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

Maud Bay† , Yves Crama† and Philippe Rigo‡ †*HEC Management School, University of Liège, Boulevard du Rectorat 7 (B31), B-4000 Liège, Belgium*; ‡ *Naval Architecture and Transportation Systems (ANAST), University of Liège, Chemin des Chevreuils 1 (B52-3), B-4000 Liège, Belgium*

This paper considers 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 paper, 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.

Keywords: structural design optimization; black-box functions; discrete variables; mixed-integer nonlinear programming

1. Introduction and motivation

In the 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 in 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.

In structural design problems, the constraints of the structural analysis are defined by implicit functions which model mechanical properties and have highly nonlinear behaviors. 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. In our study they are viewed as black-boxes that evaluate implicit constraints.

This paper focuses on difficult large-scale structural optimization problems where some of the variables are restricted to assume *discrete* values only. Even without this restriction, the continuous version of the problem is highly non-convex, and the evaluation of a solution (i.e., a structural analysis) is time consuming. The algorithms proposed in this paper carry out a limited exploration of the space of discrete solutions. They allow to compute one or more discrete solution(s) and require a relatively small number of calls to a subroutine 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.

2. The structural design problem

We consider here the structural design of large passenger ships such as cruise liners or carriers. The 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. The metallic structures are ship sections produced by assembling dozens of stiffened panels. The design variables defined for each panel are the thickness of the stiffened plates, the spacing between stiffeners and the dimensions of the stiffeners. 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.

The objective of the optimization problem is to minimize the weight or cost of the structure, subject to constraints relative to its mechanical behavior. The constraints are of three types: technological, geometrical and structural. Technological constraints define upper and lower bounds on the dimensions of the constitutive elements, such as the minimal thickness of plates. Geometrical constraints express relationships between the design variables, for example, ratios between the height and width of stiffening elements. Structural constraints represent limits on mechanical behaviors like deflection, yielding, buckling and stress in each element. These constraints are grouped in behavior models, the complexity of which leads to the impossibility of explicitly expressing the relationships between the constraint under consideration (deflection, stress) and the design variables (dimension and position of the members); see, e.g., Hughes (1988) and Rigo (2005).

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

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

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 be using confidential databases. The compu-

tation 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 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 $(k = 1, \ldots, m)$, 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 paper 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 $K \subseteq \{1, \ldots, m\}$ 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.

3. Literature review

Problems like $P(x, y)$ are discrete implicit optimization problems with time-expensive function evaluations. To solve the discrete problem $P(x, y)$, 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 structural continuous optimization, 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 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 2001a, Rigo and Fleury 2001) and is used to solve the continuous subproblems arising in our heuristics; see Section 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 *et al.* 1998, 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 (2001b) 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 an issue, 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. 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.* 2006) and mesh adaptive direct search methods (Audet and Dennis 2006); see Kolda *et al.* (2003) for a survey.

For problems with mixed (discrete and continuous) variables, like $P(x, y)$, common to all systematic global optimization algorithms is a branching scheme which recursively splits 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 are found in various domains besides naval structure optimization: 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 considered in this paper involve hundreds of discrete and continuous variables, and thousands of implicit constraints. For these computationally expensive implicit problems, 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.

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:

$$\lfloor r \rfloor = \max\{\ y_k \in Y_k \mid y_k \leq r\};$$

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:

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

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

$Round(r, Down) = \lfloor r \rfloor$,
$Round(r, Up) = \lceil r \rceil$,
$Round(r, Closest) = \lfloor r \rfloor$ if $(r - \lfloor r \rfloor < \lceil r \rceil - r)$,
$Round(r, Closest) = \lceil r \rceil$ if $(r - \lfloor r \rfloor \geq \lceil r \rceil - r)$.

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.

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)

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 $P^{K,\alpha}$;
b) the convergence properties of the optimization algorithm *Solve* which is used to handle these subproblems.

We discuss these issues hereunder.

4.1.1. *Existence of feasible solutions*

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.

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[1]. The monotonicity hypothesis will be further validated by the computational experiments reported in Section 5.

Thus, in practice, the Dive and Fix heuristic always produces 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

[1]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.

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 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 5.3.

4.1.2. *Solution of continuous relaxations*

The optimization of the implicit continuous problems with time expensive function evaluations, $P^{K,\alpha}(x, y)$, must be performed by an appropriate algorithm. We denote by $Solve(P^{K,\alpha}(x, y), x^0, y^0)$ the output of this optimization algorithm, given an initial solution (x^0, y^0) (see Algorithm 2). *Solve* requires a number of structural analyses.

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, the algorithm *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 (1998, 2001a, 2001b, 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 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 (see Fleury and Braibant

1986, Fleury 1989a):

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

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.

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 2001a,b, Rigo and Fleury 2001).

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 $\alpha_k := Round(y_k^q, rounding\_mode)$ 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 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.

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 $k \notin K, \tilde{y}_k^q := y_k^q$
  for all $k \in K, \alpha_k := Round(y_k^q, rounding\_mode)$ and $\tilde{y}_k^q := \alpha_k$
  $(x^*, y^*) := Solve(P^{K,\alpha}(x, y), x^q, \tilde{y}^q)$
  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 $Solutions$

In view of this hypothesis, when the *Closest* rounding mode leads to 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 eventually, the algorithm always finds at least one discrete solution of $P(x, y)$.

