BIOS
an object-oriented framework for Surrogate-Based Optimization using bio-inspired algorithms

Barroso, Elias Saraiva; Ribeiro, Leonardo Gonçalves; Maia, Marina Alves; Rocha, I.B.C.M.; Parente, Evandro; de Melo, Antônio Macário Cartaxo

DOI
10.1007/s00158-022-03302-0

Publication date
2022

Document Version
Final published version

Published in
Structural and Multidisciplinary Optimization

Structural and Multidisciplinary Optimization (2022) 65:203
https://doi.org/10.1007/s00158-022-03302-0

RESEARCH PAPER

BIOS: an object-oriented framework for Surrogate-Based Optimization using bio-inspired algorithms

Elias Saraiva Barroso[1] · Leonardo Gonçalves Ribeiro[1] · Marina Alves Maia[1,2] · Iuri Barcelos Carneiro Montenegro da Rocha[2] · Evandro Parente Jr.[1] · Antônio Macário Cartaxo de Melo[1]

Received: 31 October 2021 / Revised: 11 March 2022 / Accepted: 11 June 2022 / Published online: 7 July 2022

Abstract
This paper presents BIOS (acronym for Biologically Inspired Optimization System), an object-oriented framework written in C++, aimed at heuristic optimization with a focus on Surrogate-Based Optimization (SBO) and structural problems. The use of SBO to deal with structural optimization has grown considerably in recent years due to the outstanding gain in efficiency, often with little loss in accuracy. This is especially promising when adaptive sampling techniques are used. However, many issues are yet to be addressed before SBO can be employed reliably in most optimization problems. In that sense, continuous experimentation, testing and comparison are needed, which can be more easily carried out in an existing framework. The architecture is designed to implement conventional nature inspired algorithms and Sequential Approximated Optimization (SAO). The system aims to be efficient, easy to use and extensible. The efficiency and accuracy of the system are assessed on a set of benchmarks, and on the optimization of functionally graded structures. Excellent results are obtained.

Keywords Object-oriented framework · Structural optimization · Surrogate-Based Optimization · Sequential Approximated Optimization · Composite structures

1 Introduction

In recent years, Surrogate-Based Optimization (SBO) has been gaining popularity due to its capability of approximating otherwise time-consuming functions with much cheaper surrogates at the expense of a small accuracy loss (Queipo et al. 2005; Forrester et al. 2008; Stork et al. 2020b). Hence, SBO can play an important role in structural optimization problems, where the high computational cost of finite element analysis is typically the main bottleneck (Chen et al. 2014; Do et al. 2019).

In SBO, a given dataset is used to build a model, which is then employed to find the global optimum. For constant surrogates, this model is fixed and, since no *a priori* knowledge is available, the sample must be well distributed so that good overall accuracy is obtained (Forrester et al. 2008).

In optimization problems, however, it might be wise to improve accuracy in promising regions of the design space, which corresponds to the Sequential Approximate Optimization (SAO) (Schmit and Farshi 1974; Kitayama and Yamazaki 2011; Ribeiro et al. 2020; Maia et al. 2021). In this approach, the surrogate model is continuously updated by the addition of new sampling points. These additions aim at improving the accuracy near the global optimum (Jones et al. 1998; Sobester et al. 2005; Forrester et al. 2008). This is also known as adaptive sampling, and usually demands a smaller number of High-Fidelity (HF) evaluations than constant models, where accuracy throughout the whole design space is vital (Liu et al. 2018; Chunna et al. 2020). Several methods have been proposed for choosing the infill points. Liu et al. (2018) argue that the combination of variance-based adaptive sampling and Gaussian Processes (GP) models is typically a good choice.

Responsible Editor: Axel Schumacher

✉ Evandro Parente Jr.
evandro@ufc.br

1 Laboratório de Mecânica Computacional e Visualização, Departamento de Engenharia Estrutural e Construção Civil, Universidade Federal do Ceará, Campus do Pici, Bloco 728, Fortaleza, CE CEP 60455-760, Brazil

2 Faculty of Civil Engineering and Geosciences, Delft University of Technology, P.O. Box 5048, 2600GA Delft, The Netherlands

The Efficient Global Optimization (EGO) algorithm, proposed by Jones et al. (1998), was essential to the widespread use of SAO as a robust technique to solve costly optimization problems. The authors employ Kriging to approximate the exact costly function, and the sample is updated by the addition of the point that maximizes the Expected Improvement (EI) at each iteration. Later, Sobester et al. (2005) presented a similar approach, but using a Radial Basis Functions (RBF) model. The authors also proposed a modification in the EI function which allows the user to explicitly balance the relative importance of exploitation and exploration.

To this day, there are still important open issues in SAO design (Queipo et al. 2005; Stork et al. 2020b), e.g., what is the best approach to perform the selection of new data points, or how to deal with constrained optimization and discrete variables. Thus, further experimentation, testing, and comparison are required to establish SAO as a reliable technique for global optimization of real-world engineering problems (Simpson et al. 2002; Wang and Shan 2007; Steponavičė et al. 2016; Liu et al. 2018; Ribeiro et al. 2020; Maia et al. 2021). For that purpose, the availability of a framework that allows further development and comparison between different methods is essential.

A multitude of frameworks has been proposed to deal with different problems in optimization (Giunta and Eldred 2000; Jacobs et al. 2004; Wagner and Affenzeller 2005; Meza et al. 2007; Durillo and Nebro 2011; Passos and Luersen 2018; Krishnamoorthy et al. 2002; Sivakumar et al. 2004; Zadeh et al. 2009; Martins et al. 2009; Blank and Deb 2020). The jMetal framework (Durillo and Nebro 2011), written in Java, is one of the most renowned. jMetal can deal with multi-objective optimization and a number of algorithms and methods is available. It is also possible to add new problems and algorithms, extending the framework features. In particular for computationally expensive applications, open-source software libraries such as Keras (Gulli and Pal 2017) and Tensorflow (Abadi et al. 2015) in Python, help disseminating the use of machine learning techniques, with particular focus on Deep Neural Networks, to tackle the issue.

This scenario is much more limited when it comes to SAO. Along with some frameworks dedicated to trust-region based approaches (Giunta and Eldred 2000; Jacobs et al. 2004), most open-source software packages available can only be employed for simple or very specific problems. For Python, scikit-optimize (Kumar 2017) is a very popular choice, as it is able to solve box-constrained Bayesian optimization using different acquisition functions. However, it is not able to solve problems with implicit constraints or a discrete data structure. SURROGATES toolbox (Viana 2010) is a general purpose MATLAB code which performs EGO for box-constrained single-objective continuous problems, providing different Design of Experiments (DoE) and modeling techniques. MATSuMoTo (Muller 2014) is another MATLAB toolbox which allows for box-constrained continuous and discrete optimization using different surrogate models. For R, DiceKriging and DiceOptim (Roustant et al. 2012) are packages that can be used together to perform the optimization of expensive functions via the EGO algorithm. Another alternative for R users is the Moko package. The framework can handle multi-objective optimization and approximates all objective functions and constraints using Kriging. On that note, it is worth stressing that a general framework must be able to consider, in the same problem, exact and approximate responses. This way, the approximate model is only evaluated when necessary and the cheap-to-evaluate response can be exactly assessed without an overly complex approximation that might result in loss of accuracy.

This paper presents BIOS (Biologically Inspired Optimization System), a framework capable of handling conventional and sequential approximate optimization of unconstrained and constrained problems with continuous and discrete variables. The use of the Object-Oriented Programming (OOP) paradigm and software design patterns (Gamma et al. 1994; Alexandrescu 2001) allow the framework to be compact and all-purpose, providing cleaner projects and codes. Relying on standard libraries, BIOS is a cross-platform (Windows, Mac, and Linux) software and supports hybrid-parallel computing using OpenMP and MPI.

BIOS was initially developed for performing the optimization of composite structures using bio-inspired optimization algorithms, e.g. Genetic algorithms (GA) and Particle Swarm Optimization (PSO). Its effectiveness and robustness was demonstrated in various engineering applications (Rocha et al. 2014; Barroso et al. 2017). In these works, parallel computing enabled a faster optimization process, which is an useful feature if multiple cores are available. BIOS also provides relevant routines to help assess the response of composite structures.

Later, SAO was integrated into BIOS as a way of providing a more efficient optimization approach for computationally expensive engineering problems involving numerical simulations, using, for example, the Finite Element Method (FEM) or the Isogeometric Analysis (IGA). The SBO approach allows the use of more complex analysis procedures and enables the solution of practical composite design problems, which is a field that demands further study (Luersen et al. 2015; Jaiswal et al. 2018). BIOS implements SAO for different sampling methods, surrogate models, and infill criteria and proved to be highly efficient in this type of problem (Ribeiro et al. 2020; Maia et al. 2021).

The contributions of this work are summarized as follows. A framework for structural optimization is presented, using SAO algorithms based on Radial Basis Functions and Kriging. Conventional optimizations with evolutionary and swarm intelligence algorithms are also supported.

The framework architecture is discussed in detail, denoting its flexibility, modularity, and reusability. Our framework contrasts with the alternatives in the literature since it is a open-source project using OOP concepts, thus allowing for extension and code reuse. Besides, BIOS provides different optimization algorithms and modeling techniques, is able to deal with continuous and discrete problems and to handle constrained and unconstrained optimization, is capable to perform SBO approximating only computationally expensive functions (while non-expensive functions are evaluated exactly), and allows for parallel computing. Finally, the framework application in benchmarks and in the optimization of functionally graded structures are demonstrated.

The rest of the paper is organized as follows. In Sect. 2, the main optimization algorithms implemented in BIOS are presented. In Sect. 3, the Surrogate-Based Optimization is further discussed. In Sect. 4, the architecture of BIOS is presented, while Sect. 5 describes the steps required to use the framework SAO algorithms to optimize a computationally expensive structural problem. Application examples are presented in Sect. 6 and the conclusions are discussed in Sect. 7.

2 Bio-inspired Optimization Algorithms

Bio-inspired heuristic algorithms have been widely employed in structural optimization in recent years due to their robustness and simplicity. In this type of algorithm, only objective and constraint functions need to be defined. Hence, they are particularly suitable when gradient information is not available, as in optimization problems with discrete (or categorical) variables. Furthermore, these algorithms are usually less prone to become trapped in local minima (Arora 2017).

On the other hand, heuristic algorithms tend to be computationally expensive in comparison with gradient-based approaches. The use of parallel computing and surrogate modelling to tackle this issue has been explored in many papers (Rocha et al. 2014; Do et al. 2019; Ribeiro et al. 2020; Zhu et al. 2012; Rouhi et al. 2015; Díaz et al. 2016; Keshtegar et al. 2020; Jaiswal et al. 2018), and some of these are available in BIOS.

The optimization procedure consists of randomly generating an initial set of solutions, or *population*, which is continuously improved towards the optimum until a stopping criterion is met (Arora 2017). The solutions are improved using a set of operators related to the algorithm meta-heuristic logic. The main steps for the optimization procedure are depicted in Fig. 1.

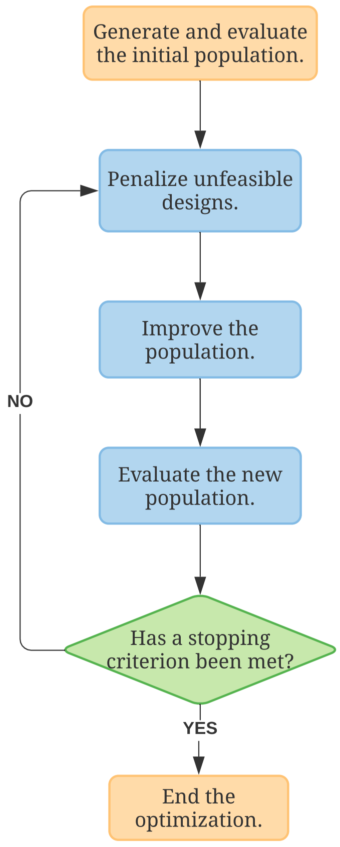

Generate and evaluate the initial population.
Penalize unfeasible designs.
Improve the population.
Evaluate the new population.
Has a stopping criterion been met?
NO
YES
End the optimization.

Fig. 1 Flowchart of bio-inspired heuristic optimization algorithms

A general single-objective optimization problem can be described as:

$$\begin{cases} \text{minimize} & f(\mathbf{x}) \\ \text{subjected to} & g_i(\mathbf{x}) \le 0 \quad i = 1, 2, \ldots, n_c \\ \text{with} & \mathbf{x}_l \le \mathbf{x} \le \mathbf{x}_u \end{cases} \tag{1}$$

where n_c is the number of inequality constraints and $\mathbf{x}_l$ and $\mathbf{x}_u$ are the lower and upper bounds of the design variables, respectively. In BIOS, constraints are handled by a penalty approach, such as the static or adaptive penalty methods (Deb 2000; Lemonge and Barbosa 2004).

The heuristic operations employed to the population improvement vary according to each algorithm. The main algorithms available in BIOS are the Genetic Algorihtm (GA) (Goldberg 2012), the Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995; Bratton and Kennedy 2007), and the Differential Evolution (DE) (Storn and Price 1997; Price et al. 2005). Moreover, the Artificial Bee Colony (ABC) (Karaboga 2005) and Artificial Immune System (AIS) (Castro and Zuben 2002) are also available.

GA is an evolutionary algorithm often employed in discrete optimization. The improvement of the population is performed by applying three operators at each iteration: crossover, mutation, and the selection. It is worth emphasizing that the BIOS implementation of this algorithm can handle both discrete and continuous optimization. For the interested reader, further discussion can be found in Rocha et al. (2014).

PSO is a swarm-intelligence based algorithm first proposed to deal with continuous optimization. Again, in BIOS, both discrete and continuous optimization are supported. At each iteration, particles move based on their inertia (previous velocity), their cognitive factor and their social factor. A mutation operator can also be applied to improve the global convergence (Barroso et al. 2017).

Finally, DE is an evolutionary strategy also aimed at dealing with continuous optimization. Here, the operators employed to improve the population are differentiation, crossover, and selection. Further details can be found in Price et al. (2005).

That being said, bio-inspired algorithms often require hundreds or even thousands of function evaluations to achieve the optimum solution (Steponavičė et al. 2016). Furthermore, even though they are more reliable in their exploration of the design space, there is no guarantee that the algorithms will find even a local optimum solution. That means that their best design may not satisfy the Karush-Kuhn-Tucker (KKT) conditions (Kuhn and Tucker 1951).

3 Surrogate-Based Optimization

Surrogate-Based Optimization can be employed to reduce the computational cost of an optimization process. Put simply, a SBO may be described by two basic stages: sampling and building of the chosen surrogate model. There are multiple ways to define the surrogate model hyperparameters, ranging from simple cross validation techniques and analytical expressions to the optimization of a given likelihood function.

A SAO algorithm is obtained by introducing a stage where new point(s) (i.e. infill point) are added to the dataset so that the surrogate model is improved iteratively in promising regions of the design space. This process is repeated until a stopping criterion is met, as depicted in Fig. 2. Generally, two criteria can be considered: the maximum number of evaluations $n_{\max}$ or maximum number of stall iterations It_{stall}.

The combination of an initial sampling method, surrogate model and infill criteria defines a SAO algorithm. The following sections further detail each stage. For a more complete review on commonly employed surrogate modeling techniques for optimization and recent advances, the reader is referred to proper literature (Queipo et al. 2005; Forrester et al. 2008; Forrester and Keane 2009).

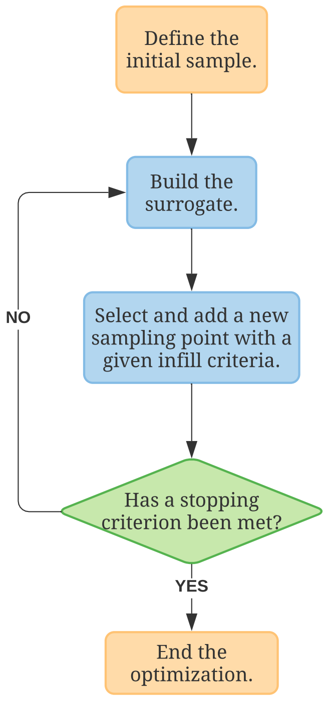

Define the initial sample.
Build the surrogate.
Select and add a new sampling point with a given infill criteria.
Has a stopping criterion been met?
NO
YES
End the optimization.

