Modular optimization framework for mixed expensive and inexpensive real-world problems

Winter, R. de; Bäck, T.H.W.; Stein, N. van

Modular Optimization Framework for Mixed Expensive and Inexpensive Real-World Problems

Roy de Winter
r.de.winter@liacs.leidenuniv.nl
Leiden Institute Of Advanced Computer Science
Leiden, Zuid Holland, The Netherlands

Thomas Bäck
t.h.w.baeck@liacs.leidenuniv.nl
Leiden Institute for Advanced Computer Science
Leiden, Zuid Holland, The Netherlands

Niki van Stein
n.van.stein@liacs.leidenuniv.nl
Leiden Institute Of Advanced Computer Science
Leiden, Zuid Holland, The Netherlands

ABSTRACT

A modular optimization framework is proposed that can deal with different problem characteristics encountered in real-world design problems. The common denominator of problems considered in this work is the computationally demanding objective function used to evaluate the designs, while some of the constraint functions can be inexpensive. The framework uses radial basis functions as surrogates for the computationally expensive objectives while inexpensive functions can directly be used for searching promising solutions on the surrogates. The framework is adjustable for the number of continuous design parameters, the number of constraints, the number of solutions that can be evaluated in parallel, and different strategies can be selected on how to deal with the computationally inexpensive functions. We propose guidelines on how to set up the modular framework depending on the problem characteristics one has to deal with. To verify the working of the optimization framework, two different real-world ship design optimization problems have been optimized with different characteristics such as the number of constraints and how expensive they are. The results show that it is beneficial to directly exploit the inexpensive functions when possible and that parameterization of the problem is the most important step of the process.

CCS CONCEPTS

• **Computing methodologies** → *Simulation evaluation*; *Parallel algorithms*; • **Applied computing** → **Computer-aided design**.

KEYWORDS

Optimization, Computationally Expensive, Constraint Optimization

ACM Reference Format:

Roy de Winter, Thomas Bäck, and Niki van Stein. 2024. Modular Optimization Framework for Mixed Expensive and Inexpensive Real-World Problems. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638530.3664133

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0495-6/24/07.
https://doi.org/10.1145/3638530.3664133

1 INTRODUCTION

Real-world optimization problems often are computationally expensive and are subject to constraints [4, 30]. However, some problems have a computationally expensive objective and inexpensive constraints, while for other problems some or all of the constraints are also computationally expensive. Evaluation of these problems is usually done with commercial simulation software that requires a license. In cases where multiple licenses are available, multiple solutions can be evaluated in parallel. Selecting a unique well-performing optimization algorithm per optimization problem can be a labor-intensive task as they are often programmed in different languages, the communication between the algorithm and the simulation software needs to be set up, and skepticism towards a new algorithm must be overcome before it will be used in practice. Therefore, in this paper, a modular adaptive global optimization framework (MAGOF) is introduced that can deal with different problem characteristics encountered in real-world optimization problems. The common denominator of problems considered in this work is the computationally demanding objective function, while the constraints function(s) can be expensive or inexpensive, and multiple solutions can be evaluated in parallel.

To illustrate the performance of the MAGOF framework, four different configurations are tested on the G-problem test suite [24]. After the strengths have been identified the MAGOF framework is used to optimize two real-world ship design optimization problems.

A formal problem notation is given in Equation 1.

$$\begin{aligned} &\text{minimize: } \mathbf{f} : \Omega \rightarrow \mathbb{R}\,,\ f(\mathbf{x}) \\ &\text{subject to: } g_i(\mathbf{x}) \leq 0\ \forall i \in \{1,\ldots,m\} \\ &\mathbf{x} \in \Omega \subset \mathbb{R}^d. \end{aligned} \tag{1}$$

Here m is the number of constraints, d is the number of parameters, f is the objective function, g_i is the i^{th} constraint function, and $\mathbf{x}$ the vector of decision parameters.

The remainder of this paper is organized as follows: In Section 2 the related work is presented. In Section 3 the modular framework is described in detail. In Section 4 the experiments and two real-world ship design problems are described. The results on the benchmark functions and the two problems are presented in Section 5. Finally conclusions and future work directions are given in Section 6.

2 RELATED WORK

Developing optimization algorithms and their variants in isolation with limited analysis of module settings and hyper-parameter configurations, limits the applicability of these algorithms. Modular

approaches to optimization algorithm development mitigate these limitations partly by enabling comparisons across a wide array of modifications to a core algorithm's implementation and linking the performance of these modifications to different problem characteristics [31]. Such modularization is achieved by providing multiple options for each component (or module) of the algorithm, allowing for a fully independent selection of options per module. The concept of modular algorithms, exemplified by frameworks such as paradisEO [15] and MLO [36] has been applied across various algorithm families [12, 32], including multi-objective optimization algorithms [16, 25]. The framework proposed in our study leverages this modular approach, focusing on different problem characteristics of real-world expensive constraint optimization tasks and introducing adaptability in managing different problem scales and constraints. This approach offers a new perspective on optimization for complex real-world design challenges.

The modular framework in this paper is built on several ideas from different earlier works. Since this work deals with constraint single objective problems with the common denominator that at least one of the constraint and/or objective functions are computationally expensive, inspiration is taken from surrogate-assisted optimization algorithms and constraint optimization algorithms.

2.1 Surrogate Assisted Optimization

When solving computationally expensive black-box optimization problems, often Bayesian optimization is chosen as the method to find the optimal solutions. A popular and common Bayesian optimization algorithm for single objective optimization problems is the Efficient Global Optimization (EGO) algorithm [23]. Bayesian optimization consists of 4 steps:

(1) Bayesian optimization starts with a Design of Experiments (DoE). The solutions from the DoE are evaluated with the objective and constraint functions and stored in an archive.
(2) Surrogate models are fitted with all data from the archive.
(3) With an acquisition function the promising solution(s) are selected based on the surrogate(s) predictions. The acquisition function is optimized with an optimization algorithm.
(4) The new solutions are evaluated with the objective and constraint functions and added to the archive.

Finally, the algorithm terminates if a stopping criterion is met, otherwise the algorithm goes back to step 2. The elements that are varied the most in Bayesian optimization are the surrogate choice, and the acquisition function.

