ALMA MATER STUDIORUM

UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Dottorato di Ricerca in Ingegneria Biomedica, Elettrica e dei Sistemi

Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi"

Ciclo XXX

Settore concorsuale: 01 - A6 Ricerca Operativa
Settore scientifico disciplinare: MAT/09 Ricerca Operativa

MODELS AND ALGORITHMS FOR OPERATIONS MANAGEMENT APPLICATIONS

Presentata da:
ANDREA BESSI

Coordinatore Dottorato:
Chiar.mo Prof. Ing.
DANIELE VIGO

Supervisore:
Chiar.mo Prof. Ing.
DANIELE VIGO

ESAME FINALE 2018

INDEX TERMS

Derivative Free Optimization

Black Box Optimization

Computer Vision Algorithm

Sequential Approximate Optimization

Parameter Tuning

Contents

Abstract

In the last decades, there has been a huge evolution in the development and application of Derivative Free Optimization (DFO) techniques. One of the major emerging DFO application fields are the optimal design of industrial products and machines and the development of efficient computer software. This Ph.D. thesis focuses on the exploration of the emerging DFO techniques, oriented especially in the optimization of real-world industrial problem. In this thesis is firstly presented a brief overview of the main DFO techniques and application. Then, are reported two works describing the development of DFO approaches aimed to tackle the optimization of Computer Vision Algorithms (CVA), employed in the automatic defect detection of pieces produced by a real-world industries. At the end, are discussed the conclusions related to the results obtained, their potential additional applications, and the further promising area of theoretical research.

Chapter 1

Introduction

1.1 Motivation

In this dissertation we examined the most known Derivative Free Optimization (DFO) approaches focusing on their application to a real world industrial problem, such as the emerging field of CVA employed in automated defect detection. The industrial relevance of this application area is highly increasing thanks to the great progress achieved by CVA accuracy and reliability. The fine tuning of the CVA is highly time consuming when done manually, thus enhancing the need of reliable and automated procedures to define the optimal parameters in various use conditions. To the best of our knowledge, no specific optimization method has been proposed in the literature for this type of applications. This make the parameter tuning of CVAs an interesting

class of emerging problems. The fundamental motivations of this thesis are thus resumable as follows.

- Analyze the state of the art of DFO approaches, in order to verify which class of them is the more suitable in order to tackle the CVA tuning problem.

- Search for the study and development of an effective optimization algorithm within the chosen family of approaches, able to effectively tackle the CVA tuning problem and to take advantage, where possible, of its specific characteristics.

- Conduct a deep on-field experimentation of the developed algorithm, in order to achieve representative results able to validate the proposed approach and verify if it is suitable for this class of problems.

1.2 Thesis Contributions

The original contribution of this dissertation concerns (i) a survey of the state of the art of DFO techniques, (ii) the presentation of a competitive DFO approach able to tackle the CVA tuning problem, and (iii) the demonstration that DFO, with application in CVA optimization, constitute a very effective and promising methodology.

1.3 Thesis outline

- **Chapter 2** is focused on a brief introduction to the Derivative Free Optimization. An overview is given of the main resolutive methodologies. Some example o the main DFO application are also presented.

- **Chapter 3** describe the optimization of a Computer Vision Algorithm trought an implemented DFO approach. CVA are widely used in several applications ranging from security to industrial processes monitoring. In recent years, an interesting emerging application of CVAs is related to the automatic defect detection in some production processes for which quality control is typically performed manually, thus increasing speed and reducing the risk for the operators. The main drawback of using CVAs is represented by their dependence on numerous parameters, making the tuning to obtain the best performance of the CVAs a difficult and extremely time-consuming activity. In addition, the performance evaluation of a specific parameter setting is obtained through the application of the CVA to a test set of images thus requiring a long computing time. The problem falls into the category of expensive Black-Box functions optimization. Here, is described a simple approximate optimization approach to define the best parameter setting for a CVA used to determine defects in a real-life industrial

process. The algorithm computationally proved to obtain good selections of parameters in relatively short computing times when compared to the manually determined parameter values.

- **Chapter 4** describe an application of Sequential Approximate Optimization (SAO) to solve a Black-Box Optimization Problem arisen during the calibration of several Computer Vision Algorithms (CVAs) used in the defects detection of pieces produced by an industrial plant. The performance of the CVAs depend on numerous parameters, making the search for their optimal configuration extremely difficult. In addition, linear constraints involving parameters are present, and some of these can be of integer type, thus complicating both the selection of the initial Design of Experiment necessary to initialize the search and the subsequent parameter optimization phase. The performance evaluation of a specific parameter configuration is obtained applying the CVA to a test set of images for which the defectiveness state is known. The comparison between the actual defectiveness state of the items and the relative CVA's evaluations produces the true and false positive detection ratio. These ratios are then linearly combined through weights to obtain a unique objective function, that turned out to be highly multimodal with respect to the input parameters. Moreover, since the

evaluation of the objective function is computationally costly due to the high dimension of the dataset and the CVAs long running time itself, the problem falls into the category of Expensive Black-Box Functions Optimization. To tackle this, being the objective function evaluation not a monolithic experiment but a series of different images elaborations, an effective time-saving strategy based on partial elaborations of the images dataset is implemented. In this chapter it is described the SAO approach developed to define the best parameter setting for different CVAs used to detect defects in a real-life industrial process. The overall approach is extensively tested first on an analytical benchmark function and then on data coming from a real-world application, experimentally proving to be able to obtain good parameters sets in relatively short computing times.

Chapter 2

Overview of Derivative-Free Optimization Approaches and Applications

2.1 A short introduction to DFO

Derivative-Free Optimization is identified in literature as the collection of methods, within Operational Research, that does not make use of the information about the derivative of the Objective Function $f(\cdot)$ to search the optimal solution. In the target problem of the optimization typically the derivative is not available. Examples of such class of problem are those one in witch $f(\cdot)$ is computed by:

- Running of computer code;
- Performing a physics experiments;
- Running (software) simulations.

In this class of problems, due to their nature, several complicating factors related to the objective function may occur:

- It can be multimodal;
- Its value can be affected by noise;
- Its evaluation can be computationally expensive (time-costly).

Regarding DFO applications, they have grown exponentially in the last years, and nowadays a lot of industrial problem can be solved only with Derivative-Free Algorithms (DFA). This because the associated objective function derivative, and even sometimes the analytical expression fo the objective function itself, are unknown. A detailed overview of the main practical applications of DFO, with particular attention to the industrial field, is reported in Section 2.5. In the following, a brief overview of the main DFO approaches is presented.

As well described by Rios and Sahinidis [1], the study of DFO techniques dates back many decades, starting with the work of Nelder and Mead [2] and

their simplex algorithm. A great number of approaches were presented in literature till nowadays, for an exaustive classification of them we refer the reader to [3].

DFAs can be mainly classified by three characteristics. The first one regards which part of the search domain is considered by the DFA at each iteration: depending on this, we can have *local* or *global* algorithms. The second characteristic regards if a Surrogate Model (SM) is used in the search process or not. As described later in Section 2.4, a SM is an approximation of $f(\cdot)$ representing the relation between its value and the problem's variables. A *SM-based* DFA builds and uses the SM to guide the search process, whereas a *direct* DFA determines directly the domain points to sample. A complete overview of Direct Search approaches can be found in [4]. The last characteristic regards if the optimization process is *deterministic* or, instead, *stochastic*-based decision are taken during the iterations. The following sections will provide a brief summary of the most known DFAs in literature that fall in the above described categories.

2.2 Local DFA

Local Search Algorithms (LSA) are a class of optimization algorithms able to perform the search only in a limited part of the domain, thus are not able

to perform a global optimization and therefore cannot guarantee to reach the global optimum. This because the candidate solution identified at each iteration relies only in a neighborhood of the previously sampled ones. LSA typically present an easier architecture and lower computational cost than the global ones. LSA, in the end, can even be sub-classified in direct and SM-based.

Nelder-Mead simplex algorithm is a direct LSA introduced in 1965 (see [2]). At the beginning, a set of points that form a simplex in the domain space is selected. Then, at each iteration, the objective function at each corner is computed and the corner with the worst one is selected. This is then substituted by another vertex, that is searched in the domain in such a way that the resulting polytope is still a simplex. Indeed, the core of the algorithm is to try to relocate the simplex at each iteration, reaching vertex points that allow to achieve the best objective function value. As to the convergence property of the method, since McKinnon [5] proved that it can stop at a point with non zero gradient even when optimizing a convex $f(\cdot)$, successive improvement of the method were proposed in order to prevent stagnation. Tseng [6] proposed a simplex based method that guarantee a global convergence with convex functions.

Trust-region methods are SM-based LSA techniques. These make use of a surrogate model in order to represent only the part of the domain, called the Trust Region (TR), where the search process, iteration after iteration, focus on. The typical model used is the polynomial (often quadratic) interpolations. The core characteristic of this methods is that, at each iteration, the size of the TR (its radius δ) can be modified. This decision is based on the level of reliability that the SM is supposed to have. Being $\mathbf{x}_k$ the incumbent solution at iteration k, at iteration $k+1$ the minimizer $\mathbf{x}^*$ of the SM predictor $s(\cdot)$ inside the TR is evaluated:

$$\mathbf{x}^* = arg\ min_{||\mathbf{x}^* - \mathbf{x}_k|| \leq \delta}\ s(\mathbf{x}^*) \tag{2.1}$$

then a check is performed in order to decide if the SM representing the TR can still be considered reliable or not. This decision is taken by computing the ratio r of the actual reduction on the f with respect to the predicted one:

$$r = \frac{f(\mathbf{x}^*) - f(\mathbf{x}_k)}{s(\mathbf{x}^*) - s(\mathbf{x}_k)} \tag{2.2}$$

Fixed a lower threshold l and an upper threshold u, three possibility can occur: 1) $r < l$ 2) $r \geq l \wedge r < u$ 3) $r \geq u$. In the case of the former, $\mathbf{x}^*$ is rejected and δ is reduced, since r suggest that the TR is not reliable

and must be focused on a lower space. In the second and in the latter, $\mathbf{x}^*$ is accepted as the new incumbent solution:

$$x_{k+1} = \mathbf{x}^* \tag{2.3}$$

Moreover, in the latter case, δ is augmented expanding the TR. The various proposed TRM differs in the choose of the SM, the thresholds, and the way in which the TR is redimensioned each times. In literature, two famous approaches using a quadratic surrogate model are presented in [7] and [8].

2.3 Global DFA

As for LSA, even Global Search Algorithms (GSA) can be sub-classified. Beside they can make use of a SM or not, and at the same time they can be deterministic or stochastic. A brief overview of the main approaches in literature is now presented.

The DIRECT algorithm is a direct and deterministic GSA proposed by Jones in the 1993 (see [9]). It belong to the class of algorithms that systematically subdivide the search space using a particular branching scheme. DIRECT, in particular, not allows the presence of constraints between the

design variables and, thus, consider as feasible all the hypercube given by their lower and upper bound. This hypercube is recursively subdivided in hyper-rectangles in such a way that they are not overlapping each others and the union of their volume entirely cover the domain. At each iteration, DIRECT decide which of the current existing rectangles must be split in two. For every new generated rectangle, its center point c is evaluated. The decision of which rectangles must be split is taken basing on two criteria: (i) the rectangle size (ii) the more or less promising values of its base points. The former criterion represents *exploration*, while the latter *exploitation*. DIRECT guarantee to reach a point arbitrarily close to the global optimum if sufficient time is given and no constraint to the partitioning procedure are imposed.

Genetic Algorithms are a direct and stochastic family of approaches that aims to reply the natural selection and reproduction processes assuring the survival of the fittest individual in a population. In this view, the sampled points represents the individuals, and the value of the variables at each points represents the individual's characteristic, i.e., genes. Genetic algorithms (GA) generate, at each iteration, a new set of s individuals (i.e. candidate points) and delete another one of the same size in order to maintain constant the size n of the population. Tree key operation determine the way

in which the new individuals are generated: (i) Selection (ii) Crossover and (iii) Mutation. With Selection, a subset of $2s$ individuals are chosen as "parents" for the creation of the next generation. The probability to be chosen is a function of the fitness value (i.e. objective function) of the points. In this manner, the next generation will be generated basing on the current most fitting elements. With Crossover, form each couple of parents two new individual are created, by randomly mixing the genes of the parents under a specified rule. With Mutation, a portion of the genes of the new individuals is flipped with a specified, and typically low, probability. While Selection and Crossover tends to concentrate the search near the best solution sampled *(exploitation)*, this leading to local optimum, Mutation determine the *exploration* of the search space. GA were initially proposed by Holland (see [10]) and a huge amount of contributions followed.