Fig. 2 General SAO framework

3.1 Initial sampling

On computer experiments, the initial sampling points should be well distributed in the design space, since, *a priori*, no information about the behavior of the function is available (Simpson et al. 2002; Kleijnen et al. 2005; Tenne 2014). Thus, the initial sampling is often performed by a Design of Experiments (DoE) technique.

In BIOS, deterministic methods (e.g. Hammersley and Sobol sequences) and stochastic methods (e.g. random sampling and Latin Hypercube Sampling), are available (Tenne 2014; Steponavičė et al. 2016). While the former group can provide more uniform sampling spaces in some cases, the latter can introduce a variability which can be helpful when performing multiple optimizations.

Comparison between sampling methods is presented in a variety of papers (Jin et al. 2001; Simpson et al. 2002; Steponavičė et al. 2016; Cho et al. 2016). Typically, it can be said that the method chosen is of minor importance, as long as it is capable of providing an uniform dataset (Steponavičė et al. 2016). However, it is worth mentioning that the Hammersley sequence uniformity is compromised in high-dimensions (Steponavičė et al. 2016; Cho et al. 2016). Steponavičė et al. (2016) also argue that stochastic techniques such as the random sampling and, to a lesser degree, the Latin Hypercube Sampling (LHS) might also occasionally not achieve the desirable uniformity.

In general, the number of points of the initial dataset increases exponentially with the number of variables m of the problem (Jin et al. 2001; Forrester et al. 2008). This aspect is known as the *curse of dimensionality* (Forrester et al. 2008), and is a major concern for surrogate-based optimization. In BIOS, the initial number of points is a user-defined parameter, and the user should select an adequate value for a specific problem. A variation of the LHS is also implemented where N different datasets are generated, and the one where the minimum distance between two sampling points is the highest is chosen (*maximin* criterion) (Forrester et al. 2008). The distance d_p between sampling points may be evaluated by:

$$d_p(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(\sum_{k=1}^{m} |x_k^{(i)} - x_k^{(j)}|^p \right)^{\frac{1}{p}} \tag{2}$$

where d_p is the usual Euclidean distance for $p = 2$. Here, this method is referred to as LHS$_\mathrm{N}$. Fig. 3 illustrates some of the techniques discussed in this section for a two-dimensional problem. In this particular case, the Hammersley sequence and the LHS$_{20}$ presented the highest d_p.

3.2 Surrogate modeling

The surrogate models available in the current version of BIOS are Radial Basis Functions (RBF) and Kriging. These are very robust models able to perform accurate predictions in a wide range of problems (Forrester et al. 2008), and are often regarded as the best surrogate modelling techniques in comparative studies (Jin et al. 2001; Simpson et al. 2002; Hussain et al. 2002; Wang and Shan 2007; Kim et al. 2009; Díaz-Manríquez et al. 2011; Nik et al. 2014; Williams and Cremaschi 2021). Both models are further discussed in the following sections.

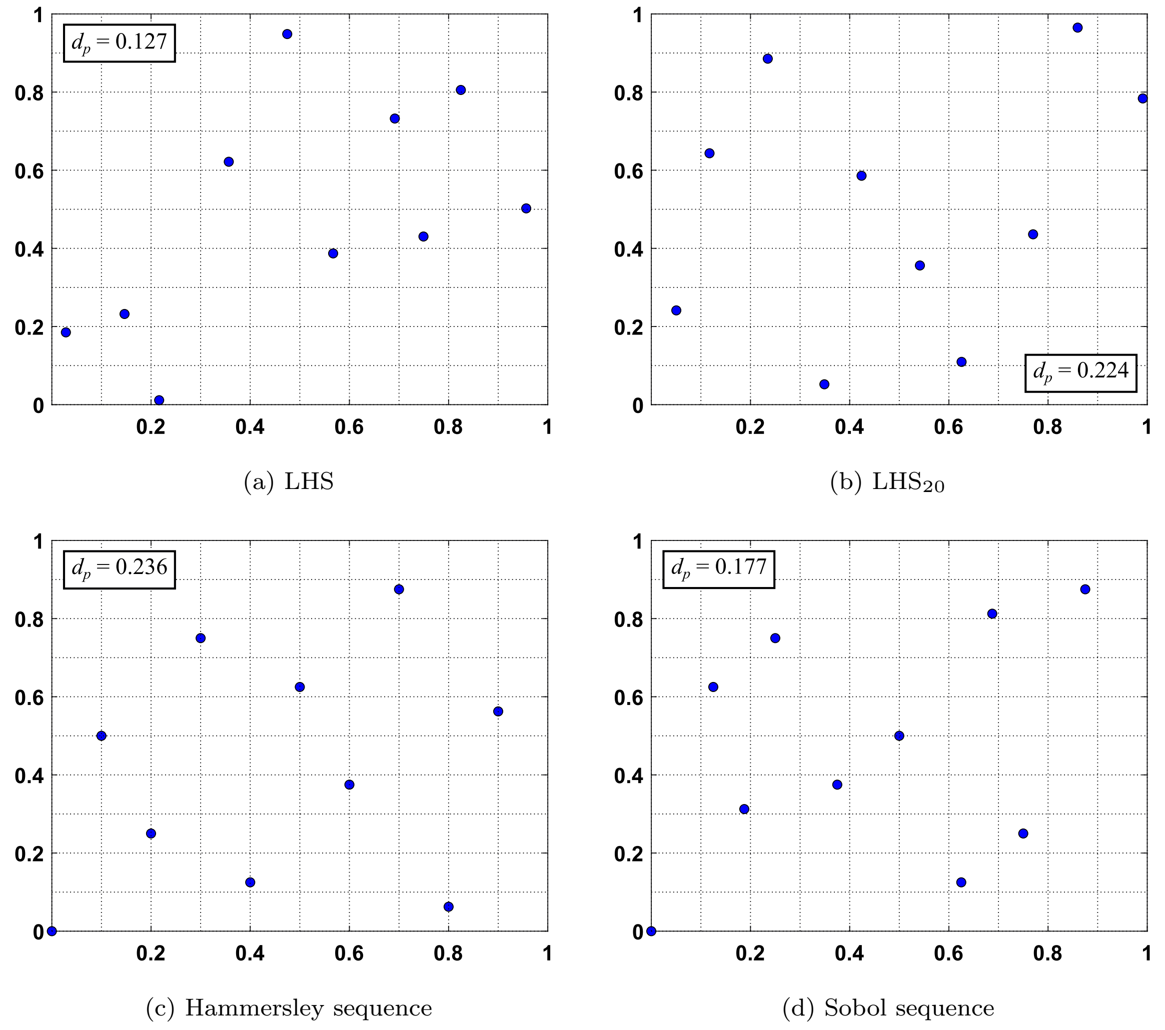


Fig. 3 Comparison between sampling methods

3.2.1 Radial Basis Functions

The Radial Basis Functions model (Hardy 1971) is a linear combination of radially symmetric functions centered around a set of points (Forrester et al. 2008):

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\psi} = \sum_{j=1}^{n} w_j \, \psi_j(|\mathbf{x} - \mathbf{c}_j|) \tag{3}$$

where $\mathbf{w}$ is the weight vector, $\mathbf{c}_j$ are known as the basis functions centers, and $\boldsymbol{\psi}$ is the basis function vector. Different basis functions may be employed (Forrester et al. 2008), but the Gaussian is the most popular choice (Sobester et al. 2005; Kitayama and Yamazaki 2011):

$$\psi_j(r) = \exp\left(-\frac{r^2}{\sigma_j^2}\right) \tag{4}$$

where r is given by $|\mathbf{x} - \mathbf{c}_j|$ and σ_j is a width parameter. This particular basis function is interesting for SAO since it allows the use of the theory behind Gaussian Processes (Sobester et al. 2005; Liu et al. 2018) to select new infill point(s).

Note that the width parameter σ_j has to be defined *prior* to the model fitting. Figure 4 illustrates the influence of this hyperparameter in the response of the Gaussian function. For small values, the surrogate model resembles a "needles in a haystack" function, where only regions near sampling points provide accurate predictions (Forrester et al. 2008), while greater values can make the approximate surface nearly flat. Such values may also lead to a Runge phenomenon in the interpolation, as the Gram matrix can become ill-conditioned (Wu et al. 2016).

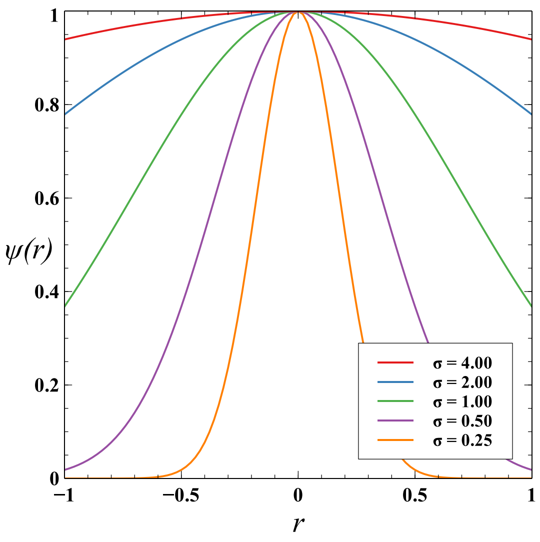


Fig. 4 Gaussian basis functions with different widths

In BIOS, this parameter can be defined using analytical approaches, such as the ones proposed by Nakayama et al. (2002) and Kitayama and Yamazaki (2011), or by cross validation techniques, such as the Leave-One-Out Cross Validation (LOOCV) (Sobester et al. 2005) and the k-Fold Cross Validation (k-FCV) (Müller and Shoemaker 2014; Ribeiro et al. 2020). By previous testing and experience, the k-FCV, with $k = 5$, usually provides accurate and efficient results.

The fitting of the model is performed by an interpolation procedure. Thus, on data points, $\hat{y}(\mathbf{x}) = y(\mathbf{x})$:

$$\begin{bmatrix} \psi_{11} & \psi_{12} & \cdots & \psi_{1n} \\ \psi_{21} & \psi_{22} & \cdots & \psi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{n1} & \psi_{n2} & \cdots & \psi_{nn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{\Psi}\mathbf{w} = \mathbf{y} \tag{5}$$

here, $\boldsymbol{\Psi}$ is the Gram matrix, whose elements are given by Eq. (4), where $\psi_{ij} = \psi_j(|\mathbf{x}_i - \mathbf{x}_j|)$.

3.2.2 Kriging

Kriging is a widely employed surrogate that models the responses as stochastic processes (Forrester et al. 2008). Its predictor is given by:

$$\hat{y}(\mathbf{x}) = \hat{\mu} + \boldsymbol{\psi}^T \boldsymbol{\Psi}^{-1} (\mathbf{y} - \mathbf{1}\,\hat{\mu}) \tag{6}$$

where the first term is related to the global trend and the second term refers to the localized autocorrelated deviations.

The correlation matrix $\boldsymbol{\Psi}$ is given by:

$$\boldsymbol{\Psi} = \begin{bmatrix} \mathrm{cor}[\mathbf{y}_1, \mathbf{y}_1] & \mathrm{cor}[\mathbf{y}_1, \mathbf{y}_2] & \cdots & \mathrm{cor}[\mathbf{y}_1, \mathbf{y}_n] \\ \mathrm{cor}[\mathbf{y}_2, \mathbf{y}_1] & \mathrm{cor}[\mathbf{y}_2, \mathbf{y}_2] & \cdots & \mathrm{cor}[\mathbf{y}_2, \mathbf{y}_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{cor}[\mathbf{y}_n, \mathbf{y}_1] & \mathrm{cor}[\mathbf{y}_n, \mathbf{y}_2] & \cdots & \mathrm{cor}[\mathbf{y}_n, \mathbf{y}_n] \end{bmatrix} \tag{7}$$

where $\mathbf{y}_i$ is the i-th sampling point response and $\mathrm{cor}[\mathbf{y}_i, \mathbf{y}_j]$ denotes the correlation between points i and j. Usually, the correlation is given by:

$$\mathrm{cor}[\mathbf{y}_i, \mathbf{y}_j] = \exp\left(-\sum_{l=1}^{m} \theta_l \, |x_{i,l} - x_{j,l}|^{p_l}\right) \tag{8}$$

This basis function is equivalent to the Gaussian if $p_l = 2$ and θ_l is the same for all variables. In fact, p_l is often set at 2 to ease out the model fitting. Another way to describe the correlation between data points is the Matérn 5/2 function (Maia et al. 2021):

$$\mathrm{cor}[\mathbf{y}_i, \mathbf{y}_j] = \prod_{l=1}^{m} \exp\left(-\frac{\sqrt{5}r}{\theta_l}\right)\left(1 + \frac{\sqrt{5}r}{\theta_l} + \frac{5\,r^2}{3\,\theta_l^2}\right) \tag{9}$$

where $r = |x_{i,l} - x_{j,l}|$.

To build the surrogate model, θ_l must be defined for each variable l. This is obtained through Maximum Likelihood Estimation (MLE):

$$\ln(L) = -\frac{1}{2}\left(n \ln(2\pi) + n \ln(\sigma^2) + \ln|\boldsymbol{\Psi}|\right) - \frac{(\mathbf{y} - \mathbf{1}\mu)^{\mathrm{T}} \boldsymbol{\Psi}^{-1} (\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2} \tag{10}$$

Based on it, the maximum likelihood estimates for μ and σ^2 are given by:

$$\hat{\mu} = \frac{\mathbf{1}^{\mathrm{T}} \boldsymbol{\Psi}^{-1} \mathbf{y}}{\mathbf{1}^{\mathrm{T}} \boldsymbol{\Psi}^{-1} \mathbf{1}} \text{ and } \hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^{\mathrm{T}} \boldsymbol{\Psi}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu})}{n} \tag{11}$$

Substituting these into Eq. (10) and removing the constant terms, the concentrated ln-likelihood function is obtained:

$$\ln(L) \approx -\frac{n}{2} \ln(\sigma^2) - \frac{1}{2} \ln|\boldsymbol{\Psi}| \tag{12}$$

Finally, the maximization of this function is solved as an unconstrained optimization problem. In BIOS, it is possible to instantiate an heuristic optimization algorithm to solve it, such as the ones shown in Sect. 2.

3.3 Model update

This stage is of major importance to ensure efficiency in the optimization process. The infill criterion needs to be able to explore the design space to correctly locate the optimal region, which must then be exploited until the global minimum is found.

Thus, a good infill criterion must be able to balance local exploitation and global exploration (Yao et al. 2014; Stork et al. 2020a). In some cases, more than one point per iteration is sampled to account for both concepts individually (Nakayama et al. 2003; Kitayama et al. 2010; Pan et al. 2014; Xiang et al. 2016).

The infill criterion is often represented by the optimization of a given acquisition function. A basic example is the iterative addition of the point which minimizes the model prediction $\hat{y}(\mathbf{x})$. However, while this approach may work for unimodal problems, it is a pure exploitation method that tends to be too greedy (Jones et al. 1998; Mlakar et al. 2015; Bouhlel et al. 2018).

Variance-based adaptive sampling methods are also a popular choice to guide the model improvement (Jones et al. 1998; Sobester et al. 2005; Liu et al. 2018). In Gaussian Processes (GP), the posterior variance can be assessed by (Chunna et al. 2020):

$$\hat{s}^2(\mathbf{x}) = \hat{\sigma}^2\left[1 - \boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\psi + \frac{\left(1 - \mathbf{1}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\boldsymbol{\psi}\right)^2}{\mathbf{1}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\mathbf{1}}\right] \tag{13}$$

For Kriging, the evaluation of $\hat{\sigma}$ is described in Sect. 3.2.2, while for the RBF model one may assume that $\hat{\sigma} = 1.0$ (Sobester et al. 2005; Xiang et al. 2016). For trial designs close to sampling points, $\hat{s}(\mathbf{x})$ tends to 0 (low uncertainty) and, as it gets farther from them, $\hat{s}(\mathbf{x})$ goes to $\hat{\sigma}^2$ (high uncertainty).

BIOS currently works with models based on Gaussian Processes (GP), and allows the use of four different infill criteria of this type: Lower Confidence Bound (LCB), Probability of Improvement (PI), Expected Improvement (EI), and Weighted Expected Improvement (WEI).

The LCB criterion is a straightforward approach where the infill point is found by minimizing the confidence bound (Srinivas et al. 2010; Brochu et al. 2010):

$$y_{LCB}(\mathbf{x}) = \hat{y}(\mathbf{x}) - \kappa\,\hat{s}(\mathbf{x}) \tag{14}$$

where κ is a user-defined parameter. Higher values of κ favor the exploration aspect.

