# A Multi-Objective Genetic Algorithm for Jacket Optimization

Jan Burak
Norwegian University of Science and Technology
jan@kodeworks.no

Ole Jakob Mengshoel
Norwegian University of Science and Technology &
Carnegie Mellon University
ole.j.mengshoel@ntnu.no

## ABSTRACT

Jackets are massive steel towers supporting offshore installations such as oil platforms and wind turbines. Due to the high costs of material, construction, and installation, there is an interest in optimizing such jacket designs. This is an example of the broader problem of structural design optimization.

In this paper, we describe underlying concepts related to the problem of jacket design as well as previous research on jacket design optimization. Motivated by the complexity of the problem, including the multiple objectives typically involved, we develop a novel multi-objective genetic algorithm, NSGA-J, which is tailored to jacket design optimization. NSGA-J is based on the prominent Non-Dominated Sorting Genetic Algorithm (NSGA-II), but tailors it to the problem of designing jackets. Experimentally, we study a cloud-based implementation of NSGA-J and present our results and experiences. The paper ends with a discussion of lessons learned and sketches opportunities for future research. We hope to inspire future work on complex applications of structural design optimization including jacket design optimization.

## CCS CONCEPTS

• **Computing methodologies** → **Randomized search**; **Continuous models**; • **Applied computing** → **Computer-aided design**;

## KEYWORDS

Jacket Design, Structural Design Optimization, Multi-Objective Optimization, Genetic Algorithms, Non-Dominated Sorting Genetic Algorithm, Simulation, Cloud Computing.

**ACM Reference Format:**
Jan Burak and Ole Jakob Mengshoel. 2021. A Multi-Objective Genetic Algorithm for Jacket Optimization. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3449726.3463150

## 1 INTRODUCTION

**Context.** Offshore structures are exposed to varying environmental conditions, leading to different types of structures being developed [6, 20]. Tubular steel truss structures, often called jackets, support drilling rigs and offshore wind turbines. Jacket heights vary from tens to hundreds of meters and weigh thousands of tonnes, based on the topside and sea depth. Jacket design is a subfield of structural design, which concerns itself with the strength, rigidity, and stability of structures. Kicinger et al. [10] outline three domains of structural optimization, which are topology, shaping, and sizing. When simplifying a structure to a graph, *topology* optimization [7] decides the number of nodes and connections between them, corresponding to joints and beams of a jacket. *Shaping* decides the angles between elements and their lengths, while *sizing* adjusts the diameters and thicknesses of elements. The majority of a jacket's construction cost comes from the cost of steel. There is also significant welding cost, determined by the angles at which the elements are connected [3].

The amount of steel in a jacket can be algorithmically optimized whilst still meeting the design requirements for an expected structure lifetime. To this end, the company Kvaerner has developed a genetic algorithm (GA) to optimize jacket designs. This GA is well-suited for geometrical optimization of jackets [11]. There are two main types of structural design optimization methods, namely traditional methods [9, 24, 26] and evolutionary algorithms (EAs) [8, 11, 12, 19, 21, 25]. We focus on EAs in this work.

**Challenges.** Structural design is a complex task, and challenges facing optimization of jackets include: (i) the complexity of the analysis required to evaluate each design; (ii) the accuracy of the input data available at the early stages of design; (iii) the fact that a design needs to be construction-friendly; and (iv) the many objectives and constraints that a design needs to satisfy. Since the multi-objective perspective is often important in jacket design optimization, we focus mainly on problem (iv) in this paper.

**Contributions.** We develop and test a MOEA, NSGA-J [2]. NSGA-J is based on the Non-Dominated Sorting GA (NSGA-II) [4], which is considered one of the most prominent MOEAs. We evaluate NSGA-J in jacket design optimization experiments. NSGA-J uses exactly the same software as engineers designing jackets. However, while engineers bring substantial skills and intuition to the design process, they may stop too early in a manual design optimization process. The reason is the time it takes to re-evaluate jacket designs with only manual or semi-automatic design optimization aids. In contrast, our automated approach enables a more iterative design process, where there is more flexibility in where you stop in the iterative process. Further, our software uses DNV GL's OneCompute cloud, and is thus quite scalable.

**Overview.** The structure of the rest of the paper is as follows. Section 2 discusses related work with an emphasis on structural optimization using EAs. Section 3 presents notation and theory behind multi-objective GAs as well as NSGA. Section 4 describes how NSGA-J implements a GA for jacket design optimization. Section 5


*GECCO '21 Companion, July 10–14, 2021, Lille, France*

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463150

presents experimental results for NSGA-J, while Section 6 provides discussion and lessons learned with ideas for future work.

## 2 BACKGROUND AND RELATED RESEARCH

Here we discuss related work on jacket design optimization, as well as on similar applications of EAs in structural optimization including truss optimization. In particular, jacket optimization can be considered a special case of truss optimization, namely optimization of massive (weight of 100s or 1000s of tonnes) trusses for offshore deployment. Many types of multi-objective metaheuristics, including EAs, have been applied to structural optimization and truss optimization [28].

There are several studies on jacket optimization for offshore wind turbines (OWT). Some work has been done using evolutionary methods, but much of the literature employs gradient methods. There is no universal consensus on which methodology is preferred; our focus in this paper is on EAs, in particular GAs.

**Single-objective.** Pasamontes et al. [21] use a binary single-objective GA (SOGA) to optimize the mass of the OC4 jacket by sizing the beams and adjusting the elevation heights, thus obtaining a 30% reduction in mass. Extending the work of Pasamontes et al., Schafhirt et al. [22] focus on the thickness and diameter parameters. The number of iterations until convergence is reduced to one third. Martens et al. [16] use a GA to optimize both the topology and sizing of an OWT jacket. Jacket topology is created by picking beams from a predefined set; GA operators are implemented similarly to in Pasamontes' system. Häfele et al. [8] performed extensive modelling of the jacket design search space and employed a particle swarm optimization (PSO) algorithm, the Augmented Lagrangian PSO (ALPSO). ALPSO's focus is on representing the problem in a way that does not restrict the formation of solutions. A test was run for the OC4 reference jacket, taking 47 days.