Response Surface Methods (RSMs) is a collection of SM-based techniques, typically deterministic, although they can make use of stochastic steps in order to prevent the stagnation of the search if necessary. RSM includes a large number of methods, whose behavior can be quite complicated depending on the implementations.

2.4 The Response Surface Methodology

A dedicated section needs to be reserved for the so called Response Surface Methodology, that represent one of the higher area of research inside the DFO techniques. Three key aspect characterize RSMs: (i) the initialization of the SM search through a selected strategy called Design Of Experiment; (ii) the use of a SM, called Response Surface, in order to approximate the optimizing functions f; and (iii) the optimization over the SM and its update at each iteration. The process starts with the DOE, in order to carry out an efficient initial sampling of the domain space. Then the selected SM is firstly built and the iterative process starts. At each iteration the optimization over the SM is performed. Then the new candidate point is sampled and the SM is updated. The process stops when the termination criteria are fulfilled. Several techniques of DOE were proposed in literature, spacing from *classical* to *modern* ones. The former were mainly proposed to control the presence of noise in physics experiments, by performing a geometric sampling of the space. The latter were introduces for the sampling of deterministic experiments like computer simulations, and are aimed to the optimal filling of the design space with an arbitrary number of points. However, apart to the type of DOE used, as described in [11] a main classification of RSMs can be obtained considering the two other aspect: the *type of SM* used and the

search strategy adopted.

2.4.1 The Surrogate Models

SM can be subdivided in two types: interpolating and not-interpolating the sample points. In the former, the resulting Response Surface (RS) pass through all the sampled points, thus not allowing the presence of noise. In the latter, the RS it is built in such a way to minimize the sum of the squared errors from these, and noise is tolerated. A typical non-interpolating SM is the fitting of a quadratic polynomial functions. Several possibility instead exists for interpolating SM, and the most used are: Radial Basis Functions (RBF) and Kriging. While the former represents an effective way to achieve an interpolating surface, the latter are based on a statistical interpretation that, at a cost of an higher computational cost, enables advanced search strategies that make use of this information.

Radial Basis Functions Surrogate Models make use of a network of basis functions $\phi(\cdot)$ in order to exactly interpolate the n sampled points $\mathbf{x}_j, j = 1, ..., n$ at their values $f(\mathbf{x}_j)$, and easily predict the value of an unsampled points $\mathbf{x}^*$. The resulting SM is here defined as the weighted sum of n basis

functions $\varphi(\cdot)$:

$$\hat{f}(\mathbf{x}^*) = \sum_{j=1}^{n} \gamma_j \varphi(\mathbf{x}^*, \mathbf{x}_j) \tag{2.4}$$

where $\mathbf{x}_j, j = 1, ..., n$ are taken as the centers of the radial functions and $\boldsymbol{\gamma} = [\gamma_1 ... \gamma_k]^T$ is a corresponding vector of weights to be determined. Several types of radial functions are used in practice, the most common ones are reported in Tab. 2.1:

RBF	formulation
Cubic	$\varphi(\mathbf{x}, \mathbf{c}) = (\parallel \mathbf{x} - \mathbf{c} \parallel + p)^3$
Thin plate spline	$\varphi(\mathbf{x}, \mathbf{c}) = \parallel \mathbf{x} - \mathbf{c} \parallel^2 \ln(\parallel \mathbf{x} - \mathbf{c} \parallel p)$
Multiquadric	$\varphi(\mathbf{x}, \mathbf{c}) = \sqrt{\parallel \mathbf{x} - \mathbf{c} \parallel^2 + p^2}$
Gaussian	$\varphi(\mathbf{x}, \mathbf{c}) = \exp(-p \parallel \mathbf{x} - \mathbf{c} \parallel^2)$

Table 2.1: Radial Basis Functions

Here, $\mathbf{c}$ represent the center of the basis function and p is a *shape* parameter whose value highly influence the smoothness or bumpiness of the resulting response surface, and thus must be carefully determined.

The $\boldsymbol{\gamma}$ coefficients are easily determined by solving the system:

$$\mathbf{A}\boldsymbol{\gamma} = \mathbf{F} \tag{2.5}$$

where:

$$\mathbf{A} = \begin{bmatrix} \varphi(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \varphi(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \varphi(\mathbf{x}_n, \mathbf{x}_1) & \cdots & \varphi(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \quad \mathbf{F} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_n)] \tag{2.6}$$

Kriging belongs to the family of Gaussian Process Regression (GPR) methods, that are based on a statistical assumption: the response of the system to optimize is considered as the realization of a stochastic process, even if it is actually deterministic. Each sampled points $\mathbf{x}_j, j = 1, ..., n$ is viewed as a random variable with known realization $f(\mathbf{x}_j)$, and these variables are correlated each others. In Kriging, the correlation function is analytically expressed as follows:

$$\begin{aligned} Corr(\mathbf{x}_i, \mathbf{x}_j) = \exp[-\sum_{h=1}^{H} \theta_h |{\mathbf{x}_i}^h - {\mathbf{x}_j}^h|^{\rho_h}] \\ \theta_h \geq 0, \quad \rho_h \in [0, 2] \end{aligned} \tag{2.7}$$

where h is the index of the space dimensions (i.e. the number of variables of the optimization problem), θ_h and ρ_h are *shape* parameters whose value can be optimally determined by maximizing the likelihood of the observed data.

The resulting Kriging predictor can be expressed in RBF formulation:

$$\hat{f}(\mathbf{x}^*) = \mu + \sum_{i=1}^{n} b_i Corr(\mathbf{x}^*, \mathbf{x}_j) \tag{2.8}$$

where μ is the center of the Gaussian Process, and b_i are the weight coefficients to be determined like in the RBF interpolation before described. Kriging, being a GPR method, is capable to compute the confidence interval associated to the prediction of an unsampled points, and thus estimate the predictor error $s^2(\mathbf{x}^*)$. This allow the use of more sophisticated search strategy, as following described.

2.4.2 The search strategies

Search strategy can be mainly classified in two category: (i) One Stage Approaches and (ii) Two Stage Approaches.

One Stage Approaches carry out the building of the SM and the search of the candidate solution in a single step. This because the SM itself is built basing not only upon the already sampled points, but also over an hypothesis about where the optimum solution could be located. In this way, the parameter of the SM turn out to be as consistent as possible with both the sampled data and the hypothesis made about the optimum. The next candi-

date solution is identified as the point where the built SM results to be more credible as possible. This credibility is typically estimated by analyzing the shape of the response surface resulting by the model.

Two Stage Approaches carry out the search of the next candidate solution in two phases. In the first, the SM is built by estimating its parameter basing only on to the set of sampled points. For example, the parameters of the Kriging predictor are estimated by the Maximum Likelihood, in order to maximize the consistence with the sampled data. In the second phase, the parameters of the model are assumed to be correct and the search of the global optimum is performed. This approach is typically less computational costly then the former but also less affordable, especially in the case that the sampled points does not furnish a good sampling of the domain and the resulting response surface is misleading.

2.5 Applications

The most known DFO applications in literature refer to:

- Engineering design. Here, DFO represent a fundamental support in the optimal design of pieces to be manufactured, in order to maximize their expected performances. DFO is highly applied in the aviation

industries, for example in the Engineering Design of airplanes wings profile [12].

- Software Tuning, where the object of optimization is a computer software itself [13]. The cases studied in this dissertation fall in this category.

- Electric circuit design, were DFO is used, for example, in the optimal dimensioning of circuit constant [14].

- Dynamic pricing, were the problem consists in the optimal price assignation to different customers [15].

In the next two section is illustrated a design and application of a Response Surface Methods in order to optimize a very particular type of computer software: Computer Vision Algorithms (CVA).

Chapter 3

Initialization of Optimization Methods in Parameter Tuning for Computer Vision Algorithms

3.1 Introduction

The calibration of computer vision algorithms (CVAs) is a time consuming and critical step in the effective use of CVAs in many applications, such as the automated defect detection of pieces produced by an industrial plant. In

this case, the final quality control check at the end of the production chain consists in the optical scan of the produced pieces by a set of several CVAs. Each of them is designed to detect a specific type of defect and its behavior is controlled by a large set of parameters, which influence the CVA sensibility and accuracy and must be determined to maximize its detection efficacy on specific types of images. For a general overview of automated defect detection see [16] (see also [17] for an example in the textile industry).

More precisely, given a set of images, the efficacy of the error detection is measured as a function of the positive and negative false ratios produced by the CVA with a specific parameter set. As in many other applications parameter tuning of the CVAs is, therefore, a crucial component for the overall efficacy of the system. To the best of our knowledge, no optimization method has been developed so far for parameter tuning in defect detection.

In the context of CVAs, the computation of the efficacy requires the application of the CVA to a training set of test images. This is typically a very time-consuming operation requiring several seconds per image, hence minutes or even hours for a significant training set. Therefore, approaches based on black-box function optimization (see, e.g., [11] and [18]) must be used in this case. To this end, we developed a simple Sequential Approximate Optimization (SAO) algorithm (see, e.g., [19]) to identify the optimal parameter values for a CVA used to detect a specific error on the images.

During the optimization process, the solutions iteratively found by the algorithm are evaluated by executing the target CVA on the training set of images. The comparison between the CVA outputs obtained on the images and their real defectiveness state produce the true/false positive index ratio for the solutions tested. Our goal is the determination of the optimal input parameter combination for the CVA, leading to the best possible false positive and negative ratios for each particular type of defect.

In Section 3.2 we describe in detail the characteristic of the problem under study. In Section 3.3 the structure of the proposed algorithm is given and in Section 3.4 we present the results of an experimental validation of the algorithm on data coming from a specific real-world application.

3.2 Problem Definition

The calibration of the parameters of a CVA is an optimization problem which can be described as follows. The variables to be optimized are the input parameters of the CVA which are assumed here to be continuous and associated with a lower and an upper bound) for their variation. The performance of the CVA is measured in terms of two independent indicators, namely the number of false-negative and false-positive in the solution, to be defined later.

More precisely, we have a CVA whose behavior depends on a subset I

of parameters whose value has to be determined. For each parameter $i \in I$ we are given a lower and upper bounds l_i and u_i, respectively. For each parameter $i \in I$ let x_i be the decision variable which represents its value. Given solution $\vec{x}$ we can evaluate its quality by measuring the performance of the CVA on a training set S of images. To this end, let $f_p(\vec{x})$ be the number of false-positives returned by the CVA when applied to the set S with parameters $\vec{x}$, defined as the number of non-defective images which are classified as defective by the CVA. Similarly, let $f_n(\vec{x})$ be the number of false-negatives returned by the CVA, defined as the number of defective images which are classified as non-defective by the CVA. Finally, let α_p and α_n be two nonnegative weights associated with the two performance measures. The CVA Parameter Tuning Problem (CVAPTP) can be formulated as follows

$$(CVAPTP) \quad z = \min F(\vec{x}) = \{\alpha_p f_p(\vec{x}) + \alpha_n f_n(\vec{x})\}, \quad s.t.\ l_i \leq x_i \leq u_i \quad \forall i \in I. \tag{3.1}$$

3.3 A Sequential Approximate Optimization Algorithm

The Black Box Optimization (BBO) nature of the problem requires the use

of an inference intelligence able to predict the objective function value of an unsampled solution to guide the search process. In the literature, such representation of the BB function it is referred to as the Surrogate Model (SM, see [20]) for which a large number of types and formulations were proposed.

As frequently done in the recent literature the SM is used within a Sequential Approximate Optimization (SAO) algorithm that iteratively updates the SM by adding the solution points that are determined at each iteration. This sequential approach preserves a certain simplicity but provides some important advantages. First, the rebuilding of the SM in order to capture the incoming information improves its reliability at each iteration. Second, it guarantees to perform a global optimization over the entire solutions domain, by reducing the possibility to being trapped into local optima. The general scheme of the simple SAO algorithm we adopted is depicted in Figure 3.1.

