axis, the total moment of inertia equals the sum of the moments of inertia of the panels and is constrained to have a minimum value.

$$I_{xx} > I_{xx\ min} \tag{1.23}$$

$$I_{xx} = \sum_p\ m^p r^p \tag{1.24}$$

with:
m_p the panel weight as previously defined and
r_p the distance from the axis of the panel p to the neutral axis of the structure.

Hull girder minimum section modulus : a minimum section modulus is imposed for the panel located at the maximal distance of the gravity center. The section modulus of a panel p is the global moment of inertia divided by the largest distance (d) between any point of the panel and the position of the gravity center.

$$\frac{I_{xx}}{d} > I_{xx/v\ min} \tag{1.25}$$

Local constraints Six structural constraints must be verified at various points of each panel of the structure and for each load case. They are defined according to IACS requirements and reflects requirements on the hull girder strength, except for the last one, taken from Bureau Veritas rules that express local strength conditions:

Bending / shear strength:

- ○ *ic10* : Bending strength of plate: $\sigma_x \leq \sigma_a$
- ○ *ic19* : Shear strength: $\tau_{xy} \leq \tau_a$

Buckling strength:

- ○ *ic14* : Compressive buckling of plate: $\sigma_a \leq \sigma_C$
- ○ *ic20* : Shear buckling of plates: $\tau_a \leq \tau_C$
- ○ *ic37* : Compressive buckling of stiffeners: $\sigma_a \leq \frac{1}{\beta}\sigma_C$

Local strength (BV rules):

- ○ *ic38* : Bending strength of stiffeners: $\sigma \leq \frac{R_y}{\gamma_R \gamma_m}$

A detailed formulation of the constraints is given in Appendix C.1. As the ship structure is composed of interacting panels, the constraints show a mathematical structure that will be further developed and exploited in chapter 3.

It is worthwhile to point out that the 2D and 3D models are not equivalent: the set of design variables and the expressions of the structural constraints differ and, although they can be applied to identical ship structures, results can not be compared.

In the two following chapters we propose heuristic methods for structural optimization problems based on these models and provide results for real applications. Chapter 2 address the optimization of the three dimensional (3D) model featuring black box functions and time-expensive structural analysis, in chapter 3 we develop a method based on decomposition of the problem, that relies on the mathematical structure of the model's equations.

1.3 Ship structural optimization

Structural optimization is a domain that has been extensively studied for decades in the civil works, maritime and aerospace industries since decades.

In the scantling design of a passenger ship, the challenge is to optimize the scantling of ship sections in order to minimize various characteristics as the production cost, the weight, the moment of inertia or a combination of these.

The problem of structural optimization takes the topology of the structure as a given information: the number and the position of the panels, of the pillars and eventually the characteristics of the double hull of the ship are given. One has to optimize the thickness of the panels and the number and dimensions of the members that will be fixed on each panel, in order to improve the rigidity of the assembly. Many numerical methods to compute the structure strength and resistance to external solicitations (e.g. loading, forces applied by the waves,...) arose in the last decades, for ship structural analysis a complete reference is found in Hughes (1988).

In naval industry, the structural optimization problem arises in the early design phase of a project. Given a vessel overall dimensions and form, structural optimization consists of defining the scantling of the structure's constitutive elements so as to minimize its total weight or cost, while taking weight, robustness and security issues into account. This preliminary design phase always poses difficult problems to designers, as they have to make the most adequate choices within a very short period of time. Outside of the naval industry, such preliminary studies take place daily in numerous industries working on large projects where the product is unique and has to be custom-designed at the very beginning of the project, often before the client's order is placed (e.g., in aeronautics). The decisions made during the design phase have a major impact on the final structure and on its production cost.

When using three dimensional detailed models, the constraints of structural design problems are defined by implicit functions which model mechanical properties. Therefore, structural analysis uses computationally expensive numerical methods such as mesh models, finite element methods, or simulation to verify the constraints. A typical run of these methods may take several minutes, or even close to one hour for very large instances. This constraints formulation lead us to consider their computation code as black-box software.

Most often in structural optimization, the variables are allowed to take any real value. Practically, it is economically more interesting to use standard elements, whose dimensions are chosen in provider's catalogs. For example, standard thicknesses of plates could be 7, 8.5, 10, 11.5 and 13 mm and choosing a thickness of 10 would be less expensive than asking for tailor-made panels of dimension 9.5 mm. Setting costs are likely to result in higher production costs even if less steel is used.

Even without the discrete variables feature, the continuous version of these problem is highly non-convex, and the evaluation of a solution (i.e., a structural analysis) is time consuming. Thus, to sum up, the problems considered in this thesis are large-scale structural optimization problems where some of the variables are restricted to assume *discrete* values only. We focus on structural optimization problems which simultaneously display four main features:

- large scale, with several hundred variables and constraints;
- mix of continuous and discrete variables;
- constraints defined by implicit functions;
- expensive computational time for each evaluation of the constraints (3D model only)

The algorithms proposed in this thesis carry out a limited exploration of the space of discrete solutions. They provide one or more discrete solution(s) and require a relatively small number of calls to the structural analysis and optimization software, namely LBR-5 (see Rigo (2002)) which handles the continuous version of the structural optimization problem. The effectiveness of the algorithms is demonstrated on real-world data for the structural optimization of large passenger ships.

1.4 Multicriterion optimization

Multicriterion optimization is rooted in economics . An early approach of economic equilibrium has been developed in F. Y. Edgeworth's book *Mathematical Physics* (Edgeworth 1881): a first definition is given of an multicriterion optimum solution for a multiutility optimization problem. The reader may refer to papers by Stadler (1979, 1987) for a historical review of multicriterion optimization.

Engineering design often involves multiple conflicting criteria, objectives or goals. The design team has to minimize the weight, maximize the moment of inertia and minimize the production cost. Such conflicting objectives draw the guidelines for the designer to select one "best" design among a set of feasible designs. Dauer and Stadler (1993), Keane and Nair (2005) give overviews of multicriterion optimization approaches in engineering and historical perspectives.

The selection of a final solution in multicriterion optimization is often based on a weighted sum of the criteria: a single objective function is defined using some weighting coefficients for each objective. This straightforward method poses the difficult question of the choice of the weighting which reflects the preference of the designers for one or another objective. The most common definition of an optimum used in multicriterion optimization is Pareto optimality. This term was first articulated by the economist V. Pareto in 1906 and is also called today Edgeworth-Pareto optimality. A solution is Pareto optimal if it satisfies the constraints and is such that no criterion can be improved without causing at least one of the other criteria to decline. To elect a final solution from the Pareto set, one has to sate a compromise between the objectives, methods are presented in the survey of Marley and Arora (2004). In structural optimization, a single result is sought that can be implemented in the design. This result should be an effective compromise or trade-off among the conflicting criteria, often this result is based on additional considerations that are not part of the optimization model.

When the variables are discrete, multicriterion combinatorial problems have a finite set of feasible solutions that can theoretically be enumerated to identify all Pareto optimal solutions. However, the number of feasible solutions may grow exponentially with the number of variables and sometimes so does the number of Pareto optimal solutions. Multicriterion discrete optimization methods aim to compute approximations of discrete sets of Pareto optimal solutions.

We give the following definition of a single criterion structural optimization problem :

$$\begin{aligned} &min\, F(x) = F_1(x) \\ &\text{subject to} \\ &\qquad h_i(x) = 0,\ i = 1, \dots, I \\ &\qquad g_j(x) \leq 0,\ j = 1, \dots, J \end{aligned} \tag{1.26}$$

where $x = [x_1, x_2, \dots, x_N]^T$ is the vector of design variables and there is a single optimization criterion or objective function $F_1(x)$ that depends on x. The problem is usually subject to I equality constraints and J inequality constraints $h_i(x)$ and $g_j(x)$, respectively, that also depend on the design variables in the vector x.

The multicriterion optimization problem involves $K > 1$ criteria and can be formulated as follows:

$$\begin{aligned}
&min\, F(x) = [F_1(x), F_2(x), \ldots, F_K(x)] \\
&\text{subject to} \\
&\qquad h_i(x) = 0, i = 1, \ldots, I \\
&\qquad g_j(x) \leq 0, j = 1, \ldots, J
\end{aligned} \tag{1.27}$$

where there are K optimization criteria $F_1(x)$,..., $F_K(x)$. This problem does not have a unique solution as conflicts usually exist among the multiple criteria. To achieve an effective compromise among competing criteria a preference function may be used as a global criteria:

$$P[F(x)] = \left\{ \sum_{k=1}^{K} w_k \left[\left| (F_k(x) - F_k^0)/F_k^0 \right| \right]^{\rho} \right\}^{1/\rho} \tag{1.28}$$

$$\sum_{k=1}^{K} w_k = 1 \tag{1.29}$$

where F_k^0 is the value of the criterion F_k obtained when that criterion is the single criterion used in the optimization - the best that can be achieved with that criterion considered alone. The preference function $P[F(x)]$ replaces $F(x)$ in 1.26 for numerical solution, and concepts such as the weighted sum, the min-max and the nearest-to-the-utopian solutions are derived from these optimization problems. The weighted sum solution results from 1.28 when $\rho = 1$, whereas the nearest to the utopian solution results when $\rho = 2$ and the min-max solution when $\rho = \infty$. The numerical implementation for the min-max solution uses the equivalent of 1.28 with $\rho = \infty$,

$$P[F(x)] = \max_k \left[w_k \left| \left(F_k(\mathbf{x}) - F_k^0)/F_k^0 \right) \right| \right] \tag{1.30}$$

Moreover, a solution could be obtained for a number of values of ρ and then the design team could decide which solution best represents the design intent.

On this basis, multicriterion optimization method for naval structural optimization has been developed by Richir, Caprace, Losseau, Bay, Parsons, Patay, and Rigo (2007b) and included in the structural analysis and optimization software LBR-5. It includes the discrete multicriterion optimization method presented in section 2.6 used to find discrete optimal solutions to problem 1.27 with objective function 1.28.

Chapter 2

Local search heuristics for large-scale discrete structural optimization with expensive black-box evaluations

2.1 Introduction

This chapter considers large-scale structural optimization problems featuring discrete variables, as well as nonlinear implicit constraints which can only be evaluated through time-expensive computations. A prominent application consists in the preliminary structural design of large ships, where many of the variables take their values in discrete sets which model standard element dimensions to be selected from catalogs, and where the evaluation of the constraints involves a complex structural analysis performed by black-box software.

The resulting large-scale nonlinear combinatorial problems are particularly hard, and even finding a discrete feasible solution may prove challenging for some instances. In this chapter, we propose two heuristics that combine local search methods and a sequential optimization method based on approximations of the implicit constraints. The heuristics are applied to the structural optimization of several large ships. For these instances, the heuristics provide discrete feasible solutions whose value is close to the optimal value of the continuous relaxation obtained by disregarding the discrete nature of the variables.

Thus, to sum up, the chapter concentrates on structural optimization problems which simultaneously display four main features:

- large scale, with several hundred variables and constraints;

- mix of continuous and discrete variables;
- constraints defined by implicit functions;
- expensive computational time for each evaluation of the constraints.

2.2 Literature review

Structural optimization problems like $P(x, y)$ are large-scale, discrete optimization problems with implicit constraints and time-expensive function evaluations.

In this section, we review variants of $P(x, y)$ which have been considered in the literature.
To solve $P(x, y)$, many approaches proposed in the literature rely on the solution of continuous relaxations like $P^c(x, y)$, or consider continuous subproblems like $P^{K,\alpha}(x, y)$. The reader may consult Grossman and Kravanja (1995) for examples of other relaxations.

In this section, we review variants of $P(x, y)$ which have been considered in the literature, but for which (at least) one of these four important features is missing.

Continuous versions of $P(x, y)$ arise frequently in structural optimization. In this framework, convex linearization is widely used to provide a conservative approximation of the objective function and of the constraints; see e.g. the *Conlin* method in Fleury and Braibant (1986), Fleury (1989a,b), Zhang and Fleury (1997). These authors describe an efficient sequential convex programming (SCP) method based on *Conlin* that iterates the following steps: a convex linearization of the objective and constraints generates the approximation and a dual resolution method is used to find an optimal solution to the approximation. The solution of the approximation is the starting point of the next iterate. This efficient approach allows to obtain an optimal solution of continuous structural optimization problems after a few (10 to 15) iterations (Fleury and Braibant 1986). For the type of problems that we consider, this method has proven to be efficient on large size instances (see Rigo 2001b, Rigo and Fleury 2001) and is used to solve the continuous subproblems $P^c(x, y)$ and $P^{K,\alpha}(x, y)$ arising in our heuristics; see Section 2.4.

Other methods related to SCP and used in continuous structural optimization include the method of moving asymptotes (MMA) introduced by Svanberg (1987) (see also Bruyneel et al. 2002) and sequential quadratic programming (SQP) briefly described in Zhang and Fleury (1997). Derivative free optimization (DFO) methods have been developed more recently and use approximations (or surrogates) to accelerate engineering design optimization (see e.g. Huyer and Neumaier 1999, Booker et al. 1999). To guarantee the global convergence of the optimization process, trust regions are used in combination with various methods and maintain the quality of the approximation at each step of the procedure search (for example, see Alexandrov and Lewis 2000, Conn et al. 2000, Exler and Schittkowski 2007).

Implicit (or black-box) optimization problems arise when the algebraic formulation of some or all functions which model the behavior of the structure (objective and constraints) is not given. These implicit functions are evaluated by black-box software, and optimization approaches frequently rely on approximations (or surrogates), implying that they do not guarantee global optimality. Approximation methods for implicit problems differ according to the nature of the approximating functions, and in the way the approximations are embedded in the optimization scheme. A local approximation is valid in the vicinity of a point in the solution space, while a global approximation is valid in large regions of the space (see Barthelemy and Haftka 1993). Linear, quadratic and convex approximations are most commonly used as they lead to explicit problems for which numerous efficient methods are available; see again Rigo (2001a) or Rigo and Fleury (2001) for examples.

The time needed to evaluate the implicit functions influences the choice of the optimization method to be used. Methods applicable for fast evaluation problems include global optimization methods (see Neumaier 2004), or natural methods such as genetic algorithms (e.g. Ali et al. 2003, Jenkins 1997, Turkkan 2003).

For problems with expensive black-box evaluations when the dimension of the solution space is not too large, approximation methods are often applied (see for example Torczon and Trosset 1998). Successful global approximation methods are, for example, response surface methods (RSM) (Jones 2001, Myers and Montgomery 2002), radial basis algorithms (Regis and Shoemaker 2005, Edvall and Holmstrom 2008), and Kriging models (Simpson et al. 2005, Davis and Ierapetritou 2009). Such methods have proven global convergence on problems with two to six variables. In recent years, approximation methods have sometimes been combined with stochastic methods like genetic algorithms or evolutionary algorithms for large scale problems, with the objective of developping "near-optimal" discrete designs. Such stochastic methods may require thousands of structural analyses to locate near-optimal solution. For large scale problems, the computational burden of the analyses may therefore prevent a straightforward application of stochastic methods. It is now common for approximations to be used in lieu of accurate analysis, so as to keep the optimization computation time within acceptable limits. For example, Ong et al. (2004) present a combination of surrogate and evolutionary algorithm in engineering design. In Zhou et al. (2007), trust region and radial basis function methods are embedded in the framework to combine local and global approximation schemes.

For similar problems with time-expensive function evaluations and black-box subproblems, another broad area of research includes Derivative Free Optimization (DFO) methods like pattern search (see Abramson et al. 2006a) and mesh adaptive direct search methods (Audet and Dennis 2006a); see Kolda et al. (2003) for a survey.

For mixed problems involving both discrete and continuous variables, like $P(x, y)$, a classical family of optimization algorithms relies on branching schemes which recursively split the original problem into more tractable subproblems.

For example, Olsen and Vanderplaats (1989) apply linearization and mixed integer branch and bound. Bremicker et al. (1990) sequentially generate mixed-discrete linear subproblems which are solved by branch and bound; the authors mention that the method is adequate for problems with up to 20–30 discrete variables. Thanedar and Vanderplaats (1995) show that a nonlinear branch and bound method along with a robust nonlinear programming (NLP) algorithm turns out to be a viable approach for optimization problems involving a small number of discrete variables. For structural optimization problems, branching frameworks combined with convex linearization have been tested for implicit mixed integer NLP problems of small size by Beckers (1997). Still other general approaches combine an external strategy to drive a software that performs the time-consuming function evaluations, as in Brekelmans et al. (2005) or Fischetti and Lodi (2003).

Mixed-discrete optimization problems with expensive black-box evaluations appear in many complex engineering design problems where high-accuracy analysis models are used. Close to naval structure optimization, aerospace design problems like aerodynamic wing design (Keane and Nair 2005, Kumano et al. 2006), composite shell structure optimization (Bruyneel et al. 2002) or helicopter rotor design (Booker et al. 1998) require time expensive computational simulations (other various applications may be found in Keane and Nair (2005)). Besides engineering design, problems with expensive black-box evaluations also appear in biology (Alberto et al. 2004), engineering (Booker et al. 1998), water resource management (Hemker et al. 2008), etc. Some problems include simulation techniques to compute the implicit functions, e.g. for electric power restoration (Guikema et al. 2004), production and manufacturing (Tompkins and Azadivar 1995), shape optimization of automotive body frames (Yoshimura et al. 2005). These applications generally involve models with a rather small number of variables, say 20 to 30.

By contrast, the structural optimization problems for real ships involve hundreds of discrete and continuous variables, and thousands of implicit constraints, each evaluation of a solution is computationally expensive, and obtaining a local optimum is a challenging goal. Even in the continuous case (when computation time is not an issue), since the problems are nonconvex, only local minima can be achieved by most algorithms designed for large scale problems. When combinatorial aspects are also taken into consideration, the large number of dimensions of the solution space prevents the use of methods requiring extensive branching, or of global methods based on response surfaces or space mapping. In view of these limitations, the heuristics to be presented in the next section combine a truncated branching strategy to deal with the discrete variables, and a sequential approximation method to handle the expensive black-box subproblems.

The branching strategy is inspired from a "diving strategy" for mixed integer linear programming sketched by Pochet and Wolsey Pochet and Wolsey (2006) whereby, at each node of the branching tree, some variables are fixed and a subproblem is solved, without backtracking. The number of subproblems generated is limited as much as possible as their generation requires large computation time. The black box used for structural analysis is the LBR-5 software developed by Rigo (Rigo (1998),Rigo (2001a),Rigo (2002)).

2.3 Model

In this first model, a the complexity of the behavior models leads to the impossibility of explicitly drawing the relationships between the parameter studied (deflection, stress, etc...) and the design variables (element dimensions and position). The evaluation of these resultant effects (and of their derivatives with respect to the design variables) is possible at expensive computational cost using analytical approaches or finite element methods (FEM). Given the values of the design variables; the displacements, deformations and internal stress are computed for several loading scenarios.

The resulting structural optimization problem $P(x,y)$ can be written as follows:

$$\begin{array}{lll} \min & f(x,y) & \\ \text{s.t.} & g_i(x,y) \leq 0 & i = 1,...,I \\ & h_j(x,y) = 0 & j = 1,...,J \\ & x \in X,\ y \in Y & \end{array} \qquad P(x,y)$$

where x and y denote the vectors of continuous and discrete design variables, respectively. The vector x takes its values in a "box" $X = \{x \mid x \in \Re^n, x^L \leq x \leq x^U\}$, and y takes its values in a finite set $Y \subseteq \Re^m$ of the form $Y = Y_1 \times \ldots \times Y_m$, where each Y_k is a finite set of reals. An analytical expression of the functions f, g_i and h_j is not necessarily known; all these functions will be viewed as implicit nonlinear functions. We use the shorthand g for $(g_i, i = 1, \ldots, I)$, and h for $(h_j, j = 1, \ldots, J)$.

In ship structural design, the evaluation of the objective f is performed by software whose code may not be available and may use confidential databases. The computation of each g_i requires the solution of large systems of partial differential equations, finite element methods, or computer simulations. These methods allow evaluating the structural responses g_i (displacements, stresses) under various loading conditions, as well as their sensitivities with respect to the design variables. The constraints h_j represent nonlinear relations between variables, and their formulation may be explicit or not.

We call "structural analysis" the evaluation of f, g h, and their derivatives (or subgradients) at a point (x, y). For each value of the design variables (x, y) most of the functions f, g, h and their derivatives can only be estimated via time-expensive black-boxes computations.

The solution of discrete structural optimization problems often involves a continuous relaxation of $P(x, y)$, wherein the finite set Y is replaced by a convex box containing Y. More precisely, let $Y^c = \{\ y \mid y \in \Re^m,\ y^L \leq y \leq y^U\}$ where y_k^L and y_k^U are respectively lower and upper bounds on y_k in Y_k $(k = 1, \dots, m)$, so that $Y \subseteq Y^c$. Then, the *continuous relaxation* of $P(x, y)$ is the optimization problem $P^c(x, y)$:

$$\begin{array}{lll} \min & f(x,y) & \\ \text{s.t.} & g_i(x,y) \leq 0 & i = 1, ..., I \\ & h_j(x,y) = 0 & j = 1, ..., J \\ & x \in X,\ y \in Y^c. & \end{array} \qquad P^c(x,y)$$

Note that the optimal value of $P^c(x, y)$ is a lower bound on $P(x, y)$ but this bound may be hard to compute due to the nature of the implicit constraints.

In this chapter we will also consider subproblems in which some of the y-variables are fixed at admissible values, and the other y-variables are free and continuous. More precisely, for any subset of indices $K \subseteq \{1, \dots, m\}$ and fixed values α_k of the discrete variables y_k, $k \in K$, the problem $P^{K,\alpha}(x, y)$ is:

$$\begin{array}{lll} \min & f(x,y) & \\ \text{s.t.} & g_i(x,y) \leq 0 & i = 1, ..., I \\ & h_j(x,y) = 0 & j = 1, ..., J \\ & y_k = \alpha_k & k \in K \\ & x \in X,\ y \in Y^c & \end{array} \qquad P^{K,\alpha}(x,y)$$

where X and Y^c have been defined above. Note that $P^{K,\alpha}(x, y)$ is a problem in continuous variables only, and that $P^c(x, y) = P^{\emptyset,\alpha}(x, y)$. For any $K \neq \emptyset$, the optimal value of $P^{K,\alpha}(x, y)$ is an upper bound on the optimal value of $P^c(x, y)$. Here again, this value may be hard to compute.

We should stress at this point that in the structural optimization problem $P(x, y)$, the discrete variables y are numerical variables (defined on $\Re^m$), rather than arbitrary categorical variables (e.g., "small" or "large"). In particular, all the underlying functions f, g, h are defined everywhere in the subspace $X \times Y^c$, and their derivatives (or subgradients) can be computed in this region. Without these important features, it would not be possible to define the relaxations $P^c(x,y)$ or $P^{K,\alpha}(x,y)$, nor to implement the algorithms described below.

2.4 Algorithms

In this section, we propose two heuristics to solve structural optimization problems of the form $P(x,y)$ involving a large number of discrete variables and computationally expensive implicit functions. Both heuristics are based on branching schemes that generate sequences of continuous subproblems of the form $P^{K,\alpha}(x,y)$ by successively fixing groups of discrete variables y_k, $k \in K$. The variables are fixed by different rounding schemes to be described below.

The first heuristic is a Dive and Fix (D&F) algorithm (similar to the "diving heuristic" for mixed integer linear programming sketched by Pochet and Wolsey (2006). At each iteration, D&F incrementally fixes a group of variables; during the search a limited amount of backtracking is allowed to ensure convergence of the process toward a discrete feasible solution of $P(x,y)$. A good heuristic solution is frequently reached within a small number of subproblem generations.

The second heuristic is a Branch and Fix (B&F) algorithm: it is similar to the Dive and Fix algorithm but performs a broader exploration of the solution space by backtracking more intensively and generating more subproblems $P^{K,\alpha}(x,y)$. The B&F algorithm usually finds several discrete feasible solutions with a bounded number of subproblem generations.

In both heuristics, the main rationale for fixing *groups* of variables at each iteration, rather than *individual* variables, is to reduce the number of continuous subproblems to be solved and hence, the number of expensive structural analyses to be performed. The choice to use one or the other heuristic depends on the user's preferences in terms of the number of solutions required, and on the available amount of computing time.

Let us now proceed with a definition of the rounding schemes used by the heuristics. For any variable y_k ($k \in \{1,\ldots,m\}$) and any real $r \in \Re$, the notation $\lfloor r \rfloor$ denotes the largest feasible discrete value of y_k smaller than or equal to r:

$$\lfloor r \rfloor = \max\{\ y_k \in Y_k \mid y_k \le r\}; \tag{2.1}$$

we set $\lfloor r \rfloor = -\infty$ if the above value is not properly defined, i.e., if $r < y_k^L$. (Note that $\lfloor r \rfloor$ depends on k, but the appropriate value of k will always be clear from context.) Similarly, $\lceil r \rceil$ denotes the smallest feasible discrete value of y_k larger than or equal to r:

$$\lceil r \rceil = \min\{\ y_k \in Y_k \mid r \le y_k\} \quad (\text{or } \lceil r \rceil = +\infty \text{ if } y_k^U < r). \tag{2.2}$$

Three *rounding modes* are defined, namely *Closest*, *Down* and *Up*. The function $Round(r, rounding\_mode)$ respectively returns:

$$
\begin{aligned}
Round(r, Down) &= \lfloor r \rfloor, && && (2.3)\\
Round(r, Up) &= \lceil r \rceil, && && (2.4)\\
Round(r, Closest) &= \lfloor r \rfloor && \text{if } (r - \lfloor r \rfloor < \lceil r \rceil - r), && (2.5)\\
Round(r, Closest) &= \lceil r \rceil && \text{if } (r - \lfloor r \rfloor \geq \lceil r \rceil - r). && (2.6)
\end{aligned}
$$

2.4.1 Dive and Fix heuristic

The Dive and Fix heuristic is described as Algorithm 1 hereunder. Formally we start with a partition $\{K_1, \ldots, K_N\}$ of the set of indices of the discrete variables $(y_1, \ldots, y_m)$. At each step i of the algorithm $(i = 1, \ldots, N)$ an additional subset of variables y_k, $k \in K_i$, are fixed at values obtained by appropriately rounding the current solution, and the associated subproblem $P^{K,\alpha}(x, y)$ is solved, with $K = K_1 \cup \ldots \cup K_i$. Thus the number of time-expensive subproblems solved is proportional to the number N of subsets.

In further detail, the heuristic works as follows. Given a problem $P(x, y)$, the initial step is the solution of the continuous relaxation $P^c(x, y)$ by a black-box procedure *Solve* (to be discussed below) which takes as input an initial solution (x^u, y^u) provided by the user. (If *Solve* does not find a feasible solution, the algorithm stops.)

Then at the beginning of each iteration i, the next subset of indices K_i is selected and added to K, and the variables y_k, $k \in K$, are fixed at $y_k = \alpha_k$. The values α_k are obtained by rounding the current values y_k^{i-1} according to one of two rounding modes: either *Closest* or *Up*. The mode *Closest* is always tried first.

The procedure *Solve* is called again to solve the resulting subproblem $P^{K,\alpha}(x, y)$, starting from the initial solution $(x^{i-1}, \tilde{y}^{i-1})$ with $\tilde{y}_k^{i-1} = y_k^{i-1}$ for $k \notin K$ and $\tilde{y}_k^{i-1} = \alpha_k$ for $k \in K$. It returns a solution (x^*, y^*). If (x^*, y^*) is feasible, then the current solution is updated to this value and the algorithm moves to the next iteration. If (x^*, y^*) is not feasible, we consider that rounding the y_k-variables leads to an infeasible continuous problem, and we restart iteration i with the alternative rounding mode *Up*, the set of fixed variables $y_k, k \in K$, being unchanged.