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.

5. Computational experiments

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 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 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.

5.2. *Results*

The Dive & Fix and the Branch & Fix heuristics have been applied to the three instances described above. Table 2 displays the value[1] 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^*

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

Table 1. Numerical data for three ships

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

Table 2. Experiments: Weight optimization

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

of the best discrete solution found by B&F. The next two columns of Table 2 give the relative difference between the value of the discrete solutions and the solution of the continuous relaxation $P^c(x, y)$:

$$\%(DF^* - Z^*) = 100 \times (DF^* - Z^*)/Z^*,$$
$$\%(BF^* - Z^*) = 100 \times (BF^* - Z^*)/Z^*.$$

The last column of Table 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 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.

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 3 provides some indication of the difficulty of the instances. All computations were performed on a personal computer (Pentium 4, 3GHz, 2Gb RAM). Table 3 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 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 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.

Table 3. Experiments: Size of the search trees and computing times

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

5.3. *Parameters of the heuristics*

In both heuristics D&F and B&F, the indices of the discrete y-variables are partitioned into subsets $K_1, \ldots, K_N$. The order in which the subsets $K_1, \ldots, K_N$ 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 $K_1, \ldots, K_N$, 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 $K_1, \ldots, K_9$ 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 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 4 and 6) 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 and Table 3.

Table 4. Experiments: Influence of the number of iterations in *Solve*

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

5.4. *Numerical tolerance issues*

As discussed in Section 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.

6. Discussion and conclusions

In naval structure design, the discrete nature of the variables and the extensive computation time needed to analyze a single structure of real size 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.

In this paper, we propose two algorithms to compute near-optimal solutions of the discrete structural optimization problem.

The first algorithm is a Dive & Fix heuristic that iteratively applies partial rounding to the current solution, and 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 can be applied 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 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.

In conclusion, these algorithms have proved able to provide efficient and effective tools for the structural optimization of large size passenger ships.

Acknowledgements

We wish to thank STX (formerly *Aker Yards France*) for providing the industrial benchmark instances. We also thank several colleagues from the ANAST department for their kind assistance in this work. This research was supported by the IMPROVE Project FP6-031382.

References

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

Alberto, P., *et al.*, 2004. Pattern search methods for user-provided points: application to molecular geometry problems. *SIAM Journal on Optimization*, 14 (4), 1216–1236.

Alexandrov, N.M., *et al.*, 1998. A trust-region framework for managing the use of approximation models in optimization. *Structural Optimization*, 15, 16–23.

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

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

Barthelemy, J. and Haftka, R., 1993. Approximation concepts for optimum structural design - A review. *Structural Optimization*, 5 (2–3), 129–144.

Beckers, M., 1997. Optimisation des structures en variables discrètes. Thesis (PhD). Université de Liège, Belgium.

Booker, A.J., *et al.*, 1999. A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary Optimization*, 17, 1–13.

Booker, A.J., *et al.*, 1998. 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*, Paper no 98–4717, Saint Louis, MO, USA, 2–4 September 1998.

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

Brekelmans, R., *et al.*, 2005. Constrained optimization involving expensive function evaluations: A sequential approach. *European Journal of Operational Research*, 160 (1), 121–138.

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

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

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

Davis, E. and Ierapetritou, M., 2009. 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.

Edvall, M. and Holmstrom, K., 2008. An adaptive radial basis algorithm (ARBF) for

expensive black-box global optimization. *Journal of Global Optimization*, 41 (3), 447–464.

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

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

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

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

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

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

Guikema, S.D., Davidson, R.A., and Çağnan, Z., 2004. Efficient simulation-based discrete optimization. *In*: R.G. Ingalls, M.D. Rossetti, J.S. Smith and B.A. Peters, eds. *Proceedings of the 2004 Winter Simulation Conference*, Vol. 1, Washington, DC, USA, 5–8 December 2004. IEEE, 536–544.

Hemker, T., *et al.*, 2008. A mixed-integer simulation-based optimization approach with surrogate functions in water resources management. *Optimization and Engineering*, 9, 341–360.

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

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

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

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

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

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

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

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

Pochet, Y. and Wolsey, L., 2006. *Production Planning by Mixed Integer Programming*. Heidelberg: Springer.

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

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

Rigo, P., 2001a. Least cost structural optimization oriented preliminary design. *Journal of Ship Production*, 17 (4), 202–215.

Rigo, P., 2001b. A module oriented tool for optimum design of stiffened structures – Part I. *Marine Structures*, 14, 611–629.

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

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

Rigo, P. and Fleury, C., 2001. Scantling optimization based on convex linearizations and

dual approach – Part II. *Marine Structures*, 14, 631–649.

Simpson, T.W., *et al.*, 2005. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA Journal*, 39 (12), 2233–2241.

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

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

Tompkins, G. and Azadivar, F., 1995. Genetic algorithms in optimizing simulated systems. *In*: C. Alexopoulos and K. Kang, eds. *Proceedings of the 1995 Winter Simulation Conference*, Arlington, VA, USA, 3–6 December 1995. Washington, DC: IEEE Computer Society, 757–762.

Torczon, V. and Trosset, M.W., 1998. Using approximations to accelerate engineering design optimization. *In*: *Proceedings of the Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Paper no 98–4800, Saint Louis, MO, USA, 2–4 September 1998.

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

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

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