The algorithm starts with the identification of the initial sample of solution points, used to initialize the SM. Then, for each sample point, the corresponding value of the objective function $F(\vec{x})$ in (3.1) is computed by applying the parameter values associated with the point to the CVA over the entire images test set. The incumbent solution is defined as the best solution found so far, hence it initially corresponds to the best sample. The SM is built from the current set of points $(\vec{x}, F(\vec{x}))$ and used to determine the next

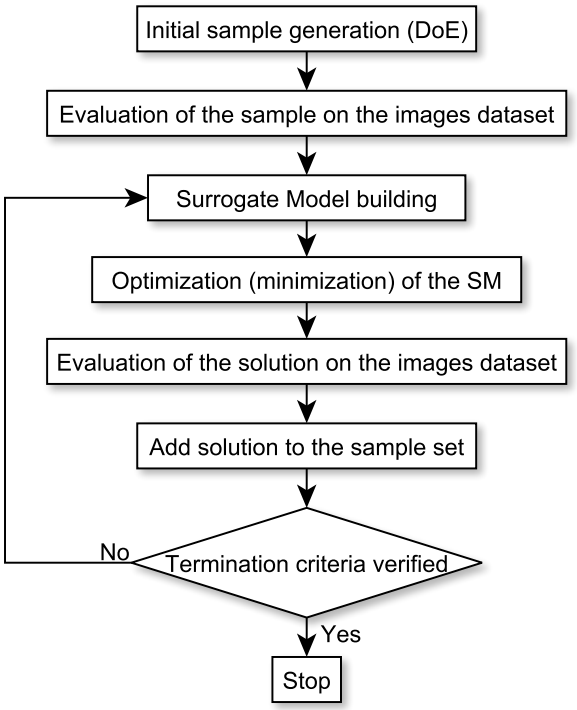

Figure 3.1: Outline of the Sequential Approximate Optimization Algorithm

candidate solution. After this, the SM is interrogated in order to search for the best candidate solution possibly improving the incumbent. This phase is generally called *adaptive sampling* criteria. The process is iterated until termination criteria based on solution quality and running time are met.

As to the type of SM used, in this work due to the BBO nature of the problem we adopted a specific type of the so called Meshfree methods, named Radial Basis Function (RBF) interpolation techniques. These are relatively easy to construct and are widely used to approximate Black Box function responses. In RBF interpolation the model, $s(\vec{x})$, is defined as the sum of a

given number K of radial functions ϕ:

$$s(\vec{x}) = \sum_{J=1}^{K} \gamma_j \phi(\|\vec{x} - \vec{\vec{x}}_j\|) \tag{3.2}$$

where $\vec{x}_j, j = 1, \cdot, K$, is the set of sampled solution points representing the centers of the radial functions $\phi()$, γ is a vector of weights to be determined, and $\vec{x}$ is the unsampled point whose value has to be predicted. Regarding the radial functions ϕ, several type are available in literature, varying from parametrized to not parametrized ones. In our case, the best trade off between a simple construction of the SM and an acceptable reliability resulted with the use cubic basis function, that assume the form:

$$\phi(\|\vec{x} - \vec{x}_j\|) = \|\vec{x} - \vec{x}_j\|^3 \tag{3.3}$$

We refer the reader to [21] for an overview of Meshfree methods, and to [22] for an example of industrial use of cubic RBF.

As to the initial sample generation through which initialize the SM, several techniques exists in literature, typically referred to as Design of Experiment (DoE). Since in our problem the parameters x_j are not subject to other constraints besides the upper and lower bounds, classical DoE as the Factorial Design are suitable. In particular, a Full Factorial Design (FFD) permit to cover the entire domain space, selecting all the points of the grid

generated by the discretization of each design variable (i.e., the parameters in our problem). This methodology is appropriate with the cubic RBF interpolation since it guarantees a sufficient reliability only inside the convex hull of the sampled points. However, in our case the computation of $F(\vec{x})$ is extremely time-consuming and using grids in which the parameter's values are discretized is not practically possible. For this reason, we decided to initialize our algorithm through a $|I|^2$ FFD, using just the domain vertices obtained with the lower and upper bounds of the parameters to be optimized. To improve the initial sample quality we also considered an initialization in which H additional random points selected inside the domain hypercube are considered. The adaptive sampling strategy that we adopt to perform the search of the candidate solution on the SM is the minimization of its predictor $s(\vec{x})$. To avoid to being trapped in a local minimum and perform an efficient search over all the domain, we implement a multi-start gradient descent algorithm and run it by using a discrete grid of starting points.
Finally, we terminate the algorithm after a maximum number of objective function evaluations or after a given number of non-improving iterations.

3.4 Experimental Validation

We applied our algorithm to the tuning of a CVA used to detect a specific type of defects on tyre images obtained in a real-world production environment. The training set is made up of 160 images for which the presence or absence of defects is known. Six parameters were selected as the target for the optimization. Each such parameter has a maximum and minimum value and a default value manually determined by the CVA designers. The behavior of the CVA with the default parameter values is used here as a benchmark reference to evaluate the performance of the optimized parameters set.

We tested the impact of three variants for the initialization step, leading to three different overall algorithms A_1, A_2, and A_3. In A_1 we used $H_1 = 100$ random points to initialize the algorithm. In A_2 the sample set is constituted by the 2^6 points of the simple FFD described in Section 3.3. Finally, in A_3, we added to the FFD set $H_3 = 36$ random internal points. The overall algorithm is run for a total of 200 objective function evaluations (including those for the initialization step). The gradient descent search for the candidate solution is performed from a 9^6 discrete grid of points.

To account for possible different relative importance of false positives and false negatives in the defect detection, we considered two different pairs of weights in the objective function (3.1). Namely, we considered $(\alpha_p, \alpha_n) =$

$\{(1,5),(1,10)\}$.

The results for the three algorithms are illustrated in Figures 3.2 and 3.3 for the (1,5) and (1,10) weight combinations, respectively. The figures report the evolution of the objective function for each algorithm compared with the benchmark reference equal to 42 for both weight combinations. By observing the figures it clearly appears that all proposed algorithms generate better parameter combinations with respect to the manual ones. In particular, for the (1,5) case A_1, A_2 and A_3 produce solutions with value 29, 29 and 27, respectively, which are 31% and 36% better that then manual ones. For the (1,10) case, they find a solution with value 38, 40 and 32, which are 10%, 5%, and 24% better, respectively. In general, we can observe that the mixed initialization of A_3 provides better final results but the simple FFD of A_2 improves quite rapidly and may constitute a good alternative when less time is available.

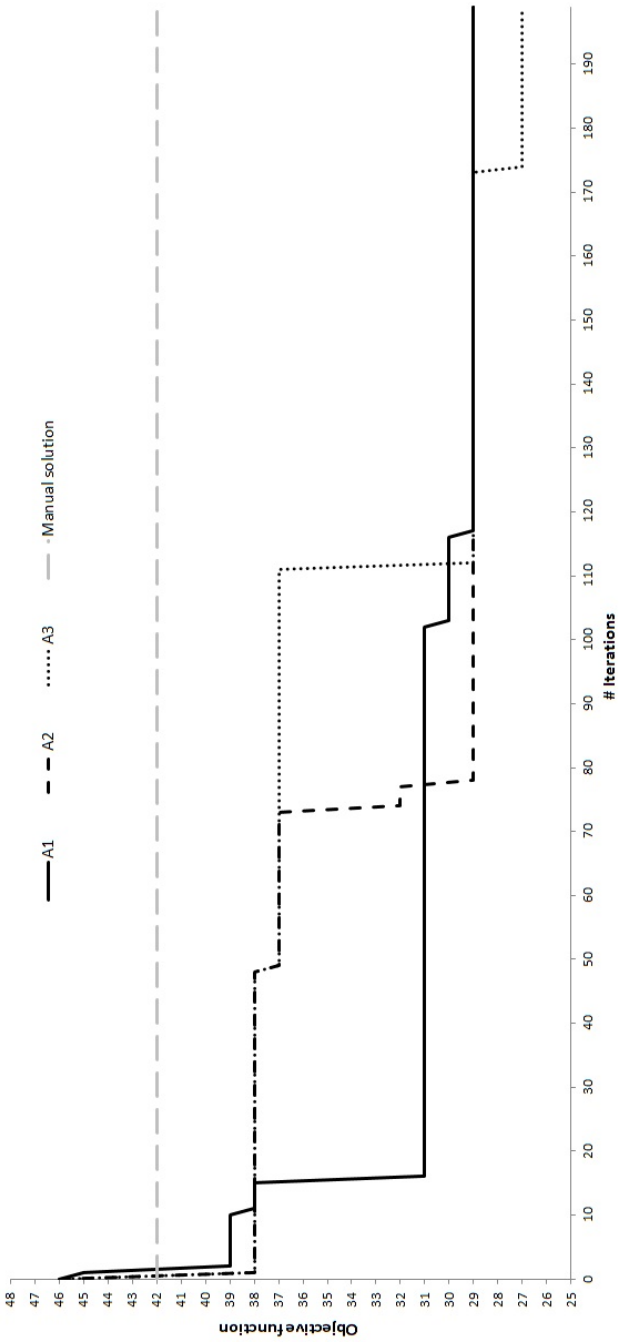

Figure 3.2: Evolution of the proposed algorithms with weight combination (1,5) in comparison with the manual tuning.

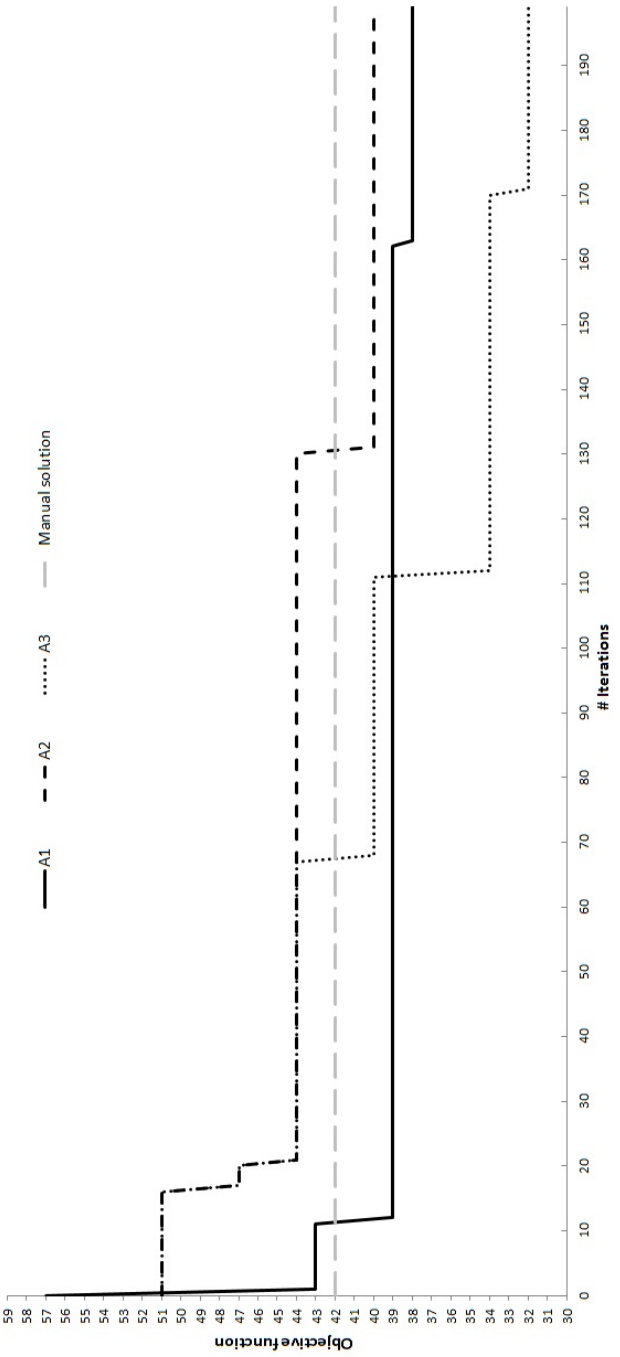

Figure 3.3: Evolution of the proposed algorithms with weight combination (1,10) in comparison with the manual tuning.