**Multi-objective.** Kunakote and Bureerat [12] evaluate different MOEAs with regards to topology optimization: (i) Pareto archive evolution strategy (PAES); (ii) population-based incremental learning (PBIL); (iii) NSGA-II; (iv) strength Pareto evolutionary algorithm (SPEA2); and (v) multi-objective particle swarm optimization (MPSO). Four environments were modelled based on a grid with specific criteria, and thus the search space admitted arbitrary structures. PBIL scored highest in computational experiments, with PAES in second place. The use of EA methods was said to be advantageous due to their robustness. However, compared to realistic jacket design problems the problems studied are small, bordering on toy problems. Noilublao and Bureerat [19] compared the performance of PBIL, SPEA2, and archived multi-objective simulated annealing (AMOSA) when optimizing the topology, shape, and sizing of a truss tower. The experimental evaluation was split into four bi-objective problems, with mass being present in every problem. PBIL turned out to be the most effective for the compliance problem. For the three other problems, PBIL produced more extended fronts, whilst the fronts of SPEA2 advanced more. The computation time was between 30 and 360 minutes, depending on the problem, with AMOSA having the shortest computation time.

**Discussion.** While there is related work to ours, much of it uses traditional (and not evolutionary) optimization methods or are single-objective (and not multi-objective). We now compare and contrast our NSGA-J to the most similar works [4, 8, 11, 12]. Compared to NSGA-II [4], NSGA-J (i) uses multiple and more complex mutation and crossover operators; (ii) contains mechanisms to avoid duplicates in the population; (iii) creates the initial population from an existing jacket design; and (iv) is tailored to jacket design optimization (see Section 4.4 for details). Kunakote and Bureerat [12] study five MOEAs for topology optimization, including NSGA-II. They study 2D-grids that are simple compared to a full-blown jacket, and employ a 2D-grid method, where they start with a grid, and fill in. Compared to Kunakote and Bureerat [12], we work directly on a realistic 3D representation of jackets, using cloud computing, as do engineers designing jackets. Häfele et al. [8] consider many design variables (material, production, coating, transition piece, transport, and installation), but combine them into a single-objective particle swarm optimization algorithm. In comparison, we use a MOEA approach. Kling et al. [11] also use a 3D representation of jackets, and the way jackets are evaluated and processed (crossover, mutation) is very similar to ours. However, they focus on single-objective optimization and not multi-objective optimization.

## 3 MULTI-OBJECTIVE OPTIMIZATION AND EVOLUTIONARY ALGORITHMS

Structural design optimization, including jacket design optimization, are among the complex optimization problems where multiple objectives (or fitness functions) often come into play. Fortunately, the evolutionary computation family includes various algorithms that take multiple objectives into consideration.

### 3.1 Notation and Representation

Let $\mathbb{W}$ be the whole numbers, $\mathbb{N}$ the natural numbers, and $\mathbb{R}$ the real numbers. Let $n \in \mathbb{N}$. A genotype (or individual) $\mathbf{x}$ is an $n$-tuple $(x_1, \ldots, x_j, \ldots, x_n)$, with $j \in \mathbb{N}$ and gene $x_j \in \mathbb{R}$ for $1 \leq j \leq n$. Each real-valued gene is lower- and upper-bounded, and has an accuracy. These things can be represented using $\mathbf{x}$'s "twin" $n$-tuple $\mathbf{r} = (r_1, \ldots, r_j, \ldots, r_n) = ((\ell_1, u_1, a_1), \ldots, (\ell_j, u_j, a_j), \ldots, (\ell_n, u_n, a_n))$, where $\ell_j, u_j, a_j \in \mathbb{R}$ and $\ell_j \leq x_j \leq u_j$. Note that $x_j = \ell_j + k \times a_j$, where $k \in \mathbb{W}$.

In words, a gene $x_j$ in an individual $\mathbf{x}$ is lower- and upper-bounded by $\ell_j$ and $u_j$ respectively, and discretized according to $a_j$. The tuple $r_j = (\ell_j, u_j, a_j)$ corresponding to a gene $x_j$ is denoted that gene's profile. For simplicity, we assume that individuals have a fixed number $n$ of genes.[1]

Genes correspond to the diameters and thicknesses of individual beams in a jacket. Further, genes also represent the main parameters of a jacket, namely base dimensions and elevation heights.

A population $\mathbf{X}$ is a tuple (such that repetition is, in general, allowed) of $m$ individuals $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$, with $m \in \mathbb{N}$ and $m \geq 2$. GAs sometimes use multiple populations, for example two populations $\mathbf{X}_1$ and $\mathbf{X}_2$.

### 3.2 Multi-Objective Optimization (MOO)

Definitions below are taken from Zavala et al. [28]. In the definition, $\mathbf{x}^*$ denotes an encoding of a solution, and $\mathbf{f}(\mathbf{x})$ the objectives. It is

[1] In the GA system, individuals can in fact have different number of genes, reflecting adding and removing of beams during GA jacket optimization.

assumed, without loss of generality, that all the objective functions are to be minimized.

*Definition 3.1.* (Multi-Objective Problem) Find a vector $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ which satisfies the $m$ inequality constraints $g_i(\mathbf{x}) \geq 0$, $i = 1, 2, \ldots, m$, the $p$ equality constraints $h_i(\mathbf{x}) = 0$, $i = 1, 2, \ldots, p$, and minimizes the vector function $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the vector of decision variables.

Definition 3.1 is tailored to NSGA-J and jacket design optimization in Definition 4.1.

We use the concept of Pareto optimality, see Definition 3.2. The feasible region $\Omega$ includes all solutions that satisfy the constraints mentioned in Definition 3.1.

*Definition 3.2.* (Pareto Optimality) Given a feasible region $\Omega$, a point $\mathbf{x}^* \in \Omega$ is Pareto Optimal if for every $\mathbf{x} \in \Omega$ and $I = \{1, 2, \ldots, k\}$ either $\forall_{i \in I}(f_i(\mathbf{x}) = f_i(\mathbf{x}^*))$ or there is at least one $i \in I$ such that $f_i(\mathbf{x}) > f_i(\mathbf{x}^*)$.

An operator $\preccurlyeq$ is used to check for dominance between two solutions. A dominating solution has at least one objective value that is better, and the remaining values are better or equal. This is formalized in Definition 3.3.

*Definition 3.3.* (Pareto Dominance) A vector $\mathbf{u} = (u_1, \ldots, u_k)$ is said to *dominate* $\mathbf{v} = (v_1, \ldots, v_k)$ (denoted by $\mathbf{u} \preccurlyeq \mathbf{v}$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e., $\forall i \in \{1, \ldots, k\}: u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$.