PI, on the other hand, is a probabilistic criterion for which the probability that a given $\mathbf{x}$ improves upon the current best design should be maximized:

$$P[I(\mathbf{x})] = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{y_{\min} - \hat{y}(\mathbf{x})}{\hat{s}\sqrt{2}}\right)\right] \tag{15}$$

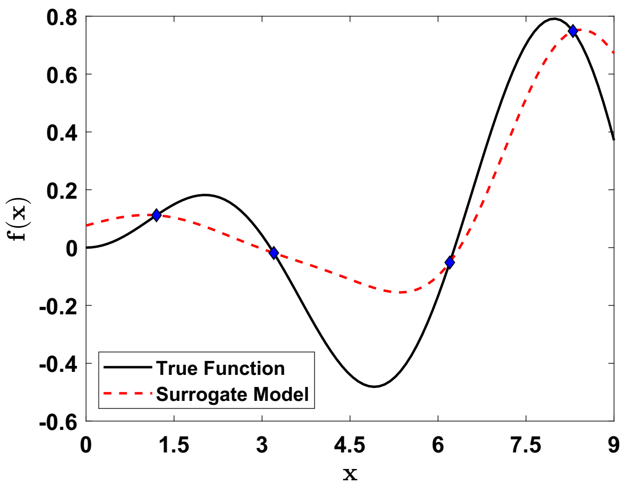
This approach is often deemed to be a pure exploitation method and, thus, may underperform in multimodal problems (Brochu et al. 2010; Jones 2001).

In this regard, the EI is often a better choice as a probabilistic approach, being given by (Forrester et al. 2008):

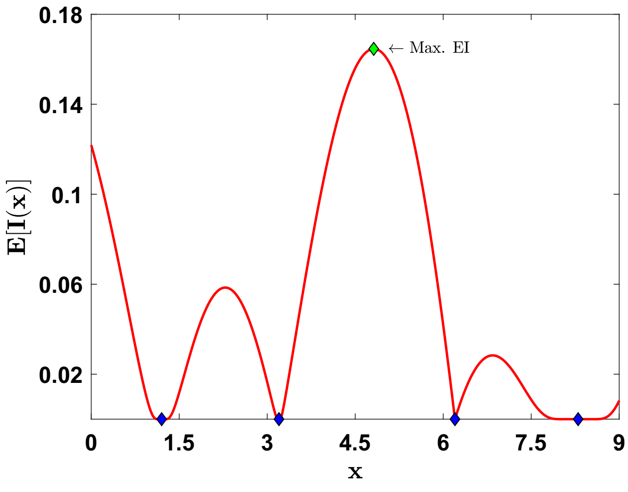
$$E[I(\mathbf{x})] = (y_{\min} - \hat{y}(\mathbf{x}))\left[\frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{y_{\min} - \hat{y}(\mathbf{x})}{\hat{s}\sqrt{2}}\right)\right] + \frac{\hat{s}}{\sqrt{2\pi}} \exp\left[\frac{-(y_{\min} - \hat{y}(\mathbf{x}))^2}{2\hat{s}^2}\right] \tag{16}$$

here, the first term is related to exploitation and the second to exploration. The EI criterion is one of the most widely used methods today due to the popularization of this alternative by the EGO algorithm, first proposed by Jones et al. (1998). Figure 5 depicts the improvement of the RBF prediction of a test function considering the EI as infill criterion. The approach is able to precisely identify the optimum region.

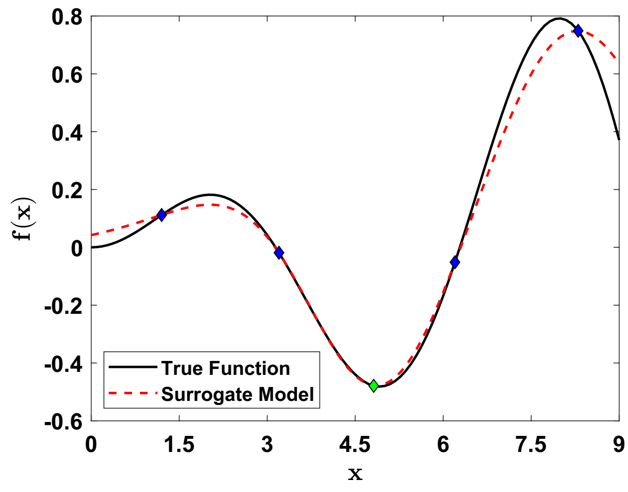
WEI consists in a variation of the EI criterion proposed by Sobester et al. (2005) with a weight parameter w being introduced:



(a) Initial RBF model



(b) Behavior of the $EI(\mathbf{x})$ function



(c) Updated RBF model

Fig. 5 Addition of point with the highest EI

$$WE[I(\mathbf{x})] = w\,(y_{\min} - \hat{y}(\mathbf{x}))\left[\frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{(y_{\min} - \hat{y}(\mathbf{x}))}{\hat{s}\sqrt{2}}\right)\right] + (1-w)\frac{\hat{s}}{\sqrt{2\pi}}\exp\left[\frac{-(y_{\min} - \hat{y}(\mathbf{x}))^2}{2\hat{s}^2}\right] \tag{17}$$

Lower values of w favor global exploration, while higher values favor local exploitation. A cycling approach can be performed, where w may assume a value that changes iteratively in a given order.

Another major concern in SAO is how to handle constrained problems, especially when probabilistic infill criteria are employed (Stork et al. 2020b). For easy, cheap to evaluate constraints, there is no need to build a surrogate model and the constraint may be exactly evaluated. In that case, if a design is unfeasible, its EI (or PI) is simply set to 0 (Sobester et al. 2005).

For constraint functions approximated by a surrogate model, the uncertainty of the process should be taken into account. Thus, a feasibility function $F(\mathbf{x})$ can be employed, such as the Probability of Feasibility (PF) (Schonlau et al. 1998):

$$P_j[F(\mathbf{x})] = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{-\hat{g}_j(\mathbf{x})}{\hat{s}(\mathbf{x})\sqrt{2}}\right)\right] \tag{18}$$

where j refers to the j-th expensive constraint function. Other feasibility functions can be found in the literature, such as the one proposed by Tutum et al. (2014):

$$F_j^{(T)}(\mathbf{x}) = \begin{cases} 2 - \mathrm{erf}\left(-\dfrac{\hat{g}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) & \text{, if } 0 < \mathrm{erf}\left(-\dfrac{\hat{g}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) \le 1 \\ 0 & \text{, otherwise} \end{cases} \tag{19}$$

or the one proposed by Bagheri et al. (2017):

$$F_j^{(B)}(\mathbf{x}) = \min\left(2\,P_j[F(\mathbf{x})], 1\right) \tag{20}$$

These feasibility functions can then be used to penalize the probabilistic infill criteria. This way, the Constrained Expected Improvement (CEI) is evaluated by:

$$E[I_c(\mathbf{x})] = E[I(\mathbf{x})]\prod_{i=1}^{n_{apc}} F_i(\mathbf{x}) \tag{21}$$

where n_{apc} is the number of approximate constraints and $F(\mathbf{x})$ is the feasibility function.

Figure 6 illustrates the behavior of $F(\mathbf{x})$ for different approaches, where:

$$\overline{g}(\mathbf{x}) = -\frac{\hat{g}(\mathbf{x})}{\hat{s}(\mathbf{x})} \tag{22}$$

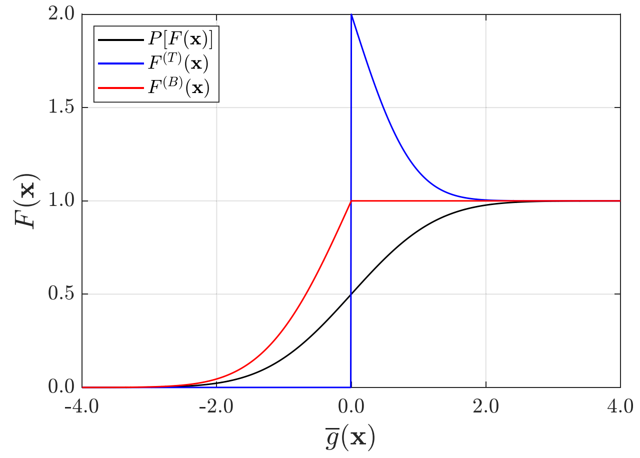


Fig. 6 Behavior of different feasibility functions

Note that the penalization of unfeasible designs is very different for each feasibility function.

In structural optimization, it is common that the objective function is cheap and easy to evaluate (e.g. cost or mass functions), while the constraints are computationally expensive (e.g. maximum deflection or failure criteria). Admittedly, this combination is another aspect in SAO that demands further investigation. In that case, BIOS evaluates the actual improvement $I(\mathbf{x})$ of a given design, which is then penalized by the feasibility function according to (Mathern et al. 2020):

$$I_c(\mathbf{x}) = I(\mathbf{x}) \prod_{i=1}^{n_{apc}} F_i(\mathbf{x}) \tag{23}$$

where:

$$I(\mathbf{x}) = \begin{cases} y_{\min} - y(\mathbf{x}) & \text{, if } y(\mathbf{x}) < y_{\min} \\ 0 & \text{, otherwise} \end{cases} \tag{24}$$

Just as in the case of the MLE maximization, the maximization of the acquisition function is carried out by instantiating heuristic algorithms in BIOS and solving it as an optimization problem. This is interesting due to the multimodal nature of these functions (Maia et al. 2021).

4 BIOS architecture

In this section, the core components of BIOS are shown in class diagrams. The roles of each component are explained in detail, as well as the hierarchy of abstract classes and their main functions.

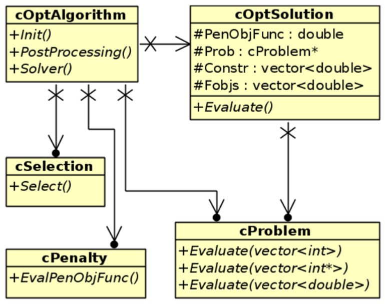


Fig. 7 Optimization module class diagram

4.1 Optimization Module

The optimization module contains the classes used to solve the optimization problem. The abstract classes and their interfaces are presented in Fig. 7. The optimization algorithms are defined in the *cOptAlgorithm* class. The abstract method *Solver()* implements all steps of the concrete algorithm class to perform N_{opt} optimizations. The output information, such as success rate and mean best, are evaluated in the *PostProcessing()* method. The *Init()* method initializes the variables required for the concrete optimization class.

The current version of BIOS has the following algorithms for single objective function optimization: Genetic algorithm (*cStandardGA*), Particle Swarm Optimization (*cStandardPSO*), Differential Evolution (*cStandardDE*), Artificial Bee Colony (*cStandardABC*) and Artificial Immunity Sytems (*cStandardAIS*). A NSGA-II implementation is also available for multi-objective optimization problems. Moreover, a set of algorithms for laminated composites problems are available: *cLaminatedGA*, *cLaminatedPSO*, and *cLaminatedNSGAII*. These algorithms use additional heuristic operators for this class of problems, with the parameters and methods employed by them being defined in the abstract class *cLamProb*. All algorithms mentioned above are depicted in Fig. 8.

The *cOptSolution* class abstracts the optimization agent of each algorithm. For instance, individuals from GA and DE are implemented in the class *cIndividual*, while particles from PSO are implemented in the *cParticle* class. Each optimization agent subclass defines its heuristic operators and implements these in their concrete subclasses depending on the optimization problem variables type. The abstract method *Evaluate()* of the *cOptSolution* class passes its optimization variables directly to the *cProblem* object for evaluation of the corresponding objective function and constraints.

We consider four variable types: double precision floating point vector for continuous optimization problems; integer vector, integer matrix, and binary variables for discrete optimization problems. The integer matrix is used for composite

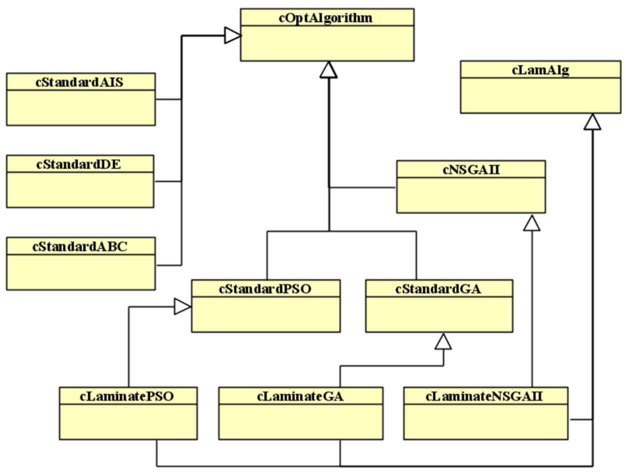


Fig. 8 *cOptAlgorithm* class hierarchy

laminated problems, where each matrix column represents ply material, thickness and angle (Rocha et al. 2014; Barroso et al. 2017). This encoding scheme facilitates the application of heuristic operators for laminates, e.g. *layer swap*, *layer addition*, and *layer deletion*. Figure 9 shows the *cOptSolution* class hierarchy, with the *cParticle* and *cIndividual* abstract and concrete classes.

The *cSelection* class implements the selection mechanism, such as the fitness proportional, rank based roulette, and tournament selection (Arora 2017).

The *cPenalty* class implements the penalty objective function approach used to handle constrained optimization problems. Static and adaptive penalty methods (Deb 2000; Lemonge and Barbosa 2004) are currently available.

The *cProblem* class implements the optimization problem. The abstract method *Evaluate()* computes the objective function and constraints for a given set of design variables. Three versions of this method are defined in the current version of BIOS, and can be implemented in the concrete problem class to handle optimization problems with continuous variables (*vector< double>*), discrete variables with integer encode (*vector< int>*), and discrete variables with matrix encode (*vector<int*>*). The latter is used in the context of composite laminated problems. The *cProblem* is also responsible to decode integer variables, within *Evaluate()* calls. Notice that additional optimization variable types can be incorporated to BIOS once a new variant of *cProblem*'s *Evaluate()* method and a new *cOptSolution* subclass for the corresponding scheme are defined.

4.2 SAO module

The SAO module contains the classes required to implement the sequential approximated optimization techniques discussed in Sect. 3. Figure 10 shows the class diagram of this module. The classes of the optimization module are reused and expanded with few abstract methods (yellow boxes in the class diagram). The *cSAO* class implements the SAO algorithm steps, the evaluation of the initial sample, the iterative model building, and the selection of new data points, as shown in Fig. 2. This class abstracts the surrogate model used, which must be defined in the concrete subclass. For instance, the *cSAORBF* and the *cSAOKRG*

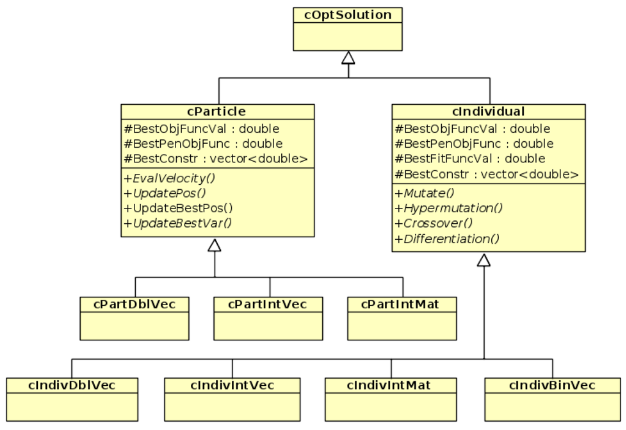


Fig. 9 *cOptSolution* class hierarchy

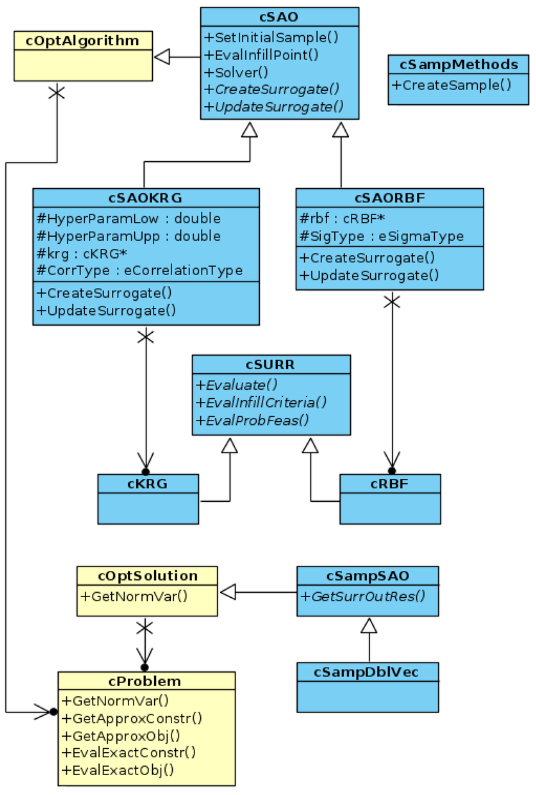


Fig. 10 SAO module class diagram

classes implement SAO using Kriging and Radial Basis Functions, respectively.