Chapter 4

Parameter Tuning of Computer Vision Algorithms Through Sequential Approximate Optimization

4.1 Introduction

In this chapter, we address the optimal calibration of Computer Vision Algorithms (CVAs) used in the automated defect detection of pieces produced by an industrial plant. The final quality control check at the end of the pro-

duction chain consists in the optical scan of the produced pieces by a set of several CVAs. Each of them is designed to detect a specific type of defect and its behaviour is controlled by a large set of parameters controlling various aspects, such as the image noise filtering and the edge detection capability. Given an image of the piece to be checked, obtained with an appropriate camera, the CVA returns a binary value indicating if it is judged as defected or not. For a general introduction to defect detection and the use of computer vision, see [23]. The various parameters influence the CVA sensibility and accuracy, and must be optimally determined in order to maximize its detection efficacy on specific types of images. As in many other applications, the parameter tuning of the CVAs is, therefore, a crucial component for the overall efficacy of the system. Given a set of test images, the efficacy of the defect detection is measured as a function of the false positive and false negative detection ratios produced by the CVA with a specific parameter set. Its computation requires the application of the CVA to each image of the set, which is typically a very time-consuming operations requiring from seconds to minutes per image. Therefore, approaches based on expensive black-box optimization (see, e.g., [24] and [11]) must be used in this case. The presence of constraints between parameters and their possible integer type represents further complicating factors. In the literature, there are some general approaches to the parameters optimization of generic software,

as illustrated, for example, by [13]. An early example of a CVA parameter tuning approach, limited to few parameters in an unconstrained continuous domain, can be found in [25]. However, to the best of our knowledge, no CVA optimization method able to handle all the issue presented by our specific application has been developed so far. We then developed a sequential surrogate based optimization algorithm to identify the optimal parameter values to be applied to a CVA used to detect a specific defect on the images. During the optimization process, each parameter solution iteratively found by the algorithm is evaluated by comparing the CVA outputs obtained on the images and their true defectiveness state, obtaining the true and false positive index ratio for the solutions tested. The set of the non-dominated parameters solutions give an approximation of the receiver operative characteristic (ROC) curve of the CVA. Since the various types of defects are very different both because of the different occurrence frequencies in the production chain and of the relative severity, every CVA must be optimized independently. Our goal is the determination, within a limited amount of computing time, of the optimal input parameter combination for a set of target-CVAs, leading the best possible false positive and false negative ratios for each particular type of defect. To reduce the large computing time required by the images elaborations, we developed an effective time-saving strategy that exploits the specific characteristic of the problem and is based on the partial elaboration

of the images dataset at each algorithm iteration. This paper is organized as follows. In chapter 4.2 we describe the characteristics of the problem under study. In chapter 4.3 we present a brief literature survey on the approaches we use in our solution algorithm: namely, the Black-Box Optimization and the main surrogate modeling techniques. The structure of the developed algorithm is presented in detail in Section 4, including the selection and validation of a surrogate model based on Gaussian process regression to support the optimization process, and discussing how the constraints on the input variables and their possible integer nature can be handled. In Section 5 we discuss the results of an experimental validation of the algorithm on both an analytical benchmark from the literature and on data coming from a specific real-world application. Finally, in Section 6 we draw some conclusions and outline directions of future research.

4.2 Problem Description

The calibration of the parameters of a CVA is an optimization problem which can be described as follows. The variables to be optimized are the input parameters of the CVA. Preliminary tests have shown a high responsiveness of the CVA's behaviour even with respect to limited variations on the input variables, and the response of the CVAs to parameter changes has shown to

be multimodal. The variables associated with the parameters to optimize are associated with a range of variation (i.e., a lower and an upper bound) and are possibly subject to linear constraints on their values involving some pairs of them. Furthermore, even though some of the parameters may assume only discrete values, in the practical application that motivates our research typically the variation ranges of such discrete variables are large (even hundreds of values) and the CVA's response to their variation can be considered relatively "smooth". The performance of the CVA is measured in terms of two independent indicators, namely the number of false-negatives and false-positives in the solution, to be defined later. It can, then, be considered a Multi-Objective optimization problem. More specifically, since these indicators are not analytically predictable but can be only empirically measured by running the CVA with a specific set of parameters on a set of target images for which the presence of the defect is known, it is a Multi-Objective Black Bok Optimization problem.

More precisely, we have a CVA whose behaviour depends on a set H of parameters. A subset $I \in H$ of parameters are subject to the optimization while the remaining $H \setminus I$ are fixed to a pre-determined value: in the following the fixed parameters will be no longer considered. For each parameter $i \in I$ we are given lower and upper bounds l_i and u_i, respectively as well as a set of linear constraints C among their values. In general, the parameter set

I is partitioned into two subsets $I = I_c \cup I_d$, where I_c is the set of continuous parameters and I_d is that of the discrete ones. Moreover, for each parameter $i \in I$ let x_i be the decision variable which represents its value. Given solution $\mathbf{x}$ we can evaluate its quality by measuring the performance of the CVA on a training set S of images. To this end, let $fp(\mathbf{x})$ be the number of false-positives returned by the CVA when applied to the set S with parameters $\mathbf{x}$, defined as the number of non-defective images which are classified as defective by the CVA. Similarly, let $fn(\mathbf{x})$ be the number of false-negatives returned by the CVA, defined as the number of defective images which are classified as non-defective by the CVA. Note that such values represent two different objective functions depending on $\mathbf{x}$ which must be minimized. Therefore, the problem can be formulated as the following two-objective problem.

$$Min[fp(x), fn(x)] \tag{4.1}$$

subject to:

$$x_i \geqslant l_i \quad \forall i \in I \tag{4.2}$$

$$x_i \leqslant u_i \quad \forall i \in I \tag{4.3}$$

$$\boldsymbol{Ax} \leq \boldsymbol{c} \tag{4.4}$$

$$x_i \quad integer \quad \forall i \in I_d \tag{4.5}$$

Where constraints (2) and (3) impose the lower and upper bounds. Constraints (4) represent the set of linear relations between the parameter values and (5) define the integer variables.

As frequently done in the literature the two-objectives problem is transformed into a single objective one by combining the two performance measures into a weighted sum of their values. Hence the objective function (1) becomes

$$Min[a * fp(x) + b * fn(x)] \tag{4.6}$$

where a and b are the non-negative weights. Moreover, since the objective function value depends on the cardinality of the test set S and the associated defectiveness state, it can be normalized so that it takes only values between 0 and 1.

4.3 Overview of Black-Box Optimization Approaches

In this section, we briefly present the main approaches presented in the literature for Black-Box optimization (BBO). A BBO problem is an optimization problem in which at least one function, typically the objective function, is a Black-Box function (BBF), i.e., a function that has the following properties:

- It is not expressed in analytical form. For example, because the function represents the response to the input the variables of a system whose internal behaviour is unknown. This implies that the computation of its value can only be obtained thought an empirical experiment on the system.

- The computation can be time-costly, especially if the evaluation of system's response consists in a physical experiment or in the run of a dedicated computer simulation software.

- The computation of the function may not return, in addition to its value, any other information of higher order such as the gradient.

- The response of the system can be multimodal with respect to the input variables.

- The response of the system can be affected by noise (i.e., it is a so-called a noisy Black-Box function).

There are several implications of the above-mentioned characteristics. For example, due to the lack of analytical description of the internal behaviour of the system, some kind of inference intelligence is necessary to select the candidate solutions to be evaluated, based on those evaluated previously. Several types of inference intelligence and different criteria to select the new solution

to be evaluated exists, and some of them will be discussed later. Moreover, the lack of higher order information, such as the gradient, increases the difficulty in the building of the inference intelligence, since it can be based on fewer information. A time-costly BBF evaluation implies that only a few evaluations can be done in a reasonable time, contrary to traditional optimization methods which rely on a huge amount of evaluations. Multimodality, implies that the BBF is not convex, hence there is the possibility of not finding the global optimum and remaining trapped in local optima. As it is well described in [24], to perform global optimization in such multimodal condition it is necessary to perform both global and local search during the optimization process, indicated as exploration and exploitation. Finally, the presence of noise makes the response of the system is non-deterministic, i.e., it can vary from different evaluations even with the same input parameter values and this should also be taken into account by the inference intelligence.

We observe that in the practical application that motivates this paper, the BBF to be optimized is not affected by noise because it is obtained by running on the test images the CVA, which is a deterministic computer code. Furthermore, the evaluation of the BBF through the CVA does not return gradient information, and is extremely time-costly requiring approximately 30 minutes per evaluation with the dataset used.

In the literature, there are several approaches to optimize single-objective Black Box Optimization problems. One way of classifying them is with respect to the type of the inference intelligence used. To this end, we can distinguish between approaches which either use or not a surrogate model (SM), see, e.g., [26]. A SM is a simplified representation of the BBF that describes the relation between inputs (i.e., the problem variables) and outputs of the real system to be optimized, based on the values of the already sampled points. An SM can be built in several ways and, in addition to the function value at an unsampled point, the SM permits to infer different information about the real system, such as the associated prevision error, which can be used to better choose the next point to evaluate. The use of an appropriate SM can greatly reduce the number of evaluations to be performed during the optimization process. On the other hand, when no SM is used, the optimization is performed through techniques that, to select the next point to be evaluated, use only the values obtained from the sampled points of the real system. General approaches not using SM such as, for example, Genetic Algorithms or Direct Search (see, e.g., [26]), require a much larger number of evaluations of the BBF, which makes these techniques impractical when the BBF to optimize is time costly, as it is in our case. Therefore, we concentrate on SM-based approaches only. In the following, we examine

the different components of such methods. In particular, we first address the strategy to select the initial set of points used to build the SM and then we consider the SM building techniques. We next address the validation of the SM and its optimal tuning in order to guarantee its higher possible reliability. Finally, we describe the optimization process which includes the search of the next candidate solution through the so-called adaptive sampling criterion, and discuss the stopping criteria.

4.3.1 The selection of the initial set of points

The common technique used to select the initial sampling points is called Design of Experiments (DOE). As explained in [27], the DOE aims at allocating sample points across the design space to maximize the amount of information derived from the real system. Classic DOE methods were developed to support physical experiments and control the effects of the associated random errors. The main example of such methods is the so-called Factorial design, whose principle is to discretize each of the d design variables into n values and then select the points from the resulting hyper-grid. Whenever all the resulting points are considered, we have the so-called Full Factorial Design (FullFD) and their number is equal to n^d, with n usually small and equal to 2 or 3. An alternative to overcome the exponential growth of the

sample space is represented by selecting only a subset of them, leading to the so-called Fractional Factorial Design (FractFD). Other examples of classical DOE methods are the Star Design (SD), where a central point and two points for each design variable, usually corresponding to the lower, upper or average values, and the Central Composite Design (CCD), which combines SD and either FractFD or FullFD. Modern DOE methods were developed to mainly support deterministic experiments unbiased by noise such as the run of computer code, as it is in our case. The most used ones are known as Latin Hypercube Sampling (LHS), Orthogonal Array Design (OAD) and Uniform Design (UD). In LHS the points are sampled randomly in the design space but, each design variable is discretized into n values, and points are selected within this hyper-grid so that for each one-dimensional projection over each variable, there is only one point for each of its n values. LHS is easy to implement, but may provide non-uniform distributions of points, hence OAD and UD are extension trying to achieve a more uniform distribution of the sample points. We refer the reader to Giunta (2003) for a complete review of modern DOE techniques. We note that all such methods assume that the design space is a hypercube only bounded by upper and lower bound constraints on the design variables.

When, as it is our case, additional constraints on the variables are present, such methods became not applicable. To address this, other methods were

proposed in the literature and are often referred as Space-Filling Design (SFD). SFD methods guarantee a good uniform distribution of points over a highly constrained convex domain. Their core mechanism is that points are iteratively moved in the space by optimizing a given distance measure such as, for example, by maximizing the minimum distance between each pair of design points. In this case, the SFD is called the MaxMin Design. If instead the objective is to minimize the distance of each points of the domain from the nearest SFD point it is called MinMax Design. Finally, several ways were proposed in the literature to decide how to move the points, that range from geometrical techniques to physics-inspired principles. For a survey on SFDs, we refer the reader to [28]. Given the nature of our problem, we decide to develop a simple but efficient MinMax SFD which will be described in Sect. 4.4.1.

4.3.2 The surrogate model building technique

Different types of surrogate model building techniques were proposed in the literature (see [27]), which can be classified into either *Physical Surrogates* (PS) or *Functional Surrogates* (FS). The construction of a PS typically assumes that some information of the internal structure of the system to model is known. In this case, the main technique to build the PS is to use a sim-

plified representation of the system. For example, by using either a coarser discretization when a simulation software is used to obtain the values of the system response, or a simplified analytical form of the relevant physics laws. PSs are generally easy to design but computationally expensive to be evaluated and typically not adaptable to other systems. In the case we consider in this paper, a PS is clearly not usable since the system to describe is actually a CVA.