A key concept is the Pareto optimal set, which is defined as the set including all non-dominated feasible solutions. Furthermore, the objective values of Pareto optimal solutions make up the Pareto front. Both of these concepts are formalized in Definition 3.4.

*Definition 3.4.* (Pareto Optimal Set; Pareto Front) For a given MOP $\mathbf{f}(\mathbf{x})$, the *Pareto optimal set* is defined as $\mathcal{P}^* = \{\mathbf{x} \in \Omega | \neg \exists \mathbf{x}' \in \Omega, \mathbf{f}(\mathbf{x}') \preccurlyeq \mathbf{f}(\mathbf{x})\}$. For a given MOP $\mathbf{f}(\mathbf{x})$ and its Pareto optimal set $\mathcal{P}^*$, the *Pareto front* is defined as $\mathcal{PF}^* = \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in \mathcal{P}^*\}$.

## 3.3 Non-Dominated Sorting Genetic Algorithms (NSGA)

One of the most popular multi-objective approaches from the EA family is the non-dominated sorting GA (NSGA) family introduced by Srinivas and Deb in 1995 [23]. NSGA concerns itself mostly with the reinsertion step, while also the selection and evaluation steps need to be adjusted to take multiple objectives into consideration. In this section, NSGA-II, an improved version of the approach, will be presented [4]. NSGA-II introduces constraint handling by augmenting domination (operator $\preccurlyeq$ in Definition 3.3) with constrained-domination:

*Definition 3.5.* (Constrained-domination) A solution $\mathbf{x}_i$ is said to constrained-dominate a solution $\mathbf{x}_j$, if any of the following is true: (i) $\mathbf{x}_i$ is feasible and $\mathbf{x}_j$ is not; (ii) $\mathbf{x}_i$ and $\mathbf{x}_j$ are both infeasible, but $\mathbf{x}_i$ has smaller overall constraint violation; or (iii) $\mathbf{x}_i$ and $\mathbf{x}_j$ are both feasible but $\mathbf{x}_i \preccurlyeq \mathbf{x}_j$.

NSGA-II reinsertion combines the parent and offspring populations into one set. That set is then ranked into fronts of non-dominated individuals, using Definition 3.5. The rank function $\rho$ maps from the space of individuals, $\mathbb{R}^n$, to $\mathbb{N}$ (lower rank is better). Then, the next population is filled up rank-wise, starting with rank $k = 1$. If the rank $k \in \mathbb{N}$ individuals do not fully fit, they are sorted using the crowded comparison operator $\prec_n$:

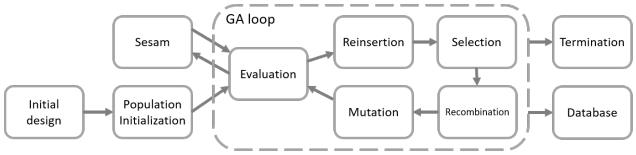



**Figure 1: A data flow diagram of the software system discussed here, consisting of two types of modules: (i) GA loop modules inside the dashed line and (ii) external modules. One instantiation of (i) is studied in this paper, namely NSGA-J (see Section 4.3).**

*Definition 3.6.* (Crowded comparison) Given individuals $\mathbf{x}_i$ and $\mathbf{x}_j$, $\mathbf{x}_i \prec_n \mathbf{x}_j$ if $(\rho(\mathbf{x}_i) < \rho(\mathbf{x}_j)) \vee ((\rho(\mathbf{x}_i) = \rho(\mathbf{x}_j) \wedge (\delta(\mathbf{x}_i) > \delta(\mathbf{x}_j)))$.

Intuitively, this operator compares two individuals based both on their (i) ranks $\rho$ and (ii) crowding distances $\delta$. The crowding distance $\delta$, which expresses how far apart individuals are from each other in objective space, seeks an even spread of individuals across the last front with regards to every objective.

In both NSGA-II and NSGA-J, classic tournament selection is used along with the partial order operator from Definition 3.6.

# 4 GENETIC ALGORITHM FOR JACKET DESIGN OPTIMIZATION

In this section, we discuss how the problem of jacket design optimization (see Section 2) can be solved using GAs. This approach produced the experimental results discussed in Section 5.

Our overall NSGA-J system is shown in Figure 1. The system's modules are detailed in this section. We first present, in Section 4.1, modules that are external to the core GA. The emphasis is on their interface to the GA. Second, the multi-objective problem formulation for NSGA-J is presented in Section 4.2 while NSGA-J is discussed in Section 4.3. Third, we discuss key differences between NSGA-J and NSGA-II in Section 4.4.

## 4.1 External Modules

DNV GL's **Sesam** software is used by both engineers and optimization software, including GAs, to evaluate designs. Thus, the files describing jacket designs are in the format used by the Sesam software.[2] Once GA individuals have been created, they are transformed to phenotype forms to be *input* to Sesam evaluation. As *output* from Sesam, analysis produces a utilization report for each GA individual. The report includes the inherent physical properties of each element, their utilizations, and angles between the elements. Since the genotype includes parameters influencing multiple elements at the same time, the information obtained from the evaluation is not readily available in the genotype.

[2]DNV GL Sesam for fixed structures https://www.dnvgl.com/services/offshore-and-marine-structural-engineering-sesam-for-fixed-structures-1096

| Notation $g_i$ | Result Function | Textual Description |
|---|---|---|
| $g_1(\cdot)$ | Total Weight | Sum weight of each individual element. It is multiplied by a cost approximating steel's cost. |
| $g_2(\cdot)$ | Total Utilization Factor | Sum of each element's utilization. |
| $g_3(\cdot)$ | Pile Violations | Number of piles that experience forces outside of a required range. |
| $g_4(\cdot)$ | Weld Weight | The complexity and cost of welding elements together. It is computed based on the area between connected elements, multiplied by an estimated cost. |
| $g_5(\cdot)$ | Wajac Penalty | If Wajac failed, $g_5 = 1$. Wajac consists of hydrostatic, hydrodynamic, and wave fatigue analyses. |
| $g_6(\cdot)$ | Angle Violations | Number of angles between elements that are not within required ranges. |
| $g_7(\cdot)$ | Utilization Violations | Number of elements with utilization above 1. |

**Table 1: The result functions $\{g_1(\cdot), \ldots, g_7(\cdot)\}$, computed from Sesam analysis reports, used for fitness and other computations in GAs. Their use depends on the GA in question; see Section 4.2 about NSGA-J fitness and constraint functions. The Wajac penalty $g_5(\cdot)$ uses Sesam's Wajac analysis, see https://www.dnvgl.com/services/hydrostatic-and-hydrodynamic-analysis-wajac-2244.**

An **initial design**, see Figure 2(a) for an example, is input to the system. An initial design, typically created by experts or derived from an expert design, consists of Sesam files describing the geometry of the jacket and the environmental conditions of the deployment site.[3] The system parses the geometry of the initial design to create a list of elements and a graph describing its connections.