If *Solve* does not find a feasible solution when the rounding mode is *Up*, then the whole D&F procedure fails. Barring this possibility (see below), after N iterations, all the y-variables have been fixed, D&F terminates and returns a discrete feasible solution $(x^N, y^N) \in X \times Y$.

Note that the global convergence of the D&F heuristic is not guaranteed; it depends in particular on:

a) the existence of feasible solutions for the continuous subproblems $P^{K,\alpha}$;

Algorithm 1 *Dive and Fix*;

```
$(x^u, y^u)$: initial solution
$\{K_1, \ldots, K_N\}$: a partition of the set of indices $\{1, \ldots, m\}$

$(x^0, y^0) := Solve(P^c(x, y), x^u, y^u)$
if $(x^0, y^0)$ is not feasible for $P^c(x, y)$ then
  fail and exit
end if
$i := 1$
$K := \emptyset$
$rounding\_mode := Closest$
repeat
  $K := K \cup K_i$
  for all $k \notin K, \tilde{y}_k^{i-1} := y_k^{i-1}$
  for all $k \in K, \alpha_k := Round(y_k^{i-1}, rounding\_mode)$ and $\tilde{y}_k^{i-1} := \alpha_k$
  $(x^*, y^*) := Solve(P^{K,\alpha}(x, y), x^{i-1}, \tilde{y}^{i-1})$
  if $(x^*, y^*)$ is feasible for $P^{K,\alpha}(x, y)$ then
    $(x^i, y^i) := (x^*, y^*)$;
    $i := i + 1$;
    $rounding\_mode := Closest$
  else
    if $rounding\_mode == Up$ then
      fail and exit
    end if
    $rounding\_mode := Up$ {$i$ and $K_i$ do not change}
  end if
until $i == N$
return $(x^N, y^N)$
```

b) the convergence properties of the optimization algorithm *Solve* which is used to handle these subproblems.

We discuss these issues hereunder.

2.4.1.1 Convergence

We initially assume that the continuous relaxation $P^c(x, y)$ has a feasible solution and that $Solve(P^c(x, y), x^u, y^u)$ returns such a solution $(x^0, y^0) \in X \times Y^c$: otherwise, searching for a discrete solution in $X \times Y$ is clearly meaningless.[1]

So, by induction, the current solution (x^{i-1}, y^{i-1}) can be assumed to be feasible at the start of every iteration i. When selecting a subset K_i of y variables and rounding y_k^{i-1}, $k \in K$, to the closest feasible discrete values, the D&F heuristic attempts to create a "near-optimal" discrete solution. However, this usually goes at the expense of feasibility and of optimality. The goal of the next optimization step, i.e. $Solve(P^{K,\alpha}(x, y), x^{i-1}, \tilde{y}^{i-1})$, is to restore feasibility while reducing the value of the objective function.

If this fails, i.e., if the rounding mode *Closest* does not lead to a feasible subproblem, then the i-th iteration is restarted using the rounding mode *Up*. An important implicit hypothesis of our approach is the following: we assume that for the structural design problem under consideration, the rounding mode *Up* always leads to a feasible subproblem when it is applied to a feasible solution (x^{i-1}, y^{i-1}). In the sequel, we refer to this as the *monotonicity hypothesis*. In the case of naval structures, this hypothesis is motivated by the empirical observation that if we start from a feasible design and we increase the dimension of the elements, then the resulting design also satisfies the structural constraints of the model, albeit at extra cost and weight[2]. The monotonicity hypothesis will be further validated by the computational experiments reported in Section 2.5.

Thus in practice, under the monotonicity hypothesis, the Dive and Fix heuristic always converges to a discrete feasible solution. If every *Closest* rounding fails to yield a feasible subproblem, then $2N$ continuous subproblems must be solved by D&F, using a multiple of $2N$ expensive structural analyses. If the user allows more time to find a better discrete solution then the next algorithm B&F can be used: it performs a denser exploration of the solution space, using a third rounding mode.

Note that the partition of the set of discrete y-variables induced by $\{K_1, \ldots, K_N\}$ also has an impact on the ability of the D&F algorithm to

[1]This is an assumption about the problem $P^c(x, y)$ itself, but also about the quality of the "external subroutine" *Solve*; we discuss this issue in Section 2.4.1.2.

[2]Similar hypotheses are likely to hold for other structural optimization problems. Of course, the respective roles of the directions "Up" and "Down" depend on the definition of the variables and may need to be reversed accordingly.

identify a feasible solution, as well as on the quality of this solution. As mentioned earlier, the main motivation for fixing variables in groups, rather than individually, is to limit the number of expensive continuous subproblems to be solved. A drawback of this approach, however, is that each rounding step considerably restricts the residual space of solutions. In our implementations, therefore, the discrete variables are considered in decreasing order of their influence on the objective and constraints: e.g., the variables in K_1 may be those for which the objective function and the constraints have the largest derivatives, and the variables in K_N would have the smallest derivatives. Thus fixing the first group of variables $y_k, k \in K_1$, has a deeper impact on the feasibility and on the quality of (x, y) than fixing the subsequent groups of variables. Typically, rounding the value of the variables in K_1 may produce a "strongly infeasible" solution; but this is counter-balanced by the fact that many variables y_k, $k \in K_2 \cup \ldots \cup K_N$, are still free at this stage and can be adjusted by the procedure *Solve* to restore feasibility. On the other hand, as the iteration counter increases, the subproblems $P^{K,\alpha}(x, y)$ have fewer and fewer free variables, but the starting solutions tend to remain "almost feasible" after rounding, since the rounded variables have a limited impact. We briefly return to this discussion in Section 2.5.4.

2.4.1.2 Solution of continuous relaxations

The optimization of the implicit continuous problems with time expensive function evaluations, $P^c(x, y)$ or $P^{K,\alpha}(x, y)$, must be performed by an appropriate optimization algorithm. We call this algorithm *Solve* and we denote by $Solve(P^c(x, y), x^0, y^0)$ or $Solve(P^{K,\alpha}(x, y), x^0, y^0)$ its output when running on problem $P^c(x, y)$ or $P^{K,\alpha}(x, y)$, respectively, given an initial solution (x^0, y^0) (see Algorithm 2).

The algorithm *Solve* could be any "subroutine" able to optimize the continuous version of $P(x, y)$. In our implementation, *Solve* is based on an iterative approximation scheme which includes structural analysis, the convex linearization and the dual solution scheme $Solve\_NLP$ proposed by Fleury and Braibant (1986) and Fleury (1989a); the elements of *Solve* are part of the LBR-5 software developed by Rigo (Rigo (1998, 2001b,a, 2002)), Rigo and Fleury (2001) (see also Hughes (1988) for an alternative to LBR5).

More precisely, *Solve* works as follows (see Algorithm 2). At each iteration of *Solve*, a black-box procedure performs the required structural analyses, i.e., it computes the value of all relevant functions (objective function and constraints) and of their partial derivatives at the current point (x^j, y^j). Then, a convex local approximation $\widetilde{P}^{K,\alpha}(x, y)$ of $P^{K,\alpha}(x, y)$ is built, wherein each function f, g, h is approximated by a convex linearization at the point (x^j, y^j). For example, $g(x, y)$ is approximated by a convex function $\widetilde{g}(x, y)$

of the following form Fleury and Braibant (1986), Fleury (1989a):

$$\begin{aligned} \widetilde{g}(x,y) \quad &= g(x^j,y^j) \\ &+ \textstyle\sum_k^{(+)}(x_k - x_k^j)\frac{\partial g}{\partial x_k}|_{(x^j,y^j)} \quad + \displaystyle\sum_k^{(-)}(\frac{x_k^j}{x_k})(x_k - x_k^j)\frac{\partial g}{\partial x_k}|_{(x^j,y^j)} \\ &+ \textstyle\sum_{k\notin K}^{(+)}(y_k - y_k^j)\frac{\partial g}{\partial y_k}|_{(x^j,y^j)} \quad + \displaystyle\sum_{k\notin K}^{(-)}(\frac{y_k^j}{y_k})(y_k - y_k^j)\frac{\partial g}{\partial y_k}|_{(x^j,y^j)}. \end{aligned} \tag{2.7}$$

For each variable z, the choice to use the direct (z) or reciprocal $(1/z)$ variable in this truncated Taylor expansion depends on the sign of the partial derivative of the function with respect to z at the current point (positive: $(+)$ or negative: $(-)$). The resulting approximation of g is separable in the direct and reciprocal variables. Since some variables $y_k, k \in K$, have fixed values in $P^{K,\alpha}(x,y)$, only the remaining variables $k \notin K$ are used to build the local approximation $\widetilde{g}(x,y)$.

The approximate problem $\widetilde{P}^{K,\alpha}(x,y)$ is then solved by a nonlinear optimization algorithm, say *Solve_NLP*. This yields a new point (x^{j+1},y^{j+1}) whose feasibility is verified by a computation of f,g,h. This new information is used to start the next iteration.

Algorithm 2 *Solve*$(P^{K,\alpha}(x,y),x^0,y^0)$;

```
j := 0
(x^*, y^*) := (x^0, y^0)
Compute $f(x^j,y^j)$, $g(x^j,y^j)$, $h(x^j,y^j)$ and their derivatives
repeat
  Build a local approximation $\widetilde{P}^{K,\alpha}(x,y)$ at $(x^j,y^j)$
  $(x^{j+1},y^{j+1}) := Solve\_NLP(\widetilde{P}^{K,\alpha}(x,y),x^j,y^j)$
  Compute $f(x^{j+1},y^{j+1}), g(x^{j+1},y^{j+1}), h(x^{j+1},y^{j+1})$
  if $(x^{j+1},y^{j+1})$ is feasible for $P^{K,\alpha}(x,y)$ and $f(x^{j+1},y^{j+1}) < f(x^*,y^*)$ then
    $(x^*,y^*) := (x^{j+1},y^{j+1})$
  end if
  $j := j + 1$
until $j == Max_{it}$
return $(x^*,y^*)$
```

In our implementation, as in Fleury and Braibant (1986) or Fleury (1989a), *Solve_NLP* combines a Lagrangian relaxation scheme and a dual approach to solve the approximate subproblem. The dual maximization problem is separable and can be decomposed in single variable problems. As the convex linearization provides a conservative approximation of the original problem (i.e., the approximation overestimates the original functions), a feasible solution of the approximate problem is always a feasible solution of the original problem.

After Max_{it} (a user-defined parameter) iterations, *Solve* eventually returns the best feasible solution found (if any). This efficient approach allows obtaining a (local) optimal solution of each subproblem $Solve(P^{K,\alpha}(x,y),x^0,y^0)$ within a few (10 to 15) iterations, starting from a nearly feasible initial solution (x^0,y^0) Fleury and Braibant (1986), Rigo (2001b,a), Rigo and Fleury (2001).

2.4.2 Branch and Fix heuristic

The Branch and Fix (B&F) heuristic is based on the same underlying principle as the Dive and Fix heuristic: at each step of the search, a group of variable values are rounded and fixed, and the resulting subproblem is solved by an approximation method requiring a finite number of expensive structural analyses; see Algorithm 3.

As in D&F, the initial step consists in solving the continuous problem $P^c(x,y)$ by the procedure *Solve*. This yields a first candidate solution (x^0,y^0).

Then, the B&F algorithm builds a partial enumeration tree, where each node of the tree is associated with a subproblem to be handled. The algorithm maintains a set *OpenNodes* of subproblems which are still to be handled. More precisely, each node is labelled by a quadruple of the form $v = (x^q, y^q, rounding\_mode, i)$, which completely characterizes a subproblem $P^{K,\alpha}(x,y)$ with $K = K_1 \cup \ldots K_i$ and $\alpha_k := Round(y^q_k, rounding\_mode)$ for all $k \in K$. When node v is explored, the algorithm *Solve* runs on $P^{K,\alpha}(x,y)$, starting from the initial solution $(x^q, \tilde{y}^q)$, with $\tilde{y}^q$ defined as previously, and it returns a solution (x^*, y^*).

If this solution is feasible then it generates exactly two successor nodes of v (and two corresponding subproblems). The first node is labelled by $(x^*, y^*, Closest, i+1)$: it is associated with a subproblem $P^{K,\alpha}(x,y)$, where we attempt to fix an additional subset of variables $y_k, k \in K_{i+1}$, to values α_k which remain as close as possible to their current optimal values y^*_k, namely $\alpha_k := Round(y^*_k, Closest)$.

If this attempt succeeds, that is, if the *Closest* rounding mode leads to a feasible solution, then the second successor node of v is labelled by $(x^*, y^*, Down, i+1)$; otherwise, it is labelled by $(x^*, y^*, Up, i+1)$. The intuition for this procedure is related to the monotonicity hypothesis, already stated in Section 2.4.1.1, according to which smaller values of the structural design variables (e.g., thickness of plates) generate smaller weight and cost than higher values, but are less likely to yield feasible solutions.

In view of this hypothesis, when the *Closest* rounding mode leads to

Algorithm 3 *Branch and Fix*;

```
$(x^u, y^u)$ : initial solution
$\{K_1, \ldots, K_N\}$ : a partition of the set of indices $\{1, \ldots, m\}$

$(x^0, y^0) := Solve(P^c(x, y), x^u, y^u)$
if $(x^0, y^0)$ is not feasible for $P^c(x, y)$ then
  fail and exit
end if
$i := 1$
$rounding\_mode := Closest$
$v^0 = (x^0, y^0, rounding\_mode, i)$
$OpenNodes := \{v^0\}$
$Solutions := \emptyset$
repeat
  Choose any node $v = (x^q, y^q, rounding\_mode, i)$ in $OpenNodes$
  $OpenNodes := OpenNodes \setminus \{v\}$   {Remove $v$ from $OpenNode$}
  $K := K_1 \cup \ldots K_i$
  for all $k \notin K, \tilde{y}^q_k := y^q_k$
  for all $k \in K, \alpha_k := Round(y^q_k, rounding\_mode)$ and $\tilde{y}^q_k := \alpha_k$
  $(x^*, y^*) := Solve(P^{K,\alpha}(x, y), x^q, \tilde{y}^q)$
  if $(x^*, y^*)$ is feasible for $P^{K,\alpha}(x, y)$ then
    if $rounding\_mode == Closest$ then
      $OpenNodes := OpenNodes \cup \{(x^q, y^q, Down, i)\}$   {Create a new node }
    end if
    if $i < N$ and $y^* \notin Y$ then
      $OpenNodes := OpenNodes \cup \{(x^*, y^*, Closest, i + 1)\}$   {Create a new node }
    else
      $Solutions := Solutions \cup \{(x^*, y^*)\}$
    end if
  else
    if $rounding\_mode == Closest$ then
      $OpenNodes := OpenNodes \cup \{(x^q, y^q, Up, i)\}$   {Create a new node }
    end if
  end if
until $OpenNodes = \emptyset$
return $Solutions$
```

a feasible solution, we try to improve this solution by rounding down the variables in K_{i+1}; on the other hand, when *Closest* fails, then we try to restore feasibility by rounding up the variables in K_{i+1}. Observe that if the continuous relaxation $P^c(x, y)$ has a solution (x^0, y^0), then the monotonicity hypothesis implies that the algorithm B&F converges, in the sense that it eventually finds at least one discrete solution of $P(x, y)$.[3]

Whenever a discrete feasible solution is found, it is stored in a *Solutions* set which is the output of the algorithm. The algorithm terminates when *OpenNodes* is empty, which necessarily happens after at most 2^{N+1} nodes have been handled.

2.5 Computational experiments

2.5.1 Industrial test cases

In this section, we present the results of the application of D&F and B&F on three real instances. These instances are naval structures produced at STX, a major naval shipyard that produces cruise ships, ferries, merchant vessels, tankers, military ships, offshore and other specialized vessels. Among these is the famous "Queen Mary 2", one of the largest cruise liners ever built. For such large and heavy structures, a small percentage variation in the weight or cost of the structure may have a significant practical impact.

Before going to sea, every ship has to be certified by a recognized organization such as Bureau Veritas, ABS, etc., which verifies that the ship complies with a number of constraints. For instance, maximal admissible values are defined for stress and deformations that occur within the structure when it is submitted to loads and external forces (sea forces on the hull). Standard load cases are defined to model combinations of sea conditions and various types of loadings (loads on decks, freight, fuel). These load cases define tests that must be passed by the structure with limited deformations and without damage or ruin. They also define the structural constraints of the design optimization problem.

We applied the D&F and B&F heuristics in order to optimize the design of three ships: a medium passenger ship (ship A), a medium cruise liner (ship B), and a large cruise liner (ship C). The overall length of these structures ranges from 100m to 250m. For each ship, the midship sections are symmetric and identical; therefore, only one half section is modelled in our work. The variables define the geometric dimensions of the elements (plates, stiffeners, etc.) and their relative spacing. Each discrete variable can typically take

[3]Clearly, this claim also hinges on the convergence properties of the external continuous optimizer *Solve*: the discrete problem cannot be easier to solve than its continuous relaxation.

Ships	Panels	Discrete variables	Continuous variables	Load cases	Constraints
A	28	176	22	1	360
B	63	271	28	2	1038
C	91	464	52	2	1709

about ten different values. The constraints of the model are the limitations on the maximal stress and deformations of the steel elements. Each instance also includes a constraint on the vertical position of the gravity centre of the sections. A model may include hundreds of variables and more than a thousand constraints; see Table 2.5.1. The objective function that we consider here is the weight of the structure.

The first ship studied is a 100m long passenger ship. Each midship section is composed of 28 panels and beam columns, but the design variables of the model are relative to the panels only. There are 176 discrete variables and 22 continuous variables. The only load case considers loads on each deck and sea pressure on the hull; it yields 360 constraints.

The second ship is a cruise liner, 200 meter long, whose structure is modelled by 30 meter long midship sections, each composed of 63 panels. The model includes 299 variables, among which 271 are discrete variables, and 1038 constraints. The two active load cases consider the hogging or sagging effect associated with the sea pressure, and loads on the decks.

The third application is a larger cruise liner with a more complex structure. The 250 meter long structure has been modelled by 32.5m sections. The model represents 91 panels and beam columns, leading to 516 design variables among which 464 discrete variables. We consider two load cases which take into account the sea pressure, the loads on the decks, and either the sagging or the hogging effect, leading to a total of 1709 constraints.

2.5.2 Results

The Dive & Fix and the Branch & Fix heuristics have been applied to the three instances described above. Table 2.5.2 displays the value[4] Z^* of the best solution of $P^c(x,y)$ computed by *Solve*, the value DF^* of the best discrete solution found by D&F, and the value BF^* of the best discrete solution found by B&F. The next two columns of Table 2.5.2 give the relative difference between the value of the discrete solutions and the solution of the continuous relaxation $P^c(x,y)$:

$$\%(DF^* - Z^*) = 100 \times (DF^* - Z^*)/Z^*, \tag{2.8}$$
$$\%(BF^* - Z^*) = 100 \times (BF^* - Z^*)/Z^*. \tag{2.9}$$

[4]All these values have been scaled for confidentiality purposes.

	Z^*	DF^*	BF^*	$\%(DF^*\text{-}Z^*)$	$\%(BF^*\text{-}Z^*)$	$\#BF^*$
A	209 299	211 486	210 459	1.04	0.55	8
B	2 991 974	3 092 439	3 085 541	3.36	3.13	12
C	4 683 688	4 771 694	4 760 374	1.88	1.64	16

The last column of Table 2.5.2 ($\#BF^*$) indicates the number of discrete solutions found by B&F.

Quite remarkably, both D&F and B&F prove able to identify discrete solutions of the structural design problem which are extremely close to the best solution Z^* obtained for the continuous relaxation of the problem. The additional weight induced by the discreteness requirement remains smaller than 3.4% of the total weight in the worst case.

If we assume that Z^* is the optimal value of $P^c(x,y)$, and since this optimal value is a lower bound on the optimal value of $P(x,y)$, we can actually conclude that both D&F and B&F provide *near-optimal* solutions of the *discrete* structural optimization problem. Of course, this conclusion depends on the quality of the continuous solutions computed by *Solve* and, in particular, on the ability of this subroutine to converge to a global optimum of the continuous relaxation $P^c(x,y)$; but it shows, at least for our test instances, that the discrete version of the problem can be solved "almost as efficiently" as its continuous version. Moreover, the gaps $\%(DF^* - Z^*)$ and $\%(BF^* - Z^*)$ provide measures of performance for D&F and B&F which are independent of any comparison with alternative algorithms. (As we mentioned earlier, we are actually not aware of other algorithms which could handle the same problem instances.)

It is also interesting to note that B&F frequently finds several feasible solutions: this may prove valuable in an industrial setting where the designers would like to compare the features of alternative designs. Indeed, a detailed analysis of the results shows that the best scantlings produced by B&F may be quite different in terms of the combinations of values assumed by the discrete variables.

Table 2.5.2 provides some indication of the difficulty of the instances. All computations were performed on a personal computer (Pentium 4, 3GHz, 2Gb RAM). Table 2.5.2 displays approximate computing times for the subroutine *Solve* on $P^c(x,y)$, for D&F and for B&F. It also shows the number of continuous subproblems (nodes) generated by D&F and by B&F heuristics. The results in Table 2.5.2 are presented in increasing order of complexity of the problems. For Ship A, the optimization of $P^c(x,y)$ by *Solve* at the root node takes 40 seconds, while the running time of B&F is around 6 minutes. For Ship B, *Solve* takes 6.5 minutes and B&F runs for 51 minutes. Ship C is the most complex instance presented here due to the multiple load cases and the large number of variables: *Solve* takes 13 minutes, while the B&F

	Solve time	*D&F* tree size	*D&F* time	*B&F* tree size	*B&F* time
A	40s	9	3m23s	26	6m03s
B	6m28s	9	24m10s	35	50m48s
C	13m11s	12	2h47m02s	64	6h1m50s

algorithm runs for more than 6 hours. These computing times, though relatively high, are quite acceptable in the context of the design phase of a large ship.

2.5.3 Comparison with branch-and-bound

As mentioned above, the quality of the solutions computed by D&F and B&F for our test instances is provably good, since they come within a couple of percents of the bounds provided by the continuous relaxations. This suggests that a branch-and-bound algorithm may also perform well on our problem; as explained in Section 2.2, however, straightforward application of branch-and-bound is prevented by the large amount of computation time required by structural analyses.

In order to provide a comparison between our results and those obtained by alternative approaches, we have developed a branch-and-bound algorithm that can be applied to solve an approximate model of $P(x, y)$. More precisely, we solve the discrete version of the convex local approximation model $\widetilde{P}^c(x, y)$ of $P^c(x, y)$ built at the point $(x^0, y^0) := Solve(P^c(x, y), x^u, y^u)$ as explained in Section 2.4.1.2. This type of approach is classical in other areas of structural design, for instance, in aeronautical engineering. (But we are not aware of available implementations of exact or approximate algorithms which could handle our ship design problem instances.)

Table 2.1 presents the results obtained with this branch-and-bound algorithm on a particular test instance, namely Ship A. Each line is associated with a different starting point (x^u, y^u) for the procedure $Solve(P^c(x, y), x^u, y^u)$. The algorithm generates several discrete solutions of the approximate problem $\widetilde{P}^c(x, y)$, all of which present important violations of a number of constraints of the original problem. Among these discrete solutions, we denote by (x^b, y^b) the solution whose value (in the approximate model $\widetilde{P}^c(x, y)$) is closest[5] to the value $f(x^0, y^0)$. In Table 2.1, we also display the objective value $\tilde{f}(x^b, y^b)$ of the approximate problem at (x^b, y^b), and the value $f(x^b, y^b)$ of the original objective function at this point. Column p displays the proportion of constraints of $P(x, y)$ violated by (x^b, y^b), n gives the number of

[5]Similar conclusions apply if we take (x^b, y^b) to be the best solution obtained by branch-and-bound, but the violations of the constraints are even worse in this case.

Table 2.1: Experiments: branch and bound for Ship A

$f(x^0, y^0)$	$\tilde{f}(x^b, y^b)$	$f(x^b, y^b)$	p	n	v	(v_{min}, v_{max})
209 231	205 202	246 632	1.4 %	5	31%	(26,36)%
209 249	210 361	242 152	1.1 %	4	33%	(28,38)%

violated constraints, v is the mean constraint violation due to the approximation error at (x^b, y^b), and v_{min}, v_{max} are the extreme percentages of violations over all constraints of $P(x, y)$.

The main conclusion to be drawn from Table 2.1 is that the approximation provided by model $\widetilde{P}^c(x, y)$ at (x^0, y^0) is apparently too poor to provide acceptable solutions. Indeed, in all cases, the branch-and-bound algorithm converges to discrete solutions, but these solutions present important violations of a small number of constraints, and their value $f(x^b, y^b)$ is much worse than the initial value $f(x^0, y^0)$, or than the values obtained by DF^* and BF^* (see Table 2.5.2). Moreover, we should also mention that the running time of branch-and-bound becomes prohibitive on larger instances.

A main advantage of DF^* and BF^*, thus, is probably that these algorithms are able to carry out an *intensive* exploration of the discrete solution space in the neighborhood of the initial continuous solution (x^0, y^0). This approach – which shares some similarity with the heuristic procedures used by human designers – proves sufficient to efficiently identify a number of good solutions. By contrast, branch-and-bound provides a more *exhaustive* search of the feasible space, but since it runs on a local model, the quality of the model deteriorates quickly outside of the vicinity of (x^0, y^0). (Alternative approaches working on adaptive models may be able to alleviate this difficulty, but it is not clear whether this is worth the effort, in view of the quality of the solutions computed by DF^* and BF^*.)

2.5.4 Parameters of the heuristics

In both heuristics D&F and B&F, the indices of the discrete y-variables are partitioned into subsets $K_1, \dots, K_N$. The order in which the subsets $K_1, \dots, K_N$ are used for rounding greatly influences the quality of the results produced by these heuristics: as a matter of fact, in our experiments, we observed that the B&F algorithm can find several solutions for some appropriate orderings of $K_1, \dots, K_N$, and no solution at all for other orderings. As we mentioned earlier, it seems most effective to consider first the variables having the largest impact on the objective and on the constraint functions. For the naval structures that we consider here, the order of the subsets is determined in agreement with the engineers' knowledge of the problem. Thus, for every ship design instance, nine subsets $K_1, \dots, K_9$ are defined according to the nature of the variables: the variables y_k with $k \in K_1$ determine the thickness of the plates, those with $k \in K_2$ determine the number of frames on each panel, those with $k \in K_3$ determine the web dimension of the frames, and so on.

Another important parameter of the heuristic algorithms is the number of iterations performed by *Solve* for the optimization of each subproblem $P^{K,\alpha}(x, y)$ (parameter Max_{it} in Algorithm 2). Indeed, each such iteration requires computationally expensive function evaluations.

Table 2.5.4 displays Z^*, the value of the best solution of $P^c(x, y)$ obtained by *Solve*, and BF^*, the value of the best discrete solution of $P(x, y)$ obtained by B&F, when Max_{it} is respectively set to 6, 10 or 15 iterations in *Solve*.

The column labelled $\%(BF^* - Z^*)$ displays the ratio $100 \times (BF^* - Z^*)/Z^*$ for each setting of Max_{it}, and $\%(BF^* - Z^{**})$ shows the ratio $100 \times (BF^* - Z^{**})/Z^{**}$, where Z^{**} is the best value found by *Solve* for $P^c(x, y)$, for any setting of Max_{it}. (Thus, Z^{**} is our best estimate of a lower bound on the optimal value of the instance, and $\%(BF^* - Z^{**})$ can be viewed as a reliable overestimate of the gap between BF^* and the true discrete optimum.) The last column shows the size of the search trees.