The abstract *cSURR* class defines the surrogate models. The approximated prediction for a given input design vector is computed by the *Evaluate()* method. Moreover, the evaluation of the infill criteria and the probability of feasibility discussed in Sect. 3.3, for a given set of design variables, are carried out in the *EvalInfillCriteria()* and *EvalProbFeas()* methods, respectively. These functions are used during the evaluation of the infill points by the *cSAO* class. Model parameters and hyperparameters, as well the evaluation of these for given samples, are defined in each concrete class of *cSURR*, such as *cKRG* and *cRBF*. Note that some input parameters, such as the correlation scheme (*CorrType*) and hyperparameter bounds (*HyperParamLow* and *HyperParamUpp*) for Kriging, and the width evaluation scheme (*SigType*) for Radial basis functions are passed from the corresponding *cSAO* concrete class.

It is important to note that the number of responses evaluated by surrogate models is not necessarily equal to the sum of the objective functions and constraints defined in the optimization problem. In practice, the surrogate model should be employed only in the prediction of computationally expensive objective functions and constraints. Hence, the *cProblem* class defines which objective functions and constraints are approximated in the *GetApproxConstr()* and *GetApproxObj()* methods. These methods set a boolean flag for each objective function and constraint, indicating if they are approximated or not. The default implementation sets up flags to approximate all functions. In practical applications, the concrete *cProblem* should overload these methods to approximate only costly functions, increasing computational efficiency. The evaluation of a single objective or constraint function is performed, respectively, by the *EvalExactObj()* and *EvalExactConstr()* methods.

Furthermore, the surrogate model input variables are normalized in the [0,1] range. This normalization is carried out in the method *GetNormVar()* of the *cProblem* class. The *cSampDblVec* class implements the sample optimization agent used in *cSAO* when continuous variables are used, which is the case of the examples presented in Sect. 6. The addition of integer variable samples is easily done considering the class architecture presented so far. However, this topic is not in the scope of the presented discussion and will be explored in future studies.

The pseudocode of the *Solver()* method of *cSAO* class is illustrated in Fig. 11. The evaluation of the initial dataset is done in the *SetInitialSample()* method, with aid of the helper class *cSampMethods*, which implements the sampling techniques discussed in Sect. 3.1. The abstract methods *CreateSurrogate()* and *UpdateSurrogate()* are implemented in the concrete classes, as it operates with the concrete *cSURR* object and their specific parameters. The penalty approach is used here only to select the current best sample, which may be unfeasible depending on the initial sample generated. The *GetCycleWeight()* sets w^i and β^i values used for the evaluation of the WEI and LCB, respectively, in the *EvalInfillPoint()* routine.

```
$\mathbf{x} \leftarrow$ InitialSample($n_s$);
Evaluate($\mathbf{x}$);
Surr $\leftarrow$ *CreateSurrogate*($\mathbf{x}$);
for *Each iteration until MaxIter* do
    Penalty($\mathbf{x}$);
    $\mathbf{x}^{best} \leftarrow$ BestSample($\mathbf{x}$);
    GetCycleWeight($w^i$,$\beta^i$);
    $\mathbf{x}^i \leftarrow$ EvalInfillPoint(Surr,$\mathbf{x}^{best}$,$w^i$,$\beta^i$);
    Evaluate($\mathbf{x}^i$);
    Surr $\leftarrow$ *UpdateSurrogate*($\mathbf{x}_i$,Surr);
    $\mathbf{x} \leftarrow \mathbf{x} \cup \mathbf{x}^i$;
    if *Stopping criteria is satisfied* then break;
end
```

*Only abstract methods are written in italic.

Fig. 11 Pseudocode of the *Solver()* method of the *cSAO* class

```
class cCircularPltFGM : public cProblem,
                        public cFGM
{
 public:
           cCircularPltFGM(void);
  virtual ~cCircularPltFGM(void) { }

  void   Analysis(cVector,double&);
  void   Evaluate(cVector&,cVector&,cVector&);
  double EvalExactConstr(int,cVector&);
  void   GetApproxConstr(bool*);
};
```

Fig. 12 *cCircularPlateFGM* class definition

5 Solving optimization problems using BIOS

In this section, we describe the steps to use BIOS's SAO algorithms to solve an engineering problem with computationally expensive functions. The problem is discussed in Sect. 6.4, and consists of the maximization of the fundamental frequency of a circular plate, made of a functionally graded material (Al/Al_2O_3), subjected to maximum mass and maximum ceramic volume fraction constraints, as formulated in Eq. (34).

The first step is to define a concrete *cProblem* class, which for this example is the *cCircularPltFGM* class shown in Fig. 12. This class also inherits from *cFGM* class, an auxiliary class with common routines employed in optimization FG structures (see Fig. 13). It is worth pointing out that a similar class for laminated composite optimization is also available in BIOS.

```
class cFGM
{
  void MoriTanaka(vector<double>,vector<double
     >&);
  void CalcABDG(cMatrix&);
  void CalcMb(cMatrix&);
  void QMatrix(double,cMatrix&);
  void EvalVolumeRatio(cVector,double&);
};
```

Fig. 13 *cFGM* class definition

The optimization problem data, such as the number of variables, objective functions, and constraints, are set in the constructor of the concrete problem class. The evaluation of the objective functions and constraints for a given design vector is performed by the *Evaluate()* method shown in Fig. 14. The *Analysis()* method (line 14) processes the expensive numerical response, which in this case consists in finding the first natural frequency of the circular plate model defined by a given input design variable vector. Next, the mass and ceramic fraction are evaluated, and the output constraint vector **c** (lines 16–34) is computed. Finally, the objective function is returned.

The routines described above are enough to process the optimization problem with conventional optimization algorithms. If no further specification is provided for the SAO algorithms, the default settings will build approximations for all objectives and constraints. However, in this particular problem, only the objective function evaluation is computationally expensive. Hence, we implement *GetApproxConstr()*, which sets an approximation flag for each constraint, and *EvalExactConstr()*, which evaluates the *i*-th constraint for a given design variable vector. The implementation of these methods is depicted in Fig. 15.

With the concrete problem established, the optimization is performed using the BIOS modules described earlier. The code can be easily integrated into any C++ program. Nevertheless, BIOS includes a main function file for the optimization of a single problem, for reading input text files and writing results into an output file. The input file containing the optimization parameters of the circular FGM plate problem is shown in Fig. 16. Additionally, an input file for specific problem data can be defined. In this case, the FGM material properties are specified in a new file with extension .fgm, as shown in Fig. 17.

It is worth pointing out that software design patterns are used to simplify usability, i.e., the creational patterns singleton and abstract factory are employed (Gamma et al. 1994; Alexandrescu 2001). One benefit in the use of these patterns is that all code required to add a new problem to the system can be placed into new files, and no modifications need to be done to the main code. In the problem described here, the registration of concrete problem class into the problem factory object is performed on its implementation file, Fig. 14 at line 1, where the extension of the problem data file is also specified.

```
 1 static const bool r = cProblemFactory ::
       Register("CircularPltFGM", MakeProb<
       cCircularPltFGM>,".fgm");
 2
 3 double cCircularPltFGM :: cCircularPltFGM( )
 4 {
 5   NumVar    = 6;
 6   NumObj    = 1;
 7   NumConstr = 2;
 8 }
 9
10 double cCircularPltFGM :: Evaluate(vector<
       double> &x, vector<double> &c)
11 {
12   // Frequency evaluation.
13   double w;
14   Analysis(x, w);  // IGA modal analysis.
15
16   // Get volume fraction at control points.
17   int numcp = NumVar*2 - 1;
18
19   vector<double> Vcp(numcp);
20   for (int i = 0; i < NumVar; i++)
21     Vcp[i] = Vcp[numcp-1-i] = x[i];
22
23   // Evaluate mass and ceramic volume fraction.
24   double rho = 0;
25   EvalDens(Vcp, rho);
26   double area = PI*0.5*0.5;
27   double mass = rho*area*x[0];
28
29   double vcrat;
30   EvalVolumeRatio(Vcp, vcrat);
31
32   // Evaluate problem constraints.
33   c[0] = (mass - 100)/100; // Eq. (35).
34   c[1] = (vcrat - 0.35)/0.35; // Eq. (36).
35
36   // Return problem objective function.
37   return -w; // Frequency maximization.
38 }
```

Fig. 14 Circular FGM plate implementation file

```
void cCircularPltFGM :: GetApproxConstr(bool*
     approxc)
{
  approxc[0] = 0;
  approxc[1] = 0;
}

double cCircularPltFGM :: EvalExactConstr(int i,
     vector<double> &x)
{
  // ...
  // Lines 8-24 from Evaluate method (Fig. 14).
  // ...

  if (i == 0)
    return (mass - 100)/100;
  else
    return (vcrat - 0.35)/0.35;
}
```

Fig. 15 Implementation of approximated constraint flags and exact constraint evaluation

```
%OPTIMIZATION.ALGORITHM
'SAORBF'

%INDIVIDUAL.TYPE
'DoubleVector'

%OPTIMIZATION.NUMBER
10

%INITIAL.SAMPLE.SIZE
40

%MAXIMUM.ITERATIONS
150

%STALL.ITERATIONS
20

// continue.
```

```
%CONSTRAINT.TOLERANCE
1.0e-5

%USE.CYCLIC.WEI
'true'

%PROBLEM.TYPE
'CircularPltFGM'

%SIGMA.TYPE
'KFCV'

%PENALTY.METHOD
'Adaptive'

%END
```

Fig. 16 Optimization data input file of the Circular FGM plate problem

```
%MATERIAL
2

%MATERIAL.ISOTROPIC
2
1 70e9    0.30    0.00    0.00
2 380e9   0.30    0.00    0.00

%MATERIAL.DENSITY
2
1 2707
2 3800

%FGM.MATERIALS
2
1
2

%FGM.MODEL
'Mori-Tanaka'

%FGM.VOLUME.DISTRIBUTION
'BSpline'

%FGM.CONTROL.POINT.NUMBER
5

%FGM.THICKNESS.RANGE
0.01 0.0001 0.05
```

Fig. 17 Problem data input file of the Circular FGM plate problem

6 Applications

In this section, the capabilities of BIOS are assessed over several well-known test functions and three structural engineering problems to highlight the potential of SAO in increasing the efficiency of the optimization process.

Here, two surrogate models were considered, namely RBF with the Fivefold Cross Validation (5-FCV) technique to define the width parameter and Kriging with a Gaussian correlation function and the MLE approach to define its hyperparameters. The lower and upper bounds of the Kriging hyperparameters were set to $\log(\theta_L) = -1$ and $\log(\theta_U) = 2$, respectively. In addition, two different criteria to choose the new infill points were investigated, the EI and the WEI. In the latter, the cycling approach considered weights $w \in [0.2, 0.35, 0.5]$.

Thus, four different algorithms were investigated. They were named RBF-EI, RBF-WEI, KRG -E I, and KRG-WEI. The optimization of the acquisition functions (EI and WEI) was performed using the DE algorithm. Table 1 presents the values of the parameters used for optimization the acquisition functions and the likelihood estimator (for fitting the Kriging model). Moreover, the parameters used for the PSO are also shown, which is employed in the conventional optimization carried out for comparison purposes in Sect. 6.4.

The SAO algorithms were terminated when the number of high-fidelity evaluations exceeded n_{max}, or when the algorithm failed to improve upon the optimal design for It_{stall} consecutive iterations. These parameters depend on the dimensionality and complexity of each problem. Since the EI and the MLE are cheap to evaluate functions, the size of the population and the number of iterations used by DE were increased to 100 and 500, respectively, with no additional stopping criterion.

Due to the stochastic nature of the optimization algorithms used in this paper, each problem was run 10 times. The comparison between the SAO algorithms is given in terms of accuracy and efficiency.

To assess the accuracy of the optimum found, the NRMSE was considered, which corresponds to the average of the Normalized Root Mean Squared Error (NRMSE) of all runs. The error of each run was evaluated by comparing the best sampling point found by the algorithm and the true optimum.

To assess the efficiency, the average number of high fidelity evaluations (n_{ev}) performed until reaching a stopping criterion was considered. In the case of the structural

Table 1 Parameters for the bio-inspired algorithms

General	N_p	60
	G_{max}	150
	G_{stall}	20
DE	Differentiation method	Current-to-Best
	Crossover rate (C_r)	0.80
	Scale factor (F)	0.85
PSO	Topology	Global
	Inertia (w)	0.70
	Cognitive factor (c_1)	1.50
	Social factor (c_2)	1.50
	Mutation (ρ_{mut})	0.05

engineering problem, the gain in computational efficiency was measured using:

$$\beta = \frac{T_{slw}}{T_{alg}} \tag{25}$$

where T_{alg} is the average time spent by a given algorithm and T_{slw} is the time spent by the slowest algorithm. Thus, β represents how much faster a given algorithm is compared to the least efficient one.

All numerical computations were performed on a computer with two processors Xeon® with 2.8 GHz clock speed and 32 GB of RAM, each with 10 cores, resulting in 20 cores in total. In BIOS, paralellization affects essentially two procedures: the evaluation of the population in heuristic algorithms and the evaluation of the initial sample for SAO methods.

6.1 Branin function

The first example is the minimization of the Branin function, a two-dimensional problem commonly employed in SBO (Jones et al. 1998; Sobester et al. 2005; Forrester et al. 2008; Song et al. 2019):

$$f(\mathbf{x}) = \left(x_2 - \frac{5.1}{4\,\pi^2}\,x_1 + \frac{5}{\pi}\,x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\,\pi}\right)\cos x_1 \;+ 10 \tag{26}$$

Figure 18 depicts its surface. This function has 3 global minima: $f(\boldsymbol{x}) = 0.3979$ at $\boldsymbol{x} = [-\pi, 12.275]$, $\boldsymbol{x} = [\pi, 2.275]$, and $\boldsymbol{x} = [9.425, 2.475]$.

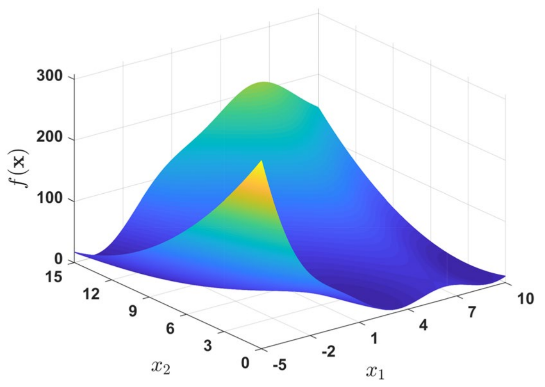


Fig. 18 Branin function

In this example, for replication purposes, nine initial sampling points are generated via the Hammersley sequence, a deterministic sampling method. In this case, the algorithm is terminated if it does not improve its best solution for $It_{\mathrm{stall}} = 10$ consecutive iterations. In addition, the algorithm is also stopped when the maximum number of samples is achieved, which is set to $n_{\max} = 40$.

Figure 19 illustrates one of the optimization runs using the RBF-EI approach. In the first iteration, the EI maximization leads to a point at the edge of the design space, at $\boldsymbol{x} = [10, 0]$. For the next several iterations, the algorithm tends to exploit the region around one of the global optima (right corner). After some more exploration and exploitation of the optimum region (note how the solution begins to cluster around the optimum), one of the optimum designs is found at Iteration 18.

Table 2 compares different SAO algorithms on the Branin function minimization. The Kriging-based algorithms performed exceptionally well, but the RBF also presented small errors. While the conventional approaches would require hundreds or even thousands of evaluations of the true function, these methods were able to find very good results with less than 50 evaluations. The results found are compared to the ones shown by Jones et al. (1998), where the EGO algorithm is applied, showing that the results are in agreement with what is expected. It is worth to note that the EGO algorithm uses a different stopping criterion, related to the maximum Expected Improvement (EI) found in each iteration, which may explain the difference in n_{ev}.

Finally, Fig. 20 presents the boxplots showing the NRMSE of each approach. The blue × depicts the average response. The results show the robustness of the SAO approaches, which are able to reliably find individuals very close to the global optimum.