Each gene has an ID (the index) that shows whether two genes from two different individuals relate to the same structural element, which is needed for the computation of diversity and recombination. Further, all genes have values within predefined ranges, and are discretized with a given accuracy, as described in the profile $\mathbf{r} = (r_1, \ldots, r_n) = ((\ell_1, u_1, a_1), \ldots, (\ell_n, u_n, a_n))$.

**Population initialization** amounts to cloning the individual created from the initial design and adjusting every gene by a normal distribution.[4] New individuals are created in this way until the population is filled to a given maximum size $m$. As shown in Figure 1, the initial population goes straight to the evaluation. Thus, individuals have fitness values before going to selection. (The reinsertion used on the initial population does not affect it, as at that point the number of individuals is equal to the maximum population size.)

The theoretical benefits of initialization, in terms of improved runtime, are not fully understood for EAs, although there are initial results [1]. However, initialization using expert designs or problem-specific heuristics are well-known methods to empirically reduce runtime both in EAs [5] and stochastic local search (SLS) [18].

**Termination** takes place after a fixed number of generations, $n_G$. NSGA-J is run for $n_G$ generations and the individuals in the last generation are considered the final computational results.[5]

## 4.2 NSGA-J: Fitness and Constraint Functions

Sesam outputs are available for **NSGA-J evaluation**, see Table 1. From these results, two objectives $f_1$ and $f_2$ are formulated to be minimized by NSGA-J, along with one equality constraint. Specifically, Definition 3.1 is instantiated to the following:

*Definition 4.1.* (Multi-Objective Problem in NSGA-J) Find a vector $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ which satisfies one equality constraint $h_1(\mathbf{x}) = 0$ and minimizes the vector function $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))^T$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the vector of decision variables.

In Definition 4.1, $f_1(\mathbf{x}) = g_1(\mathbf{x}) + g_4(\mathbf{x})$ is the sum of total weight and weld weight (see descriptions of $g_1$ and $g_4$ in Table 1), which were grouped together due to being closely correlated. The second objective, $f_2$, is the total utilization factor, $f_2(\mathbf{x}) = g_2(\mathbf{x})$ (see $g_2$ in Table 1). The two objectives $f_1$ and $f_2$ are competing, as lower utilization is obtained by using more steel, thus giving a larger safety margin. This in turn increases the total weight of the design.

Angle, utilization, and pile violations, as well as the Wajac penalty, form an equality constraint $h_1(\mathbf{x})$. Specifically, the sum $h_1(\mathbf{x}) = g_3(\mathbf{x}) + g_5(\mathbf{x}) + g_6(\mathbf{x}) + g_7(\mathbf{x})$ is an equality constraint per Definition 3.1 and Definition 4.1. If $h_1(\mathbf{x}) = 0$, individual $\mathbf{x}$ is feasible, else if $h_1(\mathbf{x}) \neq 0$ $\mathbf{x}$ is infeasible (see Definition 3.5). Descriptions of Sesam result functions $g_3$, $g_5$, $g_6$, and $g_7$ are in Table 1.

## 4.3 NSGA-J: Optimization Loop

Once the population is initialized, **NSGA-J** enters a loop that is run until termination. The loop proceeds in the same way as for a classic GA.

**NSGA-J evaluation**, see Figure 1, is multi-objective fitness evaluation following Definition 4.1. NSGA-J's two fitness functions $f_1$ and $f_2$ as well as the equality constraint $h_1$ are computed from Sesam's analysis results as discussed in Section 4.2.

**NSGA-J reinsertion** closely follows the NSGA-II reinsertion scheme, which uses constrained domination (see Definition 3.5). Early tests of ours suggested that some duplicate solutions end up in the population. NSGA-J reinsertion was therefore adjusted to remove individuals with combinations of objective values equal to those of other individuals. Thus after every reinsertion, a population is obtained where every individual has a unique combination of objective values. Such duplicate removal is in fact not part of NSGA-II [4], however the idea of non-revisiting GAs and EAs to avoid re-computing fitness values is known [15, 27].

**NSGA-J selection** inputs are tournament size $n_T$ and population $\mathbf{X}$; output is parent population $\mathbf{P}$. The rivals (or competitors) $\mathbf{P}$ are compared against each other using the crowded comparison operator (see Definition 3.6). The best among rivals $\mathbf{R}$ is added to the set of parents. With a tie among rivals, an individual $\mathbf{w}_i$ is picked at random. Given that the recombination operator that comes after

[3] The geometry of a design is described by a few key numerical parameters like elevation heights, and foot and head dimensions that decide the overall shape of a jacket. After objects representing legs are created, horizontal braces are added at the elevation heights. Subsequently a bracing pattern is formed by adding members in positions relative to the elements already created. The design files include a list of possible pipe sizes to choose from, and every member is given a specific pipe size.

[4] Early tests of NSGA-J, using a random initial population, suggested that the constraint handling method struggles to make progress from a population of random, infeasible individuals.

[5] Alternatively, the final non-dominated front could be computed by considering individuals from all $n_G$ generations [14].

| Math | Operator | Prob. | Object |
|---|---|---|---|
| $p_S^C$ | Crossover gene switch | 0.5 | Element |
| $p_I^C$ | Crossover gene interpolation | 0.1 | Element |
| $p_D^M$ | Mutation beam disposal | 0.025 | Individual |
| $p_C^M$ | Mutation beam creation | 0.0025 | Individual |

**Table 2: Probabilities for the operators, except for uniform adaptive mutation (see Table 3).**

selection picks parents in pairs from the set, pairs of duplicates are avoided by temporarily removing the last tournament winner from the set of candidates for the next iteration. This was to avoid parents with equal genotypes, as this would cause recombination to not produce new genomes.

Picking of parents is done with replacement. The procedure is a mixture of classic tournament selection with the partial order operator from NSGA-II, together with an NSGA-J-specifc addition of no consecutive duplicates. The value of the tournament size is a trade-off between exploration and exploitation. With lower values, it is more likely for worse solutions to reproduce. With higher values, the better solutions are prioritized, but the number of possible unique pairings is lower.