Well-known surrogate modeling methods are Gaussian Process regression (GP) [18] (also known as Kriging) and Radial Basis Function (RBF) surrogates [4, 7]. In this work only RBFs are considered since they are computationally less expensive but very similar [3]. The surrogates are models that approximate the relationship between input and output variables and this way learn from the already evaluated solutions. After the relationship is learned the acquisition function is optimized [34]. A set of existing acquisition functions that already exist are for example the Expected Improvement (EI) acquisition function [23] and the Generalized Expected Improvement (GEI) acquisition function [26]. The Expected Improvement acquisition score uses the predicted value of the surrogate and then subtracts the standard deviation or uncertainty which is higher in regions where only a few solutions have been evaluated. This acquisition function therefore automatically prefers solutions in unexplored regions with promising objective scores and this way balances exploration and exploitation. The generalized expected improvement acquisition function balances exploration and exploitation differently by taking the standard deviation (or uncertainty) and raising it to the power g_{EI}. The value of g_{EI} determines the balance between exploration and exploitation. A high value of g_{EI} increases exploration while a small value of g_{EI} increases exploitation. These acquisition functions, however, only work for a surrogate that can also provide an uncertainty quantification. GP by default has this built-in, RBFs by default do not. Fortunately, an uncertainty quantification method has recently been introduced for RBFs [3]. This allows the use of fast RBFs in combination with an acquisition function that requires an uncertainty quantification method to work.

2.2 Constraint Optimization

Constraints can be dealt with in many different ways [11]. Common constraint handling approaches [2] are (1) feasible solution preference during the search, (2) unconstrained optimization with a penalty function for constraint violations, (3) gradient-based repair mechanisms, (4) constraint separation where the objective and constraint satisfaction is seen as two different optimization problems, (5) and model-assisted approaches that use machine learning or surrogates to model the constraints and this way only try to propose feasible solutions. The last approach is the most interesting one as this can be used for computationally expensive problems. Examples of algorithms that already apply such a constraint handling methodology are the single objective SACOBRA algorithm [4] and the multi-objective SAMO-COBRA algorithm [13]. These algorithms model the constraints with RBFs and then optimize the acquisition function while at the same time, the constraint surrogates are used to guide the solution to a feasible region.

Besides using surrogates for constraints, in some other algorithms, the constraints are computationally inexpensive and can be used directly in the search for promising and feasible solutions. One such algorithm is the Mode Pursuing Sampling (MPS) algorithm [33] which models the objective function with quadratic regression. It uses the constraints functions directly and then proposes solutions around the estimated feasible optimum. By iteratively updating the regression model it tries to find the global optimal feasible solution. Another algorithm that uses the constraints directly is the IC-SA-NSGA-II algorithm [5]. This algorithm is a multi-objective variant that uses the constraints directly to check if the solutions proposed by NSGA-II [14] are feasible according to the inexpensive constraints, and to check the dominance of the solution the algorithm uses RBFs as a surrogate for the objective functions.

3 MODULAR OPTIMIZATION FRAMEWORK

The pseudocode of the Modular Adaptive Global Optimization Framework (MAGOF) is presented in Algorithm 1. The evaluation method and strategy are described in more detail in Algorithm 2 and Section 3.3. The input, the overall explanation of the pseudocode, and the working of the framework are described in more detail in the following subsections.

3.1 Input parameters

The input arguments for the modular framework are: (1) **Objective function** $f(\mathbf{x})$ that is to be minimized. The objective function is defined by the user as expensive $f_e(\mathbf{x})$, or inexpensive $f_c(\mathbf{x})$. (2) **Constraint function(s)** $\mathbf{g}(\mathbf{x})$ that consist of m separate constraint functions, where $m \geq 1$. The constraint function(s) are either defined by the user as expensive $\mathbf{g}_e(\mathbf{x})$, or inexpensive $\mathbf{g}_c(\mathbf{x})$. The constraint functions return the constraint violation, meaning that constraint values $g(\mathbf{x}) \leq 0$ are defined as feasible. (3) **Input space** $\Omega \subset \mathbb{R}^d$ that is limited by the lower and the upper boundary $[\mathbf{x}_{lb}, \mathbf{x}_{ub}]$. (4) **Initial sample strategy and sample size** N_{init} and DoE define how many samples are evaluated in the design of experiments. This should at least be larger than $d+1$. (5) **Evaluation budget** N_{max} defines how many expensive function evaluations are allowed to be evaluated. (6) **RBF strategy domain**, $\Phi = \{Cubic, Gaussian, Multiquadric, InverseQuadratic, InverseMultiquadric, ThinPlateSpline\} \times \{\text{Plog}, standardized\}$. The RBF strategy domain defines the different surrogates that are used in every iteration to model the computationally expensive functions. (7) **Parallelism** p, is the number of solutions that can be evaluated in parallel. Parallelism is not required as p can also be 1. (8) **Acquisition function** α that is used to find promising solutions. The acquisition function uses the surrogates or the inexpensive functions directly to find p promising solutions for evaluation. (9) **Constraints first indicator** that defines if the constraints should all be satisfied before the objective function can be evaluated.

3.2 Design of Experiments

The framework in Algorithm 1 starts in line 2 by creating a Design of Experiments (DoE). The size and the strategy for the DoE can be chosen by the user and can be random, a latin hypercube sample, solutions on the boundaries, or a Halton sample. Each of these sample strategies has its strengths, however, an empirical comparison by Bossek et al. [6] showed that an as small as possible initial Halton sample [22] is in most cases the most efficient strategy. It is also possible to start with an initial sample that is already evaluated.

3.3 Evaluation of the solutions

On lines 3 and 14 of Algorithm 1 the solutions that are proposed by the DoE, or after optimizing the acquisition function, are evaluated as described in the evaluate Algorithm 2. The approach is dependent on the inexpensiveness of the constraint and objective functions. There are 3 levels of expensiveness. (1) the function can be evaluated almost instantly and can therefore be evaluated millions of times (it must at least be faster than fitting and interpolating an RBF surrogate model). (2) the function requires a little bit of evaluation time and can therefore not be evaluated numerous times and evaluating them millions of times is too costly. Function evaluations of level 2 for example require a few seconds up to a few minutes and are significantly less costly compared to the most expensive evaluations. (3) the function is computationally expensive and the evaluation budget is very limited e.g. computational fluid dynamic simulations or finite element analysis that can take up to hours on a cluster to evaluate.