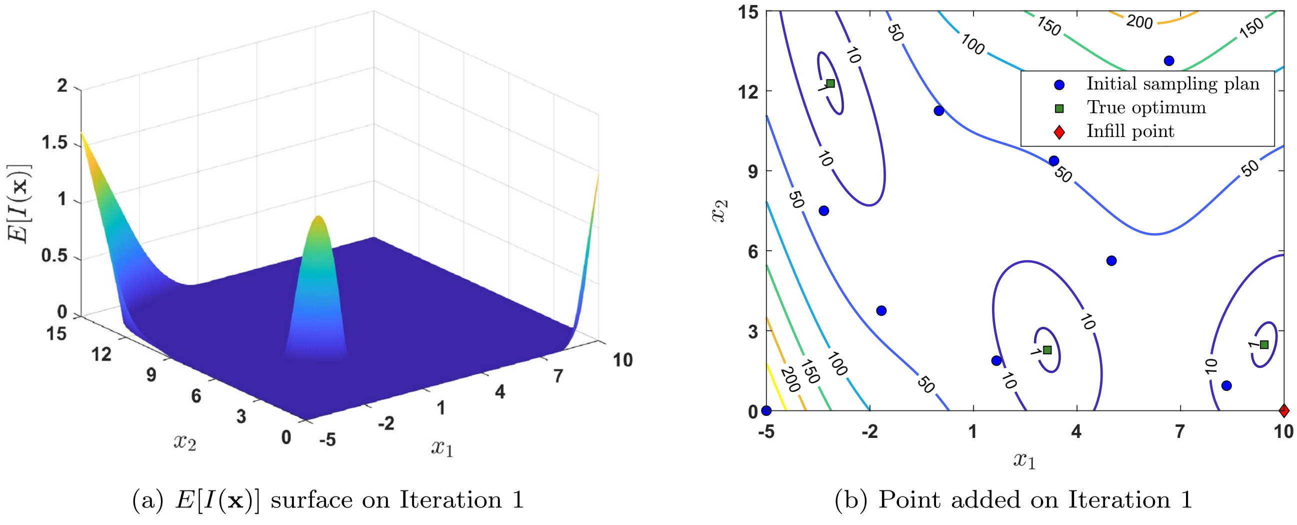
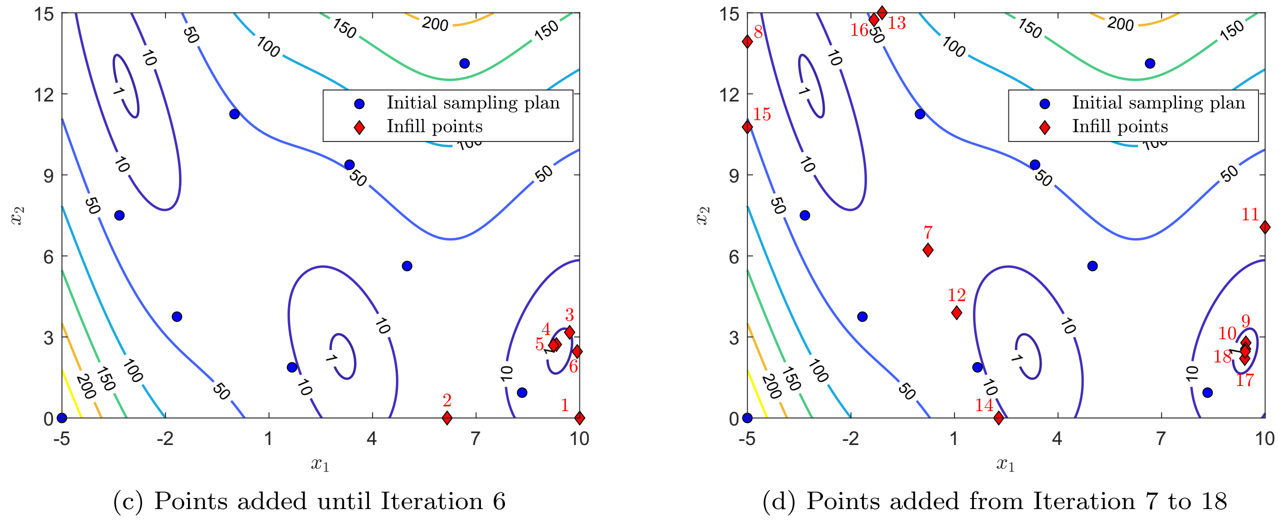
6.2 Hartmann 6 function

The Hartmann 6 function is a six-dimensional problem also commonly employed in SBO (Jones et al. 1998; Sobester et al. 2005). The function is given by:

$$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \,\exp\left(-\sum_{j=1}^{6} A_{ij}\,(x_j - P_{ij})^2\right) \tag{27}$$

where $\boldsymbol{\alpha} = [1.0, 1.2, 3.0, 3.2]$ and:

$$\boldsymbol{A} = \begin{bmatrix} 10.0 & 0.05 & 3.00 & 17.0 \\ 3.00 & 10.0 & 3.50 & 8.00 \\ 17.0 & 17.0 & 1.70 & 0.05 \\ 3.50 & 0.10 & 10.0 & 10.0 \\ 1.70 & 8.00 & 17.0 & 0.10 \\ 8.00 & 14.0 & 8.00 & 14.0 \end{bmatrix}^T \tag{28}$$



(a) $E[I(\mathbf{x})]$ surface on Iteration 1

(b) Point added on Iteration 1

(c) Points added until Iteration 6

(d) Points added from Iteration 7 to 18

Fig. 19 Addition of infill points for the Branin function

Table 2 Average results for the Branin function

Method	$\overline{\text{NRMSE}}$	n_{ev}
RBF-EI	0.50%	37
RBF-WEI	0.71%	33
KRG-EI	0.01%	37
KRG-WEI	0.03%	40
EGO (Jones et al. 1998)	0.20%	28

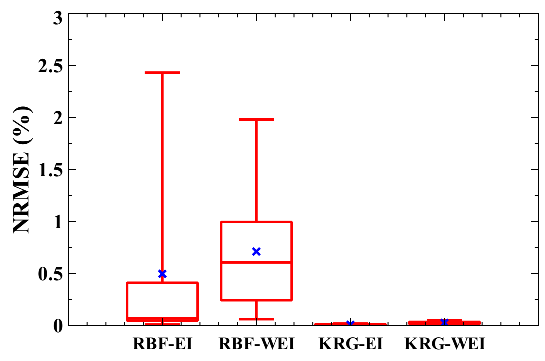


Fig. 20 NRMSE boxplot for the Branin function

$$\boldsymbol{P} = 10^{-4}\begin{bmatrix} 1312 & 2329 & 2348 & 4047 \\ 1696 & 4135 & 1451 & 1451 \\ 5569 & 8307 & 3522 & 8732 \\ 124 & 3736 & 2883 & 5743 \\ 8283 & 1004 & 3047 & 1091 \\ 5886 & 9991 & 6650 & 381 \end{bmatrix} \tag{29}$$

This function has 6 local minima. The global optimum is located at $\boldsymbol{x} = [0.202, 0.150, 0.477, 0.275, 0.312, 0.657]$, where $f(\boldsymbol{x}) = -3.322$. Here, the initial sample consists of 30 points, selected via the LHS$_{20}$ method. In this example, the stopping criteria are given by $n_{\max} = 150$ and $It_{\text{stall}} = 10$. The results using BIOS are shown in Table 3, along with the results found by Jones et al. (1998). The best results were found with the KRG-WEI approach, where a very small NRMSE was found. Also, the RBF-based approaches seem to be slightly more efficient, as they required fewer evaluations of the true function.

Figure 21 presents the boxplots of the NRMSE for this problem. Due to the higher dimensionality and the multiple local minima, this function is more complex and harder to optimize. The KRG-WEI approach stands out as the most robust method.

6.3 Kitayama's constrained problem

This problem was shown by Kitayama et al. (2010) as a illustrative example for constrained SAO. The objective function is the negative of a two-dimensional sphere function, centered at $\boldsymbol{x} = [1.0, 0.5]$:

$$f(\mathbf{x}) = -\left(x_1 - 1.0\right)^2 - \left(x_2 - 0.5\right)^2 \tag{30}$$

This function is subjected to the following constraints:

$$g_1(\mathbf{x}) = \frac{\left[\left(x_1 - 3\right)^2 + \left(x_2 + 2\right)^2\right]\exp(-x_2^7)}{12} - 1 \le 0 \tag{31}$$

$$g_2(\mathbf{x}) = \frac{\left(x_1 + 0.5\right)^2 - \left(x_2 - 0.5\right)^2}{0.2} - 1 \le 0 \tag{32}$$

Table 3 Average results for the Hartmann 6 function

Method	$\overline{\text{NRMSE}}$	n_{ev}
RBF-EI	0.71%	74
RBF-WEI	2.47%	72
KRG-EI	1.80%	79
KRG-WEI	0.04%	79
EGO (Jones et al. 1998)	1.90%	84

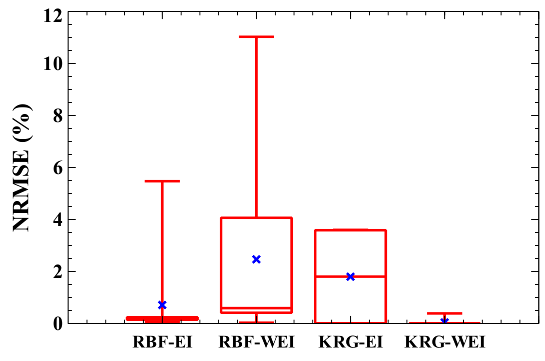


Fig. 21 NRMSE boxplot for the Hartmann 6 function

$$g_3(\mathbf{x}) = \frac{10\,x_1 + x_2}{7} - 1 \le 0 \tag{33}$$

In this problem, all constraints will be approximated by surrogate models, similar to the objective function, and the feasibility function proposed by Tutum et al. (2014), Eq. (19), is used to deal with the approximate constraints. Figure 22 presents the constrained design space, where the unfeasible space is being highlighted. The feasible region is nonconvex and consists of two separate small regions in the design space.

The problem has a local optimum at $\boldsymbol{x} = [0.262, 0.122]$, with $f(\boldsymbol{x}) = -0.687$, but the global optimum is at $\boldsymbol{x} = [0.202, 0.833]$, with $f(\boldsymbol{x}) = -0.748$. For constrained optimization, it is interesting to increase the initial sampling size, especially for functions with multiple constraints and such complex feasible space. It is very hard for SBO to identify the feasible region if there are no points in it.

The initial model is built using 12 sampling points, generated via the Hammersley sequence. Here, the stopping criteria for the SAO algorithms are given by $n_{\max} = 80$ and

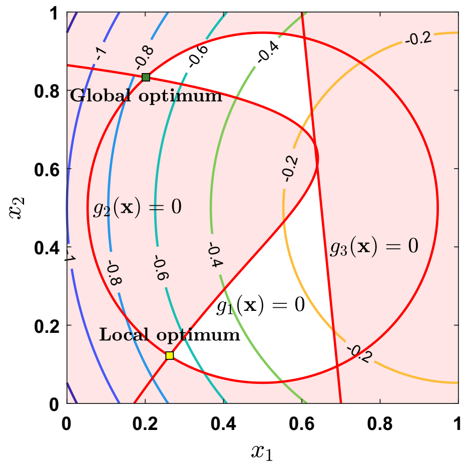
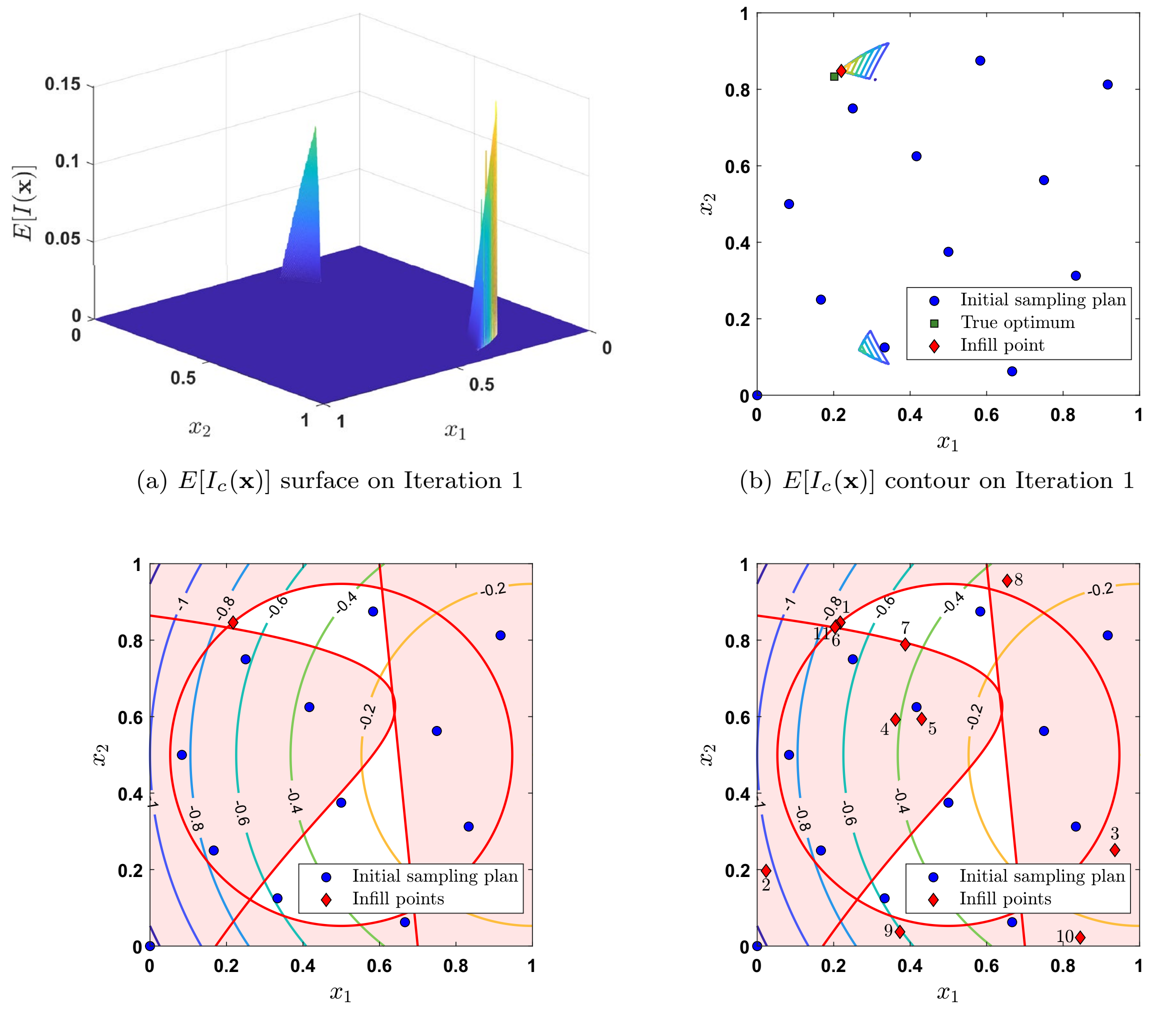


Fig. 22 Kitayama's problem (Kitayama et al. 2010) constrained space



(a) $E[I_c(\mathbf{x})]$ surface on Iteration 1 (b) $E[I_c(\mathbf{x})]$ contour on Iteration 1

(c) Point added on Iteration 1 (d) Points added until Iteration 11

Fig. 23 Addition of infill points for Kitayama's constrained problem

$It_{\text{stall}} = 20$. Figure 23 illustrates one of the optimization runs using RBF-EI. Note that only three points are in the feasible design space in the initial sample and one of those is actually very close to the local optimum. Nevertheless, the algorithm is able to select, in the first iteration, an infill point very close to the global optimum. The actual global optimum is found on Iteration 11, and the algorithm stops after 20 iterations with no noticeable improvement upon the objective function. One should note that, on very small feasible spaces, it might be hard to locate a non-zero region for the Constrained Expected Improvement defined by Eq. (21), which further explains why heuristic algorithms are a good choice for these applications.

A comparison of the performance of the different SAO algorithms for this problem is presented on Table 4. Once again, the Kriging-based algorithms performed exceptionally well. The RBF also presented minor errors and, as will become more evident in the next section, at a lower computational cost. The results are compared to the ones found by Kitayama et al. (2010) using their proposed SAO algorithm.