**NSGA-J recombination** takes parents $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$ in order in pairs from the parent list to crossover their genes and produce two offspring.

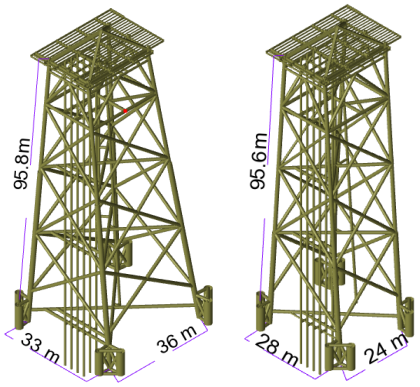
There are two types of crossover: gene switch crossover and gene interpolation crossover. Notation and probabilities used in experiments are summarized in Table 2. For each pair of genes across two individuals relating to the same structural element, the scheme can perform two operations with given probabilities. Gene switch crossover exchanges the gene values. Gene interpolation crossover interpolates the values by assigning new values to the genes of both individuals taken from a uniform distribution with the two original values as endpoints. Offspring $\mathbf{u}'$ and $\mathbf{v}'$ are automatically brought to the precision determined by the gene profile $\mathbf{r} = (r_1, \ldots, r_n)$.

| Math | Parameter | $p_{A1}^M$ | $p_{A2}^M$ |
|---|---|---|---|
| $p_E^i$ | Element mut. prob. | 0.005 | 0.005 |
| $p_S^i$ | Step percentage | 0.1 | 0.6 |
| $p_U^i$ | Upper prob. (ceiling) | 0.5 | 0.5 |
| $p_L^i$ | Lower prob. (floor) | 0.005 | 0.0005 |
| $\Delta p_I^i$ | Prob. increase rate | 0.1 | 0.05 |
| $\Delta p_D^i$ | Prob. decrease rate | 0.2 | 0.1 |

**Table 3: Parameter for the two uniform adaptive mutations $p_{A1}^M$ and $p_{A2}^M$, defined by tuples $p_{Ai}^M = (p_E^i, p_S^i, p_U^i, p_L^i, \Delta p_I^i, \Delta p_D^i)$, where $i \in \{1, 2\}$. For other operators, see Table 2.**

**NSGA-J mutation** comes in several types: beam disposal mutation, beam creation mutation, joint creation mutation, and adaptive mutation. Details including probabilities used in experiments are summarized in Table 2 and Table 3.

The mutations used in experiments were create beam, dispose beam, and uniform adaptive mutation. The create beam mutation adds new elements to the structure by either utilizing existing joints or creating new joints for the beam to connect to. The genes of the new beam are randomized. Conversely, the dispose beam mutation removes an existing element at random. These two operations carry out topology optimization. Uniform adaptive mutation adjusts gene




**(a) Input** **(b) Output**

**Figure 2: Jacket input and output for NSGA-J.**

values by a given step percentage and with a given probability. The step percentage gives a step value based on the gene profile for the type of the element being adjusted. Additionally, the probability of performing that mutation is adjusted in every iteration of the NSGA-J loop. If no improvement in the best individual has been observed in the previous generation, the probability of mutation is increased by a given increase rate. If improvement has been observed, the probability is lowered. Maximum and minimum probability values are given to limit the range of adjustment. Given that this mutator adjusts the genes relating to the diameters and thicknesses of elements, as well as the main parameters, it constitutes shaping and sizing optimization.

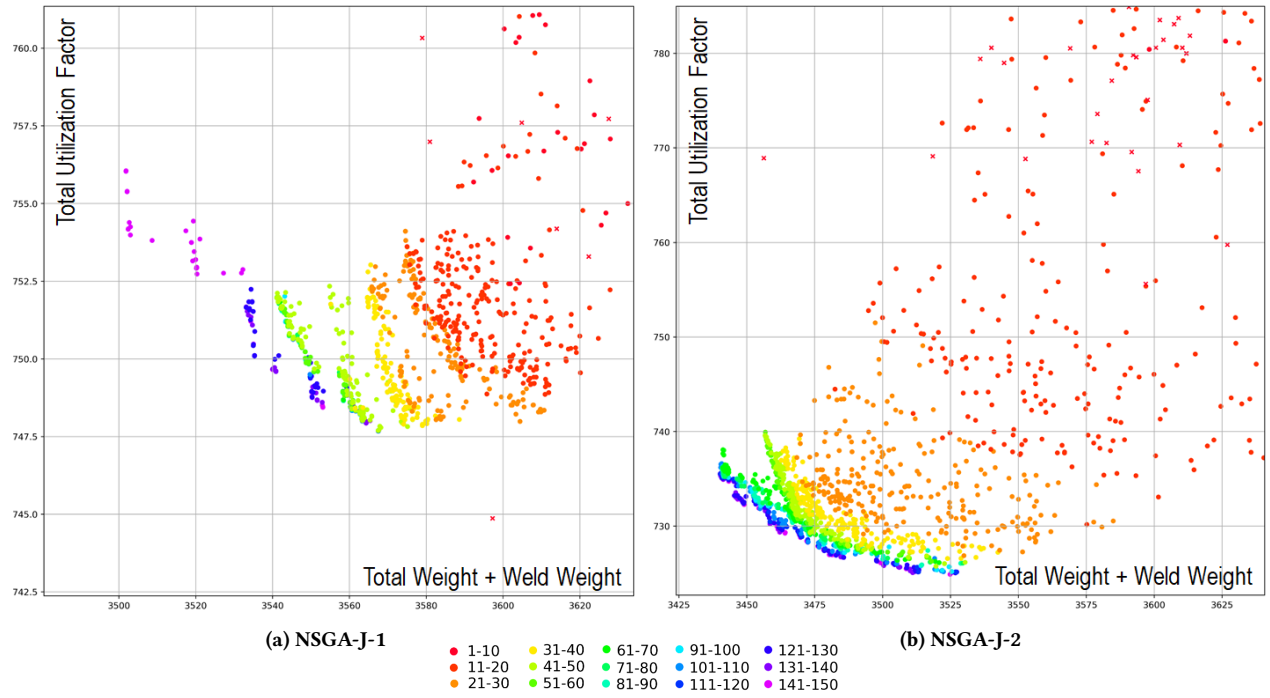
Two versions of uniform adaptive mutation, defined by tuples $p_{A1}^M$ and $p_{A2}^M$, were used concurrently during each test run. Their parameter values are summarized in Table 3. The most significant difference here is that one of the mutations adjusts the gene values by a step percentage of $p_S^1 = 0.1$, the other one by $p_S^2 = 0.6$.