The inexpensive functions level 1 are used directly in the optimization algorithm, see Section 3.5. If the constraints are inexpensive level 2, they are evaluated first before the computationally expensive functions. Only if the constraints are satisfied, the expensive functions (level 3) are evaluated. If the constraints are violated, a null is stored instead of the expensive outcome. This way, in the next iteration of MAGOF the RBF surrogates for the expensive functions remain the same as in the previous iteration while for the inexpensive functions level 2 the RBF surrogates are updated.

3.4 Radial Basis Functions

In every iteration of MAGOF, surrogates are fitted to approximate the constraint and objective functions (line 9 of Algorithm 1). However, there are many kernel options and scaling techniques available when fitting RBF surrogates and each option can be good for different scenarios. Therefore, RBF surrogates are fitted with the following kernels: *Cubic*, *Gaussian*, *Multiquadric*, *InverseQuadratic*, *InverseMultiquadric*, *ThinPlateSpline* and two different scaling strategies are used to scale the constraint and objective values. The standardization method is used so that the uncertainty quantification method can be used for the RBFs. The Plog transformation from Equation 2 is selected so that the RBFs can better model steep slopes. For each combination of these kernels and transformation methods, a surrogate is fitted which results in a total 12 RBF surrogate models per expensive function. In every iteration, the RBF strategy with the smallest approximation error is selected (line 5 and 15 of Algorithm 1) and the RBF approximation errors are stored.

$$\text{Plog}(y) = \begin{cases} +\ln(1+y), & \text{if} \quad y \geq 0 \\ -\ln(1-y), & \text{if} \quad y < 0 \end{cases} \tag{2}$$

3.5 Acquisition Function Optimization

The acquisition functions integrated into the framework are: the expected improvement acquisition function [23], the generalized expected improvement acquisition function [26] for parallel evaluations when $p > 1$, and the purely exploitative acquisition function that predicts the objective value with the RBF surrogate without uncertainty. This acquisition function is optimized with the COBYLA algorithm [27]. COBYLA is a single objective optimization algorithm that optimizes an optimization problem with constraints by linearly approximating the acquisition function and the most violated constraint in a small trust region. COBYLA finds the most promising solution in this trust region, then checks the constraint and objective values, and iteratively adjusts the trust region until the trust region is so small and the local optimum is found.

For the functions with expensiveness levels 2 and 3, COBYLA is instructed to use the surrogate models when optimizing the acquisition function. The inexpensive functions (level 1) that can be calculated instantly, are directly used by COBYLA when optimizing the acquisition function. The usage of the inexpensive functions is beneficial because they don't make approximation errors that surrogate models make. Note that during the optimization of the acquisition function, the inexpensive functions (or surrogate for expensive functions) are evaluated many times.

Because COBYLA is a local optimizer, the COBYLA algorithm starts searching from multiple random locations. This makes it more likely that the global optimum is found.

Algorithm 1: MAGOF.
Input: Objective function $f(\mathbf{x})$, that can be computationally expensive $f_e(\mathbf{x})$ or computationally inexpensive $f_c(\mathbf{x})$, constraint function(s) $\mathbf{g}(\mathbf{x})$, split where required into expensive constraint function(s) $\mathbf{g}_e(\mathbf{x})$, computationally inexpensive constraint function(s) $\mathbf{g}_c(\mathbf{x})$, decision parameters' lower and upper bounds $[\mathbf{x_{lb}}, \mathbf{x_{ub}}] \subset \mathbb{R}^d$, sampling strategy DoE, number of initial samples N_{init}, maximum evaluation budget N_{max}, RBF strategy domain consisting of 12 RBF strategies $\Phi = \{Cubic, Gaussian, Multiquadric, InverseQuadratic, InverseMultiquadric, ThinPlateSpline\} \times \{\text{Plog}, standardized\}$, number of solutions that can be evaluated in parallel p, acquisition function α, constraint first indicator c_{first}.
Output: Evaluated solutions.

Function MAGOF(f, $\mathbf{g}$, $\mathbf{x}_{lb}$, $\mathbf{x}_{ub}$,*DoE*, N_{init}, N_{max}, Φ, p, α, c_{first}):
  $\mathbf{x}^* \leftarrow \{\mathbf{x}_1, \cdots, \mathbf{x}_{N_{init}}\} \leftarrow$ DoE $(\mathbf{x}_{lb}, \mathbf{x}_{ub}, N_{init})$ ▷ Generate initial Design of Experiments, $\mathbf{X} \in \mathbb{R}^{d \times N_{init}}$
  $\mathbf{F}, \mathbf{G}, \mathbf{X} \leftarrow$ Evaluate$(\mathbf{x}^*, f, g, p, c_{first}, N_{init}, \mathbf{F} = [], \mathbf{G} = [], \mathbf{X} = [])$ ▷ Evaluate initial sample and initialize archives $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{X}$
  $\mathbf{h} \leftarrow \{f_e \cup \mathbf{g}_e\}$ ▷ Union of expensive objective and constraint functions
  $\varphi^* \leftarrow \left(\varphi_1, \ldots, \varphi_{|\mathbf{h}|}\right) \leftarrow (Cubic, standardized)^{|\mathbf{h}|}$ ▷ Initialize RBF strategy for all expensive functions, $\varphi^* \in \Phi$
  $\mathbf{E}_{i,j} \leftarrow 0 \; \forall (i,j) \in \mathbf{h} \times \Phi$ ▷ Initialize RBF approximation errors for each possible RBF configuration(Φ) for all expensive functions($\mathbf{h}$)
  $j \leftarrow N_{init}$ ▷ Initialize expensive evaluation counter
  while $j < N_{max}$ **do**
    $\mathbf{S}^{\Phi} \leftarrow \left(S^{\Phi_1}_{h_1}, \ldots, S^{\Phi_{12}}_{h_{|\mathbf{h}|}}\right) \leftarrow \{\text{FitRBF}(\mathbf{X}, h, \Phi, \mathbf{x_{lb}}, \mathbf{x_{ub}}) \mid \forall h \in \mathbf{h}\}$ ▷ Fit RBFs using all strategies(Φ) for all expensive functions($\mathbf{h}$)
    $\mathbf{S}^{\varphi^*} \leftarrow \left(S^*_1, \ldots, S^*_{|\mathbf{h}|}\right)$ ▷ Select best RBF strategy based on line 5 or 15
    $\mathbf{x}^*_1, \ldots, \mathbf{x}^*_p \leftarrow$ Max$(\alpha, p, \mathbf{S}^{\varphi^*}, f_c, \mathbf{g}_c)$ ▷ Get p new solutions based on acquisition function α, use cheap functions f_c and $\mathbf{g}_c$ directly
    $j \leftarrow j + p$ ▷ Increase iteration counter to new matrix sizes
    $\mathbf{X} \leftarrow \left[\mathbf{X}, \; \mathbf{x}^*_1, \; \ldots, \; \mathbf{x}^*_p\right]$ ▷ Add p new solution vectors, $\mathbf{X} \in \mathbb{R}^{d \times j}$
    $\mathbf{F}, \mathbf{G}, \mathbf{X} \leftarrow$ Evaluate$(\mathbf{x}^*, f, g, p, c_{first}, j, \mathbf{F}, \mathbf{G}, \mathbf{X})$ ▷ Evaluate new solutions
    $\varphi^*, \mathbf{E} \leftarrow$SelectBestRBFStrategy$(\mathbf{E}, S^{\Phi}, \mathbf{F}, \mathbf{G}, \mathbf{X})$ ▷ Update RBF approximation errors $\mathbf{E}$, and new best RBF configuraiton φ^*
  end
return (F, G, X)