These results suggest that increasing the number of iterations of *Solve* does not significantly improve the quality of the best solution found for $P^c(x, y)$, and does not necessarily improve the best solution obtained by B&F, nor even the number of discrete solutions generated by B&F. However, the running time of the B&F algorithm increases, from 6 hours when $Max_{it} = 6$ to more than 8 hours when $Max_{it} = 15$.

In our experiments, we observed that the procedure *Solve* frequently converges to a good feasible solution in less than four iterations. In consequence, the number of iterations can be set dynamically when *Solve* is used in the heuristics: for each continuous subproblem $P^{K,\alpha}(x, y)$, *Solve* stops after a small number of iterations Max_{it} (typically, between four and six) if a feasible solution has been found, and it keeps running up to a extra number of iterations $Extra_{it}$ only if no feasible solution is found earlier. (A similar approach was proposed by Borchers and Mitchell (1994) in a mixed-integer nonlinear branch-and-bound procedure where the NLP subproblems are not solved to optimality.) This dynamic procedure reduces the solution time while maintaining the quality of the solutions computed by D&F and B&F. It was used for the experiments reported in Table 2.5.2 and Table 2.5.2.

Finally, regarding the number of active constraints, we observed that the number of active constraints increases fast during the first four iterations of *Solve* (see Algorithm 2) and then remains almost unchanged in the next iterations as the objective functions keeps on decreasing slightly. This suggests that *Solve* converges quickly to a local optimum.

It is also interesting to note that at each run of *Solve* on $P^{K,\alpha}(x, y)$, i.e., at each node of the enumeration tree in D&F or in B&F, the percentage of

Iterations	Z^*	BF^*	$\%(BF^* - Z^*)$	$\%(BF^* - Z^{**})$	Tree size
6	4 692 866	4 753 169	1.29	1.50	52
10	4 683 688	4 760 374	1.64	1.65	64
15	4 683 066	4 763 312	1.71	1.72	47

Table 2.2: Results for Ship C

active constraints remains almost constant (about 10% of the total number of constraints), but the set of active constraints does change significantly (about 20% of the active constraints are modified at each node), meaning that the search process actually generates rather diversified structures.

2.5.5 Numerical tolerance issues

As discussed in Section 2.4.1.1, D&F and B&F fix the discrete variables in groups, rather than individually, with the objective of reducing the number of structural analyses to be performed. A drawback of this approach, however, is that the same rounding mode (*Closest*, *Up*, or *Down*) is applied blindly to all variables of a group. When the value of a given variable y_k (in a group selected for rounding) is very close to one of its admissible discrete values, say $r_k \in Y_k$, then a better choice may be to round y_k to r_k, whatever the current rounding mode is. We have implemented this idea by setting a "discretization tolerance factor" ε such that, if $|y_k - r_k| < \varepsilon$, then any rounding operation sets $y_k = r_k$.

Also, we have seen that in naval structural design problems, the constraints express admissible values for stress, deformation, etc. In practice, some security margins are included in the definition of these values. Therefore, a solution may be deemed acceptable by the designer even if some constraints are slightly violated. This issue has been taken into account in our algorithms through the introduction of a tolerance factor in the evaluation of the constraints. This parameter can be set by the designer before each run.

2.6 Multicriterion optimization

6

[6]This section is a collaborative work with Thomas Richir, Jean-David Caprace and Philippe Rigo at the ANAST department of the University of Liège and N. Losseau, M.G. Parsons and S. Patay, from STX shipyards. An earlier version of this section has been presented at the 10th International Symposium on Practical Design of Ships and Other Floating Structures (PRADS) in Houston, Texas (Richir et al. 2007b). Any mistake eventually appearing in this section is only my responsibility.

In this last section, we present a multiobjective optimization of a cruise ship. Two objectives are considered : the production cost and the moment of inertia of the hull. The production cost is computed from a detailed shipyard cost database, linked to the advanced cost module of LBR5 software. In this cost module about 60 different fabrication operations are considered, covering the different construction stages, as girders and web-frames prefabrication, plate panels assembling, blocks pre-assembling and assembling, as well as 30 types of welding and their unitary costs (Richir et al. 2007a). The other objective, namely the moment of inertia, is proportional to the weight of the structure and its maximization is thus a conflicting objective regarding the cost minimization.

Multicriterion optimization has been introduced in Section 1.4. The multicriterion problem presented in Section 1.4 has been optimized with the Branch and Fix algorithm of section 2.4. Computing experiments have been performed on a passenger ship composed of 118 stiffened plate elements and 19 pillars, shown on Figure 2.1. The midship section of the ship is characterized by 14 decks, a 40 m breadth and a 45 m height.

Five scantling design variables are defined for each stiffened plate element: plate thickness, flange width, web height and thickness of longitudinal stiffeners and spacing between two longitudinal stiffeners. The solution space for the discrete design variables is defined with a step of 1 mm for the thicknesses and 10 mm for the web height and flange width. The size of the solution space for the discrete variables is roughly 38. 10^{439} solutions. Due to equality constraints imposed between the spacing of elements which are vertically aligned, the stiffener spacing is defined as a continuous design variable.

The model is thus characterized with:

- 118 stiffened plate elements,
- 383 design variables (including 308 discrete variables),
- 10 load cases
- 1418 constraints
- 56 equality constraints.

Four global constraints on the hull girder are considered, they are the weight (bounded above), the section modulus and moment of inertia (bounded below) and the gravity center shift (doubly bounded). This last constraint aims at avoiding stability trouble.

Ten load cases were considered in the calculation:

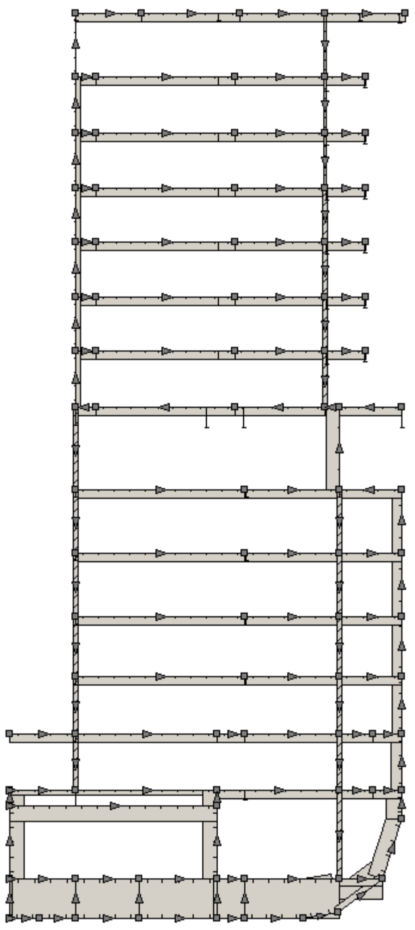


Figure 2.1: LBR-5 Midship section of a passenger ship

- two “IACS load cases” (hogging & sagging): still water bending + wave bending with a probability of exceedence = 10^{-8}
- eight “BV load cases” (hogging & sagging)
  - Load case “a”: still water bending + wave bending with a probability of excess = 10^{-5} + sea pressure (scantling draft & ballast draft)
  - Load case “b”: still water bending + wave bending with a probability of excess = 10^{-5} + sea pressure (scantling draft & ballast draft) + inertial pressure

An approximation of the Pareto front was obtained using a process that randomly alters the weights in the weighted sum solution and solves the optimization problem for each of these problems. The non dominated solutions form an approximation of the Pareto front, and are shown in Figure 2.2, with F1 the moment of inertia and F2 the production cost. More than 200 points were calculated. To avoid large computing time only raw scantling optimizations were performed. The Pareto front was generated in about 100 minutes with a Pentium 2.40 GHz and 512 Mo of RAM desktop. The equal weights min-max and nearest-to-the-utopian solutions are also shown in Fig.2.2. This representation helps the designer to select a solution which is a compromise between the two objectives, possibly taking into account additional criteria, not included in the model.

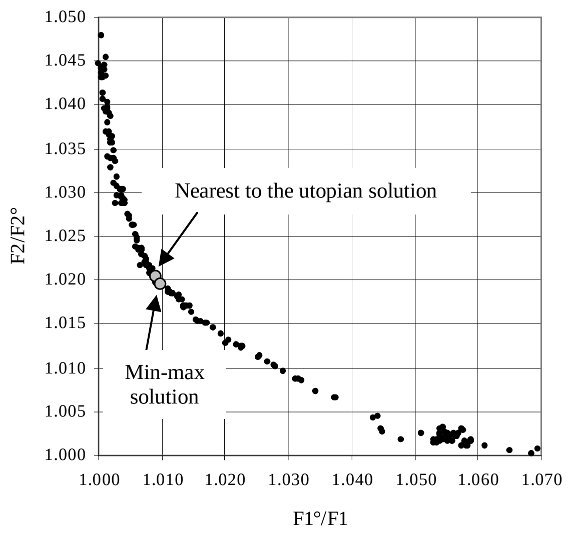


Figure 2.2: Pareto Front

The equal weights min-max solution was also calculated by performing a discrete optimization. The cost and stiffness savings, obtained by comparison with the initial scantling, are of 1.84 percents for the production cost and 14.89 percents for the moment of inertia. Note that the initial scantling does not satisfy some structural constraints; for this reason the cost saving would probably be more important if the final solution was compared to a feasible initial solution. Moreover the weight associated with the cost objective could be increased in the multicriterion formulation in order to improve the cost saving.

2.7 Discussion and conclusions

In naval structure design, the discrete nature of the variables and the extensive computation time needed to analyze a single structure are major issues to be faced by any optimization approach. Actually, for real life instances, even finding a discrete feasible solution of "good quality" represents a considerable challenge. A common industrial practice consists in optimizing first a continuous relaxation of the model, and performing next a one-step (brute force) rounding of the variables followed by manual adjustments to restore feasibility. No existing algorithm seems to be able to simultaneously handle the presence of a large number of continuous and mixed variables, and of a large number of nonlinear constraints with expensive black-box evaluations.

In this chapter, we have proposed two algorithms to compute near-optimal solutions of discrete structural optimization problem.

The first algorithm is a Dive & Fix heuristic that iteratively applies partial rounding to the current solution, and performs a continuous optimization of nonlinear subproblems, until a feasible discrete solution is found. Dive & Fix allows to obtain a discrete feasible solution by solving at most $2N$ continuous optimization subproblems (involving expensive evaluations of the implicit constraints), where N is the number of variable groups defined by the user.

A second Branch & Fix heuristic has been developed when longer computation time is available. This heuristic performs a more intensive search in the neighborhood of the intermediate solutions of the Dive & Fix algorithm. In the process, it solves at most 2^{N+1} continuous subproblems.

As illustrated in Section 2.5, both the Dive & Fix and the Branch & Fix heuristics produce discrete feasible solutions of the structural optimization problem whose value is very close to the optimal value of the continuous relaxation. Moreover, Branch & Fix usually finds several feasible solutions of high quality, among which the designer can choose according to additional criteria. Both heuristics can also be fine-tuned to accommodate a number of practical issues identified by the designers, such as tolerance on the constraint bounds.

Finally we note that since the main contribution of our work is on handling discreteness requirements, it is worthwhile to think of the associated continuous optimization algorithm *Solve* as of an "external subroutine" which could be replaced by another solver in an alternative implementation of D&F or B&F. In this sense, any progress in the solution of the continuous structural optimization problem could immediately be transposed into the solution of the (mixed) discrete problem.

Multiobjective optimization has also been tested on a large passenger ship. The Branch and Fix method developed in the previous chapter has been used to obtain a large set of non-dominated solutions for the weight and inertia objectives. This set approximates the Pareto front and provides the designer with an efficient tool to help his choice of the final design.

In conclusion, we believe that the algorithms described here provide efficient and effective tools for the structural optimization of large ships.

Chapter 3

Decomposition and Direct Search methods for explicit problems with fast function evaluations

3.1 Introduction

The optimization of naval structures is a large-scale optimization problem characterized by hundreds of variables and constraints. In engineering design, the variables are the dimensions of the structural elements and often take integer values or values chosen within a specified set. For instance, the elements thickness (panels, stiffeners, frames) can be considered as continuous, but it preferably takes values within finite sets of standard thicknesses. Choosing the dimensions of elements among available and predefined values contributes to reduce the raw material cost of the structure. The number of stiffeners welded on a plate is a design variable that must be integer.

A current practice to handle large scale problems with discrete variables is to perform a nonlinear optimization followed by a post processing phase: the designer has to "round" by hand the values of the optimal solution of the nonlinear problem. Such an adaptation of the optimal scantling usually results in a significant increase of the cost or weight of the structure.

In this chapter we explore a new method that exploit the mathematical structure of the two-dimensional model presented in chapter 1.2.2. We propose an iterative optimization method based on a decomposition of the problem in more tractable subproblems. The method iteratively solves a continuous relaxation of a main problem and then a set of discrete subproblems. The main problem is related to the beam ship optimization and the subproblems are closely related to panels, considered as the basic elements of the ship. As the subproblems have a reduced number of discrete variables (up to five) direct search methods may be a good choice to explore the discrete solution space. To address the subproblems, we develop a new direct search (DS) method based on lattice-based Generating Set Search (GSS) and Mesh Adaptative Direct Search (MADS) for mixed variable optimization. The decomposition and direct search (DSS) method is tested on a real instance and shows encouraging preliminary results.

3.2 Literature review

Many methods have been proposed in the literature to handle structural optimization problems among which descent and gradient-based methods have been of common use since the 60's. Evolutionary and stochastic approaches were developed in the late 70's then in the 80's and 90's response surface methods and design of experiments have been adapted to computer simulation and engineering design with an extensive use of surrogate modeling to accelerate the search for an optimum. In this work we explore the combination of problem decomposition and methods to handle the discreteness of the problem.

Problem decomposition
Multilevel optimization is a suitable method to reduce the size of large scale engineering problems. It decomposes a problem (hardly tractable because of the large number of variables and nonlinear constraints) into a set of much smaller independent subproblems. These methods have been successfully applied for nonlinear optimization in the early seventies by Kirsch, Reiss, and Shamir (1972), Schmit and Ramanathan (1973) and have been further developed in numerous papers since then. In Sobieszczanski (1993) decomposition is applied to structural design and the authors propose two efficient formulations for the coupling of subproblems. Following the same idea, we define a master problem (at system level) for the weight optimization of the entire structure (beam ship), and subproblems (or sublevel systems) for the optimization of structure components (panels). Unlike Sobieszczanski (1993) we consider discrete subproblems and use a coordinating function to obtain solutions compliant with the system level solution.

Nonlinear optimization
Optimization methods to solve the nonlinear master problem (featuring continuous variables only) may be chosen among the numerous methods available in global optimization. We refer to the previous chapter for a presentation of these methods. In this chapter we use the convex linearization method associated with a dual procedure developed by Fleury et al. to find an ini-

tial solution to the nonlinar master problem (Fleury and Braibant (1986), Fleury (1989a,b), Zhang and Fleury (1997)). The authors describe an efficient sequential convex programming (SCP) method based on *Conlin* that iterates the following steps: a convex linearization of the objective and constraints generates the approximation and a dual resolution method is used to find an optimal solution to the approximation. The solution of the approximation is the starting point of the next iterate. This approach finds an optimal solution of nonlinear structural optimization problems after a few (10 to 15) iterations (Fleury and Braibant 1986). This method has proved to be efficient on large size structural optimization problems with hundreds of continuous variables and constraints (see Rigo 2001b, Rigo and Fleury 2001).

Direct Search
The original motivation underlying the development of Direct Search (DS) methods is to use the values of the objective function only, with no knowledge of derivative information. DS methods are characterized by a sequential examination of trial solutions and a strategy to determine what the next trial solution will be. Each iterate lies on a mesh built using a finite set of directions called positive basis or generating set, that positively spans the solution space (i.e. such that any vector of the space can be written as a positive combination of the directions in the set). There has been an active stream of research and developments of DS methods in last years, we present some recent evolutions of these methods. Our interest in DS methods is to exploit the mesh structure in order to iterate on feasible discrete solutions only.

Generalized Pattern Search (GPS) is a DS method introduced by Torczon (1997) for nonlinear unconstrained optimization with some adaptations to constrained optimization problems. GPS has been extended to bound constraint problems by Lewis and Torczon (1999) and to linear constraint problems by Lewis and Virginia (2000). The method relies on the generation of a tangent cone of directions with respect to any nearby (or active) constraint.

For bound constraints and general constraints problems Lewis and Torczon (Lewis and Torczon 2002) introduce a scheme that uses an augmented Lagrangian method. At the same time another pioneer work is due to Fletcher and Leyffer who proposed a filter approach to treat nonlinear constraints without derivatives by a generalized pattern search (see Fletcher and Leyffer (2002)). Audet and Dennis (Audet and Dennis 2004) formulate a pattern search method for general constrained optimization : they use a filter method that aggregates the objective and the constraints violations and treat the resulting problem, without use of any derivatives, penalty constants, or Lagrange multipliers. These methods generalize previous work on unconstrained, bound constrained, and linearly constrained generalized pattern search.

In general pattern search methods, the local exploration uses a predefined set of search directions that positively span the solution space. This limitation has been removed in the Mesh Adaptative Direct Search (MADS) class of algorithms, where search directions are modified during the search, so that the variable space is explored in an asymptotically dense set of directions. MADS algorithms were introduced as an alternative to filter-GPS and show much stronger convergence properties. MADS have been adapted in various ways to handle constraints, in particular using a progressive barrier approach in (Audet and Dennis 2006b), and make use of derivative information, as described in papers authored by Abramson, Audet et al. (Abramson et al. 2004, 2006b, 2009). Recently, new evolutions of MADS algorithms have been developed that take into account periodicity of non continuous variables and mix of continuous and discrete variables (see Audet and Digabel (2009), Abramson et al. (2009)).

Although developed as derivative free methods, GPS Methods have been adapted to take advantage of derivative information when available (Abramson et al. 2004) for unconstrained or linearly constrained nonlinear optimization problems. If gradient information is partially or not available, GPS is sometimes combined with gradient estimators approaches, this method is of common use in problems involving simulation, see Fu, Glover, and April (2005) for a survey.

The reader may consult Kolda et al. (2003) for a comprehensive review of optimization by direct search and Lewis et al. (2000) for an historical perspective.

A drawback shared by DS, MADS and GPS methods is that they can tackle a limited number of variables only. To our knowledge, their use in engineering design optimization has been so far always linked to some kind of surrogate modeling that reduces the number of variables of the design optimization problem.

In this chapter we develop a new method that combines problem decomposition and direct search method. Problem decomposition tackles the high dimension issue by generating subproblems of lower dimensions. The subproblems generated by decomposition have mixed variables (continuous and discrete), general constraints and a complete knowledge of derivatives. We propose a new DS method that iterates on rational lattice (or mesh) to consider only discrete values of the variables, a mesh adaptative approach is used to ensure a dense local exploration of the solution space and the search makes use of the derivative information available for the objective and the constraints.

3.3 Model

We reformulate here the model described in section 1.2.2 to adopt a notation suitable for the decomposition and direct search algorithms. We define an aggregate variable χ and show the function dependencies in term of the variables. We consider the discrete nonlinearly constrained problem $P(x, y)$:

$$\begin{array}{lll} \min & f(x,y) & \qquad\qquad\qquad\qquad P(x,y) \\ \text{s.t.} & g_j(x,y) \le 0 \qquad j=1,...J & \\ \text{with} & x \in X,\ y \in Y & \end{array}$$

where $x = (x_1, ...x_K) = (\Delta_1, ..., \Delta_K)$ denotes the vector of continuous variables[1] and $y = (y_1, ..., y_4)$ is the vector of discrete variables associated to panels $k = 1, ..., K$. For each panel k we have $y_k = (\delta_k, h_k, d_k, w_k)$ denoting respectively the plate thickness and the stiffeners dimensions (i.e. web height, web thickness, flange weight), as described in section 1.2.2.

The domain of definition of variables x is a "box" $X = \{x \mid x \in \Re^n, x^L \le x \le x^U\}$, the variables y take their values in a finite set $Y \subset \Re^m$ of the form $Y = Y(\delta) \times Y(h) \times Y(d) \times Y(w)$, where each Y is a finite set of reals multiple of a given quantity *step* (for example $Y(\delta) = n_0 \times step(\delta), \ldots, n_N \times step(\delta)$ with $n_0, n_N \in Z$ and $n_0 \le n_N$). It is important to notice that Y is a set of preferable values for y variables but choosing intermediate values in $[x^L, x^U]$ has a physical meaning, and all the functions of the problem and their derivatives exist everywhere on these intervals.

The objective function $f(x, y)$ can be the weight (eq. 1.18), the cost (eq. 1.20), the moment of inertia (eq. 1.24) or a combination of these functions. Ten types of constraints are defined and are presented in equations 1.22 and following.

We define for further reference a continuous relaxation $P^c(x, y)$ of the original problem $P(x, y)$, by relaxing the discrete definition of variables y :

$$\begin{array}{lll} \min & f(x,y) & \qquad\qquad\qquad\qquad P^c(x,y) \\ \text{s.t.} & g_j(x,y) \le 0 \qquad j=1,...J & \\ \text{with} & x \in X,\ y \in Y^c & \end{array}$$

where $X = (\left[x_1^l, x_1^u\right], ..., \left[x_K^l, x_K^u\right])$ and $Y_k = (\left[y_1^l, y_1^u\right], ..., \left[y_4^l, y_4^u\right])$ are continuous bounded intervals.

We define now an aggregate variable χ_k to denote the weight of a stiffened panel k, where $\chi_k \in X$ is a continuous variable defined on a finite interval in $\Re$. Using equivalently the two notations (x, y) and $(\Delta_k, \delta_k, h_k, d_k, w_k)$, χ_k

[1]For modeling reasons the stiffeners spacing Δ_K is considered as continuous in these applications, as motivated in chapter 2.

writes as follows:

$$\chi_k = \delta_k + \frac{h_k.d_k + w_k.t_k}{\Delta_k} \tag{3.1}$$

$$\chi_k = y_{1\,k} + \frac{y_{2\,k}.y_{3\,k} + y_{4\,k}.t_k}{x_k} \tag{3.2}$$

We also define $P(\chi)$ another relaxation of the continuous problem $P^c(x, y)$, where the variable χ is defined using 3.2, with $\chi = \chi_1, .., \chi_K$, and χ_k is the weight of the panel k, $k = 1 \dots K$. Solving $P(\chi)$ provides the panels optimal weights that minimize the objective (i.e. the weight or cost of the beam ship) and satisfy all the constraints of the model.

$$\begin{array}{lll} \text{min} & f(\chi) & \\ \text{s.t.} & g_j(\chi) \leq 0 & j = 1, ...J \\ \text{with} & \chi = (\chi_1, ..., \chi_K) & \chi \in \Re \end{array} \qquad P(\chi)$$

In this formulation, constraints g_j depend on the χ variable, as detailed in the sequel. In this design problem, two quantities play a special role in the structural analysis: the vertical position (v_S) of the center of inertia and the inertia (i_S) of the beam ship. The formulae below indicate the variable dependency of the functions f and g with respect to the variable χ, to the vertical position v_S and to the inertia i_S. For a structure composed of stiffened plates $k = 1 \dots K$(we use Φ_f to represent a generic dependency function related to a function f) we have:

$$v_S = \Phi_{v_S}\left(\frac{\sum_k \chi_k v_k}{\sum_k \chi_k}\right) \tag{3.3}$$

$$i_S = \Phi_{i_S}\left(\sum_k c{\chi_k}^3 + \chi_k(v_k - v_S)^2\right) \tag{3.4}$$

with v_k the vertical position of the center of inertia of panel k, which is a given data.

The objective function (weight or cost) has a generic formulation:

$$f(\chi) = \sum_k \Phi_f(\chi_k) \tag{3.5}$$

The gravity center GC function (1.22) becomes

$$g_1(\chi) = \sum_k \Phi_1(\chi_k, (v_k - GC_{max})) \tag{3.6}$$

The moment of inertia I_{xx} of the beam ship (1.24) becomes

$$g_2(\chi) = \sum_k \Phi_2(\chi_k, (v_k - v_S)) \tag{3.7}$$

The section modulus I_{xx} of the beam ship (1.25) is

$$g_3(\chi) = \sum_k \Phi_3(\chi_k, (v_k - v_S)) \tag{3.8}$$

The structural constraints are computed for each panel k and each load case l. Four types of constraints (numbered j=4,5,6,7) express admissible normal stress σ (bending and buckling constraints: with references ic10, i14, ic37 and ic38 in section 1.2.2.3) and three types of constraints (j=8,9,10) are related to shear stress τ (with reference ic19 and ic20 in section 1.2.2.3). The dependencies on the variables x_k and y_k are defined as :

$$g_j^{k,l}(x_k, y_k) = \Phi_4(x_k, y_k, \frac{v_S}{i_S}) \qquad j = 4,5,6,7 \tag{3.9}$$

$$g_j^{k,l}(x_k, y_k) = \Phi_5(\sum_k (x_k, y_k), v_S) \qquad j = 8,9,10 \tag{3.10}$$

3.4 Decomposition method

The above model suggests the decomposition into two levels: a system level relative to the beam ship and subsystem levels relative to panels. At the system level, a non linear optimization of the ship is performed to obtain the optimal weights of each panel. At the subsystem level, the optimization aims at finding a discrete scantling of each panel (members dimensions and positions) such that the optimal panel weight at the system level is as close as possible to the optimal value found at the system level.

The system level problem is $P(\chi)$. It is a relaxation of the original problem $P(x, y)$ as the optimization variable χ is an aggregate variable (i.e. a non-linear combination of continuous and discrete variables, $x \in X$ and $y \in Y$). An optimal solution of $P(x, y)$ provides an optimal solution of $P(\chi)$ and conversely an optimal solution of $P(\chi)$ does not provide enough information to derive a discrete optimal solution of $P(x, y)$.

We define the subproblems $S_k(x, y)$ which consist in finding for each panel the scantling such that the panel weight is as close as possible to the optimal value of the weight (χ_k) computed at the system level. The subproblems are independent of each other and related to the master problem through a coordinating function C, presented hereunder. The generic formulation of the subproblems is stated precisely as follows:

$$\begin{array}{lllr}
\text{min} & C(x_k, y_k) & & S(x_k, y_k) \\
\text{s.t.} & \tilde{g}_j^{k,l}(x_k, y_k) \leq 0 & j = 4, \ldots, 10 \quad l = 1 \ldots L & \\
\text{with} & x_k \in X, \; y_k \in Y & & \\
\text{and} & \tilde{g}_j^{k,l}(x_k, y_k) = g_j^{k,l}(x_k, y_k) & \text{with fixed values of } \; v_S, i_S &
\end{array}$$

with k the subproblem number, L is the number of load cases defined and active for the subproblem, and for each load case l, a subset of the constraints j=4,...,10 is defined.

We consider that v_S and i_S are constants in subproblems $S(x, y)$. This approximation is justified by the large number of panels, and by the small relative influence of a single panel variables on the value of v_S and i_S (all panels are contributing, see equations 3.3). It is important to notice that in ship structural optimization, the stiffeners spacing x are subject to equality constraints because of continuity between adjacent stiffened panels. In consequence, these variables are optimized in the master problem and then considered as constants in the subproblems.

The objective function is used to relate the subproblems to the master problem. The choice of $C(x_k, y_k)$ is of dramatic importance regarding the convergence of the decomposition method. In our method the objective function aims at minimizing the gap between the components of the solution of the continuous master problem χ_k^{opt}, and the optimal solutions (x_k, y_k) (or equivalently $\Delta_k, \delta_k, h_k, d_k, w_k$) of the mixed continuous-discrete subproblems.