## 4.4 From NSGA-II to NSGA-J

NSGA-J is partly an application of NSGA-II and partly has differences from NSGA-II, motivated by the jacket design optimization problem. We now summarize key areas where the algorithms differ.

First, NSGA-II uses single-point crossover and bitwise mutation for binary-coded GAs and simulated binary crossover operator and polynomial mutation for real-valued GAs. NSGA-J, in contrast, uses multiple and more complex mutation and crossover operators (see Table 2 and Table 3) with a real-valued chromosome. Second, NSGA-II's initial population is created via randomization. In NSGA-J, initialization of the population is done from an existing jacket design (see Section 4.1). Third, NSGA-J makes an effort to avoid duplicates in the population, while in NSGA-II there are no such mechanisms. Fourth, NSGA-II when introduced was tested experimentally on synthetic test problems, while NSGA-J's experiments are performed on a practical engineering problem, namely jacket design optimization.

The three last points above reflect the fact that Sesam's analysis of a jacket design is a complex and time-consuming computation. Sesam produces extensive results (not just one or a few fitness values) from which NSGA-J fitness and constraint functions are computed.




**(a) NSGA-J-1** **(b) NSGA-J-2**

**(c) Color Scheme**

**Figure 3: Objective space plots, with $f_1$ on the $x$-axis and with $f_2$ on the $y$-axis, for evolutions of experiments with NSGA-J. Individuals from the experiment, except outliers, are included. The color scheme for individuals, presented in (c), is as follows: early generations are in red, middle generations in green, and late generations in blue.**

# 5 EXPERIMENTAL RESULTS

## 5.1 Experimental Protocol

**Experimental Data.** The experimental input was a design of the Valhall Flank West jacket, see Figure 2. The process was as follows. A variant of an existing GA [11] was run once, until generation of at least one feasible design, which was then used to seed the initial population. Thus, NSGA-J starts out with at least one feasible design (see left-most module in Figure 1).

**Computational Infrastructure.** NSGA-J was implemented using the C# programming language. A key software library used is Sesam. Sesam running in the OneCompute cloud solution is used to compute analysis results underlying evaluation of objective and constraint functions. Once the Jacket design files are prepared, they are uploaded to virtual machines running Sesam in Azure[6] cloud. Evaluation is the most computationally demanding part of the system, and employing virtual machines enables running multiple Sesam evaluations in parallel.

**GA Parameter Settings.** Several experimental runs for NSGA-J were performed; for space reasons we focus on two random ones. Each run was using $m = 50$ individuals and $n_G = 150$ generations. Results for these experimental runs are presented below, using the notation NSGA-J-$i$, where $i \in \{1, 2\}$. In NSGA-J, the tournament size was set to $n_T = 3$. Probabilities and other values used for hyperparameters for NSGA-J are shown in Table 2 and Table 3.

[6]Azure cloud https://azure.microsoft.com/en-us/overview/what-is-azure/

## 5.2 Experimental Results I: Overview

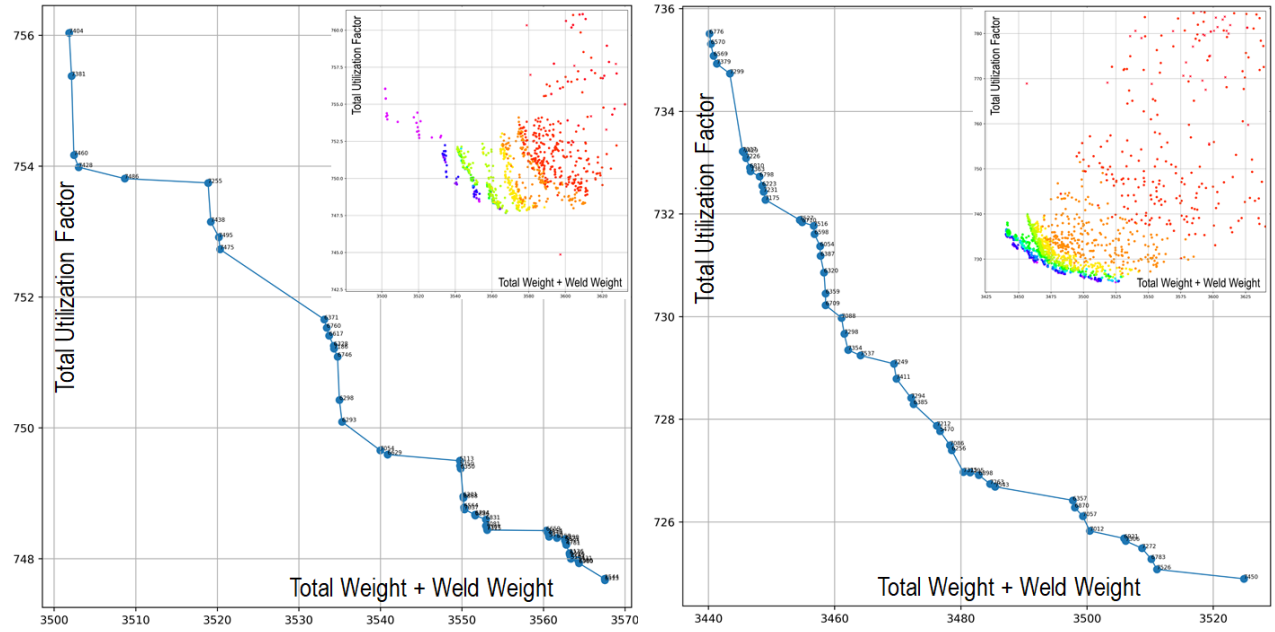
**Goal.** What are typical results, computation times, best fitness values, and lessons learned from test runs?

**Method and Data.** Early on, an Azure cloud solution was used for Sesam. The Azure cloud solution was then replaced with OneCompute,[7] developed by DNV GL. Close integration of the evaluation software with the OneCompute cloud solution allowed for more reliable execution. The switch improved the analysis time, allowed running more evaluations in parallel, and reduced the number of failed evaluations to an insignificant number. The results reported below are for the OneCompute cloud.

**Results and Discussion.** Figure 2 presents an NSGA-J input and output. NSGA-J compute times are on the order of 12 hours for each GA run of 150 generations.[8] While substantial, such compute times can reasonably fit into jacket engineering work processes.