Algorithm 2: Evaluate.
Input: Solutions $\mathbf{x}^*$ to be evaluated, Objective function $f(\mathbf{x})$, that can be computationally expensive $f_e(\mathbf{x})$ or computationally inexpensive $f_c(\mathbf{x})$, constraint function(s) $\mathbf{g}(\mathbf{x})$, split where required into expensive constraint function(s) $\mathbf{g}_e(\mathbf{x})$, computationally inexpensive constraint function(s) $\mathbf{g}_c(\mathbf{x})$, number of solutions that can be evaluated in parallel p, constraint first indicator c_{first}, objective values of evaluated solutions $\mathbf{F}$, constraint values of evaluated solutions $\mathbf{G}$, evaluated solutions $\mathbf{X}$.
Output: Evaluated solutions.

Function Evaluate($\mathbf{x}^*$, f, $\mathbf{g}$, p, c_{first}, j, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{X}$):
  $\mathbf{G} \leftarrow \left[\mathbf{G}, \; \mathbf{g}_c(\mathbf{x}^*_1), \; \ldots, \; \mathbf{g}_c(\mathbf{x}^*_p)\right]$ ▷ Add vectors of evaluated inexpensive constraints, $\mathbf{G} \in \mathbb{R}^{m \times j}$
  if not c_{first} **then**
    $\mathbf{F} \leftarrow \left[\mathbf{F}, \; f_e(\mathbf{x}^*_1), \; \ldots, \; f_e(\mathbf{x}^*_p)\right]$ ▷ If constraints do not need to be satisfied first, add vectors of evaluated objectives, $\mathbf{F} \in \mathbb{R}^j$
    $\mathbf{G} \leftarrow \left[\mathbf{G}, \; \mathbf{g}_e(\mathbf{x}^*_1), \; \ldots, \; \mathbf{g}_e(\mathbf{x}^*_p)\right]$ ▷ Add vectors of evaluated expensive constraints, $\mathbf{G} \in \mathbb{R}^{m \times j}$
  else
    for $\mathbf{x}_i \in \{\mathbf{x}^*_1, \; \ldots, \; \mathbf{x}^*_p\}$ **do**
      if $\mathbf{g}_c(\mathbf{x}_i) \leq 0$ **then**
        $\mathbf{F} \leftarrow [\mathbf{F}, \; f_e(\mathbf{x}_i)]$ ▷ If cheap constraints need to be satisfied first, only add objective value of feasible solutions.
        $\mathbf{G} \leftarrow [\mathbf{G}, \; \mathbf{g}_e(\mathbf{x}_i)]$ ▷ If cheap constraints need to be satisfied first, only add constraint value of feasible solutions.
      else
        $\mathbf{F} \leftarrow [\mathbf{F}, \; null]$ ▷ If constraints are violated, don't evaluate and add null
        $\mathbf{G} \leftarrow [\mathbf{G}, \; null]$ ▷ If constraints are violated, don't evaluate and add null
    end
return (F, G, X)

4 EXPERIMENTS

We conduct two types of experiments. The first experiment focuses on the performance of MAGOF with its different options on the G-problem test suite [17, 24]. In the second experiment, two real-world ship design optimization problems are optimized. In the first real-world problem, a bulbous bow of an existing container ship is optimized for minimum power requirements in different operating conditions. The second real-world optimization problem is a resistance minimization study conducted for a passenger vessel.

4.1 G-Problem experimental setup

The G-Problems (G1 to G11 from [17, 24]) are selected as an artificially created benchmark suite to validate the performance of MAGOF. A Python implementation of the G-Problems is taken from the CEC 2006 Special Session on Constrained Real-Parameter Optimization [19]. The G-problems considered have between 1 and 9 constraints and between 2 and 20 decision parameters, and all are to be minimized. The optimal solutions are known for all G-problems, some problems have active constraints at the optimum, while other optima are somewhere in the feasible region. The feasibility ratio of the G-Problems varies between less than 1% feasible and 99% feasible per test problem. More details regarding the G-Problems can be found in e.g. [4, 17, 24]. Evaluation of the constraint and the objective functions of the G-problems are computationally inexpensive. However, for the experiments, it is assumed that the objectives are computationally expensive to evaluate.

To test the functionality of the mixed expensiveness, four different configurations of MAGOF are tested with different infill criteria and different inexpensive function handling techniques.

1 Traditional The "traditional" configuration uses MAGOF without any special treatment and/or separation of expensive versus inexpensive functions. The objective and constraint methods are considered equally expensive which in MAGOF means that in every iteration the RBFs are fitted for the constraints and objective, the best RBF strategy is selected, and then the default acquisition function is optimized. The resulting solution is computed and evaluated with the constraints and the objectives.

2 Constraints First: The "constraints first" configuration of MAGOF utilizes the adjustment in the evaluation strategy as presented in Algorithm 2. In this configuration it is assumed that the constraints evaluations are computationally way less expensive compared to the objective evaluation. In every iteration, the RBFs are fitted, the best RBF strategy is selected, and the default acquisition function is optimized using the surrogates. The solution that is proposed is now evaluated first on the constraints. In case any of the constraints are violated, the objective function is not evaluated and the next iteration starts. In case the constraints apply the objective function is evaluated.