Stated in terms of the design problem, the objective of a subproblem is to minimize the difference between the panel weight obtained by the optimization of the beam ship in continuous variables considering all constraints, and the weight of a panel constituted of discrete size elements, compliant with the set of constraints defined for this panel. Formally we measure this difference as follows:

$$C_{(x_k, y_k)} = ((\chi_k)^{opt} - (\delta_k + \frac{h_k.d_k + w_k.t_k}{\Delta_k}))^2 \tag{3.11}$$

Using 3.11 as an objective, the minimization acts as a local search in the subspace of the panel variables. It has to be noticed that minimizing 3.11 implies minimizing the approximation error induced by considering the gravity center location and the moment of inertia as constants in the subproblems $S(x, y)$.

We now proceed to the description of the iterative algorithm developed to solve $P(x, y)$ using problem decomposition and the formulations of this section.

In further detail, the heuristic works as follows. An initial solution (x^u, y^u) is given and the initial value of the global continuous variable χ^u is computed. A feasible solution of the nonlinear problem $P(\chi)$ is sought using the procedure *Solve* described in section 2.4.1.2, if none is found then the procedure stops and the problem is declared infeasible. The iteration counter i is set to zero. Step 1 and step 2 are then performed iteratively.
At step 1 the counter is increased by one unit. When $i > 1$ the master

Algorithm 4 *Decomposition and nonlinear optimization*;

```
**Initialization Step :**
$(x^u, y^u)$: initial solution
$\chi^u = f_\chi(x^u, y^u)$
$(\chi^0) := Solve(P(\chi), \chi^u)$
**if** $(\chi^0)$ is not feasible for $P(\chi)$ **then**
    fail and exit
**end if**
$i := 0$
**repeat**
    $i := i + 1;$
    **STEP 1**
    **if** $i > 1$ **then**
        $\chi^{i-1} = f_\chi(x^{i-1}, y^{i-1})$
        $(\chi^i, g^i(\chi^i)) := Solve(P(\chi), \chi^{i-1})$
    **end if**
    $V_S := v_S(\chi^i)$
    $I_S := i_S(\chi^i, v_S^i)$
    $X_k = x_k(\chi^i)$
    **STEP 2** :
    $k := 1;$
    **repeat**
        $(C_k^i(x_k^i, y_k^i), x_k^i, y_k^i) := DS.Solve(S(X_k, y_k), \chi^i, V_S, I_S)$
    **until** $k == K$
**until** Feasibility test$(P(x^i, y^i))$ or $i == i_{max}$
**return** $(x^i, y^i)$
```

nonlinear problem is solved (for $i = 1$ $Solve(P(\chi), \chi^0)$ has been done in the initialization step) and an optimal weight χ_k^i of the panels k is found (this solution is generally not feasible for the discrete problem). Then the moment of inertia i_S^i and the vertical position of the neutral axis v_S^i are computed and the values of χ_k^i, v_S^i, i_S^i are fixed for step 2.
At step 2, the discrete subproblems are solved using a direct search method. The DS optimization method provides (x_k^i, y_k^i) a discrete scantling of the subproblem $S_k(x, y)$ such that the panel weight is as close as possible to the weight χ_k^i fixed at step one. At the end of step 2, the master problem objective f and constraints g are computed using the optimal discrete solutions (x_k^i, y_k^i) of the K subproblems. If theses solutions form a feasible solution (x, y) for the master problem then the algorithm terminates and returns this solution. Otherwise the iteration terminates and a new iteration starts at step 1, with the current solution as the initial point. This iterative procedure is performed i_{max} times at most.

3.5 Subproblems and direct search

At the subsystem level -step 2 of the algorithm-, the discrete optimization of panel k is performed with the procedure $DS.Solve(S(x_k, y_k), \chi^i, v_S, i_S)$, such that the weight of panel k is as close as possible to the panel weight computed at the system level. Each subproblem $S(x_k, y_k)$ features four discrete scantling variables y_k and as stated previously. The siffeners spacings x are subject to equality constraints that force the x variable to be fixed in the subproblems. In each subproblem, the set of general constraints depends on the variables y_k of panel k only, they are discrete and their admissible values are equally spaced, the definition set is written

$$Y = \{y_{min}, y_{min} + step, y_{min} + 2.s_k, ..., y_{max}\} \tag{3.12}$$

where s_k is constant.

Our implementation of DS.Solve is inspired from the Generating Set Search (GSS) via rational lattices described in Kolda et al. (2003) and from the Mesh Adaptative Direct Search (MADS) for mixed variables optimization by Abramson et al. (2009). GSS and MADS are both iterative feasible-point algorithms and rely on generating sets of directions (also called spanning sets or positive basis) to define a mesh used to explore the solution space. MADS methods are composed of an optional search step and a mandatory poll step. The search step may use any method to explore a finite number of points in the solution space, and is aimed to diversify the search. In this approach as we want to explore the neighborhood of the current solution, we use the poll step only. In direct search algorithms developed for constrained optimization the infeasibility is either considered in penalty functions, barrier functions, or managed using filter methods. In structural optimization, the initial solution obtained by rounding of the master problem optimal solution may be feasible or infeasible. To deal with infeasible solutions in the course of the search, our method combines ideas of the progressive barrier approach presented in (Audet and Dennis 2006b) to handle the general constraints and from (Abramson et al. 2004) to exploit the available informations about derivatives. Thus the search may start with an infeasible solution and it follows an opportunistic strategy : it accepts an iterate as soon as a a sufficient objective decrease is found or if a measure of the constraints violation is reduced. The definition of the mesh and the step size parameter are two important components of the method and we present them in the following.

3.5.1 Definition of the mesh and search directions

As a mesh adaptative direct search, the algorithm iterates on trial points located on a mesh M^i, defined as the union of neighborhoods of points y where the objective function has been evaluated before the start of iteration i. M^i is constructed from a finite set $D \in \mathbb{R}^n$ of positive spanning directions[2] $d_j \in D$ (for $j = 1, ..., n_D$). For convenience the set D is treated as a real $n \times n_D$ matrix whose columns are the vectors of directions.
We use the notation of (Abramson et al. 2004) to define the mesh :

$$M^i = \bigcup_{y \in S^i} \{y + \Delta_m^i Dz : z \in \mathbb{N}^{n_D}\} \tag{3.13}$$

with S^i the set of iterates where the objective and constraints have been evaluated in the previous iterations, and Δ_m^i the parameter describing the fineness of the mesh. A condition for convergence of mesh adaptative algorithms is that each direction $d_j \in D$ is the product Gz_j of a non singular generating matrix $G \in \mathbb{R}^n \times \mathbb{R}^n$ by an integer vector $z_j \in \mathbb{Z}^n$.

For the structural optimization problem with n discrete variables $y_1, ..., y_n$, we define the generating matrix G as a diagonal matrix of size n whose element (i, i) is the step length s_i associated to variable y_i. The matrix $Z \in \mathbb{Z}^{n \times n_D}$ whose columns are the integer vectors z_j, includes the identity matrix I, its opposite $-I$ and an additional set of directions G_g constructed using the mathematical structure of the objective function (as detailed in the following section), we have :

[2]i.e. such that non linear positie combinations of the directions span $\mathbb{R}^n$

$$D = G\,Z \tag{3.14}$$
$$Z = \begin{bmatrix} G_g & I & -I \end{bmatrix} \tag{3.15}$$

Remark that D, G, Z are fixed matrices, independent from the iteration number. The construction of G_g is discussed in the next paragraphs. Remark also that the mesh is conceptual and never actually constructed and the set S^i of solutions reduces here to a singleton y, as often in MADS applications.

With this new definition of D, the mandatory poll steps are performed as described in Audet and Dennis (2006b). At each iteration i the trial points lie on a mesh P^i characterized by its fineness parameter Δ_p^i and centered at the current iterate :

$$P^i = \{y + \Delta_m^i d : d \in D^i\} \subset M^i \tag{3.16}$$

where D^i is a positive spanning set such that $0 \notin D^i$ and for each $d \in D^i$,

- *d can be written as a nonnegative integer combination of the directions in D: $d = D.u$ for some vector $u \in \mathbb{N}^{n_D}$ that may depend on the iteration number i*

- *the distance from the frame center y to a frame point $y + \Delta_m^i d \in P^i$ is bounded above by a constant times the poll size parameter: $\Delta_m^i\,|d| \leq \Delta_p^i max\,\{\|d'\| : d' \in D\}$*

The poll size parameter Δ_p^i is a scalar to be chosen such that for every finite subset of indices I

$$\lim_{i\in I} \Delta_m^i = 0 \quad \text{if and only if} \quad \lim_{i\in I} \Delta_p^i = 0 \tag{3.17}$$

The poll size parameter Δ_p^i is updated at the end of the poll step, according to the result of the search, the update rule is further discussed in the presentation of the algorithm.

In the next paragraph we describe how the objective function is used to build G_g before the start of the algorithm, and how at each polling iteration the constraints values and derivatives are used to build a pruned set of search directions in order to exclude those leading to a deterioration of the objective or that increase the infeasibility.

3.5.1.1 Using the structure of the objective function

We use the structure of the objective function to exclude from the search the directions that lead to an increase of the objective for any value of the steps. We build the columns of the matrix G_g with the directions that may lead to an improvement of the objective, starting at any point of the solution space. G_g is constructed once at the beginning of the algorithm and is independent from the iteration number.

We consider the expression of the objective function (3.11) reproduced here for convenience :

$$C_{(x_k,y_k)} = ((\chi_k)^{opt} - (\delta_k + \frac{h_k.d_k + w_k.t_k}{\Delta_k}))^2$$

Each subexpression is considered recursively and admissible directions are defined such that the increase or decrease of the subexpressions can roughly compensate each other and the value of $C_{(x_k,y_k)}$ is kept small. In the following the term “sign” denote by a $+1$ value (respectively -1) the direction defined by an increase (resp. decrease) of the variable under consideration and 0 means that the variable is not modified.

The columns of Table 3.1 show the sign of the admissible variations of the subexpressions in the column header. Each line of the table shows an admissible combination of the signs of variations for the subexpressions and variables, such that the objective value may be unchanged for some steps taken along these directions. So, in 3.11, $(\chi_k)^{opt}$ is fixed (where k is the subproblem number as described in section 3.4) and zeroes in the first column mean that $(\chi_k)^{opt}$ is fixed (for convenience, the subscript k is omitted in the table).

We consider first the two subexpressions δ_k and $\frac{h_k.d_k+w_k.t_k}{\Delta_k}$ (remind that t_k is constant in this context). The first line of the table shows in the three first columns that if δ decreases (the element $(1,2) = -1$)then $\frac{h_k.d_k+w_k.t_k}{\Delta_k}$ must increase (the element $(1,3) = +1$) in order to keep $(\chi_k)^{opt}$ unchanged (0 in the first column).

Each subexpression of $\delta_k + \frac{h_k.d_k+w_k.t_k}{\Delta_k}$ is then considered recursively until it reduces to a single variable. So, in a next step δ_k cannot be further developed and the expression $\frac{h_k.d_k+w_k.t_k}{\Delta_k}$ is split into $h_k.d_k + w_k.t_k$ and Δ_k. Columns four and five are added ad for example, the fourth and fifth element of the first line show that for $\frac{h_k.d_k+w_k.t_k}{\Delta_k}$ to increase (increase represented by the element $(1,3) = +1$) a possible combination is keeping $h_k.d_k + w_k.t_k$ unchanged (the element $(1,4) = 0$) and decreasing Δ_k (the element $(1,5) = -1$).

The subexpression $h_k.d_k + w_k.t_k$ is then split in $h_k.d_k$ and $w_k.t_k$ and finally $h_k.d_k$ is split in h_k and d_k Tables 3.2 and 3.3 show the result of these two steps. We build an intermediate table $T1$ by replacing each line of 3.2 with the lines of 3.3 where the values of $h.d$ in table 3.3 are identical to the value of $h.d$ in the replaced line of 3.2. We repeat this operation to substitute the lines of the intermediate Table $T1$ into table 3.1, the table T obtained at the end of this operation presents one column for each subexpression and each variable of the objective function.

We define the columns of the generating matrix G_g as the transposition of the columns of T whose headers are single variables. The resulting matrix $Z = \begin{bmatrix} G_g & I & -I \end{bmatrix}$
is shown in Table 3.4, where line one is associated to δ and lines two, three, four and five to h, d, w and Δ, respectively.

$(\chi)^{opt}$	δ	$\frac{h.d+wt}{\Delta}$	$h.d+wt$	Δ
0	-1	+1	0	-1
0	-1	+1	+1	-1
0	-1	+1	+1	+1
0	0	0	-1	+1
0	0	0	+1	-1
0	+1	-1	-1	-1
0	+1	-1	-1	+1
0	+1	-1	0	+1

Table 3.1: Analysis of the objective function variations

$h.d + wt$	$h.d$	w
-1	-1	0
-1	-1	-1
-1	-1	+1
-1	0	-1
-1	+1	-1
0	-1	+1
0	0	0
0	+1	-1
+1	-1	+1
+1	0	+1
+1	+1	-1
+1	+1	0
+1	+1	+1

Table 3.2: Analysis of the objective function variations (continued)

$h.d$	h	d
-1	-1	-1
-1	-1	0
-1	-1	+1
-1	0	-1
-1	+1	-1
0	-1	+1
0	+1	-1
+1	-1	+1
+1	0	+1
+1	+1	-1
+1	+1	0
+1	+1	+1

Table 3.3: Analysis of the objective function variations (continued)

0	0	0	0	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1
1	0	1	0	-1	0	-1	0	1	0	1	0	1	0	1	0	1	0	-1	0	-1
0	1	0	1	0	-1	0	-1	0	1	0	1	0	1	0	1	0	1	0	-1	0
1	1	0	0	-1	-1	0	0	1	1	0	0	1	1	0	0	-1	-1	1	1	-1
-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1

Table 3.4: Z, columns 1–21

1	1	1	1	1	1	1	-1	-1	-1	-1	1	0	0	0	0	-1	0	0	0	0
0	-1	0	1	0	-1	0	-1	0	-1	0	0	1	0	0	0	0	-1	0	0	0
-1	0	-1	0	1	0	-1	0	-1	0	-1	0	0	1	0	0	0	0	-1	0	0
-1	0	0	-1	-1	1	1	-1	-1	0	0	0	0	0	1	0	0	0	0	-1	0
1	1	1	1	1	1	1	-1	-1	-1	-1	0	0	0	0	1	0	0	0	0	-1

Table 3.5: Z, columns 22–42

3.5.1.2 Using the constraints values and derivatives

Our method exploits the availability of the constraints derivatives to prune the set of poll directions D^i at each iteration i. We consider the set $\mathcal{A}$ of active constraints as the set of constraints not satisfied or near saturation, i.e. such that $g_{max} - g(y^i) \leq \epsilon$, where y^i is the current solution at iteration i and ϵ is a constant parameter over all iterations. $\mathcal{A}$ is used to prune the set D^i of polling directions as follows : a direction d is pruned from the set D^i if, for some constraint $g(y) \in \mathcal{A}$, $d_e.\nabla g_e(y^i) > 0$ for all component e of d and ∇g. Equivalently, the set $\overline{D^i} \subset D^i$, and is given by :

$$\overline{D^i} = \left\{ d \in D^i : \forall g(y^i) \in \mathcal{A} \ \exists e \mid d_e.\nabla g_e(y) \leq 0 \right\} \tag{3.18}$$

The definition set of each variable is also used to prune the set of search directions so that the discreteness of the variables is respected. The pruned poll set is then :

$$\overline{P^i} = \left\{ y + \Delta_m^i d : d \in \overline{D^i} \text{ and } y + \Delta_m^i d \in Y \right\} \tag{3.19}$$

3.5.1.3 Definition of the step size parameter

As for MADS, the poll step proceeds by conducting a series of exploratory moves in a neighborhood of the current iterate y^i. S is the set of successful iterations and U is the set of unsuccessful iterations. A new iterate is generated as follows :

$$y^{i+1} = y^i + \Delta^i d_i \qquad \text{if } i \in S \text{ (successful iterate)} \tag{3.20}$$

$$y^{i+1} = y^i \qquad \text{if } i \in U \text{ (unsuccessful iterate)} \tag{3.21}$$

At the poll step, the mesh fineness is given by the mesh parameter Δ_m^i and we the poll parameter Δ_p^i must satisfy $\Delta_p^i \geq \Delta_m^i$ for any iteration i with

$$\lim_{i \in I} \Delta_m^i = 0 \text{ if and only if } \lim_{i \in I} \Delta_p^i = 0 \tag{3.22}$$

for any infinite subset of indices I. At each successful iteration the mesh parameter can be increased to coarsen the mesh -which may speed up the convergence- and at each unsuccessful iteration, the mesh parameter is reduced, with a minimal value of 1. We define a simple update rule that ensures that all the iterates lie on a rational lattice compliant with Y, the definition set of discrete variables y_k :

$$
\begin{array}{lll}
\Delta_{i+1} = \phi_i \Delta_i & \text{with} & \phi_i \geq 1 \\
\Delta_{i+1} = \theta_i \Delta_i & \text{with} & 0 < \theta_i < \theta_{max} < 1
\end{array}
$$

with:

$$
\begin{array}{lll}
\theta_i = \Lambda^{m_i}, & m_i \in -1, -2, ... & \text{for all } i \in U \\
\phi_i = \Lambda^{l_i}, & l_i \in 0, 1, 2, ... & \text{for all } i \in S
\end{array}
$$

Λa constant positive integer.

Updating the iterates 3.20 using these formula leads to the following property:

$$\Delta_{i+1} = \Lambda^{\Gamma_k} \Delta_0 \quad \text{with} \quad \Gamma_k \in Z \tag{3.23}$$

Choosing a suitable Δ_0 with respect to the step s^i used in the definition of Y and imposing $\Gamma_k > 0$ ensure that all iterates are in the set Y.

3.5.2 Algorithm

With the above adaptations, the algorithm to optimize the scantling of a panel k is a straightforward implementation of DS as presented for example in Kolda et al. (2003).

The first step of the algorithm starts with the definition of an initial solution (x_0, y_0) by applying a closest rounding operation (as defined in section 2.4) to the values of the variables (y_k) of the initial solution χ_0. Δ_{tol} is the minimal value of the mesh parameter and is set to one, under this value the search is stopped. In our experiments Δ_0 takes an arbitrary initial value set to 4, 2 and 1, $\theta_{max} = 1$, and $\phi_i = 1$ and $\theta_i = 1/2\ \forall i$. The generating set G is built as described in the previous section.

For each iteration i, D_i is built using the objective structure, the variables definition sets, and the active constraints pruning. At each iteration, if the current solution is feasible, directions $d_i \in D_i$ are successively considered. If a descent direction is found that provides a feasible solution, the iterate is declared successful, the incumbent solution is updated,the mesh parameter remains unchanged (as $\phi = 1$), and the infeasible solution counter u is reset. If no such direction is found, the iteration is declared unsuccessful, meaning that the current solution is optimal for the current grid and u is incremented.

Algorithm 6 *DS.Solve*$(S(x,y), \chi_0, v_S, i_S)$

```
**Initialization Step :**
Build an initial solution $(x_0, y_0)$
Initialize $\Delta_{tol}$, $\Delta_0$, $\theta_{max}$
Initialize $G$, the generating set, a positive basis of the variables space
$i = 1$ , $u = 0$
**repeat**
  **Poll Step :**
  Let $D_i = d \subseteq G$
  **if** $(x_i, y_i) \in S$ **then**
    **if** $\exists\, d_k \in D$ such that $(x_{i+1}, y_{i+1}) \in S$ and $C(x_{i+1}, y_{i+1}) < C(x_i, y_i)$
    **then**
      $(x_{i+1}, y_{i+1}) := (x_i, y_i) + \Delta_i d_i$
      $\Delta_{i+1} = \phi_i \Delta_i$, with $\phi_i \geq 1$
      $u = 0$
    **else**
      **if** $((x_{i+1}, y_{i+1}) \in U$ or $C(x_{i+1}, y_{i+1}) \geq C(x_i, y_i))$ for all $d \in D_i$
      **then**
        $(x_{i+1}, y_{i+1}) := (x_i, y_i)$
        $\Delta_{i+1} = \theta_i \Delta_i$, with $0 < \theta_i < \theta_{max} < 1$
        $u = u + 1$
      **end if**
    **end if**
  **else**
    **if** $(x_{i+1}, y_{i+1}) \in S$ **then**
      $(x_{i+1}, y_{i+1}) := (x_i, y_i) + \Delta_i d_i$
      $\Delta_{i+1} = \phi_i \Delta_i$, with $\phi_i \geq 1$
      $u = 0$
    **else**
      **if** $cv(x_{i+1}, y_{i+1}) < cv(x_i, y_i)$ **then**
        $(x_{i+1}, y_{i+1}) := (x_i, y_i) + \Delta_i d_i$
      **end if**
      $u = u + 1$
    **end if**
  **end if**
  $i = i + 1$
**until** $\Delta_i < \Delta_{tol}$ or $u == u_{max}$
```

The mesh parameter is reduced by a factor θ and the search continues on a finer grid.
As the initial solution may be infeasible, two situations can arise such that we declare that the iteration is successfull. If the solution at the start of the iteration is infeasible and the iterate provides a feasible solution the iterate is declared successful, the incumbent solution is updated, the mesh parameter remains unchanged, and the infeasible solution counter u is reset. In the case where the solution at the start of the iteration is infeasible and the iterate fails to find a feasible solution, the iterate is declared successful if a trial -not feasible- solution has been found in the course of the iteration, that reduces some measure of the unfeasibility (noted $cv(x, y)$). Then the incumbent solution is updated, the mesh parameter remains unchanged, and the infeasible solution counter u is incremented. In any other situation the iteration is declared unsuccessful, u is incremented and the search continues on the current grid.
These steps are repeated iteratively until the mesh parameter goes beyond one, meaning that the values considered would not be in the definition sets of the variables anymore. Another stopping criterion limits the number of unsuccessful consecutive iterations that could be infinite if the algorithm visits repeatedly a sequence of trial points.

3.6 Computational experiments

3.6.1 Industrial test cases

We apply the Decomposition and Direct Search (DDS) method on a real instance modeling a passenger ship produced at STX, a major european shipyard that produces complex ships : large cruise ships, ferries, military ships, offshore and specialized vessels. Among these are the famous sister ships “Oasis of the Seas” and “Allure of the Seas”. These two world largest cruise ships are floating cities large enough for 8,000 passengers and crew, with a landscaped park, whopping theatres, casinos... and even an onboard hospital.

Before going to sea, every ship has to be certified by a recognized organization such as Bureau Veritas, ABS, etc., which controls that ships verify a number of structural constraints related to strength and stability among others. The International Association of Classification Societies (IACS Ltd.) produces a number of documents including “the Quality Management System Requirements”, “IACS Unified Requirements” (including CSR), and “Common Structural Rules” used by classification societies and shipbuilding companies to design and produce ships.

The industrial test case is a passenger ship composed of 118 stiffened plate elements and 19 pillars. Based on structure symmetry, only the half structure is modeled and shown on Figure 3.1. The midship section of the ship is characterized by 14 decks, a 40 m breadth and a 45 m height.

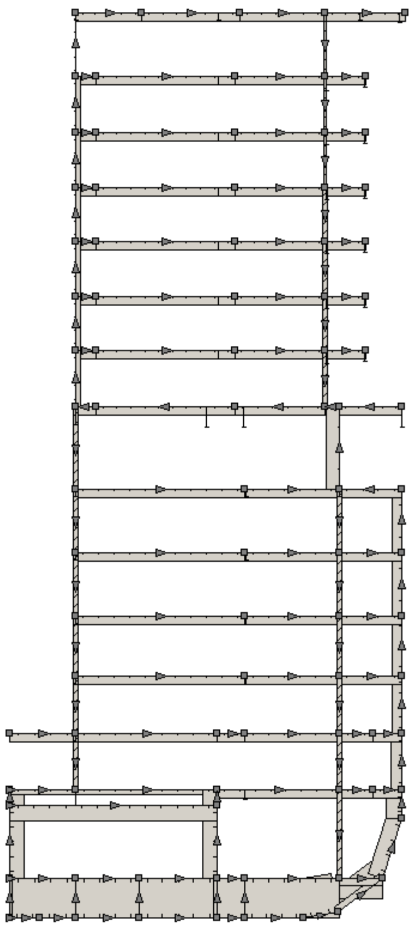


Figure 3.1: LBR-5 Model of the midship section

Five scantling design variables are defined for each stiffened plate element: plate thickness, flange width, web height and thickness of longitudinal stiffeners and spacing between two longitudinal stiffeners. The solution space for the discrete design variables is defined with a step of 1 mm for the thicknesses and 10 mm for the web height and flange width. The size of the solution space for the discrete variables is roughly 38. 10^{439} solutions. Due to equality constraints imposed between the spacing of elements which are vertically aligned, the stiffener spacing is defined as a continuous design variable.

Four global constraints on the hull girder are considered, they are the weight (bounded above), the section modulus and moment of inertia (bounded below) and the gravity center shift (doubly bounded). This last constraint aims at avoiding stability trouble.

Ten load cases were considered in the calculation:

- two "IACS load cases" (hogging & sagging): still water bending + wave bending with a probability of exceedence = 10^{-8}
- eight "BV load cases" (hogging & sagging)
  - Load case "a": still water bending + wave bending with a probability of excess = 10^{-5} + sea pressure (scantling draft & ballast draft)
  - Load case "b": still water bending + wave bending with a probability of excess = 10^{-5} + sea pressure (scantling draft & ballast draft) + inertial pressure

The model is thus characterized with:

- 118 stiffened plate elements,
- 383 design variables (including 308 discrete variables),
- 10 load cases
- 1418 constraints
- 56 equality constraints.

3.6.2 Preliminary results

The Decomposition and Direct Search (DDS) algorithm has been tested with the following parameter sets :

- DDS4-P : the initial poll step size $\Delta_p = 4$ and pruning is active.
- DDS4-NP : the initial poll step size $\Delta_p = 4$ and no pruning is performed based on the constraints.
- DDS1-P : the poll step size $\Delta_p = 1$ is constant over the entire algorithm execution, and pruning is active.
- DDS1-NP : the poll step size $\Delta_p = 1$ is constant over the entire algorithm execution, and no pruning is performed based on the constraints.

Table 3.6: Experiments: Decomposition and Direct Search Algorithm applied on a cruise ship.

	DDS4-P	DDS4-NP	DDS1-P	DDS1-NP
$W(P(\chi))$	6551117	6551117	6551117	6551117
$Z(P(x,y))$	7056146	7023685	6928474	6928474
$W\%Z$	7.71	7.21	5.76	5.76
$Feasible$	100%	98%	100%	98%
$Iterations$	41	124	22	18

Table 3.6 shows the best values[3] W of the relaxed problem $P(\chi)$, the best values Z found for the discrete problem $P(x, y)$, the increase of the objective function in percent, labeled $W\%Z$, the percentage of subproblems for wich DS finds an feasible solution starting from an infeasible initial solution, and the mean number of DS iterations performed on these panels.

The results show for the four parameters settings an increase of the objective function with respect to the optimal value obtained for the continuous relaxation $P^c(x, y)$ of the original problem $P(x, y)$.

Ninety nine subproblems are optimized with the DS procedure. We notice that the initial rounding of the optimal solution of $P^c(x, y)$, when feasible, is never improved by the DS procedure; the objective of the subproblems is to find values of the variables that induce the smallest variation of the objective function, and a solution obtained by closest rounding, is the optimal solution of the discrete subproblem if it is feasible.

Most subproblems with an infeasible initial solution are solved by the DS method with a small number of iterations and a few require a larger number of iterations, although this number stays very reasonable. For these subproblems, preliminary experiments show that a constant poll size parameter equals to 1 provide the smaller increase in the objective function, as it performs the exploration in a neighborhood of the initial solution. Preliminary tests show that when a subproblem starts with an unfeasible solution, a large initial poll size is preferable, allowing a wide exploration of the solution space.