The final problem formulation (see Definition 4.1), configuration, and results of NSGA-J, as presented here, were arrived at after many careful pilot studies.[9] Originally, and instead of what is shown in Definition 4.1, the two objectives used were total weight $g_1$ and weld weight $g_4$. However, this version produced little variation in individuals in experiments. An experiment was also performed with each result function (see Table 1) as an NSGA-J objective, but it produced too many infeasible individuals for any meaningful

[7]https://devpeuwwa01platonecomputedocumentation.azurewebsites.net/docs/v3.0/
[8]The NSGA-J-1 run took 12 hours 40 minutes, while the NSGA-J-2 run took 13 hours.
[9]Unfortunately, the configurations and results of these pilot studies are too numerous to discuss in detail here. Further, they are also deemed to be less interesting to the interested reader than the successful configurations and results being reported.




(a) NSGA-J-1 (b) NSGA-J-2

**Figure 4: Objective space plots, with $f_1$ on the $x$-axis and $f_2$ on the $y$-axis, for NSGA-J. These plots are for the last generation of the experiments, and detail the corresponding plots in Figure 3, which are also duplicated in upper-right-corner insets.**

advancement. Utilization was once used as an objective to be maximized whilst keeping the per-element utilization below 1, as it is a common jacket design guideline. But this approach also led to poor front advancement by NSGA-J, and was abandoned.

## 5.3 Experimental Results II: Convergence

**Goal.** What does the convergence behavior of NSGA-J look like?
**Results and Discussion.** Figure 3 shows, over all generations, individuals in the objective space with outliers removed for large values of both objectives. Outliers were removed as follows: the third percentile of largest $f_1(\mathbf{x})$-values and the fifth percentile of largest $f_2(\mathbf{x})$-values were removed.

Individuals are color-coded, see Figure 3(c), based on when they entered the population. The individuals are plotted in this way: If individuals from several generations occupy the same spot, the color for the earliest generation is shown. Crosses represent infeasible individuals, whilst dots represent the feasible ones.[10] Due to the strict constraint handling method of NSGA-J, infeasible individuals are only in early generations. Every experiment shows a large diversity of solutions in the early generations, and NSGA-J shows a relatively large objective space diversity even in the final generations. This follows the two design principles of NSGA-II, namely converge towards the Pareto front and diversity maintenance.

The experiments suggest several interesting points about NSGA-J. One interesting behavior relates to constraint handling. NSGA-J uses the constraint handling method presented in Definition 3.5, which always prefers feasible solutions. This scheme makes it difficult to navigate in the infeasible space, thus it is problematic with only infeasible individuals in initial population. We see in Figure 3 that NSGA-J-1 and NSGA-J-2 have both feasible and infeasible individuals in the early (red) populations, while later populations (green, blue, and purple) consist of feasible individuals only.

[10]The difference between crosses and dots may be easiest to appreciate when zooming in using a soft-copy of this paper.

## 5.4 Experimental Results III: Last Generation

**Goal.** What does the last generation look like, in experiments?
**Results and Discussion.** Figure 4 shows the last generation of the experiments described already. Both NSGA-J-1 and NSGA-J-2 show a single final front. A significant point about NSGA-J is the following. NSGA-J was changed relative to NSGA-II by implementing duplicate removal, and thus the population always contained $m = 50$ individuals that differed at least slightly.

Considering both $f_1$ and $f_2$, NSGA-J-2 is a substantially better run than NSGA-J-1. In Figure 4(a), it appears that NSGA-J-1 may undergo some type of premature convergence. However, the underlying reasons for the substantially poorer results for NSGA-J-1 are not well-understood and this calls for future research.

# 6 DISCUSSION AND FUTURE WORK

The problem of designing and optimizing offshore jackets is a complex engineering problem. Our premise in this work is that a computerized method for jacket design and optimization should be working as an assistant to the engineer, using the same software and computing infrastructure. The experience of doing so is what we report on in this paper, where we study the development and experimental validation of a multi-objective GA, NSGA-J. NSGA-J is based on the Non-Dominated Sorting GA (NSGA-II) [4], but adds to it by: (i) using multiple and more complex mutation and crossover operators; (ii) avoiding duplicates in the population; (iii) creating

the initial population from an existing jacket design; and (iv) tailoring fitness and constraint functions to jacket design optimization.

While some design constraints are included in NSGA-J, many more exist. Examples of design constraints that could be added are: legs should be splayed, leg can diameter must equal diameter of leg above, bottom leg sections must widen to increase buoyancy, and so forth. However, as requirements differ between projects, it is difficult to formalize and include all such constraints once and for all. An alternative is to shift focus to the early stage of jacket design projects. During that stage, simple designs (compared to the ones optimized here) are studied to decide on the rough topology of a jacket. In this stage, significantly fewer constraints are needed.

Another area of future work is to expand the scope of analyses performed during jacket design optimization. Examples of these are analyses of transportation, launch, and accidental limit state. In the general case, one would like to perform all such analyses after changing a design, as adjustments aimed at improving one aspect of a design might decrease its performance in other areas. Adding more analyses would increase the compute cost and time considerably, but on the other hand our use of the OneCompute cloud makes this addition realistic.

As observed by Li et al. [13], "[m]uch attention has been given to maintaining solution diversity in the objective space. However, little attention has been given to how to maintain solution diversity in the decision space." Our experimental results, including the apparent premature convergence of run NSGA-J-1 compared to run NSGA-J-2, underline the importance of this observation. Thus, we encourage future research at the intersection of multimodal and multiobjective optimization in the decision space or in both the decision and objective spaces [13, 17].

A final area of future research would be to adjust the genotype in order to extend the search space. Currently, topology and shaping optimization rely on jacket designs having parameters that influence these domains. An example of such a parameter is the base width which determines the angles of legs, and consequently the lengths of horizontal beams. This is an example of the current method's tight coupling between the genotype and phenotype representation. Making this coupling less tight and more generative would make it possible to design jackets that are substantially different from the original design. On the other hand, it may put more pressure on ensuring that a more "creative" design adheres to design constraints, which is already a complex issue.

Overall, we believe that this study contains important lessons learned for the application of MOEAs, especially multi-objective GAs, in real-world structural design optimization problems.