3 Constraints Integrated: The "constraint integrated" configuration in MAGOF is not conventional as instead of fitting RBFs for the constraints, the constraints are directly used when optimizing the acquisition function because it is assumed that the constraint function evaluations are computationally cheaper than fitting an RBF and making predictions with RBFs. The RBFs are now only used to model the objective function since the objective function evaluation is assumed to remain computationally expensive.

4 Parallel: The "parallel" configuration of MAGOF does not assume inexpensive constraints or objectives and therefore uses RBF models to model the assumed expensive constraint and objective functions. After the RBF models are fitted, the best RBF approximation is selected, and the generalized expected improvement acquisition function is optimized. The hyperparameter (g_{EI}) of the acquisition function is set in such a way that one solution proposed by the algorithm is purely exploitative ($g_{EI} = 0$), one is explorative and would be most similar to solutions proposed by the expected improvement acquisition function ($g_{EI} = 6$), and one solution is a balance between exploitative and explorative ($g_{EI} = 3$). This way, the generalized expected improvement acquisition function can be used to propose 3 different solutions. After the solutions are proposed, they are evaluated in parallel with both the objective and constraint functions.

All configurations start with an as small as possible initial Halton sample as a DoE. After the DoE the first 3 configurations are allowed to do a total of $300 - |\text{DoE}|$ iterations for the non-parallel configurations. The configuration that proposes 3 solutions per iteration was allowed to do an additional 100 iterations. This way each algorithm configuration has in theory the possibility to do 300 objective function evaluations. Note that the configuration with constraint first, does not necessarily use all these 300 objective evaluations since in the 300 iterations, this configuration also sometimes proposes infeasible solutions. The constraints integrated configuration uses a lot more constraint function evaluations since when optimizing the acquisition function, the constraints are evaluated many more times.

4.2 Bulb Optimization Problem

The first real-world ship design optimization problem is a refit of an existing container vessel with a container capacity of around 10 000 standardized containers. Due to increasing fuel prices and more strict emission regulations, this vessel must reduce its operational speed since sailing at the design speed is costly and emits too much greenhouse gas. To sail efficiently at the different operational conditions, the vessel required a bulbous bow (in short bulb) refit. This bulb refit is done by cutting the current bulb off and welding a new optimally shaped bulb back in place. However, optimizing the bulb for all different loading conditions and speeds would lead to four different optimal bulb shapes. Therefore, the goal of this study is to find the bulb that performs well on (a weighted) average on all four conditions.

MAGOF is used to optimize the bulb design. In this experiment, the optimization process used an as small as possible initial Halton sample during the design of experiments, in every consecutive iteration the best RBF transformation strategy and kernel are chosen, and a purely exploiting infill criteria is used to find promising solutions in the adaptive sampling steps. If no improvements have been found for three consecutive iterations, the algorithm switches to an exploring infill criterion.

The design freedom is defined by where the current bulb is cut off. The main cutting line is located at the design draft and 24 meters aft from the foremost section of the bulb. The remainder of the hull should remain intact and can not be modified. The new bulb of the vessel is designed and parameterized in Rhino with the rapid

hull modeling technique [20]. The bulb can be modified by varying 6 parameters that define the bulb length, height, and width at two locations, and by changing the overall contour lines of the bulb. The objective is defined by a weighted sum of the required power that is needed to reach four different speeds with different loading conditions and therefore different drafts. The weighted sum of the four different conditions is chosen as the objective function so that this bulb will perform good in all four conditions. However, this did mean that the complete hull had to be evaluated with Reynolds Averaged Navier-Stokes Equation (RANSE) analysis in the Star-CCM+ software [28] for all four conditions. The mesh for the vessel consists of between 1.7 and 3.7 million cells depending on the speed and draft of the calculation. For this optimization challenge the new algorithm is coupled with a high-performance computer on the Microsoft Azure cloud with 120 CPUs. One RANSE calculation in Star-CCM+ of the complete hull required approximately 20 minutes, since there are 4 different conditions one iteration required 1 hour and 20 minutes of computation time on the high-performance machine.

4.3 Roll-on/Roll-off Ferry Hull Optimization

After an operational data analysis study and a feasibility study for alternative fuels, an initial non-optimized design is made by C-Job naval Architects for a fully battery-powered Roll-on/-Roll-off (Ro-Ro) ferry with a capacity of 800 passengers [8, 9]. This vessel is designed to sail between the Saronic islands and the port of Piraeus. A render of this C-Job design is presented in Figure 1.

Figure 1: Render of Saronic Roll-on/Roll-off ferry designed and created by C-Job Naval Architects.

As this is a battery-powered design the vessel requires an optimized hull. To accomplish the energy-efficient hull, the hull form is optimized for minimal resistance at design speed. However, three imposed restrictions limit the search space of the to-be-designed hull: (1) the hull above the waterline is only allowed to be modified marginally, (2) the displacement of the new hull design should be equal to or larger than the original hull displacement to be able to carry the passengers, vehicles, parcels, and all equipment, (3) the Longitudinal Centre of Buoyancy (LCB) should remain within 1% of the original hull to keep a good trim, heel, and intact stability without the need to move heavy components around.

With these conditions, two independent experiments are set up. In the first experiment experienced naval architects with hydrodynamic experience manually optimized the hull for minimal resistance at the design speed. In the second completely independent experiment (no communication between the two groups), the vessel is parameterized and optimized with MAGOF. For the experiment with MAGOF the hull is parameterized below the waterline varying the following seven parameters ($d = 7$): (1) transom height, (2) the transom angle, (3) the start of the midship in the longitudinal direction, (4) the shoulder location in the longitudinal direction, (5) the bulb size (can be 0 which results in a design without a bulb), (6) the bulb width, (7) and finally the foreship width.

The design variants are again designed and parameterized in the Rhino software by using the rapid hull modeling technique [20]. After generation the three computationally relatively inexpensive constraints are also calculated in the Rhino software. Finally, the hull is exported to evaluate the computationally expensive objective in the Star-CCM+ software on a High-Performance Computer with 120 cores in the Microsoft Azure cloud. (20 minutes per evaluation). The constraints are computationally relatively inexpensive and require communication between a software packages which totals up to approximately 5 seconds. Because of this they can not be called many times directly in the optimization algorithm. However, the constraints are computationally much cheaper compared to the objective. Therefore, the constraint-first methodology of MAGOF is used by setting the $c_{first} = True$ parameter that make sure that the constraints are satisfied before the objective function is evaluated. Since the exploiting strategy proved to be effective, the predicted value is set as the acquisition function in MAGOF. The most promising solution according to the acquisituion function is evaluated first on the constraints, and only if the design variant is feasible according to the constraints the objective is evaluated. If the constraints are infeasible only the constraint surrogates are updated. This process continues until the algorithm has converged and no significant improvements are found for several consecutive iterations.