Table 4 Average results for Kitayama's constrained problem

Method	$\overline{\text{NRMSE}}$	n_{ev}
RBF-EI	0.28%	50
RBF-WEI	0.33%	47
KRG-EI	0.02%	38
KRG-WEI	0.00%	37
Kitayama et al. (2010)	0.20%	50

Finally, Fig. 24 presents boxplots with the performance of each algorithm, in terms of the NRMSE. Again, the algorithms seem to be very robust, being able to find the

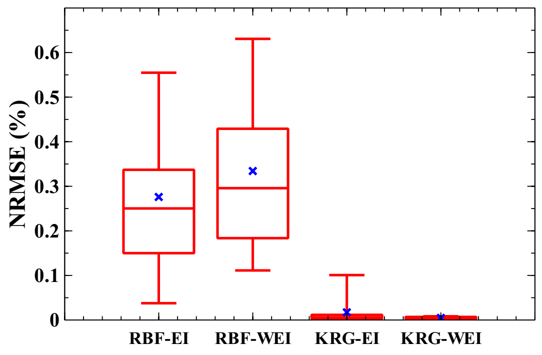


Fig. 24 NRMSE boxplot for Kitayama's constrained problem

optimum with high reliability, particularly the ones based on Kriging.

6.4 Optimization of a unidirectional FG circular plate

In this section, the fundamental frequency of a simply supported FG circular plate made of Al/Al$_2$O$_3$ is maximized, while subjected to mass and ceramic fraction constraints. The material gradation is defined by a B-Spline function with 9 control points symmetrically spread along the plate thickness. Thus, in addition to the plate thickness, 5 designs variables related to the volume fraction distribution are considered. Material properties are $E_m = 90$ GPa, $\rho_m = 2707$ kg/m^3, $E_c = 380$ GPa, $\rho_c = 3800$ kg/m^3, $\nu_m = \nu_c = 0.30$. The effective material properties are evaluated via the Mori-Tanaka scheme (Shen 2009; Do et al. 2019; Ribeiro et al. 2020). The plate geometry and boundary conditions are shown in Fig. 25. Rigid body motion is prevented by constraining the u and v displacements in two radially symmetric points on the $x-y$ plane.

The optimization problem is defined as:

$$\begin{cases} \text{maximize} & \omega(\mathbf{x}) \\ \text{subjected to} & g_1(\mathbf{x}) \le 0 \\ & g_2(\mathbf{x}) \le 0 \\ \text{with} & h_{\min} \le x_1 \le h_{\max} \\ & 0 \le x_i \le 1 \quad \text{for } i = 2, 3, \dots 6 \end{cases} \tag{34}$$

where x_1 is the plate thickness, x_2, x_3, ..., x_6 are the control points, $h_{\min} = 0.01$ m, and $h_{\max} = 0.05$ m. The constraints $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ represent the mass and ceramic fraction constraints, respectively given by:

$$g_1(\mathbf{x}) = \pi R^2 \int_{-h/2}^{h/2} \rho(z)\ dz - m_{\max} \le 0 \tag{35}$$

$$g_2(\mathbf{x}) = \frac{1}{h} \int_{-h/2}^{h/2} V_c\, dz - \overline{V}_{c,max} \le 0 \tag{36}$$

Here, $m_{\max} = 100$ kg and $V_{c,\max} = 35\%$. Since the evaluation of these constraints is cheap, there is no need to approximate them. The implementation of this problem is described in Sect. 5. The structural analysis is performed using the Finite element AnalysiS Tool (FAST), an in-house software, but any software that can be integrated with C++ can be employed. A 1024-element cubic NURBS mesh is employed as shown in Fig. 26. The IGA formulation employed, considering the First-order Shear Deformation Theory (FSDT), can be found in Maia et al. (2021).

In this problem, 40 initial sampling points are generated via the LHS$_{20}$ method and DE is used to maximize the EI or the WEI. The stopping criteria are given by $n_{\max} = 150$ and $It_{\text{stall}} = 20$. Along with SAO algorithms, the optimization is conducted using two conventional meta-heuristics suitable for dealing with continuous optimization, PSO

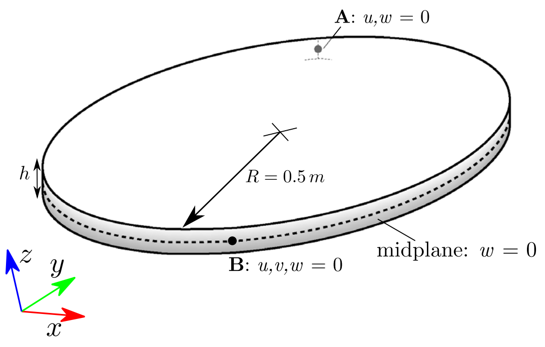


Fig. 25 Simply supported circular plate

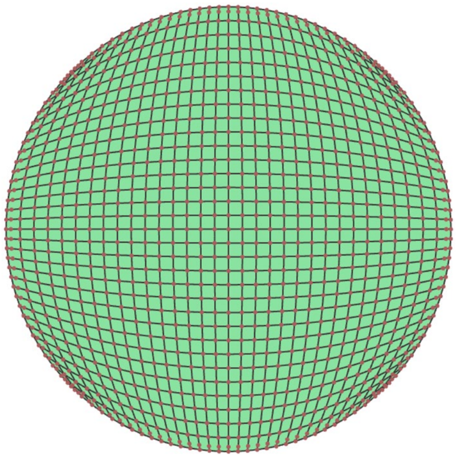


Fig. 26 Circular plate NURBS mesh

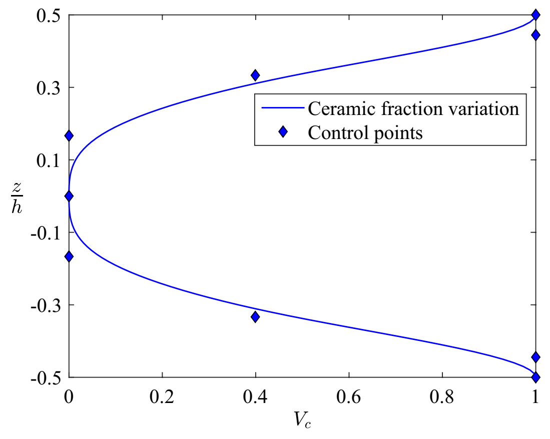


Fig. 27 Ceramic volume fraction variation of optimum design

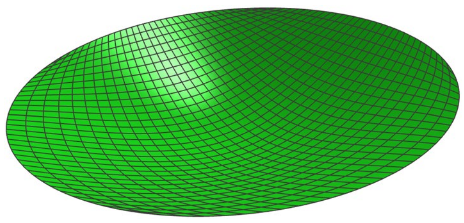


Fig. 28 First vibration mode of optimum design

Table 5 Average results for the FG circular plate

Method	$\overline{\text{NRMSE}}$	n_{ev}	β	
			Serial	Parallel
PSO	0.06%	5646	1.33	21.25
DE	0.00%	7620	1.00	15.96
RBF-EI	1.09%	64	100.93	249.86
RBF-WEI	0.50%	65	97.35	259.17
KRG-EI	0.00%	63	53.65	202.37
KRG-WEI	0.00%	61	58.89	203.06

and DE. The values of the optimization parameters considered in this problem are presented in Table 1.

The global optimum of this problem is found at $\boldsymbol{x} = [0.0412, 1.0, 1.0, 0.4, 0.0, 0.0]$, with $\omega(\mathbf{x}) = 2077.33$ Hz. Figure 27 shows the distribution of the volume fraction of the ceramic through the thickness of the optimum design. The first vibration mode for the optimum design is shown in Fig. 28. This solution is validated using ABAQUS with a 4800-element mesh of quadratic shell elements and reduced integration (S8R). The result obtained was $\omega(\mathbf{x}) = 2069.60$ Hz, which is only 0.37% smaller than the frequency calculated by FAST.

Table 5 presents a comparison on the performance of different SAO algorithms and two conventional algorithms.

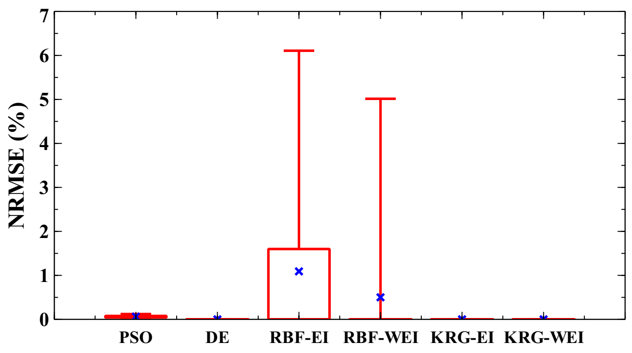


Fig. 29 NRMSE boxplot for the maximization of $\omega(\mathbf{x})$

The parameter β is given with respect to the computational cost of DE, which was the least efficient algorithm in this example. The paralellization procedure is performed by taking advantage of all 20 cores of the local machine.

Here, it is clear that the SAO algorithms are much more efficient than the conventional optimization due to the much lower number of evaluations (n_{ev}) of the high fidelity function carried out using IGA. Furthermore, Kriging-based methods were able to achieve a lower $\overline{\text{NRMSE}}$ than the conventional optimization using PSO. The RBF-based approaches also achieved small errors, while presenting even higher gains in efficiency. It is important to highlight that the serial SAO methods presented an outstanding performance even when compared to the conventional optimization using parallelization.

Finally, Fig. 29 presents the boxplots of the NRMSE for this problem. The robustness of KRG-EI and KRG-WEI approaches is evident, as they were able to find the global optimum in all cases, similar to conventional approach using DE.

6.5 Optimization of a tridirectional FG square plate

In this section, the maximization of the buckling load factor of a simply supported SUS304/Si_3N_4 FG square plate is performed, considering a ceramic volume fraction constraint. Here, the material gradation is given by a tridirectional B-Spline function with 144 control points, and material properties are $E_m = 201.04$ GPa, $\nu_m = 0.3262$, $E_c = 348.43$ GPa, and $\nu_c = 0.24$. The gradation is symmetric in all three directions, and the effective material properties are evaluated via the Mori-Tanaka scheme (Do et al. 2020). The plate geometry and boundary conditions are shown in Fig. 30.

Two different meshes are used for distinct purposes: the design and the analysis mesh. The design mesh is used to define the material gradation, and is given by a 3D $3 \times 3 \times 1$ cubic NURBS mesh. On the other hand, the analysis mesh is used to perform the structural analysis, and is given by a 2D 16×16 cubic NURBS mesh. These meshes are shown

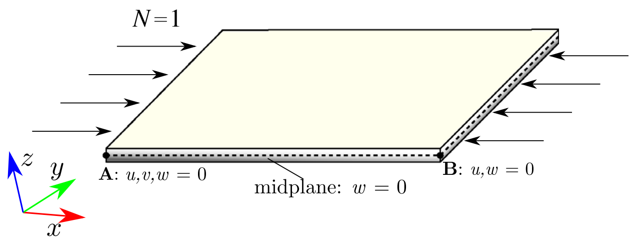


Fig. 30 Simply supported square plate

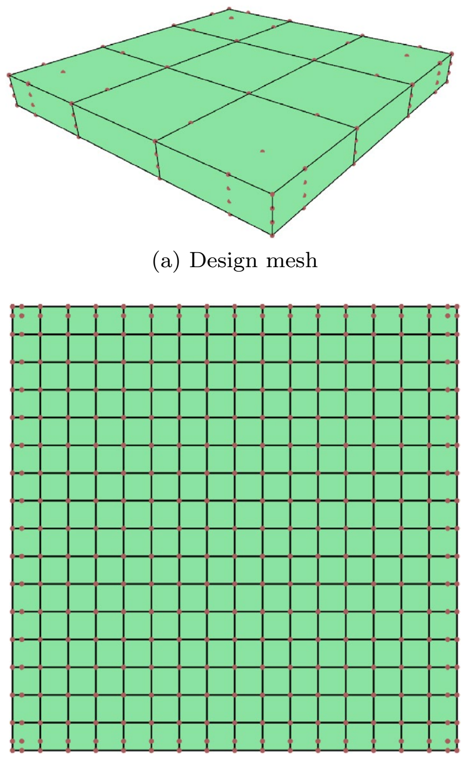


Fig. 31 Meshes used for the square plate problem

in Fig. 31. The design variables of this problem are the control points for the design mesh. This mesh has 144 control points, but due to symmetry in all three directions, the optimization problem has 18 design variables. It should be noted that this is a significant number of variables for SBO problems (Díaz-Manríquez et al. 2011). The optimization problem is defined as:

$$\begin{cases} \text{maximize} & \lambda_n(\mathbf{x}) \\ \text{subjected to} & g_1(\mathbf{x}) \leq 0 \\ \text{with} & 0 \leq x_i \leq 1 \quad \text{for } i = 1, 2, \dots 18 \end{cases} \tag{37}$$

where the constraint $g_1(\mathbf{x})$ represents the ceramic fraction constraint:

$$g_1(\mathbf{x}) = \frac{1}{V} \int V_c \, dV - \overline{V}_{c,max} \leq 0 \tag{38}$$

with $\overline{V}_{c,max} = 30\%$. Again, this constraint is not approximated by a surrogate model since its evaluation is not expensive. Moreover, the non-dimensional buckling load is given by $\lambda_n = N_{cr}\, a^2/(\pi^2 D_c)$, where $D_c = E_c h^3/[12\,(1 - \nu_c^2)]$.

This problem was first proposed by Do et al. (2020), who used a Deep Neural Network to improve the efficiency of the optimization process. Thus, 10,000 sampling points were used to train and validate the model. The authors evaluated the buckling load using a Higher-order Shear Deformation Theory (HSDT). In the present work, SAO is employed using BIOS, and 40 initial sampling points are generated via the LHS$_{20}$ method and, once again, the DE is used to maximize the acquisition function. The stopping criteria are $n_{\max} = 150$ and $It_{\text{stall}} = 20$, and conventional meta-heuristic will also be used to carry out the optimization process. The optimization parameters are shown in Table 1.

Table 6 shows the optimum design for this problem. The value for the buckling load using the HSDT was taken from Do et al. (2020), while the FSDT value was found using an in-house analysis software. It is important to note that the optimum found in this work presents a slightly higher (1.5%) buckling load when compared to the reference solution using the FSDT (Do et al. 2020). Figure 32 shows that material distribution for the optimum design is very complex in the plate domain. Note that, since load is applied in the x axis, the optimum design favors the addition of ceramic material in the loaded face. The buckling mode for the optimum design is depicted in Fig. 33.

The performance evaluation of SAO variants and conventional algorithms is presented in Table 7. The parameter β is given with respect to the computational cost of the conventional optimization with DE algorithm, since it was the least efficient alternative in this example. The boxplots of the NRMSE for this problem are presented in Fig. 34.