The execution time is closely related to the initial grid size, it remains about fifteen seconds for the optimization with a dual core CPU at 2.5GHz.

These numerical experiments are very simple and limited, but they provide insights on the algorithm behavior and on possible improvements. In

[3]Numerical values have been scaled by a constant factor for confidentiality purposes.

particular in this simple implementation, the poll mesh size parameter is initialized and modified identically in each direction of the search. Independent initial values and updates in each dimension would allow an exploration of the solution space more similar to MADS methods and would certainly give interesting results. Another interesting topic is the order in which the search directions are considered at each step of the Direct Search. Varying this order can have a deep impact on the final solution as the search is opportunistic.

3.6.3 Convergence of the algorithm

In this section we briefly discuss the convergence of the Decomposition and Direct Search method.

First we show that the DDS method tends to converge toward a discrete local optimum if the subproblem optimization method converges also towards a local discrete optimum. Then we show that the direct search method presented in Algorithm 6 converges towards a discrete optimum or reduces the infeasibility of its initial solution.

The decomposition algorithm 4 performs a nonlinear optimization of a continuous relaxation of the original problem, starting with a feasible or infeasible initial solution. The *Solve* algorithm (namely *Conlin*) converges quickly toward a local optimum, starting from a feasible or nearly feasible initial solution. For strongly infeasible initial solutions, *Solve* may not produce a feasible solution and the DDS algorithm stops.

Starting from a continuous initial solution, the decomposition method builds independent subproblems whose objective is to minimize the difference between the optimal solutions of the subproblem and the corresponding component of the optimal solution of the relaxed problem $P^c(x, y)$. If the discrete optimal solutions of all subproblems $S_k(x, y)$ form a feasible solution for the main problem $P(x, y)$ then the DDS algorithm stops and returns this solution. If some solution of the subproblems is not feasible or if the solution formed with the discrete solutions does not satisfy the original problem constraints then the DDS algorithm starts a new iteration at step 1 with the discrete solution used as the starting point. Although there is no proof of convergence of this heuristic, Sobieszczanski (1993) reports satisfactory convergence properties of the method for structural optimization.

The Direct search algorithm is the other part of the method for which the convergence must be discussed. In Direct Search Methods, convergence analysis stress that a stationary point is reached if there exist a subsequence of step size parameters that tends to zero over the course of iterations. In our approach, the step size is bounded below by 1 and we show that if a feasible solution is produced, it is a local optimum in its neighborhood, and it is reached in a finite number of iterations.

At each iteration a finite set of directions is considered. Observe that each successful iteration $i \in S$ produces a feasible solution and the decrease of the objective function is bounded below by a minimum value depending on the step size and the current mesh size : $f_{k+1}(y) < f_k(y) - \rho(\Delta_k)$. As each successful iterate must improve the objective and the objective is bounded below (by zero), the sequence of successful iterates is finite and at optimality we have :

$$f(y) \leq f(y')\text{for all}y' \in N(y) \tag{3.24}$$

where $N(y)$ is a finite set of neighbors around and including (y) on the current mesh.

In the case that the iteration does not produce a feasible solution $(i \in U)$, two situations may occur : there is or not a reduction of a given measure of infeasibility. We define a reduction of infeasibility as follows : either the number of violated constraints is reduced either it remains constant and at least one constraint violation is reduced in value. This definition implies that the set of infeasible solutions is finite (as for the objective, the modulus of a reduction of constraint violation is bounded below by a minimum value depending on the mesh size). During the course of the algorithm if the incumbent solution is not feasible and if the trial solution reduces the measure of infeasibility compared to the incumbent, then the trial solution is accepted as the new incumbent (the move is accepted), otherwise it is rejected. If no feasible solution is found when all directions have been tried since the last accepted move, the iteration is declared unsuccessful and the mesh size is reduced.

The iterative process decreases the mesh parameter Δ_m^i at each unsuccessful iteration until $\Delta_m^i = 1$. With the definition of the generating set G we can conclude that at the end of the iterative process, if a feasible solution is produced then it is a local optimum of the problem.

3.7 Discussion and conclusions

In this chapter we have developed a new methodology to solve discrete structural optimization problems whose structural analysis is fast to compute. Decomposition methods have been previously applied to solve continuous structural optimization problems. In this chapter we have combined decompostion and a discrete optimization method to the subproblems. An direct search algorithm is implemented to solve the discrete subproblems, that exploits the structure of the mathematical formulation and the knowledge of the objective and constraints derivatives.

Simple experiments have been driven that show encouraging results. More tests and parameter tuning are in our perspectives for future work, as is further validation of the method by experimentation on standard discrete structural optimization problems.

Part II

Assembly Area Scheduling

Introduction[4]

In the past decades the marine industry has undergone significant evolution and the production of large passenger ships faced dramatic changes. The size, the complexity and the security standards of the ships have increased and the ship-owners have become less and less willing to wait once the order is placed. In the meantime, to face a growing intense competition, the shipyards have had to improve their efficiency and master their production costs: they progressively moved from manufacture to automated production process. As a consequence, the shipyards are now facing this difficult challenge: produce more complex ships, cheaper and faster.

The naval industry has had to adapt its internal process while keeping its own strengths, combining the workers' experience and new techniques: new production facilities, sophisticated production equipments (such as laser sources for welding, drilling and cutting), etc. Maybe less dramatically, design methods and tools, shipbuilding process and management techniques have evolved and keep evolving as well.

The building of large ships requires the production and the assembly of tens, or even hundreds of thousands of steel elements. It includes welding hundreds of kilometers of lines, painting several hundred thousand square meters of surface, and handling sub-assemblies larger than many private houses. These complex and numerous activities require careful planning and logistics.

Formerly, most of the work took place in a dry dock, with the ship constructed almost piece by piece from the ground up. However, advances in technology and more detailed planning have made it possible to divide the vessel into subunits, called blocks and panels, which integrate electrical equipment, pipes and other systems. The blocks are assembled in dedicated halls and are composed of subparts and sub-assemblies produced in other facilities. Then, the blocks are transported to the dry dock where they are fitted together. This process is faster, less expensive and provides better quality control than previous practices. Furthermore, it leads itself to increased use of automation and robotics, which not only decrease costs, but also reduce the workers' exposure to chemical and physical hazards.

In this chapter we present new methods to improve production facility management of shipyards. We define a space and time allocation problem that arises in assembly halls producing large building blocks. Although the application is the assembly of prefabricated keel elements in shipyards, this

[4] The material presented in this chapter has been published in *Annals of Operations Research*, Volume 179 Number 1 (2010), under the title "Space and Time allocation in a shipyard assembly hall". It is a join work with Yves Langer and Yves Crama (HEC Management School, University of Liège).

method is generic enough to be applied to other types of production facilities. The building blocks are very large, and, once a block is placed in the hall, it cannot be moved until all assembly operations on this block are completed. Each block must be processed within a predetermined time window. The objective is to maximize the number of building blocks produced in the hall.

The problem is modeled as a 3-dimensional bin packing problem (3D-BPP) and is handled by a Guided Local Search heuristic initially developed for the 3-Dimensional Bin Packing problems. Our computational experiments with this heuristic demonstrate that very good results can be found within minutes on a workstation, and that the heuristic outperforms a standard constraint programming approach. We also describe some additional real-life constraints arising in the industrial application and show how these constraints can be conveniently integrated in the model.

This study was carried out at *Aker Yards France* (previously known as *Les Chantiers de l'Atlantique*), one of the major European shipyards located in Saint Nazaire, France, at the mouth of river Loire. This shipyard covers all the activities involved in the shipbuilding process, from the basic pre-design phase to the delivery of sea-ready vessels. Its main product lines consist of passenger ships (big cruise liners and car ferries), LNG tankers, military ships (frigates and logistical ships), etc. From this shipyard came many famous liners such as "*France*" (1912), "*Ile De France*" (1927), "*Normandie*" (1935) and "*Norway*" (1960). Recently, *Aker Yards France* built up one of the world's largest ocean liner, "*Queen Mary 2*". Its production, completed in 2003, took less than two years.

Chapter 4

Models for space and time allocation problems

4.1 Problem statement

We focus on the space allocation and planning decisions concerning the assembly halls. Each hall is fully dedicated to the production of blocks and is divided into several equal-sized rectangular areas (e.g., four areas for the main hall under study), each of which is large enough to contain a few blocks simultaneously (see Section 5.2.1 for additional details). The blocks are voluminous and heavy so that, once a block has been brought to the hall, it cannot be moved again until all required assembly operations have been completed on it. Then, it is transported out of the hall. The objective, as defined by the shipyard managers, was to maximize the number of building blocks produced in a given hall over a certain time horizon (we'll return to this point in Section 5.3). In the sequel, we refer to this problem as the *Space and Time Allocation* (STA) problem for shipyard assembly halls.

4.2 Mathematical model

We consider the STA problem associated with an assembly hall subdivided into a number A of identical, rectangular two-dimensional areas. Each area has width W, length L and height H. There is a set of n blocks to be produced. Each block is viewed as a parallelepiped and is characterized by its geometric dimensions (width w_j, length l_j and height h_j, for $j = 1, 2, \ldots, n$), as well as by its production requirements such as processing time t_j (the number of days needed for its assembly), release date r_j (the date when the required parts are available for assembly) and due date d_j (the date when the block is to be delivered at the dry dock).

In order to be processed, each block should be placed in any area of the hall. The only constraints are that a block j cannot be moved to the hall before its release date r_j, must remain in the hall without interruption for at least t_j time units, and must leave before its due date d_j. Moreover, each block is always positioned with its sides parallel to the walls of the hall; two blocks cannot overlap physically and cannot be placed on top of each other. As a result of the latter constraint, the height of the blocks and of the hall will not play any role in most of our discussion. (We will return briefly in Section 5.3.1 to the practical consequences of the limited height of the hall and to additional constraints on individual blocks.)

Thus, the STA problem consists in orthogonally ordering the blocks into the rectangular areas, without overlap and so as to respect the time constraints, with the objective to produce the largest possible number of building blocks.

Let us define six decision variables for each block $j = 1, 2, \ldots, n$:

- $b_j \in \{0, 1\}$, indicating whether block j will be produced or not,
- $a_j \in \{1, 2, \ldots, A\}$, indicating the area where block j will be produced,
- x_j and y_j, coordinates representing the position of the upper-left corner of block j in the selected area,
- $o_j \in \{0, 1\}$, indicating the orientation (either longitudinal or transversal) of block j in the selected area,
- s_j, the starting date for the assembly of block j.

A solution, that is to say an assignment of values to the above variables, is *feasible* if the *individual* and the *collective* constraints are met. We call individual constraints those which bear on one block only, regardless of the other blocks. The individual constraints can be modelled as follows:

(1) each block must fit within the width of an area: $x_j \geq 0$ and $x_j + [o_j w_j + (1 - o_j) l_j] \leq W$;

(2) each block must fit within the length of an area: $y_j \geq 0$ and $y_j + [o_j l_j + (1 - o_j) w_j] \leq L$;

(3) each block must fit in its time window: $s_j \geq r_j$ and $s_j + t_j \leq d_j$.

On the other hand, collective constraints deal with the interaction between the positions of different blocks. Unless we mention otherwise, the only collective constraint is that the blocks may not overlap.

4.2.1 Relation to the 3-Dimensional Bin Packing problem

The STA problem described above is closely related to the *3-Dimensional Bin Packing problem* (3D-BPP). Recall that in 3D-BPP, we are given a set of n parallelipipeds, each characterized by its width w_j, length l_j and height h_j $(j = 1, 2, \ldots, n)$, as well as an unlimited number of identical three-dimensional bins with width W, length L and height H. The 3D-BP problem consists in orthogonally packing all the items in the minimum number of bins; see e.g. Martello et al. (2000).

We have already observed that the height of the assembly hall does not play any active role in the STA problem, since building blocks cannot be stacked upon each other. Time, therefore, behaves as the third dimension of the model (similar models are mentioned by Dyckhoff (1990). We also note that a “2-dimensional" version of STA is described (without explicitly identifiying the industrial environment) by Imahori, Yagiura, and Ibaraki (2003, 2005). In these papers, the authors observe that each building block has nearly the same width as the assembly hall, so that the x-dimension does not play any active role. (One of the halls at *Aker Yards France* also is of this type.)

A major difference between 3D-BPP and the STA problem is that, in the former, items/blocks must fit in the bin height ($z_j \geq 0$ and $z_j + h_j \leq H$), whereas they must fit in their individual time window in the latter ($s_j \geq r_j$ and $s_j + t_j \leq d_j$). Moreover, the two problems deal with different objective functions (the so-called *knapsack loading* or *container packing/* problem are variants of 3D-BPP whose objective function is more closely related to the objective of STA; see e.g.Brunetta and Grégoire (2005), Dyckhoff (1990), Martello et al. (2000). The distinction between minimizing the number of bins and maximizing the number of blocks will be mitigated, in a first step of our approach, by concentrating on the search for feasible solutions, rather than on the optimization version of the problem (see Section 5.1.1 and Section 5.1.2).

3D-BPP is a generalization of the well-known (1-Dimensional) Bin Packing problem and it is therefore strongly NP-hard; see e.g.Coffman, Galambos, Martello, and Vigo (1999), Coffman, Garey, and Johnson (1997), Dyckhoff (1990), Dyckhoff, Scheithauer, and Terno (1997), or Garey and Johnson (1979) for classifications of bin packing problems and more information about their computational complexity.

Brunetta and Grégoire (2005), Faroe, Pisinger, and Zachariasen (2003), Martello, Pisinger, and Vigo (2000) are recent papers which provide brief surveys of the literature on 3D-BPP. Since the problem is hard, most efficient approaches rely on *local search metaheuristics* for the solution of large-scale instances. More specifically, Faroe, Pisinger, and Zachariasen (2003) have proposed a *Guided Local Search* (GLS) heuristic for 3D-BPP. In their computational experiments, this approach appears to outperform the best available heuristics for 3D-BPP. It also offers a high degree of flexibility in its implementation, so that it can be easily adapted to variants of the problem involving different objective functions and/or additional constraints (such as some of the real-world side-constraints discussed in Section 5.3 below). Therefore, the algorithm that we have developed for STA explicitly builds on their work. We now proceed to describe it.

Chapter 5

Methods for space and time allocation problems

5.1 A three steps algorithm

5.1.1 Finding feasible solutions

In this section, as in Faroe et al. (2003), we first concentrate on the problem of finding at least one feasible solution for the set of blocks initially given. Of course, if such a feasible solution is found, then no further optimization is needed. We will see in Section 5.1.2 what should be done in the opposite case.

5.1.1.1 General approach

Let X be any solution of the STA problem, that is, any assignment of values to the variables a_j, x_j, y_j, o_j and s_j for $j = 1, 2, \ldots, n$. (We implicitly assume that $b_j = 1$ for all j.) While trying to find a feasible schedule, our local search heuristic strictly enforces the individual block constraints defined in Section 2, meaning that X always satisfies the constraints (1)-(3). On the other hand, we do not enforce the collective constraints, but we measure the extent of their violation and these measures are summed in an auxiliary objective function to be minimized. Without additional real-life collective constraints, the extent of the violations can be measured by the total "volume" (in "square meters $\times$ days") of pairwise overlaps between the blocks. Thus, if we denote by $overlap_{ij}(X)$ the volume of the overlap between blocks i and j, then the auxiliary objective function can be formulated as

$$f(X) = \sum_{1 \leq i < j \leq n} overlap_{ij}(X) \tag{5.1}$$

Starting from an arbitrary infeasible solution where blocks can overlap, searching for a feasible solution can be achieved by minimizing the function f, since an objective value of zero indicates that all the collective constraints are satisfied.

A typical local search procedure proceeds by moving from the current solution X to another solution X' in a neighborhood $\nu(X)$ whenever this move improves the value of the objective function. Slightly adapting the framework of Faroe et al. (2003) (who do not allow rotating the boxes), we define the neighborhood $\nu(X)$ as the set of all solutions that can be obtained by translating any single block along the coordinate axes or along the timeline, or by a move to the same (relative) position in another area of the hall, or by a ± 90 degree rotation of a block around one of its four corners[1]. A neighbor of X is therefore constructed by assigning a new value to exactly one of the variables x_j, y_j, s_j, a_j or o_j. It is clear that this definition allows to move from any solution to any other solution through a sequence of neighbors.

It is well-known that local search procedures may easily get stuck in a local minimum of poor quality. Several strategies have been proposed to avoid this shortcoming of simple descent algorithms, among which simulated annealing (see e.g.Aarts and Korst (1989), tabu search (see e.g.Glover (1990), variable neighborhood search (see Hansen and Mladenovič (2001)), and many others.

Another difficulty with local search procedures is that the neighborhood of any given solution may be quite large (even if continuous, variables like x_j, y_j or s_j can be discretized for practical purposes) and therefore, exploring the neighborhood to find an improving move can be very costly in computing time.

To deal with the above issues, we rely on a the *Guided Local Search* (*GLS*) heuristic, and its accompanying neighborhood reduction scheme called *Fast Local Search* (*FLS*).

5.1.1.2 Guided local search

Guided Local Search has its roots in a neural network architecture developed by Wang and Tsang (1991), which is applicable to a class of problems known as *Constraint Satisfaction* problems. The current GLS framework, with the accompanying FLS, has been first proposed by Voudouris (1997) and Voudouris and Tsang (1997, 1999).

Generally speaking, GLS augments the objective function f of a problem to include a set of penalty terms associated with "undesirable features" of a solution, and it considers the new function h, instead of the original one, for minimization by a local search procedure. The local search procedure is

[1]In the case of a rotation around a corner, moving to a neighbor also involves corresponding changes in x_j and in y_j. To simplify the explanation, this technical issue will be ignored in the sequel.

denoted *LocalOpt* in our description of *GLS* (see Algorithm 1). Local search is confined by the penalty terms and focuses attention on promising regions of the search space. Each time *LocalOpt* gets caught in a local minimum, penalties are modified and the *LocalOpt* search is called again to minimize the modified objective function. (This is akin to the use of diversification strategies in tabu search or in variable neighborhood search.)

This general scheme has been adapted to 3D-BPP by Faroe, Pisinger, and Zachariasen (2003). In their procedure, the *features* of a solution X are the Boolean variables $I_{ij}(X) \in \{0,1\}$, which indicate whether blocks i and j overlap ($I_{ij}(X) = 1$) or not ($I_{ij}(X) = 0$). The value of $overlap_{ij}(X)$ measures the impact of the corresponding feature on the solution X.

The number of times an "active" feature has been penalized is denoted by p_{ij}, which is initially zero. Thus, the augmented objective function takes the form

$$\begin{aligned} h(X) &= f(X) + \lambda \sum_{1 \le i < j \le n} p_{ij}\, I_{ij}(X) \\ &= \sum_{1 \le i < j \le n} overlap_{ij}(X) + \lambda \sum_{1 \le i < j \le n} p_{ij}\, I_{ij}(X) \qquad (5.2) \end{aligned}$$

where λ is a parameter - the only one in this method - that has to be chosen experimentally (see Section 5.2.2).

Intuitively speaking, *GLS* attempts to penalize the features associated with a large overlap, but which have not been penalized very often in the past. More formally, we define a utility function $\mu_{ij}(X) = overlap_{ij}(X)/(1 + p_{ij})$ for each pair of blocks (i, j). At each iteration the procedure adds one unit to the penalty p_{ij} corresponding to the pair of blocks with maximum utility then calls *LocalOpt*$(X, (i, j))$ (see Algorithm 1). In a sense, the search procedure is commanded to set a priority on these features. Since features with maximum utility keep changing all the time, this guiding principle prevents *GLS* from getting stuck in local minima. The algorithm is run for T time units at most.

5.1.1.3 Fast local search

In our implementation, the procedure *LocalOpt* mentioned in Algorithm 1 is a so-called *Fast Local Search* (FLS) procedure adapted from Voudouris and Tsang (1997) and Faroe, Pisinger, and Zachariasen (2003). The main objective of FLS is to reduce the size of the neighborhoods explored in the local search phase, by an appropriate selection of moves that are likely to reduce the overlaps with maximum utility.

To describe FLS, consider any solution X and any variable m among the variables x_j, y_j, s_j, a_j, o_j with $j \in \{1, \ldots, n\}$. Informally, FLS selects at random a variable m within a list of active variables, as long as this list

Algorithm 7 *GLS*(X_0,T); {X_0 is the initial solution, T is the time limit }

```
$X := X_0$; {$X$ is the current solution}
$X^* := X_0$; {$X^*$ is the best available solution}
$t = 0$; {$t$ measures the run time}
repeat
    Select a pair $(i, j)$ with maximum utility;
    $p_{ij} := p_{ij} + 1$; {Increase the penalty of pair $(i, j)$}
    $X := LocalOpt(X, (i, j))$ {Run FLS with new penalties}
    if $h(X) \leq h(X^*)$ then
        $X^* := X$;
    end if
until $h(X) = 0$ or (elapsed time $t \geq T$)
return $X^*$
```

is not empty (active variables are those which are most likely to lead to an improvement of the current solution). Then, FLS searches within the domain of m for an improvement of the objective function. If no improvement is found, then the variable m becomes inactive and is removed from the list until the end of the current call to *LocalOpt*.

More formally, we define $\nu_m(X)$ as the set of all solutions which differ from X only by the value of variable m. The neighborhood $\nu(X)$ is thus divided into a number of smaller sub-neighborhoods:

$$\nu(X) = \bigcup_m \nu_m(X).$$

Each of the sub-neighborhoods $\nu_m(X)$ can be either *active* or *inactive*. Initially, only some sub-neighborhoods are active. (We will show at the end of this section how the selection process is designed to focus on the maximum utility overlaps.) FLS now continuously visits the active sub-neighborhoods in random order. If there exists a solution X_m within the sub-neighborhood $\nu_m(X)$ such that $h(X_m) < h(X)$, then X becomes X_m; otherwise we suppose that the selected sub-neighborhood will provide no more significant improvements at this step, and thus it becomes inactive. When there is no active sub-neighborhoods left, the FLS procedure is terminated and the best solution found is returned to *GLS*.

The size of the sub-neighborhoods related to the a_j and the o_j variables is relatively small, therefore FLS is set to test all the neighbors of these sets. On the other hand, using an enumerative method for testing the translations along the x, y and s-axes would be very time consuming, especially when areas and/or time windows are large. We may observe, however, that only certain coordinates of such neighborhoods need to be investigated. Indeed, as pointed out by Faroe, Pisinger, and Zachariasen (2003), all $overlap_{ij}(x)$

functions (respectively $overlap_{ij}(y)$, $overlap_{ij}(s)$) are piecewise linear functions, and will for that reason always reach their minimum in one of their breakpoints or at the limits of their domains. (Thinking of the geometry of the 3D-BP problem, one can easily understand that a best packing arises either when the boxes touch each other along their faces, or when they touch the sides of the bins.) As a result, FLS only needs to compute the values of $f(x)$ (respectively, $f(y)$, $f(s)$) for x (respectively, y, s) at breakpoints or at extreme values. In fact, there are at most four breakpoints for each overlap function, and only the first and the last one are evaluated.

Additionally, since changes in the total overlap function only depend on the value of the selected variable m, most of the terms of this function are constant. Thus, when evaluating the value of $f(X)$ after a move, only the n $overlap_{ij}$ terms depending on m should be computed, and so the computing time for the evaluation of one solution turns out to be linear in n. In the description of FLS (see Algorithm 2), we denote by $h_{partial}(m)$ this “partial" augmented objective function used to compare the impact of fixing m at different values.

The efficiency of FLS directly depends on the number of active sub-neighborhoods. Now, remember that $LocalOpt(X,(i,j))$ is called by GLS after some penalty p_{ij} has been adjusted with the aim to escape local minima. Thus, active sub-neighborhoods should be those which allow moves on the penalized features associated with the overlap of blocks i and j. Accordingly, Faroe, Pisinger, and Zachariasen (2003) propose to activate the moves on the two blocks i and j, as well as the moves on all blocks that overlap with block i and block j.

A schematic description of FLS is given by Algorithm 2.

5.1.2 Selecting the blocks

In the previous section, we described a GLS heuristic to find a feasible solution of the STA problem. If *GLS* works as expected, then it should return a space and time allocation with zero overlap (i.e., a feasible solution) when there is one. In general, however, no such feasible solution may exist for the set of blocks initially included in the instance, and we face the problem of selecting a maximum subset of blocks to be scheduled for assembly.

As mentioned in Section 4.2.1, this objective function differs from the usual objective function of 3D-BPP. In order to handle it, we rely on the following heuristic assumption:
(HA)*if GLS cannot find a feasible solution of* STA *within a predetermined amount of computing time T, then the heuristic assumption is that this instance is infeasible.*
As a consequence of this assumption, the search heuristic *GLS* can be used as a “blackbox" to carry out feasibility tests. We use the notation $GLS(X,T)$

Algorithm 8 *LocalOpt*$(X, (i, j))$; {X is the current solution; (i, j) is a pair of blocks }

```
  $ActiveList$ := List of the moves applicable to $i$, $j$ and to the blocks over-
  lapping either $i$ or $j$
  while $ActiveList \neq \emptyset$ do
    Pick a variable $m$ associated with a move in $ActiveList$
    Let $m^*$ be the current value of variable $m$
    $PositionList$ := List of relevant values of $m$
    for $k := 1$ to $|PositionList|$ do
      if $h_{partial}(PositionList(k)) \leq h_{partial}(m^*)$ then
        $m^* := PositionList(k)$;
      end if
    end for
    Let $X'$ be the solution obtained by performing move $m^*$ on $X$
    if $h(X') = h(X)$ then
      Remove $m$ from $ActiveList$ {No improvement}
    end if
    $X := X'$ {Execute the move}
  end while
  return $X$;
```

to indicate the output of procedure *GLS* when it is initialized with the (infeasible) solution X and executed for T time units.

Several procedures have been developed and tested based on this concept. A simple “descent method" is to initialize *GLS* with a randomly generated solution X_0 that includes the entire set of blocks (i.e., set $b_j = 1$ for all $j = 1, 2, \ldots, n$). After a search of T time units, the algorithm is stopped and returns $X_1 = GLS(X_0, T)$, the best solution found (in terms of overlap). Then, one of the blocks with the largest overlap is removed from the solution X_1, and the heuristic *GLS* is restarted from this solution. The entire procedure ends when a solution X_k with zero overlap is found; see Algorithm 3.

Algorithm 9 *BlockDescent*(X, T)

```
  Set $b_j := 1$ for $j = 1, 2, \ldots, n$;
  repeat
    Set $b_j := 0$ for any block $j$ such that $\sum_i overlap_{ij}(X)$ is maximum;
    {Remove block $j$}
    $X := GLS(X, T)$;
  until $f(X) = 0$
  return $X$;
```

A more efficient variant of this procedure, called $MaxBlocks(X,T)$, allows adding as well as removing blocks from the current set. Thus, assume that, at any iteration of the procedure, X is a solution (feasible or not) involving some subset of blocks. If the solution $GLS(X,T)$ returned by GLS is feasible, then this solution is a candidate to be the final optimal solution. So, we record it if it is better than the best incumbent solution X^*, and we try to include an additional block in the set. On the other hand, if $GLS(X,T)$ is not feasible, then a fast post-processing step is performed to produce a feasible solution X': this is achieved by simply removing blocks in a greedy fashion until all overlaps are canceled. The solution X' is recorded if is better than the incumbent X^*; then, we remove an overlapping block from $GLS(X,T)$ and the process is repeated. (Depending on the unfeasibility level of the solution, we may even want to remove more than one block at a time.) The procedure is stopped after a predetermined amount of computing time, or by any more sophisticated stopping criterion, and it returns the feasible solution X^* with the largest collection of blocks (note that, thanks to the post-processing phase, $MaxBlocks(X,T)$ always generates a feasible solution. A more precise description is given in Algorithm 4.