An FS technique uses a mix of algebraic and statistical models to represent the response of the real system. They do not require any knowledge about it, since its behaviour is implicitly represented by the coefficients of the model which are iteratively tuned. This feature make FSs generic and not bounded to the specific target system. Instead, the correct choice of the coefficients, beside the optimal selection of the algebraic model, represent a critical step in their use. In fact, the definition of the coefficients is carried out by sampling and fitting procedures. Starting from an initial set of sample points, defined through the DOE techniques already described, during a fitting step the coefficient values of the model are chosen so that they provide the best fit of the current set of points. It is then clear that the suitability of the FS in representing the real system is strictly dependent on the selection of an appropriate sampling strategy.

Several different mathematical forms have been used as FS models to approximate the unknown behaviour of the system (see, e.g., [29]), most of which are known as Response Surface Methodologies (RSM). For example, non-interpolating techniques include Polynomial Regression (PR) and Radial Basis Function (RBF) approximation. In such cases, the response of the model does not coincide with that of the real system at the sampled points and the model structure and coefficients are chosen so as to provide a simple functional structure that minimizes the sum of squared error with respect to the actual model response at the sampled points. In PR the model is actually a polynomial of given degree for which the coefficients have to be chosen, while in RBF approximation the model, $s(x)$, is defined as the sum of a given number k of radial functions $\varphi(\cdot)$:

$$s(\mathbf{x}) = \sum_{j=1}^{k} \gamma_j \varphi(\| \mathbf{x} - \mathbf{c}^j \|) \tag{4.7}$$

where $\mathbf{c}^j, j = 1, ..., k$ is the set of centres of the radial functions and $y = [y_1 ... y_k]^T$ is a vector of weights to be determined. Generally, k is smaller than the number of sample points and several types of radial functions are used in practice such as cubic, splines, Gaussian and Multiquadric functions. As to the interpolating techniques, the model is constructed so that its response coincides with the value of the sampled points of the real system. To

this end, the centres of the RBF coincide with the sampled points and the order of the model grows with the sample size. It is easily seen that interpolating techniques are applicable only if the system response is not affected by noise.

A particular family of interpolating methods is represented by the Gaussian Process Regression (GPR), also known as Kriging, which is one of the most widely used approach in the literature. GPR is based on the assumption that, even if the response of the system is deterministic, it can be viewed as the realization of a stochastic process. The core principle in predicting the response value at an unsampled point $\mathbf{x}$ is to use the previously performed evaluations to identify the value that is more consistent, under specific statistic assumptions, with the sampled data. Due to the fact that estimating of the model parameters can be computationally costly, the building of GPR require more effort compared to the other methods. However, several papers prove that GPR generally provide better system approximation. For a complete review of non-interpolating and interpolating techniques in SM building the reader is referred to [11] and [29]. Our optimization problem is characterized by a noise-free response and by a highly time-consuming computation. The first feature makes interpolating techniques usable. The second highly reduce the number of response evaluation (i.e., sample points evaluations)

that can be performed in the given time. For these reasons, we have chosen the GPR as SM building method, and we will describe our implementation in Sect. 4.4.2.

After the initial SM is built, a check of its reliability is often performed to verify if it is appropriate for the specific application. Also in this case, several techniques have been developed, among which the most used class is that of *Cross Validation* (CV) techniques. Here, the dataset of known samples of the system is split into a *training set* and a *test set*, then the first is actually used to build the SM, while the second is used for its validation by comparing the values of the SM predictions with the actual values of the system, thus obtaining relative and absolute errors. Depending on the ways in which the two sets are defined several assessment procedures can be defined, such as the *Exhaustive* CV, where all the possible ways to divide the original set must be tested and the corresponding indicators collected, or the *Leave-One-Out* CV (LOOCV), where the test set is made up by a single element chosen from the sample. If the assessment of the SM reliability is not satisfactory, different strategies exist to improve it. If the SM present parameters than can be tuned, like in parameterized RBF interpolation or in GPR, the model can be rebuilt by varying its parameters and trying to improve the LOOCV performance. Obviously, such attempt is not possible

if either the parameters are fixed during the initial building of the model, or if model parameters are not present. In this case, the possible strategy is to enlarge the training set. To this end, new points of the real system are chosen in such a way to achieve the maximum augment of the SM reliability according to a so-called *infill criteria*. In Sect. 4.4.3 we explain the strategy we developed to efficiently tune the parameters of the GPR model we implemented and to measure its reliability.

4.3.3 The search strategy for the candidate solution

The methods used to search the optimal solution within the SM built are commonly called *adaptive sampling* methods, and the most common ones are so-called two-stage approaches. Here, the SM is first built by using the existing samples and estimating all the required parameters of the model. Then, the search of the optimal solution is performed over the resulting response surface, that is densely searched. Several search strategies exist, and their usability depend on the type of SM used. The main discriminant is associated with the use or not of a stochastic approach such as the GPR in building the SM. In the negative case we cannot obtain information about the standard error associated to the predictor, then the only applicable approach is the *Minimization of the Response Surface* (MRS), which consists

in exploring the response surface to find its global minimum, for example through a gradient descent algorithm coupled with a multistart mechanism to reduce the probability of being trapped into a local optimum. Instead, whenever the SM provides information about the standard predictor error, various methods using such error of it are available, where the *Maximization of the Expected Improvement* (MEI) represents the most common one. The adaptive sampling strategy that we implemented is described in Sect. 4.4.4.

4.3.4 The overall search process

Surrogate-based optimization processes may be either one-shot or iterative, where in the latter, known as Sequential Approximate Optimization (SAO, see [19]), the SM is iteratively rebuilt, updated and used several times. SAO approaches clearly are more time consuming but generally obtain much better solutions than one-shot methods being able to perform global optimization in highly multimodal or noise-affected contexts. The process starts with a given sample of points tested on the real system, selected through a specified DoE. Then, the SM is built through a specific model-building technique, such as RBF or GPR, and is validated. If the model reliability is found unsatisfactory, the model parameters are tuned or new points to test are selected by a proper *infill criteria*, until a satisfactory reliability is achieved. An *adaptive*

sampling criteria is used to find the candidate solution to test, for example by using the MRS or MEI methods previously described. To achieve a global optimization, the adaptive sampling criterion used must balance Exploitation and Exploration components. The former component is aimed at improving the incumbent by intensifying the search in its neighborhood, thus incurring the risk of remain trapped in local minima. The second component instead aims at exploring the under-sampled regions of the solution space. The candidate solution found is then evaluated on the real system and the incumbent is possibly updated. The solution found is finally added at the samples set, and the SM is rebuilt and the process is iterated until a stop criteria is met.

4.4 The Implemented Algorithm

We now describe the implemented algorithm and the way in which we have handled the major difficulties of the problem at hand, which are summarized as follows:

- High cost of the single point evaluation, due to the fact that the single solution's evaluation is performed by running the target CVA on a large dataset of images.

- Presence of constraints between the design variables.

- Presence of integer variables;
- Two different objective function to minimize: false positive and false negative index ratio.
- High multi-modality of the system response.

Being a time-costly BBO problem, we do not apply direct optimization to the system response. In fact, because gradient information is not available, the only possible direct optimization approaches are Derivative Free methods such as Genetic Algorithm or Direct Search, that require a large number of evaluations to produce satisfactory results and, in our case, could require an impractically large computing time. Hence, we developed an optimization algorithm based on a Sequential Approximate Optimization approach which experimentally proved to obtain very good results within an acceptable computational time. In the following, we describe in detail the component of the implemented approach following the scheme outlined in Sect. 4.3.

4.4.1 A space-filling Design of Experiment

As seen in Sect. 4.3.1, the main goal in the initial DOE is the selection of n points of the solution space in such a way to extract as much information as possible from the real system to optimize. The choice of the DOE is clearly crucial for the construction of the initial SM and thus for the overall efficiency

of the SOA method.

We developed a Space Filling DoE that iteratively allocates an arbitrary number n of points in the given convex domain so that they are as much uniformly dispersed in the domain as possible. More specifically, we implemented a MaxMin SFD (MmSFD), instead of a MinMax one, because it guarantees a more uniform sampling thanks to the minimization of the size of the "unsampled regions" and, at the same time, is able to easily handle the domain constraints. As to the integrality constraints, our MmSFD procedure enforces the sample points chosen to be integer. The principle according which the MmSFD works is the simulation of the behaviour of a particles system. Each point is associated with a particle with given mass we assume the presence of repulsive forces between every pair of particles, proportional to their distance. In addition, every point is subject to an orthogonal repulsive force from each hyperplane supporting a cosntraint of the domain (including the lower an upper bound constraints). This avoids that a point violates the domain constraints, since the repulsive force became exponentially larger when the distance of the point from a constraint decreases. For each point, the direction and distance in which it should be moved is computed by considering the resultant of all the forces applied to it. At each iteration, all points are moved and the process is repeated for a given number of iterations. Let F be the complete set of linear constraints bounding the

domain including the lower and upper bound constraints on each variable corresponding to a parameter to be tuned. Moreover, let us assume for simplicity that the domain is limited and that all parameters are continuous. The procedure can be easily modified to account for integrality conditions on some parameters. The procedure starts by randomly generating n sample points into the domain, denoted as $p_t, t = 1, \ldots, n$.

At each iteration, for each point $p_t, t = 1, \ldots, n$, (called target point) we compute the resultant of the repulsive forces between the point p_t and all other points p_s,with $s \neq t$ as well as all the constraints in F, by considering the orthogonal projection of p_t on each of the constraints in F. As previously mentioned, the repulsive force is inversely proportional to the distance between the points, normalized with respect to the maximum distance between two points in the domain. More precisely, the repulsive force between p_t and another point p_s (i.e., both a sample and a projection over a constraint) is defined as:

$$f_{s,t} = (\frac{d_{max}}{\parallel p_t - p_s \parallel} - 1)^h \tag{4.8}$$

where d_{max} is the maximum distance between two points in the domain and h is a suitable parameter, typically set to a value ≥ 2 to ensure a fast convergence of the procedure. Once all repulsive forces are computed, their resultant is computed for each point which is moved in that direction by a

distance proportional to the force intensity, suitably scaled down n so as to avoid that two points exchange their relative positions as an effect of their movement.

The process is iterated until the position of the points become stable, e.g. when their coordinates do not vary by more than a predetermined threshold with respect to the previous iteration or after a given maximum number of iterations. As previously mentioned, if integer values of the variables are required these are obtained by a rounding step followed by a simple local search step where neighboring feasible integer points are evaluated.

4.4.2 The Surrogate Model

To define our Surrogate Model (SM) we first need to define precisely the objective function which represent the response value of our system. More precisely, we used equation (6) which allows us to treat the multi-objective BBO as a single-objective one and we normalized it by considering the maximum number of false positives and false negatives in the training set. This allows to compare the objective function values also when different training sets are used to evaluate different points.