5 RESULTS

In Table 1 the results are presented for the G-problem test suite. In this table, the mean smallest objective scores of the feasible solutions are presented after 10 independent runs of MAGOF with the four different options. Besides the mean objective score, the number of required function evaluations is reported that was required to reach the minimum. Please refer to [19] for the complete set of minima for all functions. A red cross (x) indicates that the optimum was not found within 300 evaluations.

Inspection of the results shows that in the majority of the problems using the cheap constraints directly in the optimization algorithm when optimizing the acquisition function is beneficial in terms of execution time and convergence. The other option where the constraints are first evaluated to check for feasibility before the objective function is evaluated also shows better results compared to the conventional approach where the objective is evaluated together with the constraints. The option to propose 3 solutions in parallel with the generalized expected improvement acquisition function does not show good results. It was expected upfront that

Function		Traditional	Constraint First	Constraint Integrated	Parallel P=3
G01	fv	−15.00	−15.00	**−15.00**	−13.54
	fe	28	24	**22**	x
G02	fv	−0.304	−0.304	**−0.383**	−0.304
	fe	x	x	x	x
G03	fv	−0.000	**−0.089**	−0.006	−0.000
	fe	x	x	x	x
G04	fv	−30665	−30665	**−30665**	−30665
	fe	26	19	**11**	111
G05	fv	5126.5	5126.5	**5126.5**	5126.5
	fe	41	12	**10**	x
G06	fv	−6957	**−6959**	**−6959**	−6956
	fe	x	x	x	x
G07	fv	24.306	24.306	**24.306**	35.479
	fe	34	22	**21**	x
G08	fv	**−0.096**	−0.096	−0.096	−0.096
	fe	**13**	30	28	100
G09	fv	680.63	1186.8	**680.64**	1597.8
	fe	240	x	**89**	x
G10	fv	7114.3	7088.6	**7049.3**	9233.1
	fe	x	x	**65**	x
G11	fv	0.7500	**0.7500**	**0.7500**	0.7500
	fe	7	**5**	**5**	12

Table 1: The mean minimum encountered objective score (fv), and the mean objective function evaluations (fe) required to find the optimal value (a x indicates the known optimal value was not reached in 300 function evaluations). Four different MAGOF configurations are compared. The results on the "traditional", "constraint first", "constraints integrated", and the "parallel" configuration are presented. The best combination of fv and fe are marked in bold per G-problem. All G-problems are optimized in 10 independent optimization runs.

when proposing multiple solutions for parallel evaluation (and this way save computation time), the number of iterations of the algorithm could be reduced. However, the number of required iterations and therefore also the number of function evaluations is higher compared to the other approaches.

On the G08 test problem, the traditional approach finds the optimum in less required objective evaluations compared to the other approaches. It is assumed that the reason for this quick convergence is that the information gathered from the evaluated infeasible solutions is of great value for this optimization problem. The information from the infeasible evaluated solutions is missing when the constraint first configuration or constraints integrated configuration is used in MAGOF.

5.1 Bulb Optimization Results

Traditionally, a bulb is designed taking into account the principle of interacting waves [29]. This principle of interacting waves involves shaping the bulb to minimize wave resistance by strategically managing the interaction between waves generated by the bulb and the hull of the ship [10]. The waves generated by the bulb are intended to cancel the other waves out so that in total, there is less wave resistance.

With this principle in mind, the first experiment is conducted. MAGOF used 7 initial Halton Samples, and 38 adaptive sampling steps before terminating the experiment. MAGOF converged to several similar optimal solutions that were all found on the boundary of the design space and the bulb was much smaller than initially expected. Analysis of the results showed that the principle of interacting waves didn't hold because typically an ideal-size bulb works best in one condition at one draft. However, when a bulb is optimized for multiple different drafts and different speeds, there is not one bulb design that generates the perfect wave for all different conditions. Upfront, it was expected by the naval architects, that a large bulb that generates a large wave would have on average the best performance in all conditions. This turned out to be not true as the interaction didn't work in all conditions the same and in some conditions made it worse. Therefore, a second experiment was set up that allowed even smaller bulb sizes.

In the second parameterization setup, the bulb is parameterized by only the height and width parameter. After a second run with a search space that allowed for smaller bulb sizes, the optimization was successful because the algorithm converged to an optimal solution that was not on the boundary of the design space. For the optimal bulb configuration found by the algorithm, some manual fairing is done to create smooth and continuous hull lines without bumps and irregularities at the transition point between the hull and the bulb. After some manual fairing and checking a few extra operating conditions, the second optimization run resulted in a bulb that in total for all conditions considered had 4.8% less required power compared to the original design. The largest reduction was found in the operating conditions that deviated the most from the original design condition which had a significantly higher speed.

In Figure 2 the free surface plot is made of the original bulb (left) and the new bulb (right) in one of the four operating conditions. The color indicates the height of the water at that location (Red shows a crest, while blue shows a trough). As can be concluded from the figure, the waves that are generated with the new proposed bulb are especially in the front less extreme compared to the original hull.

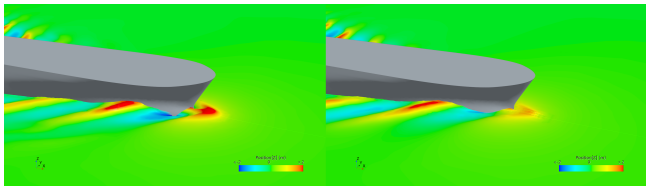

Figure 2: Free surface plot of the original bulb (left) and the new bulb (right). The color indicates the height of the water at that location (Red shows a crest, while blue shows a trough).

5.2 Ferry Hull Optimization Results

After 10 initial samples, the algorithm was allowed to do 110 more evaluations. In the design of experiments, not a single feasible solution is found, however, the first solution proposed by the algorithm

was immediately feasible. After enough feasible solutions ($d+1 = 8$) are found to fit a first surrogate model for the objectives, the algorithm started also optimizing the objective score. In total out of the 120 evaluations 21 feasible hulls are found. The convergence plot of the feasible solutions is shown in Figure 3. The convergence plot also shows exactly what was expected. The first 8 feasible designs still show a relatively high objective value as the algorithm is not minimizing the objective score yet but putting its attention to finding feasible solutions. After the algorithm has seen enough solutions to learn from (8 in this case since there are 7 parameters), the algorithm started minimizing the objective value.