Table 6 Optimum design for the tridirectional FG plate

Source	Design variables (x)																		Buckling load	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	HSDT	FSDT
Do et al. (2020)	1	1	0	1	1	0	0	0.526	1	0	0	0	0	0	0	0	0	0	2.906	2.832
This work	1	0	1	1	0.023	1	1	0	1	0	0	0	0	0	0	0	0	0		2.876

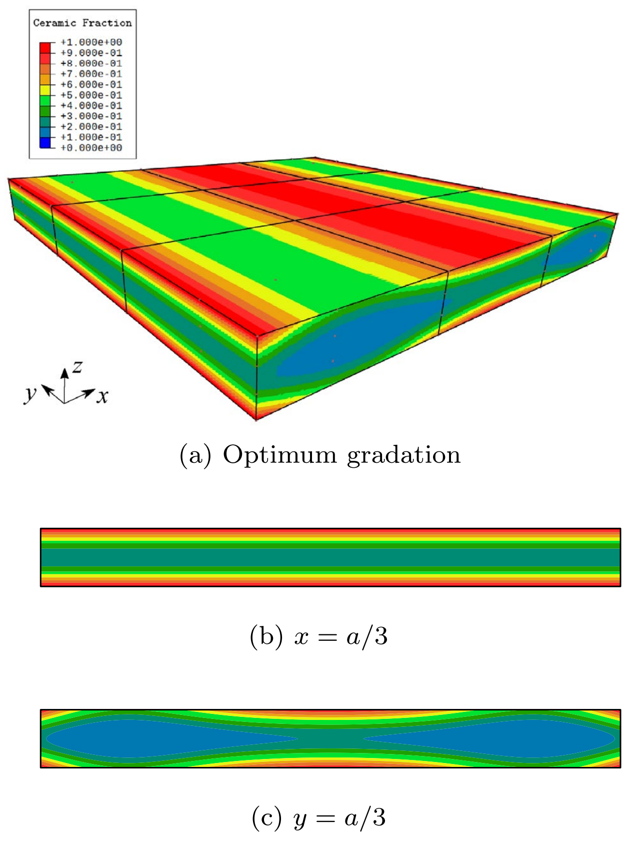


Fig. 32 Optimum design for the square plate problem

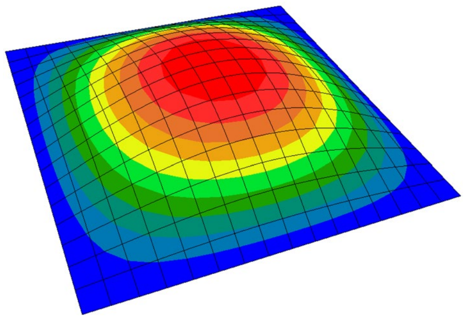


Fig. 33 Critical buckling mode for the square plate

Table 7 Average results for the FG square plate

Method	$\overline{\text{NRMSE}}$	n_{ev}	β	
			Serial	Parallel
PSO	0.45%	6786	1.30	11.73
DE	0.05%	9060	1.00	9.36
RBF-EI	0.50%	61	38.52	58.03
RBF-WEI	0.77%	61	38.08	57.87
KRG-EI	0.04%	64	15.19	46.62
KRG-WEI	0.01%	62	16.42	44.23

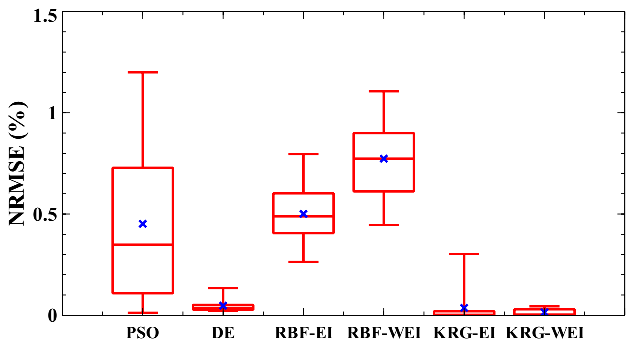


Fig. 34 NRMSE boxplot for the maximization of $\lambda_n(\mathbf{x})$

The SAO algorithms presented superior efficiency in comparison to the conventional optimization algorithms, even when comparing the serial version of SAO algorithm with parallel version of conventional algorithms. Again, Kriging-based methods outperform the PSO algorithm in terms of accuracy and achieve similar results in comparison to the DE algorithm. RBF-based methods were the most efficient alternative, and also presented good accuracy. Finally, the number of required structural analyses was drastically reduced for all SAO algorithms with respect to the conventional optimization.

6.6 Optimization of a tridirectional FG panel with a cutout

In this final problem, the multi-objective optimization of a SUS304/Si_3N_4 FG panel with a circular cutout is performed. Again, material gradation is described by a tridirectional B-Spline function, but now with 256 control points, symmetric in all three directions. Material properties are the same as the ones from the previous example. Equivalent properties are evaluated via the Mori-Tanaka scheme.

The composite panel is shown in Fig. 35. It has length $L = 1$ m, bending radius $R = 5$ m, and $\theta = 0.1$ rad. The circular cutout is located in the center of the shell with radius $r = 0.1$ m.

The analysis is performed using a finite element model consisting of 1536 quadratic eight-node shell elements with reduced integration. Due to the cutout, the model should be well-refined to guarantee that stresses are accurately computed. Figure 36 shows how the membrane force in x-direction due to a unit distributed load (N_x) for the case of $\overline{V}_c = 50\%$. We see that there are major stress concentrations near the hole, but this refined mesh allows for an accurate approximation of internal forces. Figure 37 depicts the buckling mode for this panel.

Again, a design mesh is considered to assist in defining material volume fractions in the structure's domain. This time, the design mesh is given by a 3D $5 \times 5 \times 1$ cubic

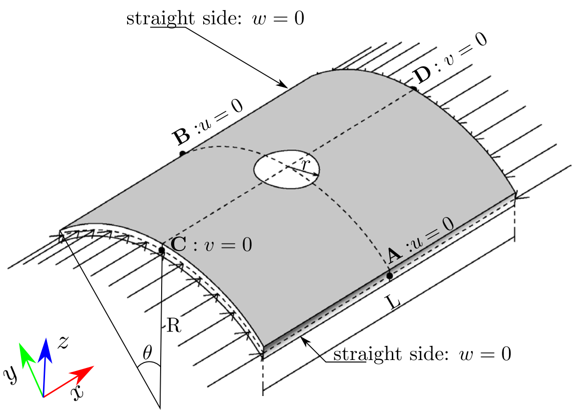


Fig. 35 FG panel with a circular cutout

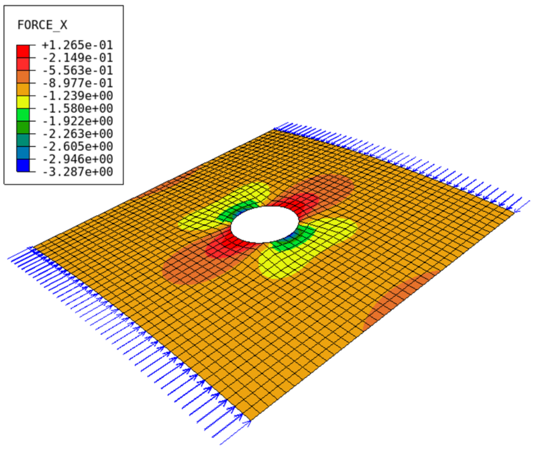


Fig. 36 Force in x-direction for unit loading

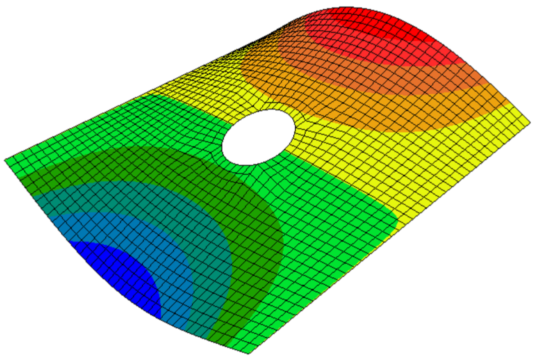


Fig. 37 Critical buckling mode for the FG panel

NURBS mesh parametrized in coordinates x and y, and in shell thickness, as depicted in Fig. 38. Note that the cutout is not explicitly included in the design mesh but is nevertheless taken into account by the analysis model. There are 256 control points but, due to symmetry in the

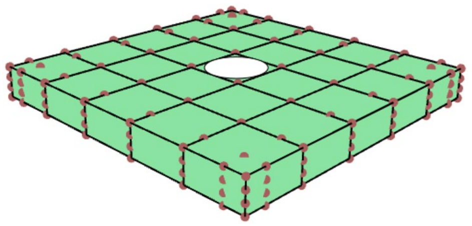


Fig. 38 Design mesh for the FG panel

three directions, 32 design variables are considered in the optimization problem. It is worth pointing out that this is a very high number of variables for SBO (Díaz-Manríquez et al. 2011).

This time, we perform the multi-objective optimization of the FG panel. Here, two objectives are considered: maximization of the buckling load λ_n and minimization of the total cost C_t. The optimization problem is defined as:

$$\begin{cases} \text{minimize} & f_1(\mathbf{x}), f_2(\mathbf{x}) \\ \text{with} & 0 \le x_i \le 1 \quad \text{for } i = 1, 2, \dots 32 \end{cases} \tag{39}$$

where $f_1 = -\lambda_n(\mathbf{x})$ and $f_2 = C_t(\mathbf{x})$. The total cost is evaluated by:

$$C_t = C_c \rho_c \frac{1}{V} \int V_c dv + C_m \rho_m \frac{1}{V} \int V_m dv \tag{40}$$

where $C_c = 50$ USD/kg and $C_m = 3$ USD/kg are the costs for the ceramic and metal, respectively (Franco Correia et al. 2021), and $\rho_c = 2730$ kg/m^3 and $\rho_m = 8000$ kg/m^3 are the material densities for the ceramic and metal, respectively. The non-dimensional buckling load is evaluated by the same expression as the previous example.

The multi-objective optimization is solved using the Weighted Compromise Programming (WCP) method (Athan and Papalambros 1996; Rouhi et al. 2015; Barroso et al. 2017), where multiple single-objective problems are defined considering weighted objectives. Thus, the final objective function is defined as:

$$f(\mathbf{x}) = \left[w \frac{f_1 - f_{1,min}}{f_{1,max} - f_{1,min}} \right]^m + \left[(1-w) \frac{f_2 - f_{2,min}}{f_{2,max} - f_{2,min}} \right]^m \tag{41}$$

where $m = 2$, $f_{1,min}$ and $f_{2,min}$ are the minimum objectives for f_1 and f_2, and $f_{1,max}$ and $f_{2,max}$ are the maximum objectives for f_1 and f_2. Since $f_1(\mathbf{x}) = -\lambda_n(\mathbf{x})$ and $f_2(\mathbf{x}) = C_t(\mathbf{x})$, we can find $f_{1,min} = -2.1406$ and $f_{2,max} = 2639.6916$ USD for the isotropic Si_3N_4 shell, and we can find $f_{1,max} = -1.1700$ and $f_{2,min} = 464.1216$ USD for the isotropic SUS304 shell. By continuously changing the weight factor w, one is able to draw the Pareto front of the multi-objective optimization problem.

In this problem, the SAO algorithms use 60 initial sampling points defined by the LHS$_{20}$ method. Here, only the Weighted Expected Improvement variant of Kriging and RBF are used as surrogates. These models are chosen since they perform well in other examples. The stopping criteria are $n_{\max} = 150$ and $It_{\text{stall}} = 20$. In addition to the SAO algorithms, the Differential Evolution meta-heuristic will also be used to carry out the optimization process, using the optimization parameters shown in Table 1. Due to the high cost of the FE simulations, only parallel optimizations are considered in this case.

The Pareto front obtained by each algorithm is presented in Fig. 39. The Differential Evolution (DE) and Kriging obtained similar optimal designs, while RBF obtained slightly worse solutions. Table 8 presents the optimal designs and their objective functions and ceramic fraction ($\overline{V}_c$) for each point in Pareto's front.

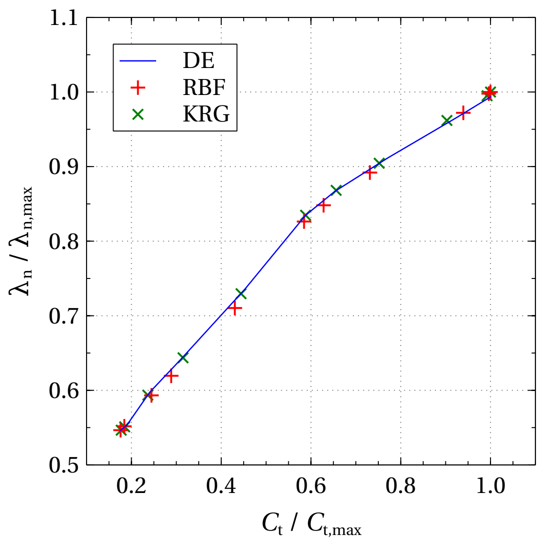


Fig. 39 Pareto's front for the FG panel

Computation times are shown results are shown in Table 9. Here, the parameter β is given with respect to the computational cost of parallel DE. The time spent by RBF and Kriging was 29 and 14 times faster than DE, respectively. Again, the RBF algorithm has superior efficiency but inferior accuracy in comparison to the Kriging algorithm. Note that RBF performed 2.1 times faster than Kriging for this problem, while being 1.3 times faster for the problem in Sect. 6.5. The efficiency of RBF in comparison to Kriging increases as bigger sample set are used since the training of RBF models has a lower computational complexity in comparison to Kriging.

The optimal gradations for three different designs of Pareto's front are illustrated in Fig. 40 for the design mesh. As the weight w decreases, ceramic becomes more present in the design, since the buckling-related objective becomes more important than the cost-related one.

7 Conclusion

This paper presented BIOS, a framework for design optimization using nature inspired search and Sequential Approximated Optimization, with focus on the optimization of composite structures. The framework architecture was described in detail, through exposition of its modules

Table 9 Processing time for the FG panel

Method	Time per optimization (s)	β
DE	6023	1.0
RBF	209	28.8
KRG	430	14.0

Table 8 Results for the FG panel

w	Design variables (**x**)	$f(\mathbf{x})$	C (USD)	λ_n	$\overline{V}_c$ (%)
0.0	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	0.0000	2639.6916	2.1406	100
0.1	[1 1 1 1 0.8683 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0.1229 1 1 0.4519]	0.0099	2619.5050	2.1298	99
0.2	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 0.6708 1 0 1 1 1 0.3646]	0.0357	2383.2830	2.0589	88
0.3	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0.2792 0 0 1 1 1 1]	0.0657	1985.1140	1.9361	70
0.4	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0.4633 0 0 0 0 0 0 0 0 0.3534 1 1 0]	0.0848	1731.9480	1.8583	58
0.5	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	0.0957	1552.1820	1.7869	50
0.6	[1 1 1 1 0 1 0.8384 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	0.0951	1172.6030	1.5616	33
0.7	[1 1 1 1 0 0.007 0.6586 0 0 0 0.2086 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	0.0696	831.4536	1.3777	17
0.8	[0 1 0.8 1 0 0 0 0 0 0 0 0 0 0.7047 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	0.0356	624.4876	1.2705	7
0.9	[0.3262 0 0 0 0 0 0 0.2054 0 0 0 0 0 0 0.0467 0.2831 0.2757 0 0 0 0 0 0 0 0 0 0 0 0.1369 0 0 0.2886]	0.0099	488.4376	1.1796	1
1.0	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	0.0000	464.1216	1.1700	0

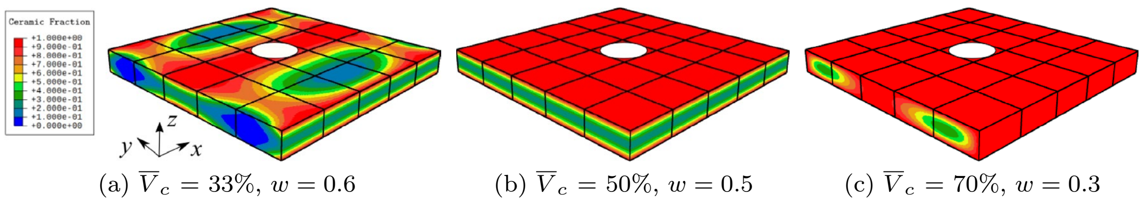


Fig. 40 Optimal gradation for the FG panel

and discussion regarding its classes responsibilities and most important methods.

The theoretical aspects concerning the SAO algorithms based in Radial Basis Functions and Kriging and their impact on the system architecture were also discussed. The effectiveness of both strategies were assessed using well-known benchmarks problems, for which excellent results were obtained. A multi-objective problem was also solved, successfully obtaining its Pareto front.

To assess the capabilities of BIOS in terms of efficiency, the optimization of functionally graded structures with costly objective functions were solved, combining parallel computing and surrogate modeling features presented here. The results show that the SAO algorithms outperform the conventional algorithms, such as PSO and DE, by orders of magnitude, with negligible differences in the optimum design.

BIOS is a powerful tool for sequential approximate methods, as well as for optimization of laminate and functionally graded structures. Moreover, BIOS can be employed in the solution of optimization problems from different engineering fields, requiring only minor modifications and adjustments to the code. It is worth emphasizing that the framework is cross-platform and open-source, which provides a resourceful alternative for future researchers in this lively field. For future works, the authors intend to extend the framework, by including new methods for solving constrained optimization problems, multi-objective optimization problems, and problems with discrete variables.

Funding This study was financed by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), and FUNCAP (Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico). The authors gratefully acknowledge the financial support provided by these agencies.

Data availability The datasets generated during the current study are available at BIOS repository: https://github.com/lmcv-ufc/BIOS.