The specific SM we used in our implementation is a GPR interpolating SM, introduced in Sec. 4.3.2. In a GPR, the value of the response $y(\mathbf{x})$ at a sam-

ple point $\mathbf{x}$ is considered as the realization of a Gaussian variable $\mathbf{Y}(\mathbf{x})$ with mean μ and variance σ^2. All the variables $\mathbf{Y}$ are then assumed to be correlated to each other, and the correlation is expressed as a parametric function of the distance of the respective sample points. Several types of such functions were proposed in the literature for GPR, all based on an exponentially decaying structure and here we use the following, well-known, one:

$$Corr(Y(x_i), Y(x_j)) = e^{\frac{-r(x_i,x_j)^2}{a}} \tag{4.9}$$

where $r(x_i, x_j)$ is the spatial distance between two points (x_i, x_j), and the parameter a determines how fast the correlation decreases with the increase of r. We refer the reader to [30] for its derivation. In our implementation, we used the weighted Euclidean distance

$$r(x_i, x_j) = \sqrt{\sum_{l=1}^{d} (w_l(\mathbf{x}_{il} - \mathbf{x}_{jl}))^2} \tag{4.10}$$

where w_l representing the scaling factor for the l-th dimension. Since the higher is the value of w_l the larger is the correlation decrease with the increase of the distance on the l-th dimension, the vector $\mathbf{w}$ is a sensitivity parameter vector associated with the domain dimensions. Given the correlation matrix $\mathbf{R}$ defined according to 4.9 we can define the GPR predictor $\hat{y}(\mathbf{x}^*)$ for a target

point $\mathbf{x}^*$ through the maximization of the likelihood function as follows (see [11] for details):

$$\hat{y}(\mathbf{x}^*) = \hat{\mu} + \rho' \mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{4.11}$$

where

$$\rho = (Corr(Y(x_1), Y(x^*)), ..., Corr(Y(x_n), Y(x^*))) \tag{4.12}$$

is the correlation vector between the target point $\mathbf{x}^*$ and the n sampled points, y being the sampled points value, and $\hat{\mu}$ the estimated mean of the Gaussian variables $\mathbf{Y}$:

$$\hat{\mu} = \frac{\mathbf{1}'\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}} \tag{4.13}$$

The parameters a and $\mathbf{w}$, used in the construction of matrix $\mathbf{R}$ are the only ones that must be determined in order to build our GPR model. Details on the determination of such parameters values are given in the next section. GPR gives also the possibility of estimate the prediction error $s^2(\mathbf{x}^*)$ associated with a target point $\mathbf{x}^*$ which is expressed as follows:

$$s^2(\mathbf{x}^*) = \hat{\sigma}^2(1 - \rho' \mathbf{R}^{-1}\rho) \tag{4.14}$$

where $\hat{\sigma}^2$ is the estimated variance of the $\mathbf{Y}$ variables:

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})' \mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})}{n} \tag{4.15}$$

4.4.3 The SM tuning and validation

We decided to use the LOOCV technique (see Sect. 4.3.2) to check the SM reliability and perform the optimal tuning of the GPR model parameters a and $\mathbf{w}$. The choice is motivated by both the quality of LOOCV indicators and by the relatively small computational effort associated with its use, because the number of tests to be performed is equal to the number of sample points. The prediction error indicator we used is the well-known Sum of the Squared Errors (SSE):

$$SSE = \sum_{i=1}^{n} |\hat{Y}_i(\mathbf{x}_i) - Y_i|^2 \tag{4.16}$$

where $\hat{y}_i$ is the predictor of sample $\mathbf{x}_i$ obtained with the SM built excluding the point $\mathbf{x}_i$ and y_i is the actual value of system response at point $\mathbf{x}_i$.

The calibration of model parameters is performed by a two-steps iterative procedure which aims at minimizing the SSE of the model. In a first step, the sensitivity parameters $\mathbf{w}$ are all set to 1, and only the tuning of a parameter is performed. The optimal value of a is then searched from a set of 1000 candidate values that we choose randomly in the interval $[10^{-2}, 10^{1}]$ (see [30]). However, since the sensibility of the model is higher for values close to 0, we adopted a logarithmic scale for the interval. Each computation of the SSE indicator through LOOCV requires $O(n^4)$ time, because it calls n times

the GPR SM algorithm which requires $O(n^3)$ time. The a parameter is fixed to the value associated with the minimum SSE value the second step of the procedure, used to calibrate the w parameters through a simple descent local search is started. At each iteration, given the current vector w we examine the neighborhood made up by increasing or decreasing by 10% each w_l value. The cardinality of the neighborhood is $O(2^d)$ and if a parameter vector improving the SSE is found the process iterates, and is stopped otherwise. The local search is anyway stopped after 50 iteration to keep the computing time controlled.

4.4.4 The adaptive sampling criteria

The calibrated SM is used to search of the next candidate solutions to be evaluated. Even though, as discussed in Sect. 4.4.2, the GPR-based SM provides the estimation of the predictor error we adopted MRS as the adaptive sampling criteria because it performed experimentally better on our problem according to a preliminary testing. In particular, we developed a multistart gradient-descent procedure, implemented by using multiple threads to reduce the overall running time.

The total number m of points from which perform the gradient search is

fixed. The domain is then discretized by splitting the values for each of the d dimensions (previously normalized in $[0, 1]$) in the same number s of interval, where s is chosen as the larger integer such that $s^d \leq m$. Finally, $m - s^d$ random points are added to those defining the grid. The gradient search is then performed from each of the m points and the final point $\mathbf{x}_g$ which minimizes the SM predictor $\hat{y}(\mathbf{x}_g)$ is determined. If some decision variable must take integer values the integrality is relaxed during the search and the final coordinate is the rounded to the nearest feasible integer value. In our experiments, we used relatively large values of m (e.g., $m = 10^6$) which allowed computing times smaller than two minutes on a normal personal computer. If x_g seems able to improve the incumbent solution x^*, i.e., if $\hat{y}(\mathbf{x}_g) < y(x^*)$, then x_g is returned as the new candidate solution to evaluate and the adaptive sampling procedure ends. Otherwise, we identify the new candidate solution as the one which maximizes the predictor error $s^2(\cdot)$ given by the GPR model over all local minima found in the search. The rationale is that the new candidate solution will improve the reliability of the SM used in the next iterations thus representing an *exploration* strategy.

4.4.5 The overall developed SAO approach

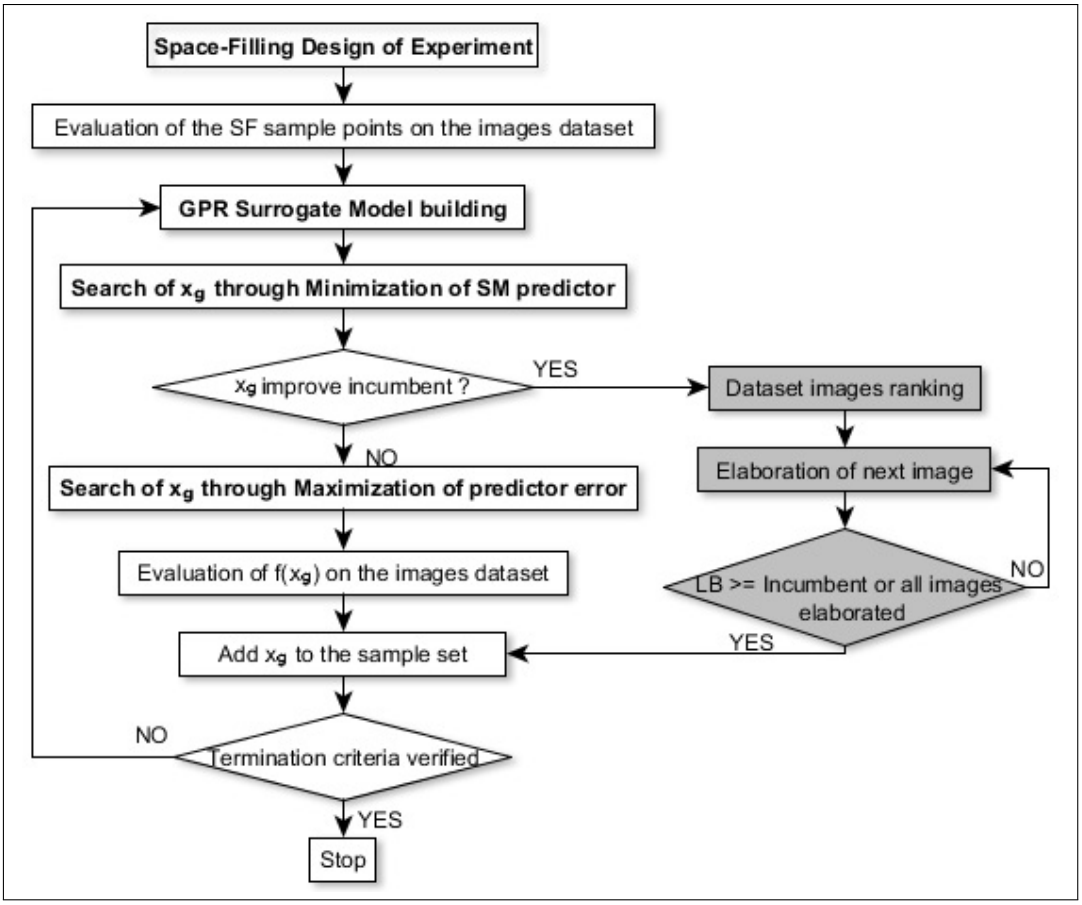

Figure 4.1: The developed SAO approach with time-saving strategy.

Fig. 4.1 illustrates the structure of the overall SOA algorithm we implemented, whose main components were described in the previous sections. At each iteration, the system response on a new candidate solution $\mathbf{x}_g$ has to be performed and this consists in running the CVA, with parameters $\mathbf{x}_g$ on all the images of the test set and computing the corresponding normalized value of the objective function (4.6). As we already discussed, the time to run the

CVA on a single image is not small and typically a large test set is used, hence the evaluation step is highly time consuming. To speed-up the O.F. computation, the images in the test set are ranked according to their average contribution to the objective function in the previously evaluated solutions. Then the images are evaluated in the ranking order and the evaluation stops as soon as the current objective function is equal to the incumbent solution. In fact, we observe that the contribution of each image evalutation to the objective function value is non-negative. Therefore, an evaluation of the objective function on a subset of images represents a lower bound LB on the value of the system response and if $LB \geq y(\mathbf{x}^*)$ this ensures that the current solution is not improving the incumbent. At the end of the evaluation the current solution is added to the sample set and the incumbent is possibly updated. The image ranking mechanism and the possible early termination of the CVA evaluation proved extremely effective in the experimental evaluation of the algorithm in the real-world case study described in the next section.

4.5 Experimental Results

In this section, we describe the experimental validation of the proposed algorithm both on test problems from the literature and on a real-world case

study of defect detection on tyre images. The overall algorithm, described in Section 4 was implemented in c++ and run on a PC with Quad-Core 3.6Ghz equipped CPU and 32Gb of RAM. In the real-world case study the image evaluation by the CVAs is performed through a legacy software for image processing which is called by the SAO algorithm. We initially tested our algorithm on a well-known standard benchmark of BBO from the literature to fine-tune its implementation and assess its quality with respect to the current state-of-the-art in BBO. Then, in Sect. 4.5.2 the algorithm is tested on the real-word industrial application of tuning CVA for defect detection of tyres at the end of the production line. In particular, we considered the optimal calibration of two used in the automated defect detection of tyres produced by an industrial plant. More specifically, the experiments were conducted in collaboration with one of the world's largest tyre production company. The target of the CVAs was the detection of defects like air bubbles, cuts and other imperfections on the surface of the produced tyre at the end of the production line.

4.5.1 Testing on a problem from the literature.

The overall SAO approach has been initially validated by optimizing a widely-used mathematical benchmark in BBO: the 6-variables Hartmann function

(see [31]). The performance of the SAO was compared with the recent approach proposed by [32] which was also validated using the same benchmark. In [32], the authors tested different initialization methods, SM building techniques and search strategies and the best performances were obtained with a GPR-based SM, an MRS adaptive sampling method, and a Quasi-Monte Carlo (QMC) initialization strategy. As described in Chapter 4.4 one of the main differences of our SAO method with respect to that of [32] is the SF-based DoE used to initialize the algorithm, while no specific details on the implementation of MRS and the GPR-based SM are given in their paper. In Fig. 4.2 we compare the evolution of the best solution found by three different runs of our algorithm, corresponding to different random seeds and denoted as BV_1, BV_2 and BV_3, with that of best algorithm developed in [32] denoted as Benchmark. As in [32] we initialized our algorithm with 50 samples and run it until either the difference with respect to the known global optimum is smaller than 10^{-3} or 300 iterations are performed. By examining Figure 3 we note that our algorithm strongly outperforms the best results obtained by [32], in all three runs we reached the global optimum in less than 70 iterations while those required by the best method presented in the literature are more than three times larger. In addition, the improvement rate is extremely steep and the behaviour of our algorithm is very similar in all three runs showing clearly the benefits of the initialization and the specific

implementation of the different components that we propose.