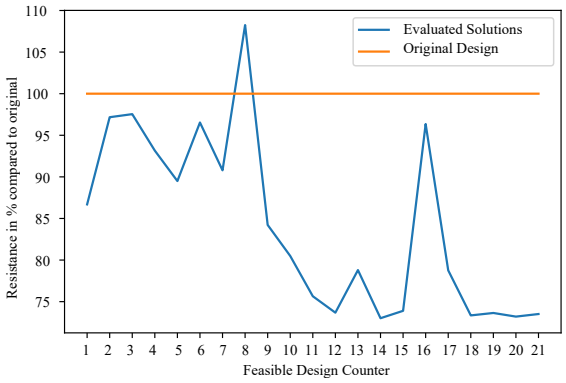

Figure 3: Convergence plot of the feasible solutions. Objective score in % compared to the original design.

The best of the 21 feasible hulls had a 26% smaller resistance value compared to the original non-optimized design. In this experiment, the principle of interacting waves worked much better since the vessel was optimized for one speed only. This resulted in a significant bulb that was added in front of the vessel as shown in Figure 4.

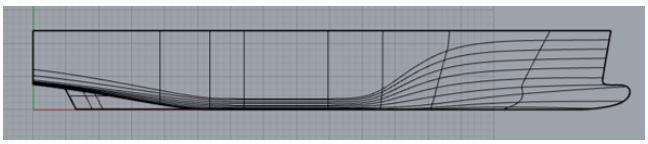

Figure 4: Hull found by the optimization algorithm.

As described earlier, besides the experiment with the algorithm, naval architects with hydrodynamic experience also independently optimized the hull of the ferry. In their experiment, they used their creativity to design the underwater part of the hull in a completely different way. The engineers chose a knuckle line and fitted the bulb under the bow flare instead of in front of the ship. After their choice, the naval architects used a total of 8 CFD evaluations and found a hull with almost identical objective value (a 26% reduction). The final optimized hull of the naval architects is displayed in Figure 5.

When compared to the design proposed by the optimization algorithm, the hull designed by the naval architects is better for a practical reason. As this is a Ro-Ro ferry, vehicles have to roll on and roll off the ship. When a bulb is in front of the ship, the ramp

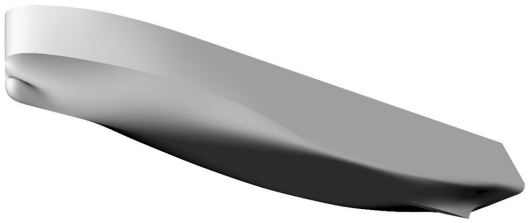

Figure 5: Proposed hull design by naval architects.

that the vehicles use to enter en to exit the ship should be longer compared to when the bulb is under the bow flare. The downside of the design with the knuckle line is that it might be more complex to build.

Although the hull from the naval architect and the optimization framework both had a 26% smaller resistance score compared to the original hull, the hull from the naval architect had practical benefits. Therefore the hull of the naval architect is preferred above the hull proposed by the optimization algorithm. The conclusion that can be drawn from this experiment is that the parameterization part of any optimization problem is the most important part when optimizing. If not all constraints are captured, the practical application is ignored, or the objective function does not perfectly represent the true goal then optimization algorithms can converge to less ideal solutions. Therefore, human experience remains important and the algorithm can only find as good solutions as the parameterization allows.

6 CONCLUSION AND FUTURE WORK

In this work, the Modular Adaptive Global Optimization Framework (MAGOF) is introduced. MAGOF can solve constraint single objective problems with a mix of computationally expensive and computationally inexpensive constraint and objective functions. MAGOF uses RBF surrogates for expensive functions, the inexpensive functions can directly be used when searching for promising solutions with an acquisition function. Besides this, a strategy is added to MAGOF that enforces the feasibility of the inexpensive constraints before computationally expensive objective and/or computationally expensive constraints are evaluated. MAGOF with the inexpensive constraints used directly when optimizing the acquisition function showed to be the most promising option when optimizing the G-Problem test suite.

Besides testing the algorithm on the G-problems, two real-world ship design problems are optimized. Provided with the parameterization, the constraint functions, and the objective function, the optimization framework was able to find feasible solutions with a good objective score. In both real-world experiments, it was concluded that the first parameterization of the solutions did not lead to the expected result which led to a different parameterization setup and a second optimization run. This shows that optimization experts and designers should continue to work together to be able to solve real-world optimization problems. In the future, more research is required on how to effectively propose multiple solutions in parallel with other batch acquisition functions described in e.g. [1, 21, 35]. Secondly, more research is required on setting up the parameterization of optimization problems.

REFERENCES

[1] Javad Azimi, Alan Fern, and Xiaoli Fern. 2010. Batch Bayesian optimization via simulation matching. *Advances in neural information processing systems* 23 (2010), 109–117.

[2] Thomas HW Bäck, Anna V Kononova, Bas van Stein, Hao Wang, Kirill A Antonov, Roman T Kalkreuth, Jacob de Nobel, Diederick Vermetten, Roy de Winter, and Furong Ye. 2023. Evolutionary Algorithms for Parameter Optimization—Thirty Years Later. *Evolutionary Computation* 31, 2 (2023), 81–122.

[3] Samineh Bagheri, Wolfgang Konen, and Thomas Bäck. 2017. Comparing kriging and radial basis function surrogates. In *Proc. 27. Workshop Computational Intelligence*, F. Hoffmann, E. Hüllermeier, and R. Mikut (Eds.). Universitätsverlag Karlsruhe, Universität Munchen, 243–259.

[4] Samineh Bagheri, Wolfgang Konen, Michael Emmerich, and Thomas Bäck. 2017. Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing* 61 (2017), 377–393. https://doi.org/10.1016/j.asoc.2017.07.060

[5] Julian Blank and Kalyanmoy Deb. 2021. Constrained bi-objective surrogate-assisted optimization of problems with heterogeneous evaluation times: Expensive objectives and inexpensive constraints. In *Evolutionary Multi-Criterion Optimization: 11th International Conference, EMO 2021, Shenzhen, China, March 28–31, 2021, Proceedings 11*. Springer, Springer, 257–269.

