DISSERTAÇÃO DE MESTRADO

Surrogate-assisted Optimization using Multi-objective Evolutionary Techniques Applied to Mechanical Structural Design

João Carlos Pereira Passos

Brasília, agosto de 2019

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO

Surrogate-assisted Optimization using Multi-objective Evolutionary Techniques Applied to Mechanical Structural Design

João Carlos Pereira Passos

Dissertação de Mestrado submetida ao Departamento de Engenharia Mecânica como requisito parcial para obtenção do grau de Mestre em Sistemas Mecatrônicos

Banca Examinadora

Prof. Dr. Carlos H. Llanos Quintero, ENM/FT/UnB _____
Orientador

Prof. Dr. Mauricio Ayala Rincón, MAT/UnB _____
Examinador externo

Prof. Dr. Adriano Todorovic Fabro, ENM/FT/UnB _____
Examinador externo

FICHA CATALOGRÁFICA

PASSOS, JOÃO CARLOS PEREIRA
Surrogate-assisted Optimization using Multi-objective Evolutionary Techniques Applied to Mechanical Structural Design [Distrito Federal] 2019.
xvi, 92 p., 210 x 297 mm (ENM/FT/UnB, Mestre, Engenharia Mecânica, 2019).
Dissertação de Mestrado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Mecânica
1. Optimização Multi-Objetivo
2. Surrogates
3. Análise de Elementos Finitos
4. Design de Estruturas Mecânicas.
I. ENM/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

PASSOS, J.C.P (2019). *Surrogate-assisted Optimization using Multi-objective Evolutionary Techniques Applied to Mechanical Structural Design* . Dissertação de Mestrado, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 92 p.

CESSÃO DE DIREITOS

AUTOR: João Carlos Pereira Passos
TÍTULO: Surrogate-assisted Optimization using Multi-objective Evolutionary Techniques Applied to Mechanical Structural Design .
GRAU: Mestre em Sistemas Mecatrônicos ANO: 2019

João Carlos Pereira Passos
Depto. de Engenharia Mecânica (ENM) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Acknowledgements

I want to thank Professor Llanos for his expert advice and encouragement throughout this challenging project, as well as my family and friends that have been supporting me despite all adverse situations and difficulties.

João Carlos Pereira Passos

RESUMO

Este manuscrito desenvolve um novo algoritmo de otimização assistida por substitutos, a fim de mitigar os problemas críticos que afeta os problemas de otimização estrutural: a) alto custo computacional e b) longo tempo de execução. A abordagem baseada em substitutos utiliza modelos substitutos que possuem baixo custo de avaliação para reproduzir o comportamento das funções de custo dos problemas, nos quais nossa técnica é capaz de aprimorar dinâmicamente os modelos substitutos sem a utilização de uma população de validação. Além disso, também implementamos um mecanismo que periodicamente tenta recuperar boas soluções que foram anteriormente descartadas devido à deficiência do modelo. Além disso, este manuscrito abrange o desenvolvimento de três novos algoritmos de preenchimento de espaço, que lidam com o problema de adicionar novos pontos de maneira eficiente a um modelo substituto já existente. Mais ainda, criamos uma métrica de preenchimento de espaço que mede a presença de lacunas em uma população de pontos dentro de um espaço de busca, o que possibilita a comparação de distribuições de tamanhos diferentes. Por fim, três estudos de caso foram usados para revelar as capacidades e limitações da nova técnica de otimização assistida por substitutos em conjunto com o algoritmo de preenchimento de espaço. Dois desses estudos de caso mostraram que o método é capaz de promover um expressivo ganho de desempenho, conciliada com uma pequena perda na precisão. No entanto, o terceiro estudo de caso revelou que a técnica também pode falhar na produção de resultados satisfatórios.

Palavras-chave: Otimização multi-objetivo, modelos substitutos, otimização estrutural, projeto mecânico.

ABSTRACT

This manuscript develops a new surrogate-assisted optimization algorithm in order to mitigate the critical issues that pertain problems of structural optimization: a) high computational cost and b) long execution time. The surrogate approach utilizes surrogate models that have cheap evaluation cost to reproduce the actual cost functions of the problems, in which our technique is capable of dynamically improve these models without the utilization of a validation population. Moreover, we also implemented a mechanism that periodically tries to recuperate suitable solutions that were previously discarded due to the model's deficiency. Besides, this manuscript covers the development of three new space-filling algorithms, which handle the problem of efficiently adding new points into an already existing surrogate model. Additionally, we created a space-filling metric that measures the presence of gaps in a population of points within a search space, which enables the comparison of distributions of different size. At last, three case studies were used to reveal the capabilities and limitations of the new surrogate-assisted optimization technique in conjunction with the space-filling algorithm. Two of these case studies show that the method is capable of promoting an expressive performance gain reconciled with a small loss in accuracy, whereas the other reveals that it could also fail to output satisfactory results.

Keywords: Multi-objective optimization, surrogate, structural optimization, mechanical design.

Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Engineering problems commonly possess more than one solution, each of them with different advantages and disadvantages. Nonetheless, these problems are inherently an optimization problem composed by three basic components: (a) the *design variables*, (b) the *state variables*, and (c) the *objective functions* (1).

The *design variables* list the set of changeable parameters of the problem. They effectively correspond to the components under optimization. In contrast, *state variables* render the response of the problem, given the selected design variables and the simulation scenario itself. At last, the *objective functions* utilize the problem's behavior or characteristics to indicate the goodness of the evaluated design. Ideally, these indicators yield comparable values and are mostly numeric values. In the literature, another conventional denotation for the objective function is cost function.

Figure 1.1 depicts an optimization process, which its iterative process has unrestricted access to the cost function (2, 3), where X and F are respectively the *design variables* and *objective functions*.

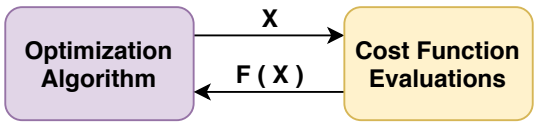


Figure 1.1 – Optimization Process

However, there are cases that unrestricted access to the cost function is a problem itself because they could demand the execution of experiments, that could be both laborious and financially expensive. Regardless, it is a fact that both of them are time-consuming, making the conventional approach impracticable as the optimization process requires several inputs of the cost function (2, 4, 5).

In that context, *structural optimization* describes a procedure capable of identifying a structure's geometry and material that enables a higher efficient usage of materials on predefined operational conditions while fulfilling a set of project restrictions and safety regulations. Nonetheless, the evaluation of these structural design problems demands the execution of simulations of high complexity, which represents a critical issue due to its high computational cost and long execution time (6), that are commonly performed by specific FEM software packages, e.g., *Ansys*, *Abaqus*, and *Autodesk Simulation*. Moreover, these difficulties only become worse at an optimization procedure, which requires the execution of many of such simulations in order to guide the search process towards the minimum/maximum global.

A promising solution for these issue is the utilization of a surrogate-assisted optimization (7), which describes an optimization process that uses surrogate models instead of the actual

cost functions. These surrogate models are functions that mimic the behavior of the actual cost functions but possess a cheap evaluation cost reconciled with a generic representation of the problems (8).

Figure Fig. 1.2 depicts a surrogate-assisted optimization process, where *X* and *F* still describe the current population of the optimization algorithm and the output of the cost functions, but *Xs* and *Fs* respectively the *training points* used to generate the surrogate models and the *output* of the surrogate models.

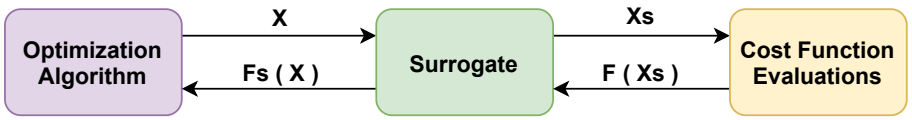


Figure 1.2 – Surrogate-assisted Optimization Process

The utilization of surrogate models by optimization algorithms grants a significant reduction of the process overall computational cost. Nevertheless, the selected population sampling strategy and model building method greatly influence the technique's performance, so that it could fail to emulate the source function, or be as costly as the direct approach.

In that context, this works develops a new surrogate-assisted optimization algorithm, which it is capable of actively updating the surrogate models based on violations on the expected cost functions' range of values. Moreover, the algorithm implements a mechanism that periodically tries to recuperate suitable solutions that were previously discarded due to the deficiency of the surrogate models.

Another contribution was the development of three new space-filling algorithms, which handle the problem of efficiently adding new points into an already existing surrogate model. Highlights that the main algorithm adopts a straightforward strategy to identify the additional points' location, which translates into a low computational cost, even at high dimension problems. Nonetheless, this approach is still capable of outperformance the more complex implementations of the space-filling algorithm.

The final contribution consists in the creation of a space-filling metric that measures the presence of gaps in a population of points within a search space, which enables the comparison of distributions of different size.

At last, three case studies were used to reveals the capabilities and limitations of the new surrogate-assisted optimization technique in conjunction with the space-filling algorithm, in which have shown that in two of them the method is capable of promoting an expressive performance gain reconciled with a small loss in accuracy, whereas the other reveals that it could also fail to output satisfactory results.