## ACKNOWLEDGMENTS

The authors wish to thank Bjørn Gunnar Eilertsen, Frode Strand, and Sigmund Mongstad Hope. They helped with introductions to Kvaerner's systems and in many other ways. The authors would also like to thank the anonymous referees for their helpful comments.

## REFERENCES

[1] D. Antipov, M. Buzdalov, and B. Doerr. 2020. First Steps Towards a Runtime Analysis When Starting with a Good Solution. In *Parallel Problem Solving from Nature (PPSN XVI)*. Leiden, The Netherlands, 560–573.

[2] J. Burak. 2020. *Multi-objective Genetic Algorithm for Engineering Design Optimization.* Master's thesis. NTNU, Trondheim, Norway.

[3] W. De Vries, N. K. Vemula, T. Passon, P .and Fischer, D. Kaufer, D. Matha, B. Schmidt, and F. Vorpahl. 2011. *Support Structure Concepts for Deep Water Sites.* Technical Report. Delft University of Technology, Delft, The Netherlands. UpWind Final Report WP4.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[5] A. E: Eiben and J. E Smith. 2015. *Introduction to Evolutionary Computing*. Springer.

[6] M. A. El-Reedy. 2019. *Offshore Structures: Design, Construction and Maintenance.* Gulf Professional Publishing.

[7] D. Guirguis, N. Aulig, R. Picelli, B. Zhu, Y. Zhou, W. Vicente, F. Iorio, M. Olhofer, W. Matusik, C. A. Coello Coello, and K. Saitou. 2020. Evolutionary Black-Box Topology Optimization: Challenges and Promises. *IEEE Transactions on Evolutionary Computation* 24, 4 (2020), 613–633. https://doi.org/10.1109/TEVC.2019.2954411

[8] J. Häafele and R. Rolfes. 2016. Approaching the Ideal Design of Jacket Substructures for Offshore Wind Turbines with a Particle Swarm Optimization Algorithm. In *Proc. 26th International Ocean and Polar Engineering Conference.* Rhodes, Greece.

[9] J. Häfele. 2019. *A Numerically Efficient and Holistic Approach to Design Optimization of Offshore Wind Turbine Jacket Substructures.* Ph.D. Dissertation. Universität Hannover.

[10] R. Kicinger, T. Arciszewski, and K. De Jong. 2005. Evolutionary Computation and Structural Design: A Survey of the State-of-the-Art. *Computers & Structures* 83, 23-24 (2005), 1943–1978.

[11] E. Y. Kling, B. Ferri, B. Bjørhovd, F. Strand, and S. M Hope. 2019. Automated Jacket Design. In *Proc. 29th International Offshore and Polar Engineering Conference.*

[12] T. Kunakote and S. Bureerat. 2011. Multi-Objective Topology Optimization using Evolutionary Algorithms. *Engineering Optimization* 43, 5 (2011), 541–557.

[13] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht. 2017. Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications. *IEEE Transactions on Evolutionary Computation* 21, 4 (2017), 518–538. https://doi.org/10.1109/TEVC.2016.2638437

[14] M. López-Ibáñez, J. Knowles, and M. Laumanns. 2011. On Sequential Online Archiving of Objective Vectors. In *Evolutionary Multi-Criterion Optimization.* 46–60.

[15] Y. Lou, S. Y. Yuen, and G. Chen. 2021. Non-Revisiting Stochastic Search Revisited: Results, Perspectives, and Future Directions. *Swarm and Evolutionary Computation* 61 (2021), 100828. https://doi.org/10.1016/j.swevo.2020.100828

[16] J. H. Martens, D. Zwick, and M. Muskulus. 2015. Topology Optimization of a Jacket Structure for an Offshore Wind Turbine with a Genetic Algorithm. In *11th World Congress on Structural and Multidisciplinary Optimization.* Sydney, Australia.

[17] O. J. Mengshoel, S. F. Galán, and A. De Dios. 2014. Adaptive Generalized Crowding for Genetic Algorithms. *Information Sciences* 258 (2014), 140—-159.

[18] O. J. Mengshoel, D. C. Wilkins, and D. Roth. 2011. Initialization and Restart in Stochastic Local Search: Computing a Most Probable Explanation in Bayesian Networks. *IEEE TKDE* 23, 2 (2011), 235–247.

[19] N. Noilublao and S. Bureerat. 2011. Simultaneous Topology, Shape and Sizing Optimisation of a Three-Dimensional Slender Truss Tower using Multiobjective Evolutionary Algorithms. *Computers & Structures* 89, 23-24 (2011), 2531–2538.

[20] Great Britain. Dept. of Energy. 1990. *Offshore installations: Guidance on design, construction and certification.* HMSO.

[21] L. B. Pasamontes, F. G. Torres, D. Zwick, S. Schafhirt, and M. Muskulus. 2014. Support Structure Optimization for Offshore Wind Turbines with a Genetic Algorithm. In *Proc. 33rd International Conference on Ocean, Offshore and Arctic Engineering.*

[22] A. Schafhirt, D. Zwick, and M. Muskulus. 2014. Reanalysis of Jacket Support Structure for Computer-Aided Optimization of Offshore Wind Turbines with a Genetic Algorithm. In *Proc. 24th International Ocean and Polar Engineering Conference.*

[23] N. Srinivas and K. Deb. 1995. Multiobjective Optimization using NSGA. *Evolutionary Computing* 2, 3 (1995), 221–248.

[24] M. Stolpe and K. Sandal. 2018. Structural Optimization with Several Discrete Design Variables per Part by Outer Approximation. *Structural and Multidisciplinary Optimization* 57, 5 (2018), 2061–2073.

[25] X. Tang, D. H. Bassir, and W. Zhang. 2011. Shape, Sizing Optimization and Material Selection Based on Mixed Variables and Genetic Algorithm. *Optimization and Engineering* 12, 1-2 (2011), 111–128.

[26] X. Tian, Q. Wang, G. Liu, Y. Liu, Y. Xie, and W. Deng. 2019. Topology Optimization Design for Offshore Platform Jacket Structure. *Applied Ocean Research* 84 (2019), 38–50.

[27] S. Y. Yuen and C. K. Chow. 2007. A Non-Revisiting Genetic Algorithm. In *IEEE Congress on Evolutionary Computation.* 4583–4590.

[28] G. R. Zavala, A. J. Nebro, F. Luna, and C. A. C. Coello. 2014. A Survey of Multi-Objective Metaheuristics Applied to Structural Optimization. *Structural and Multidisciplinary Optimization* 49, 4 (2014), 537–558.