[6] Jakob Bossek, Carola Doerr, and Pascal Kerschke. 2020. Initial design strategies and their effects on sequential model-based optimization: an exploratory case study based on BBOB. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 778–786.

[7] Martin D Buhmann. 2003. *Radial basis functions: theory and implementations*. Cambridge University Press. https://doi.org/10.1017/CBO9780511543241

[8] C-Job Naval Architects. 2023. Saronic Ferries. https://c-job.com/projects/saronic-ferries/

[9] C-Job Naval Architects. 2023. Saronic Ferries Partners With C-Job Naval Architects for the Design of the First Fully-Electric Ro-Pax Ferry in Greece. https://c-job.com/press/saronic-ferries-partners-with-c-job-naval-architects-for-the-design-of-the-first-fully-electric-ro-pax-ferry-in-greece/

[10] Po-Fan Chen, Cheng-Hung Huang, Ming-Chung Fang, and Jar-Hung Chou. 2006. An inverse design approach in determining the optimal shape of bulbous bow with experimental verification. *Journal of ship research* 50, 01 (2006), 1–14.

[11] Carlos A. Coello Coello. 2022. Constraint-handling techniques used with evolutionary algorithms. In *Genetic and Evolutionary Computation Conference, GECCO '22, Companion Volume, Boston, Massachusetts, USA, July 9 - 13, 2022*, Jonathan E. Fieldsend and Markus Wagner (Eds.). ACM, 1310–1333. https://doi.org/10.1145/3520304.3533640

[12] Jacob de Nobel, Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. 2021. Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21, Companion material)*. ACM, 1375–1384. https://doi.org/10.1145/3449726.3463167

[13] Roy de Winter, Philip Bronkhorst, Bas van Stein, and Thomas Bäck. 2022. Constrained multi-objective optimization with a limited budget of function evaluations. *Memetic Computing* 14, 2 (2022), 151–164.

[14] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[15] Johann Dréo, Arnaud Liefooghe, Sébastien Vérel, Marc Schoenauer, Juan Julián Merelo Guervós, Alexandre Quemy, Benjamin Bouvier, and Jan Gmys. 2021. Paradiseo: from a modular framework for evolutionary computation to the automated design of metaheuristics: 22 years of Paradiseo. In *GECCO '21: Genetic and Evolutionary Computation Conference, Companion Volume, Lille, France, July 10-14, 2021*, Krzysztof Krawiec (Ed.). ACM, 1522–1530. https://doi.org/10.1145/3449726.3463276

[16] Juan J Durillo and Antonio J Nebro. 2011. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software* 42, 10 (2011), 760–771.

[17] Christodoulos A Floudas and Panos M Pardalos. 1990. *A collection of test problems for constrained global optimization algorithms*. Springer.

[18] Peter I Frazier. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811* (2018).

[19] Manolis Georgioudakis. 2017. PyDE. https://github.com/geoem/pyDE/tree/master

[20] Gerard Petersen. 2015. *Powerful Ship Hull Design in Rhino with Rapid Hull Modeling Methodology*. Rhino Centre. http://rhinocentre.blogspot.com/2009/11/rhino-rapid-hull-modeling-methodology.html

[21] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. 2016. Batch Bayesian optimization via local penalization. In *Artificial intelligence and statistics*. PMLR, 648–657.

[22] John H Halton. 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* 2, 1 (1960), 84–90.

[23] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13 (1998), 455–492.

[24] Jing J Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, Ponnuthurai Nagaratnam Suganthan, CA Coello Coello, and Kalyanmoy Deb. 2006. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics* 41, 8 (2006), 8–31.

[25] Manuel López-Ibáñez and Thomas Stützle. 2010. Automatic configuration of multi-objective ACO algorithms. In *International Conference on Swarm Intelligence*. Springer, Springer, 95–106.

[26] Wolfgang Ponweiser, Tobias Wagner, and Markus Vincze. 2008. Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models. In *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)*. IEEE, IEEE, 3515–3522.

[27] M. J. D. Powell. 1994. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In *Advances in Optimization and Numerical Analysis*, Susana Gomez and Jean-Pierre Hennart (Eds.). Springer Netherlands, 51–67. https://doi.org/10.1007/978-94-015-8330-5_4

[28] Siemens Digital Industries Software. 2021. Simcenter STAR-CCM+ User Guide, version 2021.1. In *Adaptive Mesh Refinement for Overset Meshes*. Siemens, 3067–3070. https://docs.sw.siemens.com/documentation/external/PL20200805113346338/en-US/userManual/userguide/html/STARCCMP/GUID-28A739CF-6DE2-4D87-B582-E390B522011C.html#

[29] Beng-Yeow Tan. 2004. *An experimental investigation for resistance reduction on displacement-type ships by parabolization of hull form at waterline*. Ph. D. Dissertation. University of British Columbia.

[30] Koen van der Blom, Timo M Deist, Vanessa Volz, Mariapia Marchi, Yusuke Nojima, Boris Naujoks, Akira Oyama, and Tea Tušar. 2023. Identifying properties of real-world optimisation problems through a questionnaire. In *Many-Criteria Optimization and Decision Analysis: State-of-the-Art, Present Challenges, and Future Perspectives*. Springer, 59–80.

[31] Niki van Stein, Diederick Vermetten, Anna V. Kononova, and Thomas Bäck. 2024. Explainable Benchmarking for Iterative Optimization Heuristics. arXiv:2401.17842 [cs.NE]

[32] Diederick Vermetten, Fabio Caraffini, Anna V. Kononova, and Thomas Bäck. 2023. Modular Differential Evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lisbon, Portugal) *(GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 864–872. https://doi.org/10.1145/3583131.3590417

[33] Liqun Wang, Songqing Shan, and G Gary Wang. 2004. Mode-pursuing sampling method for global optimization on expensive black-box functions. *Engineering Optimization* 36, 4 (2004), 419–438.

[34] James Wilson, Frank Hutter, and Marc Deisenroth. 2018. Maximizing acquisition functions for Bayesian optimization. *Advances in neural information processing systems* 31 (2018), 9884–9895.

[35] Jian Wu and Peter Frazier. 2016. The parallel knowledge gradient method for batch Bayesian optimization. *Advances in neural information processing systems* 29 (2016), 3126–3134.

[36] Huan Zhang, Jinliang Ding, Liang Feng, Kay Chen Tan, and Ke Li. 2023. Solving Expensive Optimization Problems in Dynamic Environments with Meta-learning. *arXiv preprint arXiv:2310.12538* (2023).