Algorithm 10 $MaxBlocks(X,T)$

```
$X^* := \emptyset$;
**repeat**
  $X := GLS(X,T)$;
  **if** $f(X) = 0$ (the current solution is feasible) **then**
    **if** $|X| > |X^*|$ **then**
      $X^* := X$
    **end if**
    In $X$, set $b_j := 1$ for any block $j$ such that $b_j = 0$; {Add a random block}
  **else**
    $X' := PostProcessing(X)$; {Generate a feasible solution}
    **if** $|X'| > |X^*|$ **then**
      $X^* := X'$
    **end if**
    In $X$, set $b_j := 0$ for any block $j$ such that $\sum_i I_{ij} \geq 1$; {Remove an overlapping block}
  **end if**
**until** (Stopping criterion)
**return** $X^*$;
```

One of our aims in this study, however, was to remain as close as possible to the industrial reality and to the working methods of the shipyard under study. From this point of view, the approaches outlined above suffer from one major drawback: they are likely to have aversion for the largest blocks and to reject such blocks before any others, since they are hardest to allocate. ("Large" may mean here: either large surface $l_j \times w_j$, or large processing requirements t_j. Scheduling algorithms are known to face similar difficulties when long tasks have to be placed.)

In the real-life situation, when the entire set of blocks cannot be produced, the operator in charge of scheduling can either move specific blocks to other assembly halls, or subcontract them to external workshops, or change some of the system parameters (e.g., increase the workforce to reduce processing times, postpone due dates, etc.). However, the shipyard did not provide us with formalized information which could describe all the relevant aspects of these choices, nor with an appropriate weighing scheme to evaluate the preferences among blocks.

For this reason, the software that we have developed allows the operator to change manually the collection of blocks to be allocated. In practice, starting from any solution X_0 (feasible or not), iterative executions of the form $X_{k+1} = GLS(X'_k)$ can be performed at will by the operator, where X'_k is a solution obtained by deliberate modifications of a previous solution X_k. (In particular, by switching any variable b_j from 0 to 1 or from 1 to 0.) This indicates to the operator if a particular set of blocks is feasible or not, and provides the corresponding allocation when there is one.

Finally, it should be observed that, in practice, real instances are likely to be "almost feasible" since the collection of blocks which make up such instances are selected in preliminary planning phases which take into account, at least approximately, the actual production capacity of the assembly halls. Therefore, simple optimization procedures combined with manual updates can quickly lead to good solutions. By offering access to the three procedures mentioned above, the industrial application gives the end-user a broad control of the data and of the computed results, so that he can easily evaluate various situations and take the appropriate decision based on several trials.

5.2 Computational experiments

This section presents the results of computational experiments with the different procedures described above. The objective of these experiments was to determine guidelines to set the values of the parameters of the procedures, to evaluate their performance on various benchmarks, and to provide a comparison with the performance of a generic package based on constraint programming techniques (ILOG 2007). The algorithms, written in C^{++}, have been run on a Pentium 3GHz with 2Gb RAM.

5.2.1 Data and test instances

Let us first provide some additional information regarding the types of problem instances encountered at the shipyard.

The amount of work required by each block depends on the complexity of the block, which typically depends on its position in the ship, on the type of ship under construction, and on many other factors. The processing time is thus extremely variable for different blocks, ranging from 2 to 40 days.

A typical assembly hall consists of four working areas of approximately 70×25 meters. The crane bridge in this hall is able to carry blocks weighing up to 180 tons; blocks of such weights typically have dimensions of the same order as the width of the ship under construction (e.g., 25-40 meters). About 100 blocks are scheduled at once, for a total time horizon of 6 to 9 months.

Several test instances have been established based on the features of this assembly hall. Each instance contains 100 blocks to be allocated to one of the four areas; the dimensions of each block (spatial and temporal) are compatible with the dimensions of the areas and with the time windows. Thus, the feasibility or infeasibility of each instance is only due to the interactions among the blocks, i.e., to the “collective constraints".

The instances are of 6 different types, labeled by a letter from A to F: types A to D correspond to “realistic" instances, type E to highly structured instances, and type F to random instances. The realistic instances are derived from industrial data and are meant to exhibit the main features of real data (shapes of the blocks, processing requirements and time windows characteristics). More specifically:

A : Instances A0–A5 are based on real data.

B : Instances B1–B5 are derived from A1 by multiplying the length of each block in A1 by a factor which increases with the label of the instance (from 1.06 to 1.10). Thus, the blocks in B1 are longer than the blocks in A1, the blocks in B2 are longer than those in B1, and so on.

C : Instances C1–C5 are similarly derived from A1 by multiplying the width of each block in A1 by a factor which increases with the label of the instance (from 1.03 to 1.05).

D : The multiplicative factors applied when generating instances of type B or C have more impact on large blocks than on small ones. In order to counter this effect, we build a set of instances D1–D5 where the multiplier is applied only to the smaller blocks, thus generating more homogeneous block sizes than in B and C. Instances D1–D3 are increasingly homogeneous. Instance D4 is meant to be difficult: the length (resp., the width) of each block is exactly half the length (resp., the width) of an assembly area. In instance D5, the length (resp., the width) of each block is exactly equal to the length (resp., the width) of an area, and the time windows are such that it is very clearly impossible to allocate all the blocks.

In the instances E1–E5, all the blocks have the same dimensions and durations, and these values increase from E1 to E5. The time window is also identical for all the blocks. Instances E1 and E2 are feasible, and E3–E5 are infeasible. In particular, E5 is the instance where all the blocks have the dimensions of an assembly area and duration equal to the time window, meaning that there are 25 times too many blocks with respect to the availability of the workshop.

Instances F0–F5 are randomly generated. The spatial dimensions of the blocks are normally distributed so that on average, 16 blocks can be placed in an assembly area. The durations t_j and the release dates r_j are uniformly distributed, so as to obtain a nearly constant load of the areas. The length of the time windows decreases with the label of the instances, which are therefore increasingly difficult.

In total, this yields 32 instances labeled A0-A5, B1-B5, C1-C5, D1-D5, E1-E5 and F0-F5. Some additional hard instances will be introduced in Section 5.2.4.

5.2.2 The λ parameter

The first experiments were designed to adjust the value of the λ parameter, which appears in the definition of the augmented objective function (eq. 5.2) and which is the main parameter of the *GLS* procedure (together with its total running time). The value of λ determines to what degree a penalty modifies the augmented objective value and drives the local search out of a local minimum. A large value of λ is supposed to make the search more aggressive, to avoid solutions with penalized features and to favor large jumps in the solution space with limited attention for the overlap term $f(X)$ in the augmented objective function. Small values of λ, on the other hand, may require heavier penalties p_{ij} to escape a local minimum but should result in a more intensive exploration of the neighborhood of the current solution and to a search strategy that is more sensitive to the gradient of $f(X)$. However, small λ values might prevent a broad exploration of the solution space.

We have tested the Guided Local Search algorithm with different values of λ in a broad range from 1 to 9000, and with a high limit (1200 sec.) on its total running time. The results obtained on a representative sample of feasible instances are displayed in Table 5.1 below. (Similar results were obtained for other instances.) As the computing time of the heuristic is random and may vary from one run to the next on any specific instance, the table displays the mean values of the computing time for 10 executions on each instance, as well as the percentage of the number of trials for which GLS was able to find a solution within 1200 seconds when this percentage is smaller than 100%. (In the latter case, the average computing time is reported for the solved instances only).

λ:	1	10	50	200
A1	*(0%)	153 (30%)	113 (80%)	69.2
A2	106 (70%)	48.6	27.9	15.8
A3	61.9 (90%)	23.7	20	5.3
A4	72.8	20.9	10.1	2.4
B4	*(0%)	519 (10%)	450 (20%)	533 (70%)

Table 5.1: Mean execution time of GLS (in seconds) and percentage of solved instances

λ:	1000	3000	5000	7000	9000
A1	58.1	30.4	35.8	33.5	36.3
A2	12.5	13.4	15.3	13	10.8
A3	4	6.8	3.2	5.9	7.5
A4	1	2.5	4.4	6.6	5.9
B4	468.2	734.3	731.7	526.8	616.7

Table 5.2: Mean execution time of GLS (in seconds) and percentage of solved instances

We can see in Table 5.1 that, for small values of λ (say, λ smaller than 1000), GLS does not always reach a feasible solution. On the other hand, the performance of the algorithm does not seem to depend significantly on the choice of λ in the range from 1000 to 9000.

We also performed some experiments where the value of λ was dynamically adapted to the value of the objective functions. But this self-adjusting framework did not yield better results than those obtained with a fixed λ value.

In the following computational experiments, a λ-value of 5000 was used as default-value, since this value led to good results for most of the test instances. In the industrial application, however, the value of λ is a user-parameter which can be changed if it does not provide the expected results, and smaller values of λ are frequently used (see Section 5.3.2).

5.2.3 Performance of *GLS* as a function of its running time

In the previous section, we have shown that *GLS* is able to solve many feasible benchmark instances within $T = 1200$ seconds. In fact, it is interesting to note that, for the number of blocks considered here, the running time of *GLS* on feasible realistic instances (A0–A5) is actually quite short, in the range of 10 to 50 seconds. This is probably due to the fact, already mentioned above, that real instances are likely to be reasonably easy as the assembly hall is not excessively loaded.

To validate these observations, Table 5.3 shows the ability of *GLS* to solve a feasible instance within a given time T, when T is relatively short (which must be the case at the shipyard, where the algorithm is meant to be used frequently; see also Section 5.2.4). For each instance, we report the percentage of executions (out of 10 trials) that successfully found a feasible solution. We can see that *GLS* performs quite well for most instances, except for B4 which is clearly a much harder instance (remember Section 5.2.1).

	$T = 120$	$T = 40$	$T = 20$	$T = 10$	$T = 5$
A0	100%	90%	40%	0%	10%
A1	100%	100%	90%	30%	20%
A2	100%	100%	100%	100%	30%
A3	100%	100%	100%	100%	70%
A4	100%	10%	0%	0%	0%
B1	100%	70%	50%	10%	0%
B2	90%	70%	10%	20%	0%
B3	90%	20%	20%	0%	0%
B4	20%	0%	0%	0%	0%
D1	100%	50%	20%	20%	0%
Mean values	90%	55%	39%	25%	12%

Table 5.3: Percentage of instances solved by *GLS* as a function of running time

5.2.4 Optimization procedures

The procedures *BlockDescent* and *MaxBlocks* described in Section 5.1.2 aim at maximizing the total number of blocks produced. They iteratively generate several sets of blocks X, check whether each set is feasible (using the procedure $GLS(X, T)$), modify it accordingly, and eventually return the best solution found in the process. Of course, we expect a larger time parameter T to provide more certainty about the feasibility or infeasibility of a current

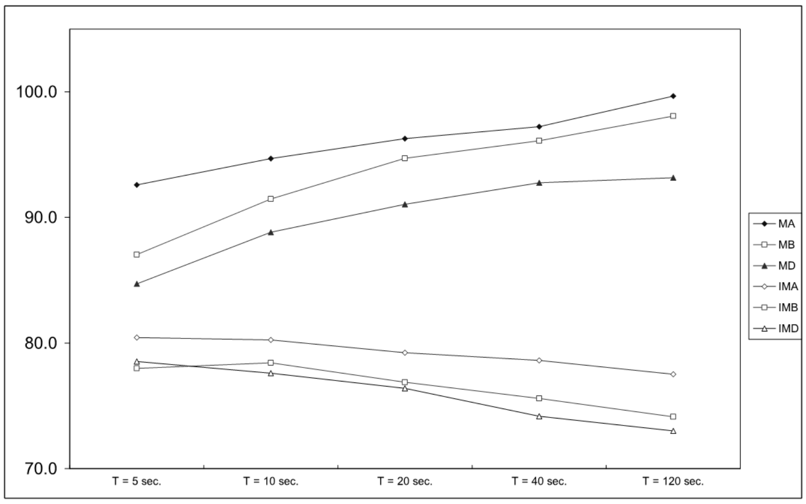


Figure 5.1: Performance of $MaxBlocks(X)$ for different values of T

solution X (cf. hypothesis (HA) in Section 5.1.2). Thus, it is intuitively better to set a rather large value for this parameter. On the other hand, when the iterative calls $GLS(X, T)$ are long, a smaller number of solutions are analyzed within the same total computing time, and this reduces the likelihood to reach the best solutions. Therefore, we have tested the trade-off between these two antagonistic impacts for the procedure $MaxBlocks(X)$, which is the more efficient of the two optimization procedures.

Figure 5.1 displays the results obtained when $MaxBlocks(X)$ is executed with a total time limit of 120 seconds and with various values of T in a range from 5 to 120 seconds. Note that, as a result of these parameter settings, $GLS(X, T)$ can be called from 1 to more than 120 times in any given experiment. For each instance and each value of T, the vertical axis shows the average number of blocks in the best solution found by $MaxBlocks(X)$.

Figure 5.1 shows the mean values over types A, B and D instances. MA is the mean value of the results over the instances A0–A5, MB over B1–B5 and MD over D1–D4 respectibely (see Section 5.2.1). Each individual instance (ex. A0) was solved 10 times for each value of T. The complexity of the individual instances has been further increased by multiplying the width of each block by a constant factor 1.5, thus giving rise to more difficult instances IA0–IA5, IB1–IB5 and ID1–ID4, respectively. The mean results obtained for the latter instances are displayed in Figure 5.1.

We may see that two behaviors appear depending on the complexity of the instances. Instances A to D are low complexity instances, including mostly feasible problems; so, the main challenge in this context is essentially to detect feasibility and, as we already observed in Section 5.2.3, larger values of T produce the best performance. For harder instances (IA, IB, ID), however, the performance of the algorithm slightly increases when T decreases. In this case, each local search phase $GLS(X,T)$ is very short, but its repetitive execution allows to reach good solutions.

The previous observations suggest that running $MaxBlocks$ for a longer time should yield even better results. Accordingly, some experiments have also been conducted with a very large time limit (6000 seconds) and led to an improvement of the best solution by about 4% (i.e., 4 blocks) on some instances. It should be stressed, however, that such long running times are excessive in the industrial context, where the algorithms are usually allowed to run for one or two minutes only.

To conclude, we can say that $GLS(X,T)$ displays a good performance on feasible instances when T is around 120 seconds. On the other hand, for infeasible instances, the optimization procedure $MaxBlocks$ allows GLS to escape local minima. In this case, a trade-off has to be made when setting the parameter T. In practice, a total computing time of 120 seconds with $T = 20$ seconds appears to offer a good compromise.

As explained in Section 5.1.2, the heart of our approach to STA is the GLS algorithm that tests the feasibility of any given set of boxes. A similar approach is typically used by *constraint programming* algorithms, which also rely on iterated solutions of feasibility subproblems; see e.g.Focacci, Laburthe, and Lodi (2003), Jussien and Lhomme (2002), Hentenryck and Michel (2005), Wang and Tsang (1991), etc. Therefore, we decided to compare the performance of our GLS algorithm with that of widely available commercial software for constraint programming, namely the *CP-Optimizer* (CPO) and the Solver packages from ILOG (2007). This comparison also allowed us to better understand the benefit of developing a specialized ad hoc algorithm for the practical problem faced by the shipyard, rather than relying on generic “off-the-shelf" solutions like those provided by $ILOG$.

We have developed a constraint programming model which is derived from the model described in Section 4.2 for STA. We define five variables for each block (area, orientation, coodinates x and y and start time) and the individual constraints and the no-overlap constraints as described in Section 4.2. We included a packing constraint: a global constraint available in CPO, to ensure that the sum of the three-dimensional encumbrance of the blocks (surface * duration) placed in each area is lower or equal to the three-dimensional avaibility of the area (surface available*total time fence of the problem).

$$load(a) \leq availability(a) \tag{5.3}$$

$$load(a) = \sum_{b:area(b)=a} encumbrance(b) \tag{5.4}$$

An issue in the placement of rectangles in rectangular areas is the symmetries inherent to the problem. First, the symmetry on the orientation of the blocs is avoided as the orientation of each block is defined to be either vertical or horizontal (no right-left nor bottom-up symmetry). A second symmetry issue is related to the placement of the blocs in an area, it is reduced by imposing, for each area, that one of the blocks has its bottom-left corner placed in the bottom left quarter of the area. A last issue concerns the symmetry of identical bins. When no temporal aspect is considered, this problem occurs when several bins are empty and thus identical: it is useful to limit the possible affectations of a block to only one of the empty bins. In the STA problem, two bins are identical if they have exactly the same loading on the overall time fence. Practically this happens only when the bins are empty on the overall time window. This issue has been taken into account by restricting the allowed positions of the first blocks to be placed.

We ran *CP Optimizer* with different sets of parameters on the available instances, it worth to notice that among the search strategies available, the "multistart" strategy appeared to provide the best performance (the other strategies not leading to any result). We used the Solver software on the same model, Solver allows to implement dedicated search strategies to use in place of the predefined search strategies present in *CPO*. The comparison between the two approaches relies on the fact that Solver and CPO are reputed to use the same underlying search engines, the difference in the results can thus be imputed to the difference in the search strategies. The dedicated search strategy rely on a firs fail approach for the choice of the variables to instantiate: we fix first the start time and the area of the blocks having the biggest ratio between their encumbrance (surface*duration) and their space and time windows (product of the size of the domains of their variables x, y an t). The start time is chosen to take the minimum value of its domain and the area is chosen to be the less loaded over the duration of the block. In the meanwhile we limit the position of the block in the bottom-left quarter of the area if pertinent. Next the orientation of the blocks are instantiated then their positions along x and then their positions along y, always trying the minimum value first. This strategy does not show better results in quality nor speed than the "multistart" strategy in *CPO* but allows to monitor the search and see that the major computing time is consumed by a systematic local search over the starttime, x and y instantiations. These variables have wide domains (approximatively 10x60x25 values) that are costly to explore: this suggests that the success of the multistart is due to its ability to escape deep explorations of confined regions of the solution space, which is a major characteristic of the *GLS* heuristic. In the following we will focus on the comparison between *CPO* and *GLS*.

We have analyzed both the quality of the results and the computing times of *GLS* and *CPO*. In our experiments, we concentrated mostly on the feasibility version of STA. Note that for any given instance, the *GLS* heuristic can only reach the conclusion that the instance is feasible or that it is unable to find a solution within the allocated time. On the other hand, the *CPO* software can either find a solution, or prove that the problem is infeasible, or reach the time limit without any conclusion.

The computing time of the *GLS* heuristic is random, therefore the results presented in Table 5.4 are averages over 5 executions. On the other hand, the execution time of *CPO* for a given instance is essentially constant. For both algorithms, a limit of 1200 seconds has been set on the computing time.

Table 5.4 displays the results obtained on the set of benchmark instances. An objective value of “1" indicates that a feasible solution has been found (in each run of the algorithm), and a “0" means that no solution has (ever) been found within the time limit. For *CPO*, a value “0*" indicates that *CPO* has been able to *prove* that the corresponding instance is infeasible. A computing time larger than 1200 indicates that the time limit has been reached.

Clearly, more instances are solved by Guided Local Search than by Constraint Programming within a given time limit. In fact, it never happens that *CPO* finds a solution but *GLS* does not. Moreover, whenever both algorithms find a feasible solution, the mean computing time required by *GLS* is always (much) smaller than the time required by *CPO*.

When no feasible solution can be found, either both algorithms reach the time limit without conclusion, or *CPO* proves that the instance is infeasible. The latest case is interesting but unfortunately, it occurs very rarely (2 instances out of 32 in our experiments), and only for instances which are “severely" infeasible.

In conclusion, the *GLS* algorithm clearly outperforms the *CPO* algorithm on our set of benchmark instances. Of course, we cannot exclude that a more sophisticated CP model and/or more advanced settings of the *CPO* software would yield better results. But at the very least, the comparison seems to justify the development of our specialized algorithm for the industrial application.

5.3 Industrial issues

The algorithms described in this chapter have been developed for three different workshops at *Aker Yards France*, where they have been put to daily use for several months. As compared to the “academic" and rather abstract

	Objective		Time	
Instance	GLS	CPO	GLS	CPO
A0	1	0	50	1201
A1	1	1	25	561
A2	1	0	11	1201
A3	1	0	7	1201
A4	1	1	4	119
A5	0	0	1201	1201
B1	1	1	43	568
B2	1	1	39	244
B3	1	0	88	1201
B4	1	0	513	1201
B5	0	0	1201	1201
C1	1	1	1	414
C2	1	0	53	1201
C3	1	0	59	1201
C4	1	0	49	1201
C5	1	0	79	1201
D1	1	0	29	1201
D2	0	0	1201	1201
D3	0	0	1201	1201
D4	0	0	1201	1201
D5	0	1*	1201	1
E1	1	0	1	1201
E2	1	0	284	1201
E3	0	0	1201	1201
E4	0	0	1201	1201
E5	0	1*	1201	1
F0	1	1	1	18
F1	1	1	1	13
F2	1	1	1	11
F3	1	1	1	16
F4	1	0	54	1201
F5	0	0	1201	1201

Table 5.4: Comparison *GLS* vs. *CPO*

description of the problem that we gave in previous sections, tailoring the algorithms to their industrial environment required several adaptations and raised new questions that we now proceed to discuss.

5.3.1 Additional constraints

Various side-constraints have to be considered in order to increase the flexibility of the industrial application. For example, in some cases, it may be necessary to restrict or to impose the position of a block (e.g., because the block is already in process when the software is launched, or because a required tool is only available in a particular area,...). Such individual constraints on blocks are easily handled by local search algorithms: forbidden positions and infeasible neighbors are simply not generated. Thus, in practice, the end-user may fix the value or reduce the domain of any variable (including a b_j-variable; see also Section 5.1.2).

Other side-constraints define some special areas in the assembly halls, to take into account the availability of some specific handling or production equipment. For example some blocks are too heavy to be carried by the cranes, those blocks have to be produced only in predefined areas located next to the border of the hall, so that can be reached by large forklift trucks. In similar way, special areas may be representative of the availability of some production or handling equipments. Among those equipments is a specific crane suited to rotate the block at the end of the production phase, just before leaving the area. Using this specific crane allow to save set-up time compared to the usage of other cranes for the same operation.

More complex collective constraints also appeared in the real-life situation. In particular, for the assembly hall described in the previous section, each working area has a single door, and the crane bridge can only carry the blocks up to a certain height C. As a result, it may happen that a tall block obstructs the door or stands otherwise in the way, and some finished blocks may not be deliverable in time because there is no feasible passageway to carry them out of the hall.

The GLS approach proved "generic" enough to deal with this issue. For each generated solution X, we added to the objective function $h(X)$ a new penalty term which accounts for exit difficulties:

$$\begin{aligned} g(X) &= h(X) + e(X) \\ &= \sum_{i<j} overlap_{ij}(X) + \lambda \sum_{i<j} p_{ij}\, I_{ij}(X) + \sum_{i,j} exit_{ij}(X), \end{aligned}$$

where $exit_{ij}(X)$ measures the overlap between block i and the "exit path" for block j. The exit path for j is restricted by security constraints which impose to use a straight path, and thus it is determined by:

- the longitudinal interval $[x_j, x_j + o_j w_j + (1 - o_j) l_j]$;
- the transversal interval $[0, y_j + (1 - o_j) w_j + o_j l_j]$, as the doors are at position $y = 0$;
- the vertical interval $[C - h_j, C]$, since each block can be carried up to the height of the crane bridge;
- the completion date $s_j + t_j$ of block j;
- the area a_j where block j is produced.

Note that the value of the *exit* terms could somehow be scaled in relation to the $h(X)$ values, but this did not appear to be useful in our procedure, as the new penalty terms proved sufficient to drive the objective function to zero.

Other collective constraints could probably be included in the GLS algorithm using the same flexible approach.

5.3.2 Robustness

The optimization procedures *BlockDescent* or *MaxBlocks* described in Section 5.1.2, just like the guided local search procedure *GLS*, always start from an initial solution X. A drawback of this approach is that the structure of X can confine *GLS* to an area of the solution space that can be difficult to escape (especially for small values of λ), and therefore, the search process may not reach the very best solution.

However, in a dynamic industrial setting, this apparent drawback turns out to be an advantage. Indeed, it may be very costly, or practically impossible for the company to adjust frequently the schedules and the allocation of blocks to the halls. By generating new solutions from previous ones, the *GLS* procedure actually ensures that the structure of previous solutions can be preserved when the production plans are updated. As a consequence, it may prove rewarding to run *GLS* with a relatively small value of λ in the industrial context.

5.3.3 Workforce capacity and requirement

At the level of planning considered in this problem, a precise design of each block has not been completely finished by the engineering department. For this reason, a statistical analysis provides an estimation of the total amount of work required to produce each block. Dividing these values by the numbers of workers planned to be dedicated to each block provides the processing times t_j.

Usually two or three ships may be produced concurrently in the halls. As the production of a ship may sometimes need to be speed-up, one can allow to work during the week-ends. To restrict this cosly usage of extra-hours to the chosen ship, a calendar has been associated to each ship, defining the open hours of the workers for the blocks of the concerned ship.

With these data, analyzing the solutions of the algorithms described above may provide an evaluation of the total workforce requirement for each day. The plots of these amounts over time can then be compared to the workforce that is available. If the capacity is insufficient, the person in charge of the planning can handle it in different ways: he may subcontract some blocks, raise the total workforce, or change the number of workers allocated to the blocks. The last case engenders a change in the processing times and may result in infeasible solutions. Additional $GLS(X, T)$ calls are then required. Note that treating this issue like a constraint would not provide enough flexibility since only the first case would have been taken into account[2]. In spite of this fact, results showed a quite irregular work requirement curve, whereas the amount of workers available should remain smooth.

Focusing on these considerations could become a further step in the automation of shipyard scheduling. The processing times should in this case be equal to the workload requirement divided by a new variable indicating the number of workers dedicated to a block. This variable would probably have a very restricted domain.

5.3.4 Subcontracting blocks

The aim of the methods described in this paper is to maximize the number of blocks produced in a particular assembly hall. When a block can not be produced, we said that it is "subcontracted". Compared to the situation in the factory under study, this formulation is an easy shortcoming. The shipyard is indeed composed by two distinct assembly halls dedicated to the production of building blocks. Additionally, blocks (or parts of blocks) can be processed at the *panel line*, another workshop with completely different processes. Blocks can indeed also be subcontracted, but producing them in an external assembly hall is more costly and thus disliked. Since the different assembly halls are divided in equal-sized rectangular areas[3], the algorithms presented in this chapter could be easily implemented in each of them.

The algorithm is also used, with no adaptation, for halls composed of rectangular areas of various sizes.

An interesting topic for additional researches about shipyard scheduling

[2]One could however consider it as a "collective constraint" and deal with it in the same way as described in section 5 with the exit problems.

[3]External halls have in fact only one single rectangular area, which is just a simplification of the initial formulation.

would be to develop algorithms that dispatch the blocks between the two assembly halls and the panel line. Dealing with the three workshops requires however specific researches about the panel line problematic. Nevertheless, the algorithms presented in this chapter can be easily adapted for the dispatch only between the assembly halls (one could indeed consider the areas of all halls together, which implies only a small revision of the method to cope with areas of different sizes) but such a tool would involve an undesired adjustment of the company's organization.

In the problem definition, the blocks have a rectangular shape and may be rotated as needed. In real life applications, margins are added on each side of the block. These margins represent the free space needed to access and work on the side of the block. They usually differ from side to side. If two blocks are contiguous their margins overlap. In a same manner the working area is surrounded by a free space and the margin relative to the side of a block located at the border of the area overlap with this free space. These considerations break the symmetry in the orientation of the blocks (a 180ř rotation is no longer without consequence and we have to consider four possible orientations instead of two), making the problem harder than when no margins are considered. The model and the algorithm have been adapted with minor changes.