Code availability BIOS source code is available at https://github.com/lmcv-ufc/BIOS.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Replication of results In addition to the material available as supplementary material, the authors believe that sufficient details have been provided in this paper, allowing the replication of experiments.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/, software available from http://tensorflow.org/

Alexandrescu A (2001) Modern C++ design: generic programming and design patterns applied. Addison-Wesley Longman Publishing Co., Inc., Boston

Arora JS (2017) Introduction to optimum design, 3rd edn. Academic Press, Cambridge

Athan TW, Papalambros PY (1996) A note on weighted criteria methods for compromise solutions in multi-objective optimization. Eng Optim 27(2):155–176. https://doi.org/10.1080/03052159608941404

Bagheri S, Konen W, Allmendinger R, Branke J, Deb K, Fieldsend J, Quagliarella D, Sindhya K (2017) Constraint handling in efficient global optimization. Proc Genet Evol Comput Conf 17:673–680. https://doi.org/10.1145/3071178.3071278

Barroso ES, Parente E, Cartaxo de Melo AM (2017) A hybrid PSO-GA algorithm for optimization of laminated composites. Struct Multidisc Optim 55(6):2111–2130. https://doi.org/10.1007/s00158-016-1631-y

Blank J, Deb K (2020) Pymoo: multi-objective optimization in python. IEEE Access 8:89497–89509

Bouhlel M, Bartoli N, Regis RG, Otsmane A, Morlier J (2018) Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method. Eng Optim 50(12):2038–2053. https://doi.org/10.1080/0305215X.2017.1419344

Bratton D, Kennedy J (2007) Defining a standard for particle swarm optimization. In: 2007 IEEE swarm intelligence symposium. https://doi.org/10.1109/SIS.2007.368035

Brochu E, Cora VM, de Freitas N (2010) A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv:1012.2599

Castro LD, Zuben FV (2002) Learning and optimization using the clonal selection principle. IEEE Trans Evol Comput 6(3):239–251. https://doi.org/10.1109/tevc.2002.1011539

Chen YT, Xiang S, Zhao WP (2014) Generalized multiquadrics with optimal shape parameter and exponent for deflection and stress of functionally graded plates. Appl Mech Mater 709:121–124. https://doi.org/10.4028/www.scientific.net/amm.709.121

Cho I, Lee Y, Ryu D, Choi DH (2016) Comparison study of sampling methods for computer experiments using various performance measures. Struct Multidisc Optim 55(1):221–235. https://doi.org/10.1007/s00158-016-1490-6

Chunna L, Hai F, Chunlin G (2020) Development of an efficient global optimization method based on adaptive infilling for structure optimization. Struct Multidisc Optim. https://doi.org/10.1007/s00158-020-02716-y

Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186(2–4):311–338. https://doi.org/10.1016/s0045-7825(99)00389-8

Díaz J, Cid Montoya M, Hernández S (2016) Efficient methodologies for reliability-based design optimization of composite panels. Adv Eng Softw 93:9–21. https://doi.org/10.1016/j.advengsoft.2015.12.001

Do D, Lee D, Lee J (2019) Material optimization of functionally graded plates using deep neural network and modified symbiotic organisms search for eigenvalue problems. Composites B 159:300–326. https://doi.org/10.1016/j.compositesb.2018.09.087

Do DT, Nguyen-Xuan H, Lee J (2020) Material optimization of tri-directional functionally graded plates by using deep neural network and isogeometric multimesh design approach. Appl Math Model 87(107):501–533. https://doi.org/10.1016/j.apm.2020.06.002

Durillo JJ, Nebro AJ (2011) jMetal: a Java framework for multi-objective optimization. Adv Eng Softw 42(10):760–771. https://doi.org/10.1016/J.ADVENGSOFT.2011.05.014

Díaz-Manríquez A, Toscano-Pulido G, Gómez-Flores W (2011) On the selection of surrogate models in evolutionary optimization algorithms. In: 2011 IEEE congress of evolutionary computation (CEC), pp 2155–2162

Forrester AI, Keane AJ (2009) Recent advances in surrogate-based optimization. Progress Aerosp Sci 45(1–3):50–79. https://doi.org/10.1016/j.paerosci.2008.11.001

Forrester AIJ, Sobester A, Keane AJ (2008) Engineering design via surrogate modelling: a practical guide. Wiley, Hoboken

Franco Correia V, Moita JS, Moleiro F, Soares CMM (2021) Optimization of metal-ceramic functionally graded plates using the simulated annealing algorithm. Appl Sci. https://doi.org/10.3390/app11020729

Gamma E, Helm R, Johnson R, Vlissides JM (1994) Design patterns: elements of reusable object-oriented software. Addison-Wesley Professional, Boston

Giunta AA, Eldred MS (2000) Implementation of a trust region model management strategy in the DAKOTA optimization toolkit. In: 8th symposium on multidisciplinary analysis and optimization. https://doi.org/10.2514/6.2000-4935

Goldberg DE (2012) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Boston

Gulli A, Pal S (2017) Deep learning with Keras: Implementing deep learning models and neural networks with the power of Python. Packt Publishing

Hardy RL (1971) Multiquadric equations of topography and other irregular surfaces. J Geophys Res 76(8):1905–1915. https://doi.org/10.1029/jb076i008p01905

Hussain MF, Barton RR, Joshi SB (2002) Metamodeling: radial basis functions, versus polynomials. Eur J Oper Res 138(1):142–154. https://doi.org/10.1016/s0377-2217(01)00076-5

Jacobs JH, Etman LF, Van Keulen F, Rooda JE (2004) Framework for sequential approximate optimization. Struct Multidisc Optim 27(5):384–400. https://doi.org/10.1007/s00158-004-0398-8

Jaiswal P, Patel J, Rai R (2018) Build orientation optimization for additive manufacturing of functionally graded material objects. Int J Adv Manuf Technol 96(1–4):223–235. https://doi.org/10.1007/s00170-018-1586-9

Jin R, Chen W, Simpson TW (2001) Comparative studies of metamodelling techniques under multiple modelling criteria. Struct Multidisc Optim 23(1):1–13. https://doi.org/10.1007/s00158-001-0160-4

Jones DR (2001) A taxonomy of global optimization methods based on response surfaces. J Glob Optim 21(4):345–383. https://doi.org/10.1023/a:1012771025575

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim. https://doi.org/10.1023/A:100830643

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Tech. rep. Erciyes University

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN95—international conference on neural networks, vol 4

Keshtegar B, Nguyen-Thoi T, Truong TT, Zhu SP (2020) Optimization of buckling load for laminated composite plates using adaptive Kriging-improved PSO: a novel hybrid intelligent method. Def Technol. https://doi.org/10.1016/j.dt.2020.02.020

Kim BS, Lee YB, Choi DH (2009) Comparison study on the accuracy of metamodeling technique for non-convex functions. J Mech Sci Technol 23(4):1175–1181. https://doi.org/10.1007/s12206-008-1201-3

Kitayama S, Yamazaki K (2011) Simple estimate of the width in gaussian kernel with adaptive scaling technique. Appl Soft Comput 11(8):4726–4737. https://doi.org/10.1016/j.asoc.2011.07.011

Kitayama S, Arakawa M, Yamazaki K (2010) Sequential approximate optimization using radial basis function network for engineering optimization. Optim Eng 12(4):535–557. https://doi.org/10.1007/s11081-010-9118-y

Kleijnen JPC, Sanchez SM, Lucas TW, Cioppa TM (2005) State-of-the-art review: a user's guide to the brave new world of designing simulation experiments. INFORMS J Comput 17(3):263–289. https://doi.org/10.1287/ijoc.1050.0136

Krishnamoorthy CS, Prasanna Venkatesh P, Sudarshan R (2002) Object-oriented framework for genetic algorithms with application to space truss optimization. J Comput Civil Eng 16(1):66–75. https://doi.org/10.1061/(ASCE)0887-3801(2002)16:1(66)

Kuhn HW, Tucker AW (1951) Nonlinear programming. In: Proceedings of the second Berkeley symposium on mathematical statistics and probability, 1950, University of California Press, Berkeley, pp 481–492

Kumar M (2017) scikit-optimize: sequential model-based optimization toolkit. https://scikit-optimize.github.io/stable/

Lemonge AC, Barbosa HJ (2004) An adaptive penalty scheme for genetic algorithms in structural optimization. Int J Numer Methods Eng 59(5):703–736. https://doi.org/10.1002/nme.899

Liu H, Ong YS, Cai J (2018) A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. Struct Multidisc Optim 57(1):393–416. https://doi.org/10.1007/s00158-017-1739-8

Luersen MA, Steeves CA, Nair PB (2015) Curved fiber paths optimization of a composite cylindrical shell via kriging-based approach. J Compos Mater 49(29):3583–3597. https://doi.org/10.1177/0021998314568168

Maia MA, Parente E, de Melo AMC (2021) Kriging-based optimization of functionally graded structures. Struct Multidisc Optim. https://doi.org/10.1007/s00158-021-02949-5

Martins JRRA, Marriage C, Tedford N (2009) pyMDO: an object-oriented framework for multidisciplinary design optimization. ACM Trans Math Softw 36(4):1–25. https://doi.org/10.1145/1555386.1555389

Mathern A, Steinholtz OS, Sjöberg A, Önnheim M, Ek K, Rempling R, Gustavsson E, Jirstrand M (2020) Multi-objective constrained Bayesian optimization for structural design. Struct Multidisc Optim. https://doi.org/10.1007/s00158-020-02720-2

Meza JC, Oliva RA, Hough PD, Williams PJ (2007) OPT++: an object-oriented toolkit for nonlinear optimization. ACM Trans Math Softw 33(2):12-es. https://doi.org/10.1145/1236463.1236467

Mlakar M, Petelin D, Tušar T, Filipič B (2015) GP-DEMO: differential evolution for multiobjective optimization based on Gaussian process models. Eur J Oper Res 243(2):347–361. https://doi.org/10.1016/j.ejor.2014.04.011

Muller J (2014) Matsumoto: the matlab surrogate model toolbox for computationally expensive black-box global optimization problems. arXiv:1404.4261

Müller J, Shoemaker CA (2014) Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. J Glob Optim 60(2):123–144. https://doi.org/10.1007/s10898-014-0184-0

Nakayama H, Arakawa M, Sasaki R (2002) Simulation-based optimization using computational intelligence. Optim Eng 3(2):201–214. https://doi.org/10.1023/a:1020971504868

Nakayama H, Arakawa M, Washino K (2003) Optimization for black-box objective functions. In: Series on computers and operations research optimization and optimal control, pp 185–210, https://doi.org/10.1142/9789812775368_0013

Nik MA, Fayazbakhsh K, Pasini D, Lessard L (2014) A comparative study of metamodeling methods for the design optimization of variable stiffness composites. Compos Struct 107:494–501. https://doi.org/10.1016/j.compstruct.2013.08.023

Pan G, Ye P, Wang P, Yang Z (2014) A sequential optimization sampling method for metamodels with radial basis functions. Sci World J 2014:1–17. https://doi.org/10.1155/2014/192862

Passos AG, Luersen MA (2018) Multi-objective optimization with Kriging surrogates using 'moko', an open source package. Lat Am J Solids Struct. https://doi.org/10.1590/1679-78254324

Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a pratical approach to global optimization. Springer, Berlin

Queipo NV, Haftka RT, Shyy W, Goel T, Vaidyanathan R, Kevin Tucker P (2005) Surrogate-based analysis and optimization. Progress Aerosp Sci 41(1):1–28. https://doi.org/10.1016/j.paerosci.2005.02.001

Ribeiro LG, Maia MA, Parente E Jr, Melo AMC (2020) Surrogate based optimization of functionally graded plates using radial basis functions. Compos Struct. https://doi.org/10.1016/j.compstruct.2020.112677

Rocha IB, Parente E Jr, Melo AMC (2014) A hybrid shared/distributed memory parallel genetic algorithm for optimization of laminate composites. Compos Struct 107(1):288–297. https://doi.org/10.1016/j.compstruct.2013.07.049

Rouhi M, Ghayoor H, Hoa SV, Hojjati M (2015) Multi-objective design optimization of variable stiffness composite cylinders. Composites B 69:249–255. https://doi.org/10.1016/j.compositesb.2014.10.011

Roustant O, Ginsbourger D, Deville Y (2012) Dicekriging, diceoptim: two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. J Stat Softw 51(1):1–55. https://doi.org/10.18637/jss.v051.i01

Schmit L, Farshi B (1974) Some approximation concepts for structural synthesis. AIAA J 12(5):692–699. https://doi.org/10.2514/3.49321

Schonlau M, Welch WJ, Jones DR (1998) Global versus local search in constrained optimization of computer models. New Dev Appl Exp Des 34:11–25. https://doi.org/10.1214/lnms/1215456182

Shen HS (2009) Functionally graded materials: nonlinear analysis of plates and shells. CRC Press, Boca Raton

Simpson TW, Lin DKJ, Chen W (2002) Sampling strategies for computer experiments: design and analysis. Int J Reliab Appl 2:209–240

Sivakumar P, Rajaraman A, Samuel Knight GM, Ramachandramurthy DS (2004) Object-oriented optimization approach using genetic algorithms for lattice towers. J Comput Civil Eng 18(2):162–171. https://doi.org/10.1061/(ASCE)0887-3801(2004)18:2(162)

Sobester A, Leary SJ, Keane AJ (2005) On the design of optimization strategies based on global response surface approximation models. J Glob Optim. https://doi.org/10.1007/s10898-004-6733-1

Song X, Lv L, Sun W, Zhang J (2019) A radial basis function-based multi-fidelity surrogate model: exploring correlation between high-fidelity and low-fidelity models. Struct Multidisc Optim 60(3):965–981. https://doi.org/10.1007/s00158-019-02248-0

Srinivas N, Krause A, Kakade S, Seeger M (2010) Gaussian process optimization in the bandit setting: no regret and experimental design. In: ICML 2010—proceedings, 27th international conference on machine learning, pp 1015–1022. https://doi.org/10.1109/TIT.2011.2182033

Steponavičė I, Shirazi-Manesh M, Hyndman RJ, Smith-Miles K, Villanova L (2016) On sampling methods for costly multi-objective black-box optimization. In: Advances in stochastic and deterministic global optimization springer optimization and its applications, pp 273–296. https://doi.org/10.1007/978-3-319-29975-4_15

Stork J, Eiben AE, Bartz-Beielstein T (2020a) A new taxonomy of continuous global optimization algorithms. Nat Comput. https://doi.org/10.1007/s11047-020-09820-4

Stork J, Friese M, Zaefferer M, Bartz-beielstein T, Fischbach A, Breiderhoff B, Naujoks B, Tusar T (2020b) Open issues in surrogate-assisted optimization. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-18764-4

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359. https://doi.org/10.1023/a:1008202821328

Tenne Y (2014) Initial sampling methods in metamodel-assisted optimization. Eng Comput 31(4):661–680. https://doi.org/10.1007/s00366-014-0372-z

Tutum CC, Deb K, Baran I (2014) Constrained efficient global optimization for pultrusion process. Mater Manuf Processes 30(4):538–551. https://doi.org/10.1080/10426914.2014.994752

Viana F (2010) SURROGATES toolbox user's guide. Gainesville, FL, USA, version 2.1 edn, http://sites.google.com/site/felipeacviana/surrogatestoolbox

Wagner S, Affenzeller M (2005) HeuristicLab: a generic and extensible optimization environment. Adaptive and natural computing algorithms. Springer, Vienna, pp 538–541. https://doi.org/10.1007/3-211-27389-1-130

Wang GG, Shan S (2007) Review of metamodeling techniques in support of engineering design optimization. J Mech Des 129(4):370. https://doi.org/10.1115/1.2429697

Williams B, Cremaschi S (2021) Selection of surrogate modeling techniques for surface approximation and surrogate-based optimization. Chem Eng Res Des 170:76–89. https://doi.org/10.1016/j.cherd.2021.03.028

Wu Z, Wang D, Patrick Okolo N, Jiang Z, Zhang W (2016) Unified estimate of Gaussian kernel width for surrogate models.

Neurocomputing 203:41–51. https://doi.org/10.1016/j.neucom.2016.03.039

Xiang H, Li Y, Liao H, Li C (2016) An adaptive surrogate model based on support vector regression and its application to the optimization of railway wind barriers. Struct Multidisc Optim 55(2):701–713. https://doi.org/10.1007/s00158-016-1528-9

Yao W, Chen X, Huang Y, Tooren MV (2014) A surrogate-based optimization method with RBF neural network enhanced by linear interpolation and hybrid infill strategy. Optim Methods Softw 29(2):406–429. https://doi.org/10.1080/10556788.2013.777722

Zadeh PM, Toropov VV, Wood AS (2009) Metamodel-based collaborative optimization framework. Struct Multidisc Optim 38(2):103–115. https://doi.org/10.1007/s00158-008-0286-8

Zhu W, Meng Z, Huang J, He W (2012) Optimization design for laminated composite structure based on kriging model. Appl Mech Mater 217–219:179–183. https://doi.org/10.4028/www.scientific.net/AMM.217-219.179