1.1 THE PROBLEM AND THE METHOD USED

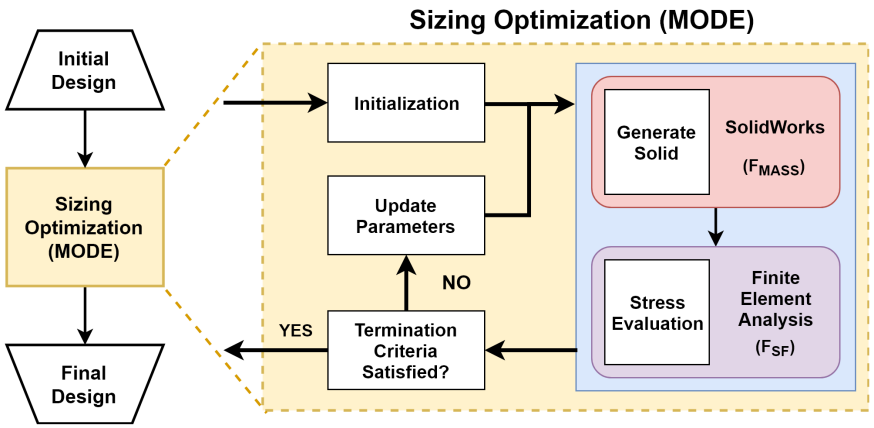


Figure 1.3 – The Method used in this work

Figure 1.3 depicts the data flow of the overall method initially proposed in this work and presented in (9), where different tools were integrated, such as Multi-objective Optimization based on Differential Evolution (MODE) algorithm, SolidWorks tool, and Matlab's FEA toolbox. Of which, the SolidWorks and Finite Elements Analysis (FEA) were used for the purpose of evaluating the cost function (fitness) in order to guide the optimization process.

In the context of optimization tasks, Multi-objective Optimization Problem (MOOP) algorithms are used in problems with two or more conflicting criteria. The sizing optimization (used in this work) was modeled as a MOOP because both safety factor and mass are conflicting objectives. Therefore, the notion of the dominance concept is important to measure which solutions are better, because both objective functions in MOOP can have equivalent importance to a decision-maker.

It can be observed that the design problem of a mechanical part (in the context of *e.g., structural optimization*) can be seen as an Optimization Problem (OP), guided by a cost function. In this case, the optimization algorithm (e.g., a bio-inspired one) must evaluate the cost function in each iteration. But in this case, the cost function has a high mathematical complexity (e.g., based on large system of equations equations), and/or evaluated by using computationally-expensive simulation tools, namely *Ansys*, *Abaqus*, *Autodesk Simulation*, and *Matlab*.

For such problems the use of surrogate models is vital and recommended given that the optimization process to be used, in order to obtain a suitable project, can spend hours, weeks or months, depending on the complexity of the project to be faced.

1.2 RELATED WORKS

The applications that utilize surrogate model techniques in the design and analysis of computer experiments is a promising field of research, which has continually gain attention in the last 30 years (10). In that context, several publications were make to discuss the establish approaches and future trends in this research area (11, 12, 2, 8, 13), in which the works overview the present state of the art of constructing surrogate models, optimization strategies, feasibility analysis, and many other aspects of pertaining the utilization of surrogate models. More so, these techniques have already been applied in several optimization problems in order to mitigate the high computational of engineering simulations (14, 15, 16, 3, 17).

Thus, it is possible to find similar approaches to our work in literature, which Gong and Duan (18) proposes an adaptive surrogate modeling-based sampling strategy for parameter optimization and distribution estimation, Wang et al. (5) develops a data-driven surrogate-assisted multi-objective optimization based on gaussian process, and Voutchkov and Keane (19) discusses the idea of using surrogate models to reduce the number of cost function calls for multi-objective optimization.

1.3 OBJECTIVES

1.3.1 General Objective

The main objectives of this manuscript are to develop a new surrogate-assisted optimization algorithm, which it is capable of actively updating the surrogate models based on violations on the expected cost functions' range of values. With our technique, we aim to answer whether it is possible to represent the dynamics of the cost functions with a small set of points.

1.3.2 Specific Objectives

- Develop a procedure to assist in the identification of new points for updating the surrogate model.
- Develop a metric to measure the search space's coverage of the surrogate model

1.4 CONTRIBUTIONS

The main contributions of this manuscript are:

- Development of a new surrogate-assisted optimization algorithm, which it is capable of actively updating the surrogate models based on violations on the expected cost functions'

range of values. Moreover, the algorithm implements a mechanism that periodically tries to recuperate suitable solutions that were previously discarded due to the model's deficient.

- Development of three new space-filling algorithms, which handle the problem of efficiently adding new points into an already existing surrogate model. High-lights that the main algorithm adopts a straightforward strategy to identify the additional points'location, which translates into a low computational cost, even at high dimension problems.
- The creation of a space-filling metric that measures the presence of gaps in a population of points within a search space, which enables the comparison of distributions of different size.

1.5 MANUSCRIPT ORGANIZATION

This manuscript introduces, in Chapter 2 detail the groundwork fundamentals that acts a the base of this work. There we discuss the creative process of products and its particularities in an engineering context. We also detail an approach to solve complex problems and explains the fundamental concepts of structural optimization.

Chapter 3 provides an in-depth overview of the internal operation of a typical surrogate model, which explains the process of selection of the training points and its update strategies. This chapter is also responsible for detailing the techniques applied to construct a surrogate model and the challenges of its utilization.

Chapter 4 explains the main difference between surrogate-assisted optimization and conventional optimization. It shows the internal operation of the proposed surrogate-assisted optimization process, detailing the peculiarities of cost functions and the solutions itself.

Chapter 5 devotes to the development of three space-filling algorithms, which details the advantages and disadvantages of them. Likewise, develops a space-filling metric in order to measure the presence of gaps in populations and the challenges imposed by high dimension problems.

Chapter 6 develops three structural optimization problems as case study in order to validate and test the functionalities of surrogate-assisted optimization algorithm.

Chapter 7 provides the concluding remarks as well as the direction for future works.

2 THEORETICAL BACKGROUND

The following chapters detail the groundwork fundamentals that acts a the base of this work. Of which, Chapter 2.1 discuss about the creative process of products. Chapter 2.2 explains the creative process in a engineering context. Chapter 2.3 details a solution approach to solve complex problems. At last, Chapter 2.4, 2.5, 2.6, and 2.7 explains key concepts of structural optimization.

2.1 DESIGN PROCESS

Design Process denotes a functional sequence of steps used in the conception of processes and products, i.e., a methodology focus on the creative process itself (20, 21). This process features high interactivity and cycle behavior, which ensures the meeting of its attainable objectives requirements. Thus, "learning from failure" is a key concept of this methodology, for the "journey" towards the solution molds its upshot and maybe even the problem itself (22, 23). Figure 2.1 depicts the flowchart of a typical eight-step design process.

Figure 2.1 – Design Process - Flowchart

The *Problem Definition* is the methodology's first step, procedure responsible for the definition of the set of goals that conditions the task' fulfillment. The method is innately flexible, being the objectives a product, answering to a question, or even the creation of another methodology altogether. Either way, during this stage, the project designers endeavor integrally focus on un-

derstanding as much as possible about the problem's endgame. Likewise, the designer has to take special care of its requester, because they may not be able to convey their needs nor comprehend their feasibility.

Once both parties reach an agreement about the project's goals, follows *Background Research* on the problem. The objective of this step consists of locating comparable predicaments to seek reference from their pre-existing solutions, should they exist. At this stage happens the first project revision, which makes expectations and outcomes increasingly more realistic. This revision is a procedure in which the gathered information serves to refine the problem definition further. Moreover, the cycling process of research and refining occurs until the meeting of the client's expectations and project constraints.

The *Design Requirements*, respectively, the methodology's third step, translates the requester's expectations and project constraints into requirements. These design requirements describe essential characteristics that the proposed solution must comply to indicate the project conclusion. Likewise, it serves to guide the advance during its development process. In here are defined features or specifications that the product or process must satisfy to meets its minimum passing grade.

The trickiest question in a design process is: The requirements mold the solutions or the solutions mold the requirements? The correct answer would that the methodology makes a compromise of both. Realistic speaking, some problems are currently unfeasible, or its known solution violates the problem's requirements. In those situations, there must happen a problem revision itself or loosen up its demands.

As there are problems without a single solution, there are others that have several established solutions. In that context, the *Brainstorming* aims to list all possible methods to solve the problem, including solutions that do not respect the problem's actual requirements or constraints. The core concept of this step lies in solving the problem at any cost, not bound by restrictions or a biased mindset. That happens because the problem may not be as well-defined as given credit, or its requirements are still unrealistic or do not conform with current technology limitations.

Once the bank of solutions is complete, the designer can gauge their pros and cons given the project constraints and design requirements. Additionally, this compendium enables the visualization of the current state of the art, thus allowing an understanding of potential contributions and selling points of the final product. During the execution of the *Solution Selection*, there will be straight rejections, but also solution that can entirely or partially fulfill the restrictions. The final selection will not be only based on those observations, because conventionally the designer attributes different weights to each of the problem's constraints. So that, even a solution that does not attend a set of criteria could be the chosen one.

Figure 2.1 designates the lasts three steps of the methodology as *Solution Refinement*. This grouping aims to indicate that these steps operation differs from the rest of the others. Although cyclic behavior marks the whole process, this section has a particular loop interaction. That happens because inputs regarding the solution's outputs are necessary to keep the refinement

in track with the problem's constraints. For that reason, those steps are the leading workload initiators in the entire process.

The first step of the *Solution Refinement* is responsible for the development of the solution itself. The designer implements the technical solution to the problem with the available information. However, due to the existence of problems that have data scarcity, unknown behavior, or simplified model representations, the experiment's results may differ from the expected outcomes (2, 5). Thus, trial and error its one of the leading methods to obtaining insights and at the same time verify the solution (24).