It frequently happens that a series of similar blocks have to be produced for a ship. For example the blocks that include the emergency boats require similar production tools to install the on-board handling equipment. In a similar way, two blocks that are intended to be adjacent when mounted on the ship may need to be adjacent during their production phase, to perform a fine positioning of the elements such as members or piping tracks, that will be connected on the ship. An easy way to cope with the second example is to define a super-block that group the two (ore more) adjacent blocks and to replace the set of concerned blocks with the super-block in the data of the problem. For the first example this method is too restrictive, we preferably added an 'distance' constraint that restrict the relative distance between two blocks. The distance has been defined of three manners (using side-to-side and corner-to-corner measures). The processing dates and time windows are not taken into account by this constraint but in the input data when required (An additional constraint to fix a common processing start date for two adjacent panels may be added if necessary).

5.4 Conclusion

In this chapter, we have presented a real-world space and time allocation problem arising in a large shipyard, and we have modeled this problem as a 3-dimensional bin packing problem. We have demonstrated the practical efficiency and usefulness in this industrial context of the GLS approach proposed by Faroe, Pisinger, and Zachariasen (2003) for the 3D-BPP. This generic approach allows to incorporate various real-life constraints and led to the successful implementation of a flexible and robust application which is nowadays in use at the shipyard.

Part III

Conclusions

We now review the main results presented in the thesis and propose some perspectives for future research.

In this thesis we have proposed new algorithms for discrete optimization with a focus on applications in naval structural design and production management in shipyards. These problems have in common their combinatorial nature and the large size of the real instances.

In the first part of the thesis we consider preliminary structural design of large ships, the selection of raw parts in catalogs allows to achieve substantial savings on the production cost. The methods presented here provide the ship designer with one or more discrete optimal discrete designs among which he can choose taking into account criteria not included in the models.

The methods have been included in the preliminary design analysis and optimization software LBR-5 (Rigo 2001a,b). Two models for structural analysis are available in the software. First a three dimensional model of the structure allows to perform a structural analysis based on the resolution of systems of partial differential equations. Using this model, a structural analysis is time expensive and the optimization process performs only a limited number of analysis in a limited amount of time. Second, a two dimensional model is also available in the software. This mathematical model is based on classification rules established by an international organism of ship certification, and is very fast to compute. Both models have been adapted in the thesis to take discreteness of the variables into consideration, and are presented in chapter 2.

The contribution of chapter 3 is to propose heuristics that provide optimal discrete solutions with a small number of structural analysis and computing time. These heuristics are dedicated to the three dimensional structural analysis model and must deal with the issues of a high computing time for each structural analysis and the implicit formulation of the constraints. We have proposed new heuristics inspired by branch and bound methods and have introduced the use of variable grouping as a key element of the tree search strategy. At each node of the tree search of both heuristic, a nonlinear optimization problem is solved using an approximation scheme. The two heuristics always provide a feasible result under some monotonicity hypothesis. The Dive and Fix heuristic stops as soon as a solution is found. The Branch and Fix heuristic performs a more intensive search in the solution space, and provides a set of feasible discrete solutions at an additional cost of computing time. The two heuristics have been applied on real ship design problems, and a comparison of the results with the continuous optimal solutions have proved the heuristics to be efficient.

In chapter 4, we study the interest of combining a decomposition method and an adaptation of direct search methods to solve the two-dimensional model of structural analysis. We describe a new approach exploring two original ideas which are to apply the discreteness constraints in the subproblems generated by decomposition, and to develop a direct search method to solve the discrete subproblems in such a way that only discrete solutions are visited during the search. The convergence of the method to an optimal solution is discussed, and although the method is not guaranteed to provide a feasible solution, preliminary experiments have shown the viability of the method. Further research will be necessary to validate the method on other instances of discrete structural problems.

In the second part of the thesis we focus on the space allocation and planning decisions for the production of large blocks in shipyards assembly halls. In chapter 4 we define a space and time allocation problem (STA) as a combination of 3D bin-packing and a planning problem. Although this model is inspired by the shipbuilding industry, it is generic enough to be applied to other types of production facilities.

In chapter 5 we adapt a guided local search heuristic developed by Faroe et al. (2003) for 3D-Bin Packing problems. The guided local search and its accompanying Fast Local Search procedure are tested on real instances and on a set of test instances developed on purpose. We compare the Guided Local Search with a constraint programming model and study the influence of the parameters on the performances of the heuristic. This part of the thesis has been published in an article of Annals of Operations Research (Bay et al.) and the software developed is in daily use at STX Shipyards (France).

Part IV

Bibliography

Bibliography

E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines- A Stochastic Approach to Combinatorial Optimization and Neural Computing.* Wiley, 1989.

M. A. Abramson, C. Audet, and J. E. Dennis Jr. Generalized pattern searches with derivative information. *Mathematical Programming, Ser.B*, 100:3–25, 2006a.

Mark Abramson, Charles Audet, and John Jr Dennis. Generalized pattern searches with derivative information. *Mathematical Programming Series B*, 100(1):3–25, 2004.

Mark Abramson, Charles Audet, and John Jr Dennis. Nonlinear programming by mesh adaptive direct searches. siag. *Optimization Views-and-News*, 17(1):2–11, 2006b.

Mark Abramson, Charles Audet, James Chrissis, and Jennifer Walston. Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, 3(1):35–47, 2009.

P. Alberto, F. Nogueira, U. Rocha, and L. N. Vicente. Pattern search methods for user-provided points: application to molecular geometry problems. *SIAM Journal on Optimization*, 14(4):1216–1236, 2004.

N. M Alexandrov and R.M. Lewis. Analytical and computational aspects of collaborative optimization. Technical Report 210104, NASA/TM-2000, 2000.

N. Ali, K. Behdinan, and Z. Fawaz. Applicability and viability of a ga based finite element analysis architecture for structural design optimization. *Computers & Structures*, 81(22–23):2259–2271, 2003.

C. Audet and J.E. Dennis. Mesh adaptative direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006a.

C. Audet and John E. Jr Dennis. A pattern search filter method for nonlinear programming without derivatives. *Siam Journal on Optimization*, 14(4): 980–1010, 2004. ISSN 1052-6234.

C. Audet and John E. Jr Dennis. Mesh adaptive direct search algorithms for constrained optimization. *Siam Journal on Optimization*, 17(1):188–217, 2006b.

C. Audet and S. Le Digabel. Mesh adaptative direct search for periodic variables. Technical report, Gerad and Département de mathématiques et de génie industriel, Ecole Polytechnique de Montréal, may 2009.

J.F.M. Barthelemy and R.T. Haftka. Approximation concepts for optimum structural design -Ů a review. *Structural Optimization*, 5(2–3):129–144, 1993.

Maud Bay, Yves Crama, and Yves Langer. Space and time allocation in a shipyard assembly hall. *Annals of Operations Research.*

Muriel Beckers. *Optimisation des structures en variables discrètes.* Thesis (phd), Université de Liège, Belgium, 1997.

A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary Optimization*, 17:1–13, 1999.

Andrew J. Booker, J. E. Dennis, Paul D. Frank, Douglas W. Moore, and David B. Serafini. Managing surrogates objectives to optimize a helicopter rotor design – further experiments. In *Proceedings of the Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, number AIAA Paper no 98–4717, Reston, VA, USA, 2–4 September 1998 1998. CiteSeerX - Scientific Literature Digital Library and Search Engine (United States).

B. Borchers and J. E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programming. *Computers and Operations Research*, 21(4):359–367, 1994.

R. Brekelmans, L. Driessen, H. Hamers, and D. Den Hertog. Constrained optimization involving expensive function evaluations: A sequential approach. *European Journal of Operational Research*, 160(1):121–138, January 2005.

M. Bremicker, P.Y. Palambros, and H.T. Loh. Solution of mixed-discrete structural optimization problems with a new sequential linearization. *Computers and Structures*, 37(4):451–461, 1990.

L. Brunetta and Ph. Grégoire. A general purpose algorithm for three-dimensional packing. *INFORMS Journal on Computing*, 17(3):328–338, 2005.

M. Bruyneel, P. Duysinx, and C. Fleury. A family of mma approximations for structural optimization. *Structural and Multidisciplinary Optimization*, 24(4):263–276, 2002.

E.G. Coffman, M.R. Garey, and D.S. Johnson. Approximation algorithms for bin packing: A survey. In D.S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, 1997.

E.G. Coffman, G. Galambos, S. Martello, and D. Vigo. Packing approximation algorithms: Combinatorial analysis. In D.-Z. Du and P.M. Pardalos, editors, *Handbook of Combinatorial Optimization.* Kluwer Academic Publishers, 1999.

A.R. Conn, I.M. Gould, and P.L. Toint. *Trust-Region Methods.* SIAM, Philadelphia, 2000.

J. Dauer and W. Stadler. Multicriteria optimization in engineering:a tutorial and survey. In M. P. Kamat, editor, *Structural Optimization: Status and Promise*, pages 209–249. AIAA: Washington, D.C, 1993.

E. Davis and M. Ierapetritou. A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. *Journal of Global Optimization*, 43(2–3):191–205, 2009.

H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 1990.

H. Dyckhoff, G. Scheithauer, and J. Terno. Cutting and packing. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization.* John Wiley & Sons, Chichester, U.K., 1997.

Francis Ysidro Edgeworth. *Mathematical Physics: An Essay on the Application of Mathematics to the Moral Sciences.* M.A., Barrister-at-Law. London: Kegan Paul &Co., 1881.

M. Edvall and K. Holmstrom. An adaptive radial basis algorithm (arbf) for expensive black-box global optimization. *Journal of Global Optimization*, 41(3):447–464, 2008.

O. Exler and K. Schittkowski. A trust region sqp algorithm for mixed-integer nonlinear programming. *Optimization Letters*, 1:269–280, 2007.

O. Faroe, D. Pisinger, and M. Zachariasen. Guided local search for the three-dimensional bin-packing problem. *INFORMS Journal on Computing*, 15 (3):267–283, 2003.

M. Fischetti and A. Lodi. Local branching. *Mathematical Programming, Ser.B*, 98:23–47, 2003.

R. Fletcher and S. Leyffer. Non linear programming without a penalty function. *Mathematical Programming Series*, 91:239–269, 2002.

Claude Fleury. Conlin: an efficient dual optimizer based on convex approximation concepts. *Structural Optimization*, 1(2):81–89, 1989a.

Claude Fleury. First and second order convex approximation strategies in structural optimization. *Structural Optimization*, 1(1):3–10, 1989b.

Claude Fleury and Vincent Braibant. Structural optimization: a new dual approach using mixed variables. *International Journal for Numerical Methods in Engineering*, 23:409–428, 1986.

F. Focacci, F. Laburthe, and A. Lodi. Local search and constraint programming. In F. Kluwer and G. Kochenberger, editors, *Hand-book of Metaheuristics. International Series in Operations Research & Management Science*, pages 369–403. Kluwer Academic, Dordrecht, 2003.

Michael C. Fu, Fred W. Glover, and Jay April. Simulation optimization: a review, new developments, and applications. In *WSC '05: Proceedings of the 37th conference on Winter simulation*, pages 83–95. Winter Simulation Conference, 2005. ISBN 0-7803-9519-0.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, New York, 1979.

F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

I. E. Grossman and Z. Kravanja. Mixed-integer nonlinear programming techniques for process system engineering. *Computers and Chemical Engineering*, 19(suppl.):189–204, 1995.

S. D. Guikema, R. A. Davidson, and Z. Çağnan. Efficient simulation-based discrete optimization. In R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, editors, *Proceedings of the 2004 Winter Simulation Conference*, volume 1, pages 536–544, Piscataway, NJ, 5–8 December 2004 2004. IEEE.

Pierre Hansen and Nenad Mladenovič. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3): 449–467, 2001.

T. Hemker, K. R. Fowler, M. W. Farthing, and O. Von Stryk. A mixed-integer simulation-based optimization approach with surrogate functions in water resources management. *Optimization and Engineering*, 9:341–360, 2008.

P. Van Hentenryck and L. Michel. *Constraint-Based Local Search*. The MIT Press, Cambridge Massachusets, 2005.

O. F. Hughes. *Ship Structural Design*. Society of naval architects and marine engineers (SNAME), Jersey City, New Jersey, 1988.

W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14:331–355, 1999.

ILOG. *CP Optimizer User' Manual and Reference Manual*. ILOG, S.A., 2007.

S. Imahori, M. Yagiura, and T. Ibaraki. Local search algorithms for the rectangle packing problem with general spatial costs. *Mathematical Programming*, 97:543–569, 2003.

S. Imahori, M. Yagiura, and T. Ibaraki. Improved local search algorithms for the rectangle packing problem with general spatial costs. *European Journal of Operational Research*, 167:48–67, 2005.

W. M. Jenkins. On the application of natural algorithms to structural design optimization. *Engineering Structures*, 19(4):302–308, 1997.

D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

Narendra Jussien and Olivier Lhomme. Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence*, 139(1):21–45, 2002. ISSN 0004-3702.

Th. Von Karman. *Les méthodes mathématiques de l'ingénieur : introduction au traitement mathématique des problèmes posés à l'ingénieur*. Béranger, 1949.

Andy J. Keane and Prasanth B. Nair. *Computational Approaches for Aerospace Design. The pursuit of Excellence*. John Wiley Sons, Ltd, 2005.

U. Kirsch, M. Reiss, and U. Shamir. Optimum design by partitioning into substructures. *Structural of Structural Division ASCE*, 98(ST1):249, 1972.

T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

T. Kumano, S. Jeong, S. Obayashi, Y. Ito, K. Hatanaka, and H. Morino. Multidisciplinary design optimization of wing shape for a small jet aircraft using kriging model. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, number Paper 2006-0932, 2006.

R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.

Robert M. Lewis and Torczon Virginia. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, 10(3):917–941, 2000.

Robert M. Lewis, Virginia Torczon, and M. W. Trosset. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, 124(1-2):191–207, 2000.

Robert Michael Lewis and Virginia Torczon. A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12:1075–1089, 2002.

R. T. Marley and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural multidisciplinary optimization*, 26:369–395, 2004.

S. Martello, D. Pisinger, and D. Vigo. The three dimensional bin packing problem. *Operations Research*, 48(2):256–267, 2000.

R. H. Myers and D. C. Montgomery. *Response Surface Methodology.* John Wiley & Sons Inc., New York, 2002.

Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.

G. Olsen and G. Vanderplaats. A method for non-linear optimization with discrete variables. *AIAA Journal*, 27(11):1584–1589, 1989.

Y. S. Ong, P. B. Nair, A. J. Keane, and K.W. Wong. *Knowledge Incorporation in Evolutionary Computing*, chapter Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. J. Yaochu (ed.) Springer Berlin, Heidelberg, 2004.

Vilfredo Pareto. *Manuale d'economia politica.* Milano, 1906.

W. J. Pirson. *A Unified Mathematical Theory for the Analysis of Propagation and Refraction of Storm Generated Ocean-surface Waves, Parts I and II.* New York University, 1952.

W. J. Pirson, G. Neumann, and R. W. James. *Practical Methods for Observing and forecasting Ocean Waves by Means of Wave Spectra and Statistics.* Hydrographic Office Publication n603, 1955.

Yves Pochet and Laurence Wolsey. *Production Planning by Mixed Integer Programming.* Springer, Heidelberg, 2006.

R. G. Regis and C. A. Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global Optimization*, 31(1):153–171, 2005.

T. Richir, N. Losseau, E. Pircalabu, C. Toderan, and P. Rigo. Least cost optimization of large passenger vessels. In *Proceedings of MARSTRUCT Conference*, 2007a.

Thomas Richir, Eugene Picalabru, and Philippe Rigo. Projet r&d#13, logiciel lbr-5, aker yards (ingénierie coque). Technical report, ANAST - Université de Liège, September 2006.

Thomas Richir, J.-D. Caprace, N. Losseau, M. Bay, M.G. Parsons, S. Patay, and P. Rigo. Multicriterion scantling optimization of the midship section of a passenger vessel considering iacs requierments. In *Proceedings of the 10th International Symposium on Practical Design of Ships and Other Floating Structures (PRADS)*, September 2007b.

Philippe Rigo. *Développement d'un modèle intégré d'optimisation des structures navales et hydrauliques.* Thèse d'agrégation. Université de Liège, 1998.

Philippe Rigo. A module oriented tool for optimum design of stiffened structures – part i. *Marine Structures*, 14:611–629, 2001a.

Philippe Rigo. Least cost structural optimization oriented preliminary design. *Journal of Ship Production*, 17(4):202–215, November 2001b.

Philippe Rigo. *LBR-5 user guide, version 5.6d.* Université de Liège, 2002.

Philippe Rigo. Differential equations of stiffened panels of ship structures & fourier series expansions. *Ship Technology Research*, 52:82–100, 2005.

Philippe Rigo and Claude Fleury. Scantling optimization based on convex linearizations and dual approach – part ii. *Marine Structures*, 14:631–649, 2001.

L.A. Schmit and R.K. Ramanathan. A multilevel approach to minimum weight design including buckling constraints. *AIAA Journal*, 16(2):97–104, 1973.

T. W. Simpson, T. M. Mauery, J. J. Korte, and F. Mistree. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA Journal*, 39(12):2233–2241, 2005.

J. Sobieszczanski. Two alternative ways for solving the coordination problem in multilevel optimization. *Structural Optimization*, 6:205–215, 1993.

W. Stadler. A survey of multicriteria optimization or the vector maximum problem, part i: 1776–1960. *Journal of Optimization Theory and Applications*, 29(1):1–52, 1979.

W. Stadler. Initiators of multicriteria optimization. In J.Jahn and W.Krabs, editors, *Recent Advances and Historical Development of Vector Optimization*, number 294 in Lecture Notes in Economics and Mathematical Systems. Springer Verlag, Berlin, 1987.

K. Svanberg. The method of moving asymptotes - a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 1987.

P. Thanedar and G. Vanderplaats. Survey of discrete variable optimization for structural design. *Journal of Structural Engineering*, 121(2):301–306, 1995.

G. Tompkins and F. Azadivar. Genetic algorithms in optimizing simulated systems. In Christos Alexopoulos and Keebom Kang, editors, *Proceedings of the 1995 Winter Simulation Conference*, pages 757–762, Washington, DC, 3–6 December 1995 1995. IEEE Computer Society.

Virginia Torczon. On the convergence of pattern search algorithms. *SIAM Journal of Optimization*, 7:1–25, 1997.

Virginia Torczon and Michael W. Trosset. Using approximations to accelerate engineering design optimization. In *Proceedings of the Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, number Paper no 98–4800, Reston, VA, USA, 2–4 September 1998. CiteSeerX - Scientific Literature Digital Library and Search Engine [http://citeseerx.ist.psu.edu/oai2] (United States).

N. Turkkan. Discrete optimization of structures using a floating-point genetic algorithm. In D.T. Kashiwagi and J. Savicky, editors, *Annual Conference of the Canadian Society for Civil Engineering*, 4–7 June 2003.

C. Voudouris. *Guided local search for combinatorial optimization problems.* PhD thesis, Department of Computer Science, University of Essex, Colchester, United Kingdom, 1997.

C. Voudouris and E. Tsang. Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Operations Research Letters*, 20:119–127, 1997.

C. Voudouris and E. Tsang. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469–499, 1999.

C.J. Wang and E. Tsang. Solving constraint satisfaction problems using neural-networks. In *Proceedings of IEE Second International Conference on Artificial Neural Networks*, pages 295–299, 1991.

M. Yoshimura, S. Nishiwaki, and K. Izui. A multiple cross-sectional shape optimization method for automotive body frames. *Journal of Mechanical Design*, 127(1):49–58, 2005.

W. H. Zhang and Claude Fleury. A modification of convex approximation methods for structural optimization. *Computers & Structures*, 64(1–4): 89–95, 1997.

Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man and Cybernetics(SMC), part C*, 37 (1):66–76, 2007.

Part V

Appendix

Appendix A

Three-dimensional model

A.0.1 Structural Constraints

This section gives a detailed description of the constraints available in LBR5, the software developed at ANAST department of the University of Liège. The modeling of a ship structure implies to select a coherent subset of these constraints. A strucural analysis requires to define one or several load cases and for each of them to computes of the constraints at predefined points on each panel. A detailed description of the model is available in Rigo (1998, 2002)

Absolute displacements:
$C(XI) \leq CJMAX with CJMAX > 0$
or
$C(XI) \geq CJMAX with CJMAX < 0$

n° 1 V absolute displacement along OX (in m),
n° 2 U absolute displacement along OY (in m),
n° 3 W absolute displacement along OZ (in m),

Minimum plate thickness (buckling)
$C(XI) \leq CJMAX$
with $C(XI) = (d_{min} - d)$ and $CJMAX = 0$
Minimal net thickness of plates is determined according to Bureau Veritas rules (BV, Pt. B, Ch. 7, Sec. 1, Table 2) and depends on the location of the plate in the structure : keel plate, bottom, inner bottom, side shell, lower deck,etc. The considered plate is the unstiffened plate included between two frames and two longitudinals.The reglementary length of the ship (L), the material factor (k) and the width of the elementary plate panel (EPP) influence the minimal tickness value for each panel.

n° 4 Minimum Thickness: $d_{min} - d < 0$

with $CJMAX = 0$, d the current plate thickness and d_min the minimum thickness to avoid plate yielding and buckling. We consider for yielding the primary stresses, local bending of longitudinals and local plate bending. For buckling, the elements considered are the primary stresses (in particular, local bending of plate and longitudinals are neglected)

Relative displacements of the global structure
$C(XI) \leq CJMAX$
$C(XI)$ is the deflection and $CJMAX$ the allowable displacement in (m), with $CJMAX > 0$ or $CJMAX < 0$

n° 5 $Wrelative(Y) = Wabsolute(Y) - -Wabsolute(Y = 0)$
n° 6 $Wrelative(Y) = Wabsolute(Y) - -Wabsolute(Y = Yo)$
n° 7 $Wrelative(Y) = Wabsolute(Y) - -[Wabsolute(Y = 0) + Wabsolute(Y = Yo)]/2$

Plate yielding (von-Mises)
$C(XI) \leq CJMAX$
with $CJMAX > 0$, the allowable stress in (N/m2)

n° 11 Sigma von-Mises (Sx, Sy) plate (Z=0, mid-plate thickness) in $x = L/2$
(Primary stresses only)
n° 12 Sigma comp. (Sx, Sy) plating (Z=0) in $x = L/2$
(Primary stresses + local bending of longitudinal)
n° 13 Sigma comp. (Sx, Sy) plating (Z=0) in $x = L/2$
(Primary stresses + local bending of longitudinal + local plate bending)
n° 16 Sigma von-Mises (Txy) plate (Z=0, mid-plate thickness) in $x = 0$
(Primary stresses only)
n° 17 Sigma von-Mises (Sx, Sy) plate (Z=+d/2, upper face) in $x = L/2$
(Primary stresses only)
n° 18 Sigma von-Mises (Sx, Sy) plate (Z=-d/2, lower face) in $x = L/2$
(Primary stresses only)

Utimate strength of stiffened panels.

n° 15 $N_x/SN_{ult} \leq CJMAX$ with $0 < CJMAX \leq 1$
with N_x the current average axial load (N); N_{ult} the ultimate strength (Paik).

This constraint considers the collapse mode of a stiffened panel submitted to axial forces (at $X = L/2$).

Frames yielding (von Mises)

$C(XI) \leq CJMAX$
with $CJMAX > 0$, the allowable stress in (N/m2)

n° 21 Sigma von-Mises (Tweb) frame, at JWF/JWP at $X = 0$
n° 22 Sigma von-Mises (Sy, Txy) frame, at JWF at $X = L/2$
n° 24 Sigma von-Mises (Sy, Txy) frame, at JWP at $X = L/2$
n° 25 Sigma von-Mises (Sy, Txy) frame, in the Flange at $X = L/2$
using the notation : Web (W), Flange (F), Plate (P), Junction Web-Flange (JWF), Junction Web-Flange (JWP).

<u>Stiffener - longitudinals yielding (von Mises)</u>
$C(XI) \leq CJMAX$
with $CJMAX > 0$, the allowable stress in (N/m2)

Induced by primary bending moment:

n° 31 Sigma von Mises (Tweb) stiffener, at JWP at $X = 0$
n° 32 Sigma von Mises (Sx) stiffener, in the flange at $X = L/2$

Induced by primary bending moment and local bending of the longitudinals (stiffeners)

n° 33 Sigma comp. (Tweb) longitudinal JWP at $X = 0$
n° 34 Sigma comp. (Sx) longitudinal's flange at $X = L/2$
n° 35 Sigma comp. (Sx,Tweb) longitudinal JWP at $X = L/2$
n° 36 Sigma comp. (Sx,Tweb) longitudinal JWF at $X = L/2$

Induced by local bending (lateral pressure – tertiary effect)

Here is considered the bending of the stiffener and its attached plate between 2 frames. The load considered is MAX[XI, XF, Ploc,CHA1, CHA2, CHA3].

n° 51 Stiffener deflection with 2 clamped ends:
$F/l = 1.0 * Pl3/384EI \leq CJMAX(= 1/100, ...1/1000)$ with "l" the span of the stiffener (equals to the frame spacing),

n° 52 Stiffener deflection with 2 simply supported ends:
$F/l = 5.0 * Pl3/384EI \leq CJMAX(= 1/100, ...1/1000)$

n° 54 Maximum stress in the stiffener's flange:
$(M = Pl^2/1); Sig \leq S(allowed) = CJMAX(> 0)$ in (N/m2)

n° 55 Maximum stress in the stiffener attached plate:

$(M = Pl^2/10); Sig \leq CJMAX$,

n° 56 Maximum shear stress (Tau) in the stiffener's flange:
$\sqrt{3} * Tau \leq CJMAX$,

n° 57 Maximum unstiffened plate deflection (Clamped edges):
$W = 1.0 * PL4/384EI \leq CJMAX = W(allowed)$ (in m),
with "L" the stiffener spacing

n° 58 Maximum unstiffened plate deflection (Simply supported Edges).
$W = 5.0 * PL4/384EI \leq W(allowed)$ (in m),

Girder yielding (von-mises)
$C(XI) \leq CJMAX$
with $CJMAX > 0$, (N/m2)

n° 41 Sigma von-Mises. (Tweb) girder, at JWF in $X = 0$
n° 42 Sigma von-Mises (Tweb) girder, at JWP in $X = 0$
n° 43 Sigma von-Mises (Sx) girder, in the flange in $X = L/2$

A.0.2 Geometrical Constraints:

We use the following notations : d = Plate thickness, Tw = Web Thickness, Tf = Flange Thickness, Dw = Web height and Df = Flange width.

for the frames (transverse members)
"Plate Thickness /Web Thickness " Ratio

n° 104 $d - 2Tw \leq 0$
n° 112 $Tw - -2d \leq 0$

"Minimal web height" (minimal frame stiffness)

n° 105 $3d - Dw \leq 0$

"Flange width/web height" Ratio

n° 101 $Df - Dw \leq 0$
n° 102 $Dw - 2Df \leq 0$ (Rahman's set)

For T shape members (O. Hughes' set)

n° 109 $0.625Df - Dw \leq 0$
n° 110 $Dw - 2.5Df \leq 0$

For L shape members (O. Hughes' set)

n° 115 $1.25Df - Dw \leq 0$
n° 116 $Dw - -5Df \leq 0$

"Web slenderness"
It is assumed that the frames are taller that the stiffeners.

n° 111 $Dw - -120Tw \leq 0$ (for web supported by smaller stiffeners).