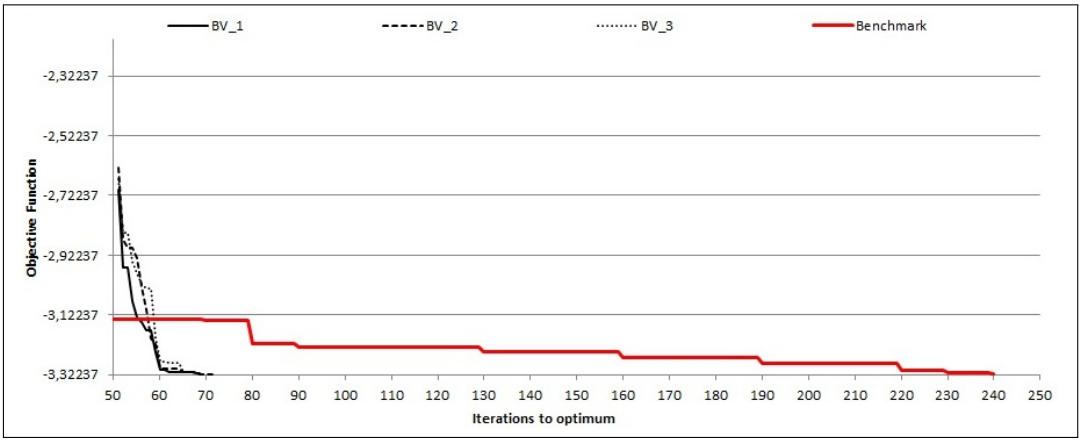

Figure 4.2: 50 points initialization - comparison with Wang et al. (2014).

4.5.2 Testing on a real-world industrial problem.

We discuss here the testing of the proposed SAO approach on the industrial problem described in Section 4.2. To validate our approach, we optimize the parameters of two different CVAs, hereafter called CVA1 and CVA2, dedicated to detect two different type of defects. In our experiments, we initially calibrated the parameters for CVA1 by using in the objective function (4.6) three different combination of weights for the false positive and negative occurrences to evaluate the sensitivity of the approach to the weights. We also examined the impact of the time-saving procedure for the image evaluation described in Section 4.4.5. Finally, we performed the tuning of CVA2 by using the complete algorithm with the time-saving procedure. In all experiments, we compare the results obtained by the SAO approach with the performance

of the CVA when using a set parameters manually optimized by an expert designer of CVAs.

Let us first consider CVA1, which is characterized by six parameters to be optimized. As training set we are given a set of 128 images for which the presence or not of defects is known. For the objective function (4.6) we considered a=1 (i.e., the weight for the false positive results) and three different weight combinations for the false negative result, namely b= 5, 10, and 20. In total, we ran six experiments by considering the three weight combinations and by enabling or not the time-saving procedure of Section 4.4.5. In all cases, we used 50 points for the space-filling initialization step, thus initially requiring a total of 6400 image evaluations. All runs were stopped when 100 iterations were performed after the initialization. The results are illustrated in Figures 4.3 to 4.8 where it can be seen that our approach with all the three weight combinations is capable of finding much better parameter settings with respect to the manual benchmark within a relatively small number of objective function evaluations. In addition, we may observe that the use of the time-savings procedure may result in a slightly larger number of objective function evaluations to reach the best result while it requires considerably less image evaluations and thus less overall computing time. These two behaviors are easily explained. First of all, the time-savings procedure uses a normalized

objective function evaluation which is actually an approximation of the true objective function. Thus, an early termination of the evaluation of a point may result in a (slightly) greater prediction error of the SM which can lead to the need of additional point evaluations before good SM approximations can be obtained. On the other hand, the early termination limits the total number of images that are actually tested thus reducing considerably the computing time for the evaluation of the new points. Both described effects are quite marginal for the case with b=5 and b=10, while are much larger for the case b=20. More precisely, for b=5 the time-saving based algorithm requires 1.2% less image evaluations to reach the best value and about 30% less evaluations in total. The corresponding values for b=10 are 4.6% and 11.3%. As to the case b=20, it requires 14% less image evaluations to reach the best value and about 22% less evaluations in total. In all cases, the algorithm considerably improves the best solution found during the initialization step. In particular for the three parameter combinations the improvement is equal to 26%, 12% and 5%.

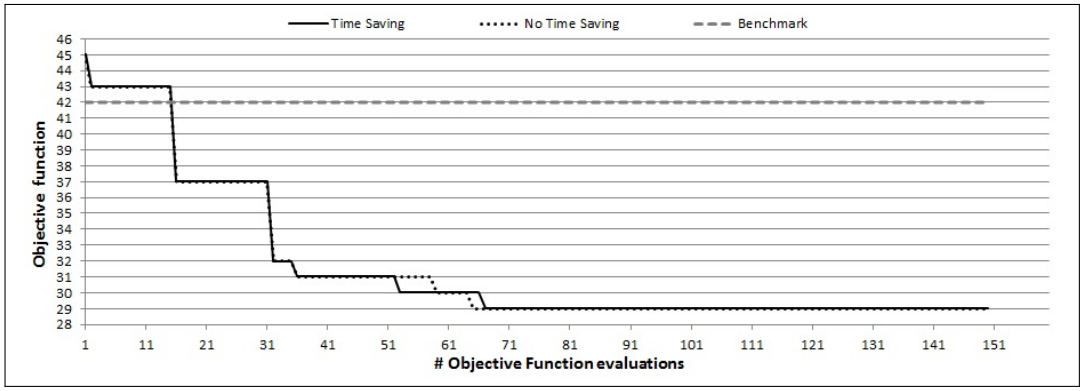

Figure 4.3: Impact of the time-saving procedure on the optimization of CVA1 parameters with a=1 and b=5. The benchmark is the manual solution and the evolution with respect to the total number of objective function evaluations is reported.

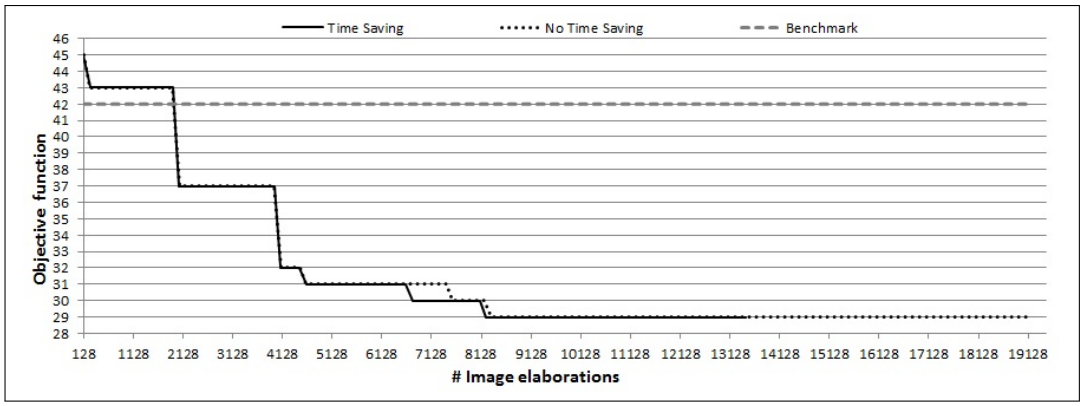

Figure 4.4: Impact of the time-saving procedure on the optimization of CVA1 parameters with a=1 and b=5. The benchmark is the manual solution and the evolution with respect to the total number of image evaluations is reported.

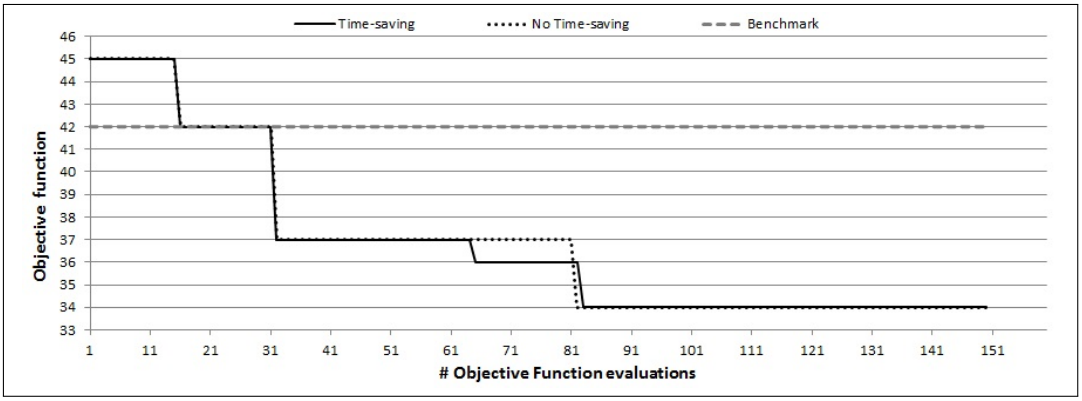

Figure 4.5: Impact of the time-saving procedure on the optimization of CVA1 parameters with a=1 and b=10. The benchmark is the manual solution and the evolution with respect to the total number of objective function evaluations is reported.

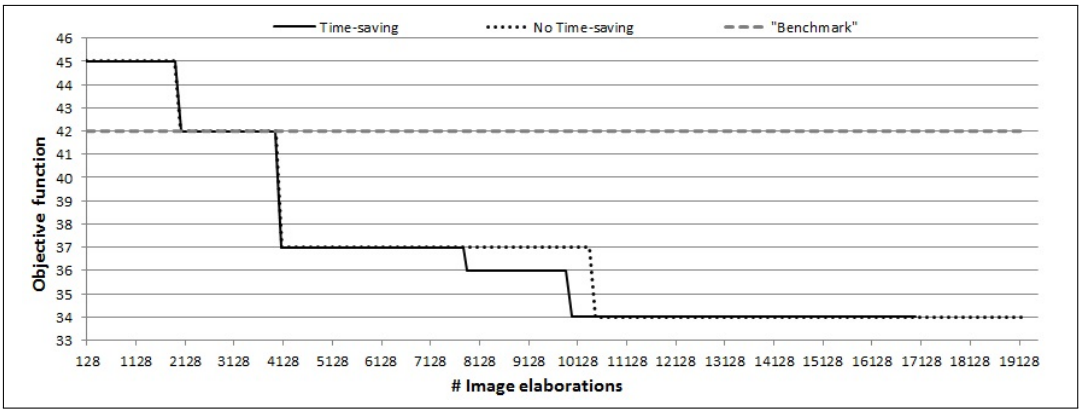

Figure 4.6: Impact of the time-saving procedure on the optimization of CVA1 parameters with a=1 and b=10. The benchmark is the manual solution and the evolution with respect to the total number of image evaluations is reported.

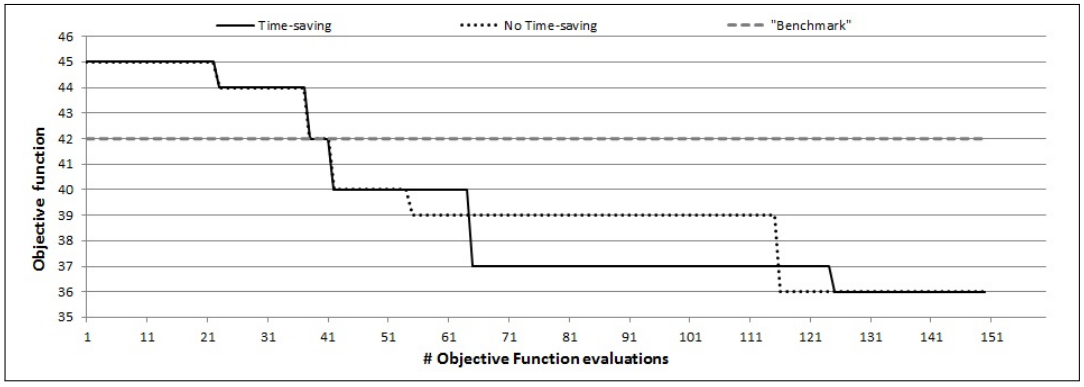

Figure 4.7: Impact of the time-saving procedure on the optimization of CVA1 parameters with a=1 and b=20. The benchmark is the manual solution and the evolution with respect to the total number of objective function evaluations is reported.