In that context, prototypes are built to evaluate those intermediate implementations of the solution in the field. However, the construction of a prototype proves to be time and resource consuming, especially when some of the tests are of the destructive kind. Thus, from the project manager's point of view, it is a money-burning process that should be seldom done, particularly considering the possibility of failures experiments.

A fundamental concept in prototyping is *Design Fidelity*, which indicates the level of detail and functionality included in a prototype, measuring how close it matches to the complete version (25). There are low-fidelity ones that could be simple paper sketches, commonly used in the earliest stages to generate feedback. On the other hand, high-fidelity prototypes are partially or fully functional versions. The latter are often employed in the later stages of the refinement process to validate the final version (26).

Nonetheless, the prototype's production cost directly correlates with its degree of fidelity. As such, the definition of a cost-effective *Design Fidelity* is essential to lower the burden of the experiments (27). The procedure aims to remove features from the product that will not heavily influence the measurements but are responsible for a significant portion of the prototype's building cost (28). As long as the prototype holds the fundamental characteristics of the solution, the gathered information is still useful to the development process (29).

To effectively close the loop on the *Solution Refinement*, there is only left to define the *Validation*. This step is responsible for building several trials to gauge the performance of the prototype and the solution regarding the designated design requirements. Highlights that the production of the prototype itself also generates valuable information regarding the building procedures and implementation feasibility of the process.

Figure 2.1 shows that the *Report* is a fundamental mechanism of this methodology because it acts as a bridge to connect all the ongoing steps. This mechanism serves to keep track of project progress and collected experience. Moreover, cataloging successes and failures of the project enable its visualization from a broader view. Therefore, simplifying the identification of features that most add value to the project (30). Likewise, a thoroughly documented solution facilities its implementation and support (31).

2.2 ENGINEERING DESIGN PROCESS

The design process of an engineering project has dramatically changed over the years. For example, optimization procedures were practically exclusive to critical applications that demanded the fittest engineering solution (32). Likewise, there was a heavy dependence on prototyping to evaluate intermediate solutions. However, due to the ever-increasing projects' complexity reconciled with technology advance, stricter regulations, and also shrinkages on schedules and budget; prototyping was superseded mainly by computational simulations (33, 17).

In that context, the advent of Computer-aided Design (CAD) in the '50s represents a breakthrough in computing and the industry. The functionalities of the earliest CAD software were reasonably limited, only been able to aid its user to draw simple sketches. However, their latest implementations have become very sophisticated, thus becoming an essential skill to the curriculum of engineers and project designer alike.

Modern CAD software features creation, parametric modeling, structural analysis, virtual assembly, dynamic simulation, technical documentation, and optimization procedures of projects on models of two and three dimensions. One of its many advantages is the systematical integration of multi-piece parts sketches, facilitating the work on complexes project. That was possible due to the creation of a set of parts' libraries and built-in methods capable of dynamically generate challenging geometries, e.g., bolts, screws, bearings, nuts, and gears.

The most significant advantage of a digital sketch against a manual one is that its duplication cost almost nothing, be that either resource or time. Thus the skillful use of a CAD software grants higher productivity and consistency to the project. This improvement also reflected in the dissemination of a standard throughout the industry, i.e., recommended practices and common components design.

However, like most new technologies, the process of broad dissemination took decades to reach the general user. This delay is due to the required hardware's high-cost and the state of Proprietary Software of the early CAD implementations. At that time, these were manly use by academics in collaborating with the space industry and large automotive assemblers, such as General Motors (34, 35). For that reason, the general public demands were not the aims of the software's functionalities — this explains why most of them were developed to assist in specific design tasks (36).

That scenario only comes to a change with the dissemination of desktops in the '70s, a period when the CAD software leaves the academical and private sector to become a product itself (37). A notable mark in the industry is the software ADAM, what makes it so unique is that 70 percent of all 3-D mechanical CAD/CAM systems' code available today traces their roots back to its source code. For this feat, its creator Patrick Hanratty is renown as the "Father of CAD/CAM" (38, 39).

The virtualization of the design process facilitates the exchange of information between the steps of the methodology. Which also enable the validation of the products on a broader range of

operational scenarios and configurations. As a direct consequence of this procedure, the amount of available information to the design process has dramatically increased, providing an unseen amount of information to the process. This breakthrough has made feasible the utilization of structural optimization on non-high profile projects, given the significant diminishment of the financial barrier (37, 19).

In the methodology, the most affected steps were those of the *Solution Refinement*, as the products' virtualization also enables the simulation of the experiments itself. Moreover, the systematization of these hand-made procedures grants faster and consistent results. However, in order to achieve quality simulations, the computational cost has escalated massively.

Therefore, an *Engineering Design Process* is an inherent optimization problem, which has become incessantly more complex and demanding over the days.

2.3 DESIGN AS OPTIMIZATION PROBLEM

There are three components in any optimization problem: (a) *design variables*, (b) *state variables*, and (c) *objective functions* (1). The *design variables* list the set of changeable parameters of the problem. They effectively correspond to the components under optimization. In contrast, *state variables* render the response of the problem, given the selected design variables and the simulation scenario itself. At last, the *objective functions* utilize the problem's behavior or characteristics to indicate the goodness of the evaluated design. Ideally, these indicators yield comparable values and are mostly numeric values. In the literature, another conventional denotation for the objective function is cost function.

Therefore, the state variables are a set of functions with the design variables as its domains, whereas the objectives are functions in which both the design variables and state variables compose their domain. Utilizing a vectorial notation in the above description yields in the following correlation:

- Design Variables: $\overrightarrow{X}$
- State Variables: $\overrightarrow{Y}(\overrightarrow{X})$
- Objectives Functions: $\overrightarrow{F}(\overrightarrow{X}, \overrightarrow{Y})$

However, this description does not fully represent the extent of the problem because it neglects the existence of parameters originated from the problem definition and the simulation environment. Although these parameters remain fixed throughout the entire process, they significantly affect the behavior of the state variables and objective functions. Thus, taking into account the existence of these implicit parameters in the previous mathematical notation results in the description below:

- Problem Parameters: $\overrightarrow{P}$
- Simulation Parameters: $\overrightarrow{S}$
- Design Variables: $\overrightarrow{X}$
- State Variables: $\overrightarrow{Y}(\overrightarrow{P}, \overrightarrow{S}, \overrightarrow{X})$
- Objectives Functions: $\overrightarrow{F}(\overrightarrow{P}, \overrightarrow{S}, \overrightarrow{X}, \overrightarrow{Y})$

Worth highlighting that, the state variables may not be injective, i.e., possess the behavior of a one-to-one function from its domain to the image. Therefore, different sets of design variables could yield similar state variables. Figure 2.2 depicts such correlation between *design variables* and *state variables*, where X_i represent a set of design variables, and Y_j their corresponding set of state variables. For this example, assumes the implicit utilization of the parameters of the problem definition and simulation scenario.

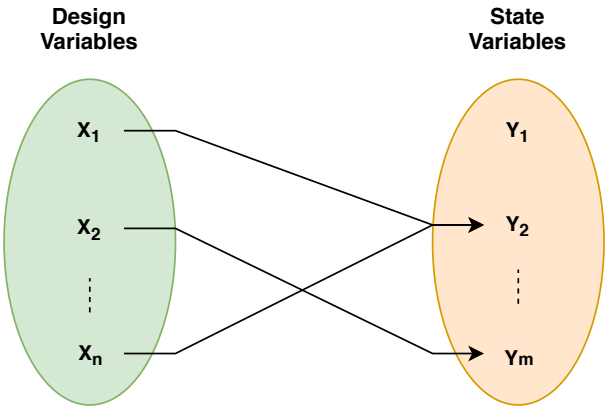


Figure 2.2 – Optimization Problem - Non-Injective Behavior

Likewise, the evaluation of the objective functions can also manifest similar behavior. However, that is a highly undesired behavior because this characteristic makes the indicator completely useless. After all, an indicator with such behavior will not be capable of differentiating solutions or gauge which one of them best fulfills the problem requirements.

The utilization of vector notation masks a characteristic of this class of problem: multi-type values (40). It is important to note that, depending on the methodology used to set up the optimization problem, a mixture of values of different types may compose its parameters and variables, e.g., integer, float, boolean. That information is crucial to the selection of the solving algorithm, as many of them does not natively support the optimization of the multi-type values.

The purpose of an optimization problem is the specification of a solution that best fulfills its prescribed requirements, given a set of constraints and restrictions (6). However, the concept of the "best" solution is relatively vague and not easily identified on all problems. A proper definition of the meaning of requirements fulfillment is necessary to handle this specific issue (1).

The idea consists of translating the optimization problem to a minimization/maximization problem. Given that its objectives commonly describe the costs, their diminishment becomes the main objective of the optimization process. For that reason, implementations in the minimization form are predominant in the literature. Whereas conflicting indicators, i.e., maximization indicators in a minimization problem, are translated into minimization indicators through the internal evaluation of their negative value.

Likewise, the same procedure applies to transform minimization indicators into its maximization form. Highlights that this procedure does not implicates in non-linear transformations in the image of these indicators. Thus, there are not solution concentrations as observed in the evaluation of non-linear transformations, e.g., the inverse of the indicators.

The mathematical formulation handles the design variables and state variables as the domain of the constrained optimization problem, and the objective function represents its image (41, 1, 6), in which the design constraints comprehend the lower and upper boundary of the vector of design variables, effectively delimiting its search space (32). Therefore, the optimization problem can be seen as follows:

$$\text{Opimization Problem} \begin{cases} \text{minimize } \overrightarrow{F}(\overrightarrow{P}, \overrightarrow{S}, \overrightarrow{X}, \overrightarrow{Y}) \\ \text{subject to} \begin{cases} \text{behavior constrains on } \overrightarrow{Y}(\overrightarrow{P}, \overrightarrow{S}, \overrightarrow{X}) \\ \text{design constrains on X} \\ \text{equilibrium constrains} \end{cases} \end{cases}$$

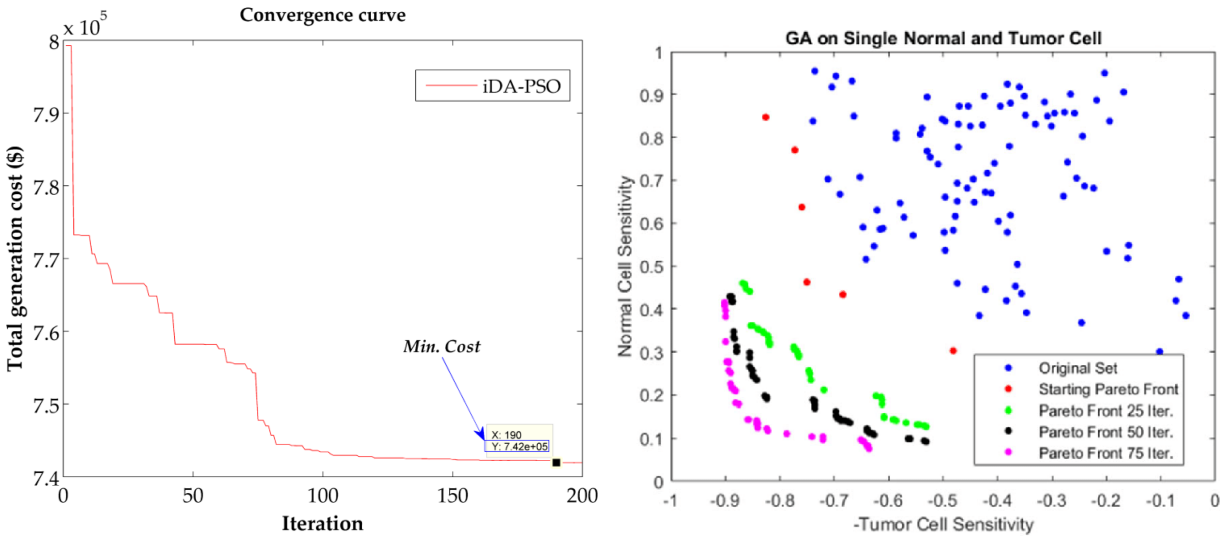
In a constrained problem, its number of degrees of freedom corresponds to the subtraction of the number of equality constraints from the number of design variables (41). A negative result indicates an overconstrained problem, i.e., its system of equations has more unknowns variables than equations.

Regarding the resolution of optimization problems, there are two schools of thoughts: (a) *single objective* and (b) *multi-objective*. The first one comprises the methods that handle problems with a single objective function, be that natively or through the grouping of multiple objective functions into a single value. While the second utilizes the concepts of *Pareto Optimality Theory* to solve a problem with conflicting objectives without the attribution of weights for them. Figure 2.3 depicts an example of the expected results of these pair of approaches.

Equation 2.1 describes the most commonly used method to transform a multi-objective into a single-objective: a weight function capable of attributing different degrees of influence to the elements coming from the same set of data, where F_i and W_i respectively describe an objective function and its influence on the final output of the weight function.

$$\overrightarrow{F}(\overrightarrow{P}, \overrightarrow{S}, \overrightarrow{X}, \overrightarrow{Y}) = \sum_{i=1}^{N} W_i F_i(\overrightarrow{P}, \overrightarrow{S}, \overrightarrow{X}, \overrightarrow{Y}) \tag{2.1}$$

However, the definition of appropriates values to these weights is quite challenging, because of



(a) Single-objective (42) (b) Multi-objective (43)

Figure 2.3 – Optimization Problem - Results

the necessity of conciliating priorities criteria with issues related to the range and scales of values of each of these indicators (44). This issue only worsens when evaluating objective functions that hold different meanings, e.g., monetary cost, weight, execution time, and sensibility. Another impasse happens while dealing with mutually exclusive objective functions, an issue that a linear transformation is not capable of handling. Nonetheless, once these issues are solved, there is a direct comprehension of the results, as indicated in Fig. 2.3 (a).

The *Pareto Optimality Theory* describes predicaments in which objective functions have equal importance and a confliction behavior in their image, i.e., improving a single indicator result in the worsening of at least one of the others objective (45, 46). In these situations, it is only possible to make a tradeoff between these objectives.

The *dominance* concept is defined as: A vector $\overrightarrow{u} = (u_1, u_2, ..., u_k)$ is said to dominate another vector $\overrightarrow{v} = (v_1, v_2, ..., v_k)$ (denoted by $\overrightarrow{u} \preceq \overrightarrow{v}$) if and only if $\overrightarrow{u}$ is partially less than $\overrightarrow{v}$, i.e. $\forall i \in \{1, ..., k\}, u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i < v_i$ (41).

Therefore, non-dominated solutions groups those that its dominance remains unchallenged in the evaluated search space. Regarding the objectives functions, *Pareto Set* and *Pareto Front* will be used to identify their respective domain and image.

In the multi-objective approach, the solver aims to identify a set of non-dominated solutions within the search space. However, its entire evaluation is generally neither feasible nor desired. In practice, most implementations only evaluate a subset of all the possible solutions. Thus, the concept of *dominance* applies exclusively to the evaluated population, not extending to the unexplored search space.

This method conventionally operates iteratively, thus in a convergent problem, there is a progressive improvement of the Pareto Front. In a minimization problem, this graphically translates

into the approximation of the Pareto Fronts to the plot's axes. Figure 2.3 (b) exemplifies this typical behavior, in which the blue and red dots compose the initial population of the problem, where the red dots mark its first Pareto Front. The remaining data pertains to the Pareto Fronts identified along the optimization process.

Therefore, unlike the previous method, a multi-objective optimization does not output a single solution (47, 41). The final call in the selection of the solution still falls in the hand of the decision-makers of the project, in which the Pareto Front acts a catalog of solutions (44, 48). Nevertheless, regardless of the selected solution, the dominance criteria guarantee that there is no better solution than them among the evaluated subset of the search space (19).

2.4 STRUCTURAL OPTIMIZATION

In this context, structural optimization comprehends the process that analyzes a structure's behavior under a set of constraints (49), e.g., external forces acting on the structure, in which its primary objective is the determination of a geometry that has higher efficient usage of materials on predefined operational conditions (50). Highlights that, this process can lead to a strict solution, i.e., only correctly operate on a specific scenario (51).

This procedure has four prerequisites: (a) a *design model*, (b) an *analysis model*, (c) a s*et of operation scenarios*, and (d) an *optimization algorithm* (49). Nowadays, the first describes a virtual representation of the designed structure, respectively, a geometric representation. This design model is usually generated by CAD software that allows the parametric manipulations of its geometric features (52).

In contrast, the analysis model describes how the external forces affect the structure, i.e., a response representation. Whereas there can be a single design model, the same can not be said to its analysis model.

The operation scenarios describe the operational conditions of the simulations, the environment that the structure is being fitted to best perform. At last, the optimization algorithm represents the methodology used to identify the design variables.

The objective of the design model and analysis model is essentially the same: represent the product. However, each of them aims to characterize a distinct aspect of this same product. The design model translates a concept product into a graphics model, whereas the analysis model enables the simulation of the real-world's structural behavior of this model. Figure 2.4 depicts the correlation between the different representations of the product.

However, these analysis models describe problems which its analytical solutions are challenging to obtain (53). Thus, through a relaxation on the requirements of the solution, numerical methods were then applied to obtain an approximated solution, of which the *Finite Element Method* is among the most classical technique, yielding simulations with both adjustable computational

Figure 2.4 – Structural Optimization - Representations

cost and trustability.

There is an extensive range of applicable object functions while dealing with structural optimization (54, 50, 55, 56, 57, 58, 59). Nonetheless, regardless of the project's purpose, most likely, it will be subject to constraints related to the below structural characteristics:

- *Compliance*: structure flexibility, typically measured in units of meters per Newton.
- *Deflection*: displacement under load.
- *Displacement*: the difference between the initial and final position of a single point of the solid.
- *Buckling*: instability that leads to structural failure.
- *Mass*: physical body's property and a measure of its resistance to acceleration under load.
- *Manufacturing Cost*: the sum of costs of all resources consumed in the process of making a product.
- *Natural Frequencies*: the frequencies in which the structure tends to oscillate in the absence of any driving or damping force.
- *Safety Factor*: the ratio of the applied load and material strength.
- *Strain*: deformation of a solid due to stress.
- *Stress*: force per unit area.
- *Temperature*: physical quantification to express the perceptions of hot and cold.

2.5 FINITE ELEMENT METHOD

The Finite Element Method (FEM) is a numerical methodology that approximates solutions for complex problems, a process that divides the problem domain into several small elements. Its core idea consists in the application of the physical laws on these elements, following the establishment of equations that models the interaction between them. The result of this process is a set of linear algebraic simultaneous equations, individually easily solvable but in a massive amount (60).

The concept of nodes is another essential component of this methodology. Graphically the nodes designate an element's boundaries and also the points of interaction between other elements (53). The definition of the coordinates of these nodes in the problem's domain is called *mesh generation*. Figure 2.5 exemplify that in the context of a structural optimization problem, in which a meshing process translates the CAD models into analysis-suitable geometries, and enables the execution of a Finite Element Analysis (FEA) (33).

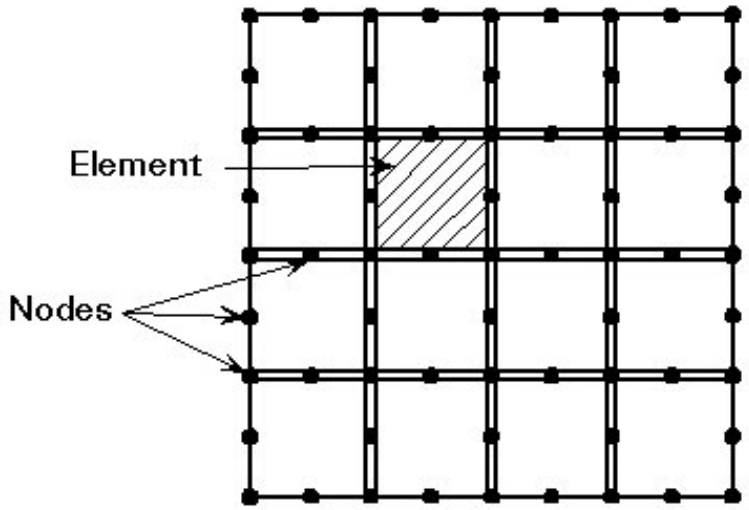


Figure 2.5 – Finite Element Method - Explanation (61)

In the literature, several algorithms implement the FEM. Nonetheless, all of them share three primitive stages: (a) input preparation, (b) formulation of the system of equations, and (c) resolution (62). The first analyzes the input structure in order to create a discrete representation, resulting in a mesh-based model. The second takes the coordinates of the nodes to define the equations that describe the interaction of this piecewise structure. The last step is the resolution of the resulting system of equations.

A mesh's number of nodes indicates its degrees of freedom, which directly correlates with the complexity of the system of equations. In that context, two aspects heavily influence resulting complexity of the meshing process: (a) the structure details, and (b) the meshing algorithm.

A *mesh* is nothing but a collection of simple elements built to resemble a single-piece structure. Thus it is reasonable to assume that a complex structure would require more elements than a simple one. Nonetheless, that becomes a fatal issue in extremely detailed structures, due to the size of its resulting mesh (63).

Meshing is an extremely complex problem and is a field of research on its own. For any given structure, there is an infinite number of mesh representations with different results and computational costs. Depending on the used algorithm, its complexity may even exceed the complexity of solving the remaining phases of the FEM (62). Thus, it is quite popular the reuse of meshes throughout simulations.

The cold truth is that a finite element mesh can never fully replicate its origin model. Moreover, it is not strange to observe the introduction of unknown behaviors in the analytical results (33), due to slight discrepancies from the CAD geometry.

The methodology FEM bypasses the condition of unsolvability of a class of complex problems up to now without a feasible analytic solution method. Controversial to its simple idea, the resulting procedure is far from trivial, being at the very least as complex as its source problem.

Therefore, its utilization in an iterative process, e.g., an optimization process, will not be without difficulties. The main expected issue is the excessive computational effort required to obtain results with reasonable quality (47, 2).

Given the high demand for information about the problem response in an optimization process, that for sure will implicate in lengthy processing time (64, 65). Likewise, due to its high hardware requirements, the complete parallelization of the process will hardly be possible without its *clusterization*. At last, due to the meshing process, the results consistency throughout the experiment becomes another issue (66).

2.6 DESIGN VARIABLES

The origin of the design variables plays a significant influence on the overall course of the optimization process (67). Two categories conventionally segregate this phenomenon: (a) CAD-based design, and (b) Finite Element based (FE-based) design. Figure 2.6 depicts the flowchart about their creation pathway.

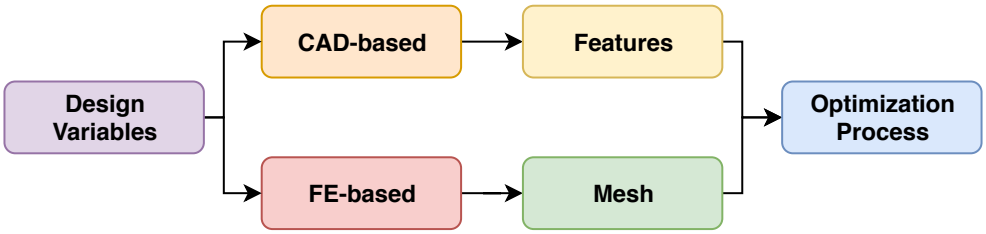


Figure 2.6 – Structural Optimization - Design Variable

In CAD-based optimization, each design variable specifies a geometric feature of the design model, i.e., an angle; a position; a width; a length; and even the amount of a given feature (68). Highlights that, due to the CAD model's parametric modeling innate functionalities, the vast majority of its features are parametrizable (69).

However, the selection process that attributes which and how many of them as design variables remains a project decision. Because as their number increases, higher is the degree of freedom and complexity of the optimization process (70, 71). That way, the alteration of a design variable issues an update on the design model itself and consequently compels the reconstruction of the analysis model - a very onerous task (54, 72, 73).

On the other hand, the design variable of an FE-based optimization pertains the properties of the fine element mesh itself, i.e., element fulfillment, nodal coordinates, and nodal connection. Nonetheless, the position of the node in the mesh indicates their function in the mesh, for they can either specify the outer shape of the model or the finite elements. In an optimization problem, the set of candidates design variables cames exclusively from the first group (68).

For that reason, this modality can achieve a significantly higher degree of freedom (67). Free-form optimization is another designation found in the literature, due to the definition of shape's boundaries without any prior explicit parameterization (69). However, it remains true that the resulting design model is inexorably a deformation of its reference design model. Hence their alteration does not demand the update of the design model but the analysis model - a very trivial task.

The core advantage of the second approach lies in the fact that updating a finite element mesh is significantly less onerous than its complete reconstruction. Because in the end, it boils down to changing values in the matrices of the finite element mesh. However, the analysis model's disassociation from the design model during the optimization process also has its particular challenges. The primary issue is its proneness to produce shapes that are not easily manufactured (32, 67, 74), leading to an increase in the production cost and problems in the scalability of the project. Likewise, this approach's search domain is vastly superior to its counterpart, conciliated with cost functions that have a higher degree of complexity.

2.7 CLASSIFICATION

In structural optimization, there are three mainstream classifications about the handling of the methodology: (a) *sizing*, (b) *shape*, and (c) *topology*. However, in order to fully differentiate them, the concepts topology and shape must be understood apriori. The formal definition of topology states that it describes invariant properties of an object upon transformation – namely, convergence, connectedness, and continuity. In the specific application, this notion pertains the way that the melding of the individual parts that composes the model, disregarding any possible transformation on its shape. In counterpart, the shape property of an object states its outer boundary, outline, or external surface.

Among the different strategies, the sizing optimization process constitutes the earliest approach researched in the topic of structural improvement (66). In here, the domain of the design model and its state variables are known a priori and remains constant throughout the entire process (50, 75). The identification fo the optimal design occurs by changing the different design variables while keeping the shape and topology of the structure untouched (76).

Initially, it is hard to perceive the difference between shape and sizing optimization. However, the distinctions lie in the fact that a shape optimization alters the boundaries of the model (50, 75). A process that typically calls for an update on the analysis model, i.e., issues the creation of a new

finite element mesh (66, 77).

In other hands, the topology optimization compromises a whole different school of thought. Here, there are two main aspects at work: (a) *connectivity*, and (b) *fulfillment*. Its core idea is to remove extraneous material from the model (55). This process significantly affects the shape of the resulting model. Achieved by altering the properties of the elements in the mesh of the analysis model by attributing solid, empty, and porous regions (78). In short, it is a strategy that effectively adds holes in the structure and operates the connectivity of the design domain (50, 76, 75). However, because it evaluates different combinations of topologies, the results are a step above the other strategies but also requires more computational efforts (77).

2.8 CONCEPTUAL SOLUTION

There are countless ways to integrate the design process, structural optimization, and FEM. Nonetheless, the following sequence of procedures reconciles different optimization approaches in order to supplement their disadvantages.

1. Concept Solution
2. Topology Optimization
3. Design Simplification
4. Parametrization
5. Shape Optimization

Regardless of its objective, the problem necessitates a concept solution that can fulfill primary requirements, though may not all of the secondary ones. The idea of this process is to remove the excess materials right at the initial phase of the design process. That way, there is a continuous improvement of the solution, which early-stage models act as a base for the development of the subsequent models (50).

The impossibility of complete automation marks its main drawback. The output solution's geometric features of a topology optimization do not conform with the conventional standard (79). Thus, a design simplification is then made necessary, followed by the selection of the design variables of the shape optimization.

Therefore, the topology optimization acts as a preprocessor of the concept structure, while the shape optimization defines the final solution through the smoothing of the jagged boundaries (80, 81, 82). Hence, this represents a procedure that initially aims to produce a rough concept, which is then fine-tuned to fulfill its requirements.

2.9 OVERVIEW

Problem solving is the core skill of an engineer, so that private organizations and even the government resort to them to solve their predicaments. Nonetheless, every problem has its peculiarities, thus even established solutions demands adjustments in order to attain its objectives. Therefore, *Design Process* and more specifically, the *Engineering Design Process*, enables even inexperienced professionals to build robust solutions efficiently.

In the context of engineering, their problems commonly possess more than one solution, each of them with different advantages and disadvantages. Nonetheless, a set of metrics could measure the goodness of these solutions, and the problem could then become a minimization/maximization problem.

Therefore, *Structural Optimization* is a procedure that implements those techniques in order to determine a structure's geometry that enables a higher efficient usage of materials on predefined operational conditions, while fulfilling a set of project restrictions and safety regulations .

In this work, multi-objective optimization techniques will be used to solve Mechanical Structural Design's problems. Nonetheless, the evaluation of the problem's cost function represents a critical issue due to its high computational cost and execution time, leading the research towards the concept of surrogates, the topic of Chapter 3.

3 SURROGATES

In an optimization context, the cost function's evaluation is crucial for its progression. That happens because there are periodic evaluations of the cost function in order to guide the search process towards the minimum/maximum global. However, often in an *Engineering Design Process*, this cost function's evaluation must be done by simulating complex mathematical systems of high computational costs. In the case of *Structural Optimization*, specific FEM software packages perform these simulations, e.g., Ansys, Abaqus, and Autodesk Simulation.

Therefore, *Surrogates* represent a set of methodologies capable of reducing the computational cost of the search process while maintaining acceptable accuracy. **The surrogate models describe functions that mimic the behavior of another function or process, in which "model" is the most common designation for these representative functions**.

There are two main reasons for its utilization: (a) handle a black-box problem or (b) a computationally strenuous problem. The first category defines problems that there is little knowledge about its internal operation, which is generally insufficient to generate a complete analytical representation. The second category describes problems that have issues regarding their execution costs. Therefore, surrogate models combine both generic representation and attractive computational simplicity (8).

The surrogates methodologies have resulted in the creation of a new branch of search algorithms: *surrogate-assisted optimization* algorithms or *model-assisted optimization* algorithms. Moreover, studies have indicated that this approach has the potential to solve some of the engineering problems' issues (7).

The principle of operation of the surrogates methodologies is quite simple, though there is a varying degree of complexity on their implementations, in which model creation methods builds a surrogate based on samples of the real function. Thus, two primitive procedures compose its entire process: (a) a *Population Sampling*, and (b) *Model Construction*.

However, worth highlights that a surrogate model is only capable of mimicking a single function. Therefore, several models are necessary for a process in which a single evaluation point output several outputs. Nonetheless, different surrogates can share the same sample (or the same input domain) but models distinct behaviors.

Therefore, the following chapters provide an in-depth overview of the internal operation of a typical surrogate. Of which, Chapter 3.1 explains selection of the stimuli inputs. Chapter 3.2 handle the update strategies of surrogates models. Chapter 3.3 details the different techniques applied to construct a model. At last, Chapter 3.4 discourse on the challenges of its utilization.

3.1 SAMPLING STRATEGIES

In the context of surrogates, there is a consolidated pattern: quality stimuli inputs are crucial to building good surrogate models (83). Moreover, that extends to all model construction techniques. Some approaches may have the capability to extract more information from its data than others, but there is a limit on that. Therefore, even simple model construction techniques achieve accurate representation once given a good set of data points (15, 84). Nonetheless, sampling the design space is not a trivial task, much less considering high-dimensional problems.

The sampling strategies are responsible for the selection of data points within the search space. In the literature, these data points could also assume the designation of *samples*, *observations*, or *training points*. However, the smaller the distance between an *evaluation point* and a *training point*, higher is the likelihood of obtaining accurate estimations. For that reason, there are two main approaches to their selection:

a) ***Domain-based Sampling Strategy***: the training points are sparsely distributed in the search space so that all evaluation points possess similar distance towards the training points. Therefore, at least theoretically, also a similar accuracy.

b) ***Response-based Sampling Strategy***: the training points are chosen according to the behavior of the source function, so the accuracy of the resulting model depends on the evaluation point's location.

The Domain-based Sampling Strategy is commonly applied with no evaluation of the cost function before the building of the surrogate. Moreover, there is no further update after the model's construction. For this characteristic, other common designations for this approach are *model-free*, *non-adaptative*, *a priori*, *one-shot*, *off-line* or *single-stage* (84).

The Response-based Sampling Strategy assumes that it is possible to identify models failing to mimic the source function's behavior (or the actual function). Thus the new training points will be strategically positioned within low-efficiency regions to enhance the efficiency of the surrogate. Similar to the other approach, there is no standard designation for this strategy, of which *model-based*, *adaptative*, *a posteriori*, *sequential*, or *online* are the most common designations (84). In here, another designation for the additional training points is *infill points* (85).

3.2 MANAGEMENT STRATEGY

The main objective of an adaptative implementation is to building surrogate models with the least amount of training points as possible while fulfilling its representation requirements (84). A practical implementation for that is the utilization of surrogate models alongside an optimization process. Where identifying the Pareto Front is far more critical than the flawless characterization

of the entire space search domain. Moreover, it may not be feasible to build a non-adaptative model or just too time-consuming.

Therefore, Management Strategy describes procedures that coordinate when and how to update the surrogate models (5). This leads to the definition of : (a) *update triggers*, and (b) *selection methods*.

The *update triggers* could either be related to the optimization process or the surrogate model itself. The first category holds triggers based in iteration-number and evaluations on the goodness of the Pareto Front, e.g., spacing, richness, diversity, or optimality (19); whereas the latter congregate those related to the accuracy of the model.

The identification of low-efficiency models occurs by analyzing the surrogate model's estimations in a set of known values of the source function (64). These test points are an independent set from the training points population. Thus, they serve exclusively to measure the performance of the surrogate model against the source function.

An error measure or correlation criteria then quantify the accuracy of the surrogate model, commonly through the identification of the root mean square error (RMSE) or the correlation coefficient (R) (86). Nonetheless, the latter provides an advantage regarding its measures interpretation. In which, the measures range from 0 to 1, where a correlation coefficient equal to 1 indicates an exceptionally accurately surrogate model. In contrast, the other measure interpretation depends on the behavior of the source function.

The drawback of this procedure is that it requires evaluations of the cost function, but at the same time, do not actively improve the surrogates, i.e., essentially wasting valuable information. Nonetheless, there is a way to achieve similar results without the definition of a set of evaluation points, thou with far less accuracy.

Given prior knowledge of the range of values of the source function, it is possible to detect anomalous behavior on the surrogate models. The principle of operation of this process lies in the fact that estimations from low-efficiency models are prone to misbehave, i.e., violate these expected ranges of values.

Otherwise, the *selection method* choose the new training points based either on (a) a design space exploration criteria or (b) a design exploitation criteria (12). At its core, both approaches enhance the surrogate model, but the first intend to improve the overall accuracy whereas the second aims to improve the most prominent design region, i.e., the regions closest to the actual Pareto Front.

In an exploitation-based approach, the most straightforward strategy is the selection of new training points among the Pareto Set. Hence, this process does not take into consideration the current training points population. Therefore, in that vicinity could exist training points capable of ensuring a higher local enhancement of the model.

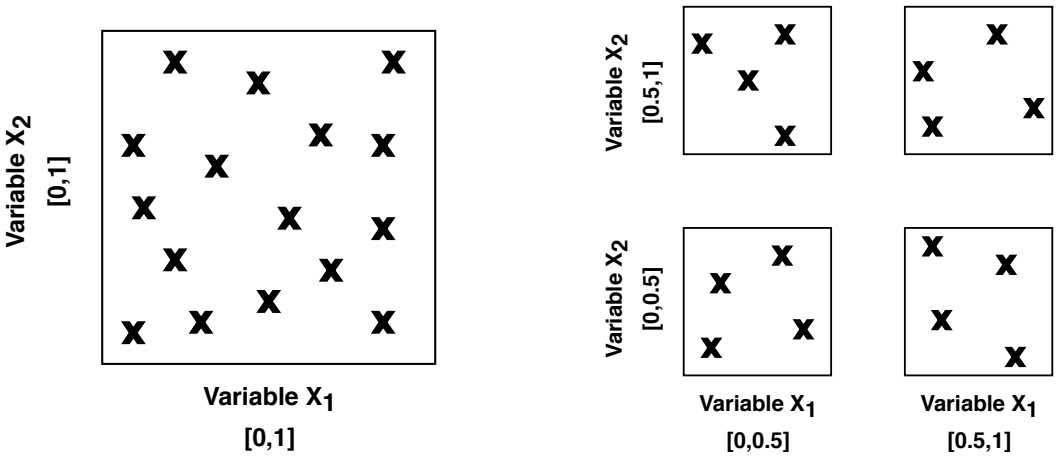
On the other hand, an exploration-based approach only takes into consideration the current training points population. It aims to settle training points in regions that most lack representation,

i.e., regions with the highest distance between training points.

3.3 THE MODEL CONSTRUCTION

Two criteria heavily influence the accuracy and computational cost of a surrogate model: *scope* and *method*. The surrogate's scope delineates the mimicked portions of the search space, and the method describes techniques that identify and reproduces the behavior of the source function.

Figure 3.1 exemplifies the two main strands of a models' scope. Where the *global approach* builds a single model for the entire domain, whereas the *local approach* builds several independent models throughout the search space.



(a) Global Model (b) Local Model

Figure 3.1 – Model Construction - Scope

From an implementation perspective, the global approach has an advantage above the local approach. In the first, the optimizations algorithms have to manage a single model. Whereas in the second, the process has to manage several models, consecutively the workload is directly proportional to their number. However, studies have shown that in problems of high dimension, those advantages cease to exist, due to the ever-increasing demands on the number of training points needed to build a global surrogate model (64).

The researches in the field of model creations have yield a considerable amount of methodologies. Nonetheless, the preeminent approaches construct their models based on criteria of interpolation, regression, projection, or a mixture of them. Hence, a classification arises from the clusterization of the techniques of a similar nature (87):

- *Data-Fit Models* (DFM)
- *Reduced-Order Models* (ROM) or Projection-Based Models (PBM)
- *Multi-Fidelity Models* (MFM)

The *Data-Fit Models* congregates methods that take a few samples of the source function to generate a surrogate model based on their regression or interpolation (84). Therefore, they do not model the physics behind its sourcing problem, i.e., their outputs are non-physics-based approximations (88). The most commonly used methods are listed below:

- Response Surface Model (RSM, Polynomial Regression)
- Gaussian Process Regression (Kriging)
- Multivariate Adaptive Regression Splines (MARS)
- Artificial Neural Networks (ANN)
- Radial Basis Functions (RBF)
- Support Vector Regressions (SVM)
- Stochastic Spectral Approximations (SSA)

In contrast, the *Reduced-Order Models* do not require data samples from the source problem but its discretized governing equations. The target of this approach are problems described in the form of large-scale systems of complex equations arising from the discretization of partial differential equations, which their large dimensionality often leads to excessive computational efforts and data storage management issues (89). Hence, this approach aims to generate low-order models that retaining the core dynamics, while neglecting irrelevant features of the large-scale system (90, 84).

The *Multi-Fidelity Models* groups methods that are capable of combining data from different levels of fidelity, i.e., the degree to which a model captures the physics of a phenomenon of interest. For this characteristic, this approach is also known as data fusion or data merging. The sourcing data came from Data-Fit Models, Reduced-Order Models, simplified simulations (equalization, resolution), and empirical methods (84).

Another approach in the rising is the simultaneous utilization of *Multiple Surrogate Models* to overcome their drawbacks. These methodologies strive to cross-validate their estimations in order to identify low-fidelity regions (91) or build an ensemble of surrogates (92).

3.3.1 Radial Basis Function

In this study, the chosen model construction method is the *Radial Basis Function* (RBF). This decision is supported by the results of several studies that have used the RBF in a similar optimization context, in which have shown that this approach provides a more accurate approximation for high-dimension problems in comparison with other *Data-Fit Models* (DFM) (93, 94, 95), such as, a) *Response Surface Model* (RSM), b) *Support Vector Regressions* (SVM), and c) *Gaussian Process Regression* (Kriging).

Likewise, the RBF is also an approximation method in which the model amplitudes are its only degrees of freedom. Thus, for positive definite RBFs, the linear systems for the amplitudes are guaranteed to be non-singular (96). Another point of consideration was the computational cost of the method, which is significantly less computationally expensive than the *Kriging*, resulting in overall faster operation (97), especially at high-dimension problems (97).

The method builds models that take into consideration the values of N known distinct points of the source function to approximates the evaluation of an arbitrary unknown point in its domain. Figure 3.2 exemplify the construction of a model with five training points.

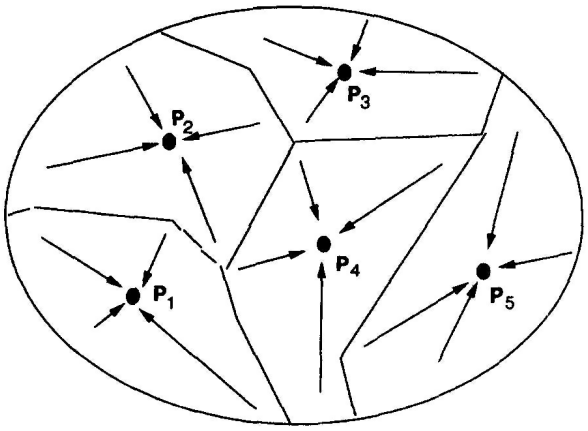


Figure 3.2 – RBF - Induction (98)

Equation 3.1 express a typical RBF implementation given N distinct training points, where $\lambda_1, ..., \lambda_N \in R$ are the weights to be determined, $\|.\|$ conventionally represent the *Euclidean norm*, ϕ is a radial function, $Xs_1, ..., Xs_N \in R^d$ are the training points, d is the dimension of the problem (number of variables of the problem), and p a complementary function.

$$f(X) \approx \tilde{f}(X) = \sum_{i=1}^{N} \lambda_i \phi(\|X - Xs_i\|) + p(X) \tag{3.1}$$

Table 3.1 list the most common radial functions, which are defined for $r \geq 0$ and γ is a positive constant. As there is no solid evidence that indicates which one of them is better than the other (8), there is no established selection criteria for radial functions either. Nonetheless, studies have shown that the cubic basis function with a polynomial tail has a promising performance (99). Moreover, this basis function does not require the definition of any additional parameter, characteristic that could induce the inclusion of behavior alien to the source function and would also require an auxiliary optimization procedure to adjust the constant parameter γ (100).

Equation 3.2 describes a RBF with a linear polynomial tail, where $a_1, ..., a_d \in R$ and $b \in R$ are the weights of the polynomial interpolation.

Name	Equation
Cubic (101)	$\phi(r) = r^3$
Thin Plate Spline (102)	$\phi(r) = r^2 ln(r)$
Gaussian (103)	$\phi(r) = e^{-(r/\gamma^2)}$
Multiquadric (104)	$\phi(r) = \sqrt{r^2 + \gamma^2}$

Table 3.1 – Radial Functions

$$f(X) \approx \tilde{f}(X) = \sum_{i=1}^{N} \lambda_i \phi(\|X - Xs_i\|) + a^T x + b \tag{3.2}$$

Equation 3.3 express the system of equations used to determine the weights λ, a and b of the Eq. 3.2.

$$\begin{pmatrix} \Phi & P \\ P^T & O_{(d+1)\times(d+1)} \end{pmatrix} \begin{pmatrix} \lambda \\ C \end{pmatrix} = \begin{pmatrix} F \\ O_{(d+1)} \end{pmatrix} \tag{3.3}$$

$$\Phi_{ij} = \|Xs_i - Xs_j\|; \quad P = \begin{pmatrix} Xs_1^T & 1 \\ Xs_2^T & 1 \\ \vdots & \vdots \\ Xs_N^T & 1 \end{pmatrix}; \quad \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{pmatrix}; \quad C = \begin{pmatrix} a_1 \\ \vdots \\ a_d \\ b \end{pmatrix}$$

$$F = \begin{pmatrix} f(Xs_1) \\ f(Xs_2) \\ \vdots \\ f(Xs_N) \end{pmatrix} \quad O_{(d+1)\times(d+1)} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}; \quad O_{(d+1)} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

The resulting system of equation is a typical linear systems $AX = B$, which its coefficient matrix in Eq. 3.3 is invertible if and only if $rank(P) = d + 1$ (96), i.e., there has to be $d + 1$ affinely independent points among the training points populations (100).

The *Singular Value Decomposition* (SVD) technique could be used to efficiently calculate these matrix inversion (105). Thus, once the coefficients λ and C are defined, the values at any position can be calculated with the Eq. 3.2. Nonetheless, it is only necessary to solve the system of equation a single time, the subsequently calls of the RBF models adopts these coefficients.

3.3.2 Examples

The model construction method's capabilities were put to test in three renown cost function in the field of optimization algorithms: a) *Three-hump camel* (camel3), b) *Ackley*, and c) *Schwefel*.

The *Three-hump camel* (see Eq. 3.4) is bi-dimensional function that describes a valley-shaped

surface, which has three local minima at its flat region, which is usually evaluated for $x_i \in [-5, 5] \forall i = 1, 2$.

$$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2 \tag{3.4}$$

The *Ackley* (see Eq. 3.5) function is widely used for testing optimization algorithms, which describes a plateau with many local minima, and a large hoke at the center, where d indicates the target dimension, and remaining parameters are keep at their default values: $a = 20$, $b = 0.2$ and $c = 2\pi$.

$$f(x) = -aexp\left(-b\sqrt{\frac{1}{2}\sum_{i=1}^{d}(x_i^2)}\right) - exp\left(\frac{1}{2}\sum_{i=1}^{d} cos(cx_i^2)\right) + a + e^1 \tag{3.5}$$

The *Schwefel* (see Eq. 3.6) is a complex N dimensional function with many local minima, which is usually evaluated for $x_i \in [-500, 500] \forall i = 1, ..., d$, where d indicates the dimension of the problem.

$$f(x) = 418.9829d - \sum_{i=1}^{d} x_i sin\left(\sqrt{|x_i|}\right) \tag{3.6}$$

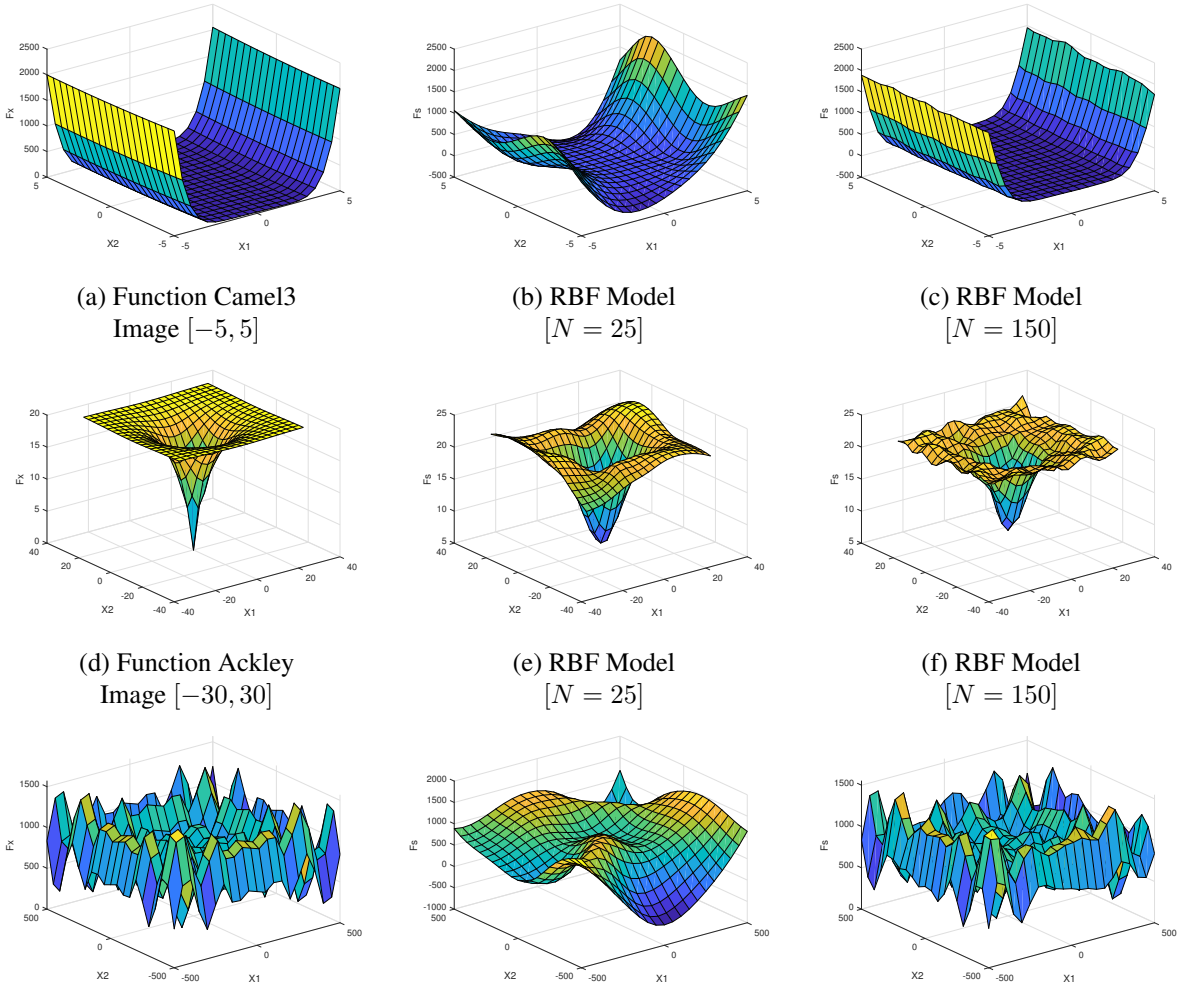
Figure 3.3 exemplify the construction of models utilizing the *RBF* technique, where the items (a), (b), and (c) holds plots the surface of function *Camel3*, RBF Model with 25 training points, and RBF Model with 150 training points respectively; the items (d), (e), and (f) holds plots the surface of function *Ackley*, RBF Model with 25 training points, and RBF Model with 150 training points respectively; the items (g), (h), and (i) holds plots the surface of function *Schwefel*, RBF Model with 25 training points, and RBF Model with 150 training points respectively.

3.4 CHALLENGES OF SURROGATE UTILIZATION

Ideally, an implementation of the surrogate methodology should reconcile both generic representations with an attractive computational simplicity (8). Nonetheless, these characteristics are natively conflicting objectives. Because building a model with reasonable accuracy of a source function with an image of high complexity requires more training points (106).

However, as the number of training increases, so does the computational complexity of the surrogate model; in which "empty space phenomenon" groups the set of issues related to modeling a source function without the right amount of training points (107).

That issue only worse while evaluating high-dimensions problems, commonly called the "Curse of Dimensionality". In here, there is a sharp increase in the number of the necessary



(a) Function Camel3 Image $[-5, 5]$ (b) RBF Model $[N = 25]$ (c) RBF Model $[N = 150]$

(d) Function Ackley Image $[-30, 30]$ (e) RBF Model $[N = 25]$ (f) RBF Model $[N = 150]$

(g) Function Schwefel Image $[-500, 500]$ (h) RBF Model $[N = 25]$ (i) RBF Model $[N = 150]$

Figure 3.3 – Examples of Surrogate Model

training points (90, 108, 85), in which the characterization of non-linear behavior becomes exceptionally inefficient for high-dimensions (106, 84). In essence, it describes a scenario where there is an exponential increase in the overall complexity (109). Hence, its computational requirements become prohibitive (110, 111, 8).

The "Curse of Dimensionality" represents the Achilles' heel of all surrogates methodologies, one that is not easily fixable. As it simultaneously affects the process of *Sampling Population* and *Model Construction* (85, 111). The issues lie in the fact that at a high-dimension, some of the presuppositions start to lose its meanings or become misleading, e.g., Euclidean distances measure (107, 112).

3.5 OVERVIEW

The utilization of surrogate models by optimization algorithms grants a significant reduction of the process overall computational cost. Nonetheless, the population sampling strategy and model building method greatly influence the technique's performance, so that it could fail to emulate the source function, or be as costly as the direct approach.

In this chapter, the principal characteristics and procedures of a surrogate methodologies implementation were thoroughly explained, detailing its different approaches and challenges. Next chapter will be devoted to discussing the interaction of such techniques in the context of an optimization algorithm.

4 METHODOLOGY

In the context of a conventional optimization algorithm, regardless of the implementation, its iterative process has unrestricted access to the cost function (2, 3). For that matter, some even perform multiple evaluations in the same iteration. Figure 4.1 depicts such an approach, where X and F are respectively the input and output of the cost function, emphasizes that the input is the *dimension* defined in the search algorithm, i.e., the set of optimized parameters. Moreover, the output is composed of its *objectives*; i.e., figuratively speaking the eyes of the said algorithm.

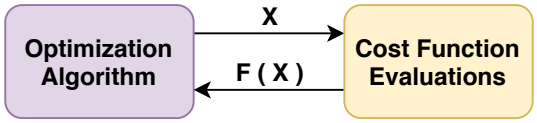


Figure 4.1 – Conventional Optimization Algorithm.

For most problems, that is usually very effective. So that the majority of the academic research concerns the development of a new algorithm or modifications of existing ones, focusing on improving the convergence to the optimal solution and overall results reproduction (77). Given that, the criteria used to rank the algorithms is their capability of reaching the optimal solution.

However, there are cases that unrestricted access to the cost function is a problem itself, as it could be quite costly. Generally, these cost criteria depend on the problem at hand. Some demand the execution of experiments, that could be both laborious and financially expensive. Alternatively, run simulations, generally computationally expensive (6). Regardless, it is a fact that either of them is time-consuming, making the conventional approach impracticable as the optimization process requires several inputs of the cost function (2, 4, 5). That way, instead of focusing on improving the optimization algorithm's search strategy, through the utilization of Surrogates' technique was possible to restrict the access to the cost functions.

Therefore, the following chapters provide an in-depth overview of the internal operation of a typical surrogate. Of which, Chapter 4.1 explains the internal operation of the optimization process. Chapter 4.2 describes how the model evaluation alters the conventional optimization algorithm. Chapter 4.3 details the handling of the cost functions. Chapter 4.4 overview some key aspects about the solutions. At last, Chapter 4.5 discourse on the challenges of population initialization.

4.1 MODEL-BASED OPTIMIZATION

A model-based optimization is an approach that the optimization process runs over models that try to mimic the behavior of the actual cost functions. In fact, during the entire procedure,

only the Surrogate can perform evaluations of the cost functions, as depicted in Fig. 4.2. Where *X* and *F* still describe the current population of the optimization algorithm and the output of the cost functions. However, in contrast to Fig. 4.1, here are introduced *Xs* and *Fs*, respectively the *training points* used to generate the models and the *output* of the models. As depicted, the Optimization Algorithm has an additional output *Iteration*, and additional input *S*. The output information regulates the internal operation of the Surrogate, and the input is how that process influences the operation of the *Optimization Algorithm*. In here, *S* represents a flag that compels the *Optimization Algorithm* to reevaluate its current iteration solutions over the models.

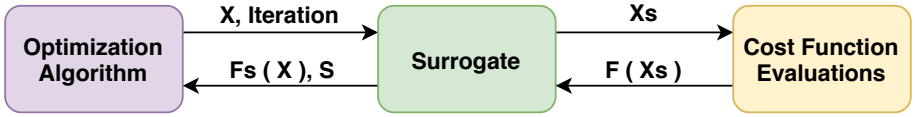


Figure 4.2 – Surrogate Approach

In the context of a surrogate-assisted optimization, there are two distinct populations. One holds the current's iteration solutions of the optimization process, and another is composed of solutions that are used to train the surrogate, i.e., the training points. Nonetheless, both populations must still follow the same search domain restrictions.

As the optimization algorithm no longer has access to the cost function, the start of optimization triggers the models' initialization. In this process, a set of solutions are picked as *training points* and then used to identify the models' parameters. The procedure used to chose those training points it is not trivial, and as such will addressed further ahead.

Figure 4.3 shows that the cost function values in the training points, *F(Xs)*, and evaluation points itself, *Xs*, are used to build the models. Highlights that, each of the cost functions F_i