If the stiffener/longitudinal height is taller than the frame height.

n° 113 $Dw - 36Tw \leq 0$ or
n° 114 $Dw - 40Tw \leq 0$

<u>For the stiffeners (longitudinal members)</u>
"Plate Thickness /Web Thickness " Ratio

n° 204 $d - 2Tw \leq 0$ and
n° 212 $Tw - -2d \leq 0$

"Minimal web height" (minimal frame stiffness)

n° 205 $3d - Dw \leq 0$

"Flange width/web height" Ratio

n° 201 $Df - Dw \leq 0$ and
n° 202 $Dw - 2Df \leq 0$ (Rahman's set)

For T shape members (O. Hughes' set)

n° 209 $0.625Df - Dw \leq 0$ and
n° 210 $Dw - 2.5Df \leq 0$

For L shape members (O. Hughes' set)

n° 215 $1.25Df - Dw \leq 0$
and
n° 216 $Dw - -5Df \leq 0$

"Web slenderness"
It is assumed that the frames are taller that the stiffeners (the stiffener web is not supported).

n° 203 $Dw - -40Tw \leq 0$ (Rahman). or
n° 211 $Dw - 36Tw \leq 0$ (Hughes).

If the stiffener/longitudinal height is taller than the frame height.

n° 213 $Dw - 120Tw \leq 0$
(Ref. Hughes. The stiffener web is supported).

<u>Interaction frames-stiffeners</u>

Compatibility "frames height" versus "stiffener height" (O. Hughes)

n° 301 $Dw(raid) - Dw(aig) \leq 0$ (standard case: frames taller than longitudinals)
n° 304 $Dw(aig) - Dw(raid) \leq 0$ (if longitudinals have to be taller that the frames)

Compatibility "web thickness of frame" versus " web thickness of stiffener" (O. Hughes)

n° 302 $Tw(raid) - 4 * Tw(aig) \leq 0$
n° 303 $Tw(aig) - 4 * Tw(raid) \leq 0$

Appendix B

Differential equations of stiffened panels

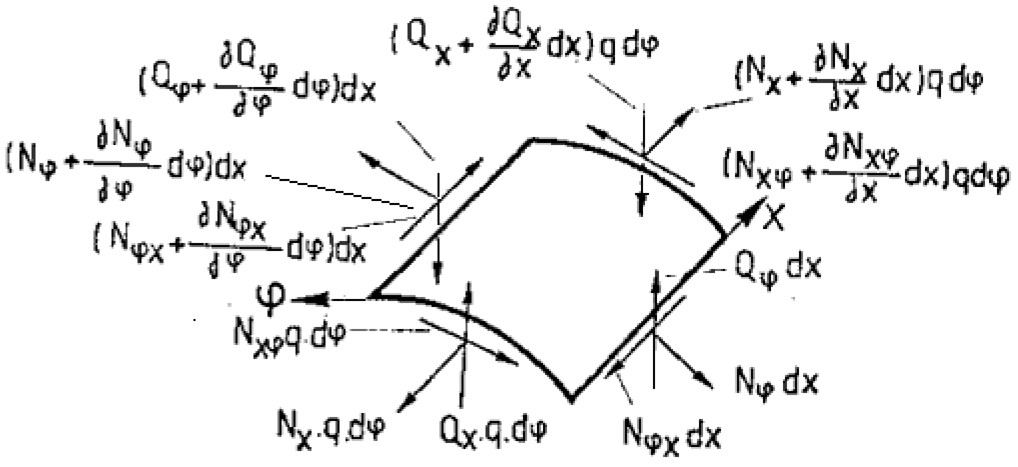
B.0.3 Differential equations of stiffened panels

This Appendix gives a insight on the analytic resolution method authored by Ph. Rigo(Rigo (1998) to solve systems of differential equations developed by Donnell, Von Karman and Jenkins (Karman (1949)). The method is based on harmonic analysis and Fourier series expansion.

For each panel, one has to compute the deformations (ϵ, γ), displacements $(u,\ v,\ w)$ and the stress (σ, τ). We consider the following hypothesis:

- thin shell theory applied to an elementary plate with a surface dimension dx.dy and a thickness of δ along z axis.
- small deformation and linear analysis
- the points located on a perpendicular axis to the mid plate surface before deformation remain on the same perpendicular axis after deformation
- the constraint σ_z perpendicular to the surface and its effects are negligible
- $\epsilon_z = 0$ there is no deformation along z

We note f' the partial derivative of f along the x direction and f° the partial derivative of f along y direction: $f' = \frac{\partial f}{\partial x}$ and $f^\circ = \frac{\partial f}{\partial y} = \frac{1}{q}\frac{\partial f}{\partial \varphi}$ In the following we do not distinguish φ and y to denote the y direction.
The linear relationships linking deformation (ϵ, γ) to displacement $(u,\ v,\ w)$



$(Q_\varphi + \frac{\partial Q_\varphi}{\partial \varphi} d\varphi)dx$
$(Q_x + \frac{\partial Q_x}{\partial x} dx)q\,d\varphi$
$(N_x + \frac{\partial N_x}{\partial x} dx)q\,d\varphi$
$(N_\varphi + \frac{\partial N_\varphi}{\partial \varphi} d\varphi)dx$
$(N_{x\varphi} + \frac{\partial N_{x\varphi}}{\partial x} dx)q\,d\varphi$
$(N_{\varphi x} + \frac{\partial N_{\varphi x}}{\partial \varphi} d\varphi)dx$
X
$Q_\varphi\, dx$
φ
$N_{x\varphi} q\, d\varphi$
$N_\varphi\, dx$
$N_x . q . d\varphi$
$Q_x . q . d\varphi$
$N_{\varphi x}\, dx$

Figure B.1: Plate yielding (von-Mises)

are for a plate:

$$\begin{aligned} \epsilon_x &= u' - zw'' \\ \epsilon_\varphi &= v^\circ - zw^{\circ\circ} \\ \gamma_{x\varphi} &= u^\circ + v' - 2\,zw^{\circ\prime} \end{aligned} \tag{B.1}$$

and the relationship between stress (σ, τ) and displacement $(u,\ v,\ w)$ relationships are :

$$\begin{aligned} \sigma_x &= \frac{E}{1-\nu^2}\left[u' + \nu(v^\circ - z(w'' + \nu w^{\circ\circ})\right] \\ \sigma_\varphi &= \frac{E}{1-\nu^2}\left[(v^\circ + \nu u' - z(w^{\circ\circ} + \nu w'')\right] \\ \tau_{x\varphi} &= G(u^\circ + v' - 2zw^{\circ\prime}) \end{aligned} \tag{B.2}$$

with E the Young Modulus, ν the Poisson coefficient and G the shear modulus $G = E/2(1+\nu)$.

The resultant forces within the plate are noted N_x, Q_x, $N_{x\varphi}$,N_φ, Q_φ, $N_{\varphi\, x}$ and the resultant moments: M_x, $M_{x\varphi}$,M_φ, $M_{\varphi\, x}$, as showed on Figure B.0.3. For a stiffened thin shell element, the relationship between stress (σ, τ) and resultant forces (N_x, Q_x, $N_{x\varphi}$,N_φ, Q_φ, $N_{\varphi\, x}$), considering the plate and stiffener components are:

$$
\begin{aligned}
N_\varphi &= \int_{-\delta/2}^{+\delta/2} \sigma_\varphi dz + f(x) \int_{+\omega_\varphi} \sigma_\varphi \frac{e_\varphi}{d_{phi}} dz \\
N_x &= \int_{-\delta/2}^{+\delta/2} \sigma_x (1 + \frac{z}{q}) dz + f(\varphi) \int_{+\omega_x} \sigma_x \frac{e_x x}{d_x} dz \\
M_\varphi &= \int_{-\delta/2}^{+\delta/2} \sigma_\varphi \, z \, dz + f(x) \int_{+\omega_\varphi} \sigma_\varphi \frac{e_\varphi}{d_p hi} dz \\
M_x &= \int_{-\delta/2}^{+\delta/2} \sigma_x \, z \, (1 + \frac{z}{q}) dz + f(\varphi) \int_{+\omega_x} \sigma_x \frac{e_x}{d_x} dz
\end{aligned}
\tag{B.3}
$$

with ω_φ and ω_x the cross sections of, respectively, the frames and the stiffeners.

Using the "stress-displacement" relationships (B.2) and the "resultant-stress" relationships (B.3) we obtain the "resultant-displacement " equations for stiffened plates, including stiffeners and frames contributions:

$$
\begin{aligned}
N_\varphi &= D(v^\circ + \frac{w}{q} + \nu u\prime) + f(x)\frac{E}{d\varphi}((v^\circ + \frac{w}{q}).\omega_\varphi - w^{\circ\circ} h_\varphi) \\
M_\varphi &= K(w^{\circ\circ} + \nu w\prime\prime) + f(x)\frac{E}{d\varphi}((v^\circ + \frac{w}{q}).h_\varphi - w^{\circ\circ} I_\varphi) \\
M_{\varphi\, x} &= K(1-\nu) w^{\circ\prime} + f(x)\frac{G}{d\varphi}(K_\varphi \, w^{\circ\circ} I_\varphi) \\
N_x &= D(u' + \nu v^\circ) + f(\varphi)\frac{E}{dx}(u' \, \omega_x - w'' h_x) \\
M_x &= K(w'' + \nu w^{\circ\circ}) + f(\varphi)\frac{E}{dx}(u' \, h_x) \\
M_{x\varphi} &= K(1-\nu) w^{\circ\prime} + f(\varphi)\frac{G}{d_x}(K_x \, w^{\circ\prime}) \\
N_{\varphi\, x} &= D(\frac{(1-\nu)}{2}).(v' + u^\circ) \\
N_{x\,\varphi} &= N_{\varphi\, x}
\end{aligned}
\tag{B.4}
$$

We consider the stiffened plate element submitted to external loads X, Y and Z acting respectively along the x, y and z axis. the element is in an equilibrium state described by the following equations:

$$\begin{aligned}
N'_x + N^{\circ}_{\varphi\,x} + X &= 0\\
N^{\circ}_{\varphi} + N'_{x\,\varphi} + Y &= 0\\
Q^{\circ}_{\varphi} + Q'_x - Z &= 0\\
M^{\circ}_{\varphi} + M'_{x\varphi} - Q_{\varphi} &= 0\\
M'_x + M^{\circ}_{\varphi\,x} - Q_x &= 0\\
N_{x\varphi} - N_{\varphi x} &= 0
\end{aligned} \tag{B.5}$$

The problem figures 13 unknowns: the displacements u, v, w and the resultant forces N_x, Q_x, $N_{x\varphi}$,N_{φ}, Q_{φ}, $N_{\varphi\,x}$; and there are 13 available equations (B.5, and B.4) Replacing the "resultant-displacement " relationships (B.4) in the equilibrium equations above, one obtains a system of three differential equations in (u, v, and w).

$$(D + \Omega_x)u'' + D(\frac{1-\nu}{2})u^{\circ\circ} + D(\frac{1+\nu}{2})v^{\circ\prime} - H_x w''' = -X \tag{B.6}$$

$$D(\frac{1+\nu}{2})u^{\circ\prime} + (D + \Omega_y)v^{\circ\circ} + D(\frac{1-\nu}{2})v'' - H_y w^{\circ\circ\circ} = -Y \tag{B.7}$$

$$-H_x u''' - H_y v\circ\circ\circ + (K + R_y)w^{\circ\circ\circ\circ} + \tag{B.8}$$

$$(2K + T_y + T_x)w^{\circ\circ\prime\prime} + (K + R_x)w'' = Z \tag{B.9}$$

with D, K related to the plate, Ω_x, R_x, H_x, T_x related to the stiffeners and Ω_y, R_y, H_y, T_y related to the frames.
The analytic method proposed by Rigo (1998) first solve the homogeneous system which requires the resolution of the 8th order differential equation

$$Aw'''''''' + Bw'''''' + Cw''''''^{\circ\circ} + Dw'''' + Ew''''^{\circ\circ} + ... \tag{B.10}$$

$$+Jw''^{\circ\circ\circ\circ\circ\circ} + Kw^{\circ\circ\circ\circ\circ\circ\circ\circ} = 0 \tag{B.11}$$

The solution is approximated as a Fourier series expansion in the variable y. the constants resulting of the integration are determined using the basic unitary load lines applied at the extremities $y = 0$ and $y = b$. The solution obtained for the basic load lines are then integrated and the constants are determined according to the external loads distribution (resolution of the non homogeneous system considering X, Y, and Z). A final transformation is then applied to adapt the results to panels with a finite width.

Appendix C

Two dimensional model

C.1 Mathematical formulation

C.1.1 Variables

For each element, five design variables are available: The five design variables chosen to be optimized on each panel are: the thickness of the panel and the dimensions and positioning of the longitudinal members (stiffeners, crossbars, longitudinals, girders, etc.):

1. $\delta_p$is the thickness of plate p
2. h_x^p web height
3. d_x^p web thickness
4. w^p flange width
5. Δ_x^p spacing between two longitudinal stiffeners.

C.1.2 Objective function

The objective function to be minimize may either be the weight of the structure, its production cost, its moment of inertia or a combination of these three objectives.

1. Weight:

$$F_W = \quad \gamma \, \mathrm{L} \sum_p \, \mathrm{B^p} \left\{ \delta^p \, + \, \frac{h_x^p d_x^p + w_x^p \mathrm{t_x^p}}{\Delta_x^p} \right\}$$

2. Cost:

$$F_C = \sum_p F_c^p$$

$$F_c^p = \mathrm{a}_1\,\delta + \mathrm{a}_2\,\delta + \mathrm{a}_3 + \mathrm{b}_1\,\frac{d_x^2\,h_x}{\Delta_x} + \mathrm{b}_1\,\frac{d_x w_x \mathrm{t_x}}{\Delta_x}$$
$$+\,\mathrm{b}_2\,\frac{d_x\,h_x}{\Delta_x} + \mathrm{b}_2\,\frac{w_x \mathrm{t_x}}{\Delta_x} + \mathrm{b}_3\,\frac{d_x}{\Delta_x} + \mathrm{b}_4\,\frac{1}{\Delta_x}$$

3. Moment of Inertia:

$$I_{xx} = \sum_p m^p r^p$$

with: $m_p = \gamma\ \mathrm{L}\ \mathrm{B^p}\left\{\delta^p + \frac{h_x^p d_x^p + w_x^p t_x^p}{\Delta_x^p} + \frac{h_y^p d_y^p + w_y^p t_y{}^p}{\Delta_y^p}\right\}$
r_p the distance from the axis of the panel p to the neutral axis of the structure.

C.1.3 Constraints

We present the generic formulation of each constraint. The constraint set of any application model is composed of subsets of these generic constraints, applied in a predefined set of points of some chosen panels.

C.1.3.1 Equality constraints

Examples: equality of the thickness of adjacent plates:
$\delta_1 = \delta_2$ quality of the longitudinal stiffener spacing of two stiffened plate elements which are vertically aligned:
$\Delta_y^3 = \Delta_y^4$

C.1.3.2 Technological constraints

These constraints are bounds limits on the design variables. The lower bounds are usually determined by technical limitations (for example a lower bound for a thickness variable to limit the impact of the corrosion phenomenon) and the upper bounds are usually set by to production requirements (for example handling capabilities). $\delta_{min} \leq \delta \leq \delta_{max}$

C.1.3.3 Geometrical constraints

These constraints limit the values of some ratios between the design variables to ensure that the structure is functional, feasible and reliable. They originate from reglementations and norms. An example is to fix a maximum ratio between the dimensions and the thickness of members (frames, stiffeners) or link the dimensions of two distinct types of members of a panel (frame height, stiffener height). An example is $\delta^p - 2\, d^p_x < 0$.

C.1.3.4 Global constraints

As external loads and forces are applied to the structure, some resultant effects such as displacements, deformations and internal stress occur. The structural constraints define the maximum admissible values of these resultant effects in order to limit the apparition of physical phenomenon such as yielding, buckling, ruin, etc.
(#1) Gravity center shift: it is desirable to maintain the vertical position of the gravity center within predefined values. A low gravity center implies that a small admissible cargo can be loaded on a ship. A high gravity center implies instability when the ship operates empty or with a light cargo.

$$GC > GC_{min}$$
$$GC < GC_{max}$$

The center of mass R of a system of objects p is defined as the average of their positions r_p (relative to a reference axis), weighted by their masses m_p defined as above. Neutral axis is by definition the horizontal axis passing through GC.

(#2) hull girder moment of inertia: for a rigid body consisting of N panels of masses m_p with r_p the distance from their own axis to the neutral axis, the total moment of inertia equals the sum of the moments of inertia of the panels

$$I_{xx} > I_{xx\,min}$$
$$I_{xx} = \sum_p m^p r^p$$

with :
m_p the panel weight as previously defined and
r_p the distance from the axis of the panel p to the neutral axis of the structure.

(#3) hull girder minimum section modulus : a minimum section modulus is imposed for the panel located at the maximal distance of the gravity center, the section modulus of a panel p is the global moment of inertia divided by the biggest distance (d) between the panel and the position of the gravity center

$$\frac{I_{xx}}{d} > I_{xx/v\ min}$$

(#4) maximum weight

$$F_W < F_{W\ max}$$

$$F_W = \quad \gamma\ \mathrm{L} \sum_p \mathrm{B}^{\mathrm{p}} \left\{ \delta^p \ + \ \frac{h_x^p d_x^p + w_x^p \mathrm{t}_{\mathrm{x}}^{\mathrm{p}}}{\Delta_x^p} \right\}$$

Structural constraints They are computed in a list of points defined by the user for each panel. The generic mathematical expression of these constraints is:

$\sigma_a \leq \sigma_{amax}$ limitation of the bending stress

$\tau \leq \tau_{max}$ limitation of the shear stress

$u \leq u_{max}$ limitation of the displacement

A structural analysis is performed to compute the value of the stress and the displacement at various point on each plate. The maximal values are calculated using standard formulas.

In this model, we use a set of rules issued by the organisms of certification of naval structures, namely IACS and Bureau Veritas.

(#10) Bending Strength of the structure is induced by the overall moment M_y on each panel:

$\sigma_x \leq \sigma_a$

with:

$\sigma_x = M_y / W_{min}$

M_y the moment along y direction

$W = I_{xx/d}$(formula of Navier)

$\sigma_a \leq 175/k$

σ_a the hull girder bending stress (N/mmš)

k the material factor

(#14) Compression buckling of plates
Elastic compression buckling of plate is computed as follows :

$$\sigma_E = 0.9mE\left(\frac{t_b}{1000s}\right)^2 \text{ (N/mmš)}$$

For a plate with longitudinal members (whose members are parallel to the compression constraints),

$$m = \frac{8.4}{\psi + 1.1}(0 \leq \psi \leq 1)$$

For a plate with transversal members (whose members are perpendicular to the compression constraints),

$$m = c\left[1 + \left(\frac{s}{l}\right)^2\right]^2 \frac{2.1}{\psi + 1.1} \qquad (0 \leq \psi \leq 1)$$

with:
E the elasticity modulus of the material
t_b the thickness of the plate, in mm
s the elementary plate width, in m
l the elementary plate length, in m
$c = 1.05$ to 1.3 according to the stiffening type
ψ the ratio between maximum and minimum compression constraints acting on the panel
(#19) Shear Strength (T_xy) of the structure The critical shear stress τ_c of the structure cannot be smaller than the applied constraint τ_a: $\tau_C \geq \tau_a$
with
$\tau_C = 110/k$ (BV rule)
k = material factor
τ_a = hull girder shear stress (N/mmš)
The method to compute τ_a is presented in the next section of this appendix.

(#20) Shear buckling of plates
The critical shear buckling stress τ_c of plates cannot be smaller than the applied constraint τ_a:
$\tau_C \geq \tau_a$
the method to compute τ_a is presented in the next section of this appendix.

For each plate the local distribution of shear buckling constraints $\tau_e^l(s)$, to be added to the mean value $\overline{\tau}_e$, is given by :

$$\tau_e^l(s) = \frac{Q}{I}\left[y_{1e}\left(s - 0.5b_e\right) + 0.5\sin\chi_e\left(s - \frac{b_e^2}{3}\right)\right]$$

with:
$\tau_C = \tau_E$ if $\tau_E \leq \frac{\tau_F}{2}$
$= \tau_F \left(1 - \frac{\tau_F}{4\tau_E}\right)$ if $\tau_E > \frac{\tau_F}{2}$
$\tau_F = \frac{\sigma_F}{\sqrt{3}}$
σ_F the elastic limit of material (N/mmš)
τ_E the elastic shear buckling constraint (N/mmš)
$\tau_e = 0.9 k_T E \left(\frac{t_b}{1000s}\right)^2 (N/mm)$
$k_T = 5.34 + 4\left(\frac{s}{l}\right)^2$
E the elasticity modulus of the material (i.e. : 2.06 x 105 N/mmš for steel)
t_b the thickness of the plate (mm)
s the elementary plate width (m)
l the elementary plate length (m)
BV rule:

$$\tau_a \leq 110/k$$

with:
k the material factor
τ_a the hull girder shear stress (N/mmš)

(#37) Compression buckling of stiffeners
The critical compression buckling σ_c can not be smaller than the admissible value σ_a :

$$\sigma_c \geq \beta \sigma_a$$

with:

$$\beta = 1 \text{ for plates and members web and}$$
$$\beta = 1.1 \text{ for longitudinal members}$$

Compressive buckling constraints are computed as follows :

$$\sigma_a = \frac{M_s + M_w}{I_n} y \cdot 10^5 \text{N/mmš}$$
$$= \text{minimum} \frac{30}{k}$$

with
M_s the bending moment in still water (kNm)
M_w the wave bending moment (kNm)
I_n the net moment of inertia of the beam ship (cm^4)
y the distance between the neutral axis and the considered point (m)

k the material coefficient, eg. 1 for standard steel.

The critical buckling constraint σ_c is computed as follows :

$$\begin{aligned}\sigma_C &= \sigma_E \text{ if } \sigma_E \leq \frac{\sigma_F}{2} \\ &= \sigma_F\left(1-\frac{\sigma_F}{4\sigma_E}\right) \text{ if } \sigma_E > \frac{\sigma_F}{2}\end{aligned}$$

with:
σ_E the elastic buckling stress (N/mmš)
σ_F the elastic limit of material (N/mmš)

(#38) Stiffener Yielding

The normal constraint due to lateral pressure σ must verify:

$$\sigma \leq \frac{R_y}{\gamma_R \gamma_m}$$

with: $R_y = 235/k$ the minimum elasticity limit of material (N/mmš) γ_R the partial security coefficient on strength γ_m the partial security coefficient on material

The maximum value for the normal constraint due to lateral pressure σ is computed as follows:

$$\sigma = \frac{\gamma_{s2}p_s + \gamma_{w2}p_w}{12w}\left(1-\frac{s}{2l}\right) sl^2 10^3 + \sigma_{x1} \text{ (N/mmš)}$$

with: σ_{x1} overall normal stress
ps still water pressure (kN/mš)
pw wave pressure (kN/mš)
γ_{s2} the partial security coefficient on still water pressure
γ_{w2} the partial security coefficient on still water pressure
w net section modulus (cmş)
$bp = s$
s spacing of ordinary members (m)
l ordinary members flange (m)

C.2 Shear stress computation method

In this section we give an insight on the computation of Shear stress. This method has been developed by Richir, Picalabru, and Rigo (2006) based on Hughes (1988) and is just reproduced here to help the reader having a good overview of the model.

In a prismatic beam which is not subject to twist the shear strain in the cross section is simply the derivative of the warping :

$$\gamma = \frac{\partial u}{\partial s}$$

with s tangential or arc length coordinates within the cross section.

If end effects are ignored the warping in a prismatic beam maintains the same cross-sectional pattern along the length of the beam, and the magnitude at any section is proportional to the transverse shear force Q(x) at that section. It is therefore convenient to introduce a warping function $\omega Q(s)$ which describes the transverse distribution of warping, and express the warping as a product

$u(s, x) = \omega Q(s)Q(x)$

First the distribution of ωQ is represented in a discretized manner by defining "nodes" at which ωQ has a specific value and by assuming a linear variation between nodes. Then the total potential energy of the system Π is expressed in terms of the nodal values ωQ, and the derivative of Π with respect to each value of ωQ is set equal to zero, thus obtaining a system of equations for the ωQ values. After solving for these the complete distribution of shear stress ($\tau = G\gamma$) can be readily calculated.

A hull segment may consist of any number of flat rectangular elements of constant thickness t_e, breadth b_e, and length L.

The assumption of a linear variation of ωQ(s) within each element implies constant shear flow in the element, whereas it is known that the shear flow varies linearly in horizontal portions ship's cross section and parabolically in all other portions. It will be shown that the constant value of shear flow which is obtained for each element is the mean value for that element, and that the other part of the total shear flow (a linear or parabolic distribution, having a zero mean value) can be obtained separately for each element and added to the mean value. Thus the assumption of a linear distribution of warping within the element does not constitute an approximation, and so a single element can be used for any straight, constant thickness portion of the cross section, even if its breadth be is quite large.

In terms of nodal value ωQ1 and ωQ2 the warping within the element is:
$\omega_{Q(s)} = \omega_{Q1} + (\omega_{Q2} - \omega_{Q1} s/be$
In which the s direction is from element node 1 to element node 2. The orientation of the element is defined by the angle χ between the s axis and the horizontal axis of the hull girder.

In terms of strain the potential energy of an element is (with N_e the number of elements):
$U = 0.5QGL \sum_{e=1}^{N_e} \int_0^{b_e} \left(\frac{\omega_{Q2}^e - \omega_{Q1}^e}{b_e} \right)^2 t_e ds$
The other component of the total potential energy is the negative of the work done by the loads as a result of the warping.
$W = QL/I \sum_{e=1}^{N_e} t_e \int_0^{b_e} \left(\omega_{Q1}^e + \frac{\omega_{Q2}^e - \omega_{Q1}^e}{b_e} \right) y(s) ds$
with $y(s) = y_1 + s\ sin\chi$
The total potential energy of the system is $\Pi = U - W$, and the system of equations for the nodal values of ω is obtained by requiring that Π must be a minimum with respect to each ωi. That is:
$\frac{\partial \Pi}{\partial \omega_i} = 0 i = 1, \ldots, N_n$
where N_n is the number of nodes.

The resulting system of equations is:
G$\sum_{m=1}^{M} t_m \left(\frac{\omega_{Qi} - \omega_{Qr}^m}{b_m} \right) = \frac{1}{2I} \sum_{m=1}^{M} t_m b_m \left(y_{1m} + \frac{n_m}{3} b_m \sin \chi_m \right) i = 1, \ldots, N_n$
In these equations the subscript e has been replaced by m to indicate that since each equation refers to one node (the ith node) the summation over the elements need only include the M elements that touch that node. For each of these M elements the subscript r denotes the node which is remote from node i, and the symbol nm is the element node number (either 1 or 2) which corresponds to node i.

It can be shown that axial equilibrium requires that $\omega_Q = 0$ at all points which lie on the neutral axis. This requirement can be imposed by placing nodes at all such points and setting these values of ωQ equal to zero. This effectively divides the system of equations into two independent subsystems.

For each element the local distribution$\tau_e^l(s)$which is to be added to the mean value$\overline{\tau}_e$obtained from a warping solution is given by:

$$\tau_e^l(s) = \frac{Q}{I} \left[y_{1e} \left(s - 0.5 b_e \right) + 0.5 \sin \chi_e \left(s - \frac{b_e^2}{3} \right) \right]$$

This expression follows the same rules than the shear flow equations, in regard to sign convention : it assumes that the s-axis is in the direction of increasing shear flow (or shear stress), otherwise the sign of $\tau_e^l(s)$must be reversed.