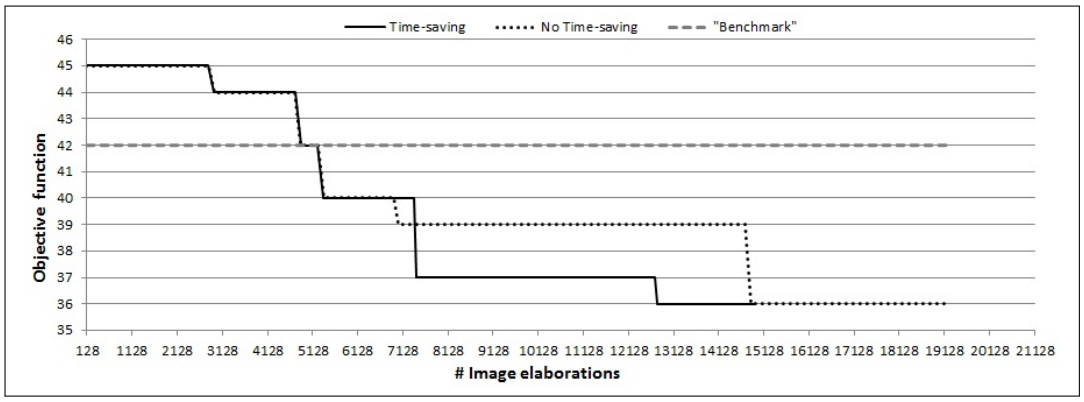

Figure 4.8: Impact of the time-saving procedure on the optimization of CVA1 parameters with a=1 and b=20. The benchmark is the manual solution and the evolution with respect to the total number of image evaluations is reported.

b	Time savings enabled	Benchmark solution (#fp, #fn)	Initial solution (#fp, #fn)	Best solution (#fp, #fn)	% impr.	# Obj. eval. for Best solution	# Obj. eval. Total	# images eval. for Best solution	# images eval. Total
5	No	42 (42, 0)	31 (21,2)	29 (24,1)	31.0	65	150	8320	19200
5	Yes	42 (42, 0)	31 (21,2)	29 (24,1)	31.0	67	150	8224	13455
10	No	42 (42, 0)	37 (27,1)	34 (24,1)	19.0	83	150	8320	19200
10	Yes	42 (42, 0)	31 (21,2)	29 (24,1)	19.0	67	150	8224	13455
20	No	42 (42, 0)	40 (40,0)	36 (36,0)	14.3	116	150	14848	19200
20	Yes	42 (42, 0)	40 (40,0)	36 (36,0)	14.3	125	150	12768	14936

Table 4.1: Summary of CVA1 experiments with the three different combination of weight ($b = 5, 10, 20$) and the Time-Saving procedure enabled and not.

	images eval. % savings	
b	for Best solution	Total
5	1.2	29.9
10	4.6	11.3
20	14.0	22.2

Table 4.2: Summary of the image evaluations savings obtained with the CVA1 optimizations with the three different weight combinations and using the Time-Saving procedure.

When comparing the optimizations of CVA1, with the Time-Saving strategy enabled, for all the three combinations of weights used, we note that the number of images elaborations necessary to achieve the optimum grows with the increase of the difference between the false positive and negative weights. The reason is that, when the difference increases the interpolating surface built by the surrogate model becomes more multimodal as the values of the objective function at the sampled points are more variable.

For all the three combination of weight, both with and without the Time-Saving procedure enabled, the algorithm always found optimal parameter solutions significantly better than the benchmarks. In particular, for b=5, b=10 and b=20, the objective function improvement is respectively 31%, 19% and 14%. Thanks to the Time-Saving procedure, the same optimal solution are found with a savings of image evaluations (i.e., computational time) of about 30%, 11% and 22%.

We also conducted a much more challenging experiment by considering CVA2 which requires eleven parameters to be optimized. Moreover, in this case the dataset used for the training of the algorithm includes 912 images, thus requiring 45,600 image elaborations just for the initialization step. In addition, CVA2 detects a defect which is much more critical than that of CVA1 thus a larger weight for the false negatives is required: we used a=1 and b=25 as weights. The evolution of the objective function with respect to the total number of image evaluations is illustrated in Figure 4.9 and the results are summarized in Table 4.3. We note that also in this case the proposed algorithm finds parameter settings much better than those of the benchmark, with an improvement equal to 22%, and that the improvement with respect to best solution found in the initialization step is 11.2%.

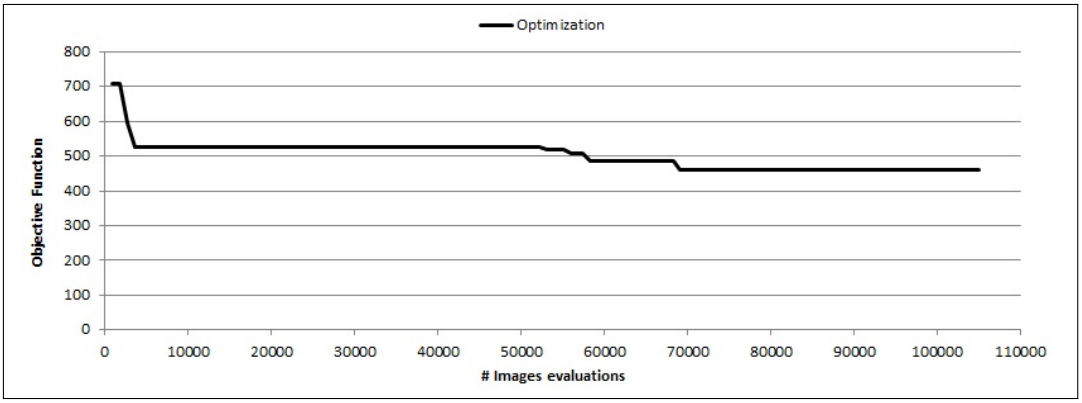

Figure 4.9: CVA2 Optimization with weight a=1 and b=25.

Benchmark solution (#fp, #fn)	Initial solution (#fp, #fn)	Best solution (#fp, #fn)	% impr.	# Obj. eval. for Best solution	# images eval. for Best solution
591 (441, 6)	525 (125,16)	461 (311,6)	22.0	83	69088

Table 4.3: CVA2 Optimization results with weight a=1 and b=25.

4.6 Conclusions

In this work, we have developed and tested a Sequential Approximate Optimization approach specifically tailored to perform the parameter tuning of Computer Vision Algorithms employed in automated defect detection. The industrial relevance of this application area is highly increasing thanks to the great progress achieved by CVA accuracy and reliability. However, the fine tuning of the CVA requires to define specific parameter settings for different defects and image types to be analysed and such activity is highly time consuming when done manually by expert designers thus enhancing the need

of reliable and automated procedures to define the optimal parameters in various use conditions. To the best of our knowledge, no specific optimization method has been proposed in the literature for this type of applications, hence we developed a SAO approach specifically designed for CVA parameters tuning. The proposed approach was tested both on a benchmark for Black Box optimization from the literature and on two real-world cases from an industrial application in the field of tyre production. Our testing shows that the proposed SAO algorithm is capable of producing, within reasonable computing times, parameter settings which are much better than those manually obtained by the designers. Furthermore, the various features of our implementation such as the space-filling initialization Design of Experiment and the time-saving procedure employed to reduce the number of image evaluations, are highly effective in the improvement of its performance. As already mentioned, this is an application area in rapid evolution. Therefore, several important directions for further research exist. For example, considering different initializations which require less sample points or more efficient adaptive sampling for the detection of the new candidate point at each iteration may be desirable. In addition, given the positive impact of the image-ranking based time saving, new techniques to reduce the number of image evaluations at each iteration should be developed.

Chapter 5

Conclusions

In this dissertation we firstly analyze the state of the art of DFO approaches, and then we developed a Sequential Approximate Optimization approach specifically tailored to perform parameter tuning of CVAs employed in automated defect detection.

Our testing shows that the proposed SAO algorithm is capable of producing, within reasonable computing times, very good parameter solutions. Furthermore, the various features of our implementation such as the time-saving procedure employed to reduce the number of image evaluations, shown to be highly effective in the improvement of its performance.

This results point out that our Derivative Free Optimization approach is suitable in facing CVAs parameter tuning problems, especially when CVAs are employed in automated defect detection and, therefore, the resulting

problem consists in the optimization of their operating characteristic curve. Moreover, the effectiveness of our method in term of solutions quality and computational effort control, due to the time-saving procedure proposed, suggest that space for further research exists. For example, the search of further techniques aimed to highly reduce the computational effort, by partial elaboration of the images dataset, seems to be promising.

List of Tables

List of Figures

Bibliography

[1] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[2] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[3] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. Siam, 2009, vol. 8.

[4] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM review*, vol. 45, no. 3, pp. 385–482, 2003.

[5] K. I. McKinnon, "Convergence of the nelder-mead simplex method to a nonstationary point," *SIAM Journal on Optimization*, vol. 9, pp. 148–158, 1998.

[6] P. Tseng, "Fortified-descent simplicial search method: A general approach," *SIAM Journal on Optimization*, vol. 10, pp. 269–288, 1999.

[7] A. R. Conn, K. Scheinberg, and P. Toint, "Recent progress in unconstrained nonlinear optimization without derivatives," *Mathematical Programming*, vol. 79, pp. 345–397, 1997.

[8] M. J. Powell, "Unconstrained optimization by quadratic approximation," *Mathematical Programming*, vol. 92, pp. 555–582, 2002.

[9] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.

[10] J. Holland, "Adaptation in natural and artificial systems, univ. of mich. press," *Ann Arbor*, 1975.

[11] D. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of global optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[12] D. Rajnarayan, A. Haas, and I. Kroo, "A multifidelity gradient-free optimization method and application to aerodynamic design," in *12th AIAA/ISSMO multidisciplinary analysis and optimization conference, Victoria, British Columbia, AIAA*, vol. 6020, 2008.

[13] K. Naono, K. Teranishi, J. Cavazos, and R. Suda, *Software Automatic Tuning: From Concepts to State-of-the-Art Results.* Springer-Verlag, 2010.

[14] J. Lu and T. Adachi, "A parameter optimization method for electronic circuit design using stochastic model function," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 75, no. 4, pp. 13–25, 1992.

[15] K. Ujjwal and J. Aronson, "Genetic algorithm based bargaining agent for implementing dynamic pricing on internet," in *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on.* IEEE, 2007, pp. 339–343.

[16] T. Newman and A. Jain, "A survey of automated visual inspection," *Computer Vision and Image Understanding*, vol. 61, no. 2, pp. 231–262, 1995.

[17] A. Kumar, "Computer-vision-based fabric defect detection: A survey," *IEEE Transaction on Industrial Electronics*, vol. 55, no. 1, pp. 348–363, 2008.

[18] D. Jones, M. Schonlau, and W. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13,

pp. 455–492, 1998.

[19] S. Kitayama, M. Arakawa, and K. Yamazaki, "Sequential approximate optimization using radial basis function network for engineering optimization," *Optim Eng*, vol. 12, pp. 535–557, 2011.

[20] N. Queipo, R. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Tucker, "Surrogate-based analysis and optimization," *Progress in Aerospace Science*, vol. 41, pp. 1–28, 2005.

[21] G. Fasshauer, *Meshfree Approximation Methods with Matlab*. World Scientific Publishing Company, 2007, vol. 6.

[22] D. McDonald, W. Grantham, W. Tabor, and M. Murphy, "Global and local optimization using radial basis function response surface models," *Applied Mathematical Modeling*, vol. 31, pp. 2095–2110, 2007.

[23] T. Newman and A. Jain, "A survey of automated visual inspection," *Computer vision and image understanding*, vol. 61, no. 2, pp. 231–262, 1995.

[24] D. Jones, M. Schonlau, and W. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[25] A. Bessi, D. Vigo, V. Boffa, and F. Regoli, "Parameter tuning for image recognition algorithms through sequential surrogate optimization," *Optimization and Decision Science: Methodologies and Applications*, vol. 217, 2017.

[26] K. Vu, C. D'Ambrosio, Y. Hamadi, and L. Liberti, "Surrogate-based methods for black-box optimization," *International Transaction in Operative Research*, vol. 24, pp. 393–424, 2017.

[27] S. Koziel, D. Ciaurri, and L. Leifsson, *Computational Optimization, Methods and Algorithms*. Springer, 2011.

[28] Garbowski and Vladimir, "Recent advances in computational mechanics," pp. 285–291, 2014.

[29] A. Forrester and A. Keane, "Recent advances in surrogate-based optimization," *Progress in Aerospace Sciences*, vol. 45, no. 1-3, pp. 50–79, 2009.

[30] A. Sobester, S. Leary, and A. Keane, "On the design of optimization strategies based on global response surface approximation models," *Journal of Global Optimization*, vol. 33, pp. 31–59, 2005.

[31] L. Dixon, "The global optimization problem. an introduction," *Toward global optimization*, vol. 2, pp. 1–15, 1978.

[32] C. Wang, Q. Duan, W. Gong, A. Ye, Z. Di, and C. Miao, "An evaluation of adaptive surrogate modelling based optimization with two benchmark problems," *Environmental Modelling and Software*, vol. 60, pp. 167–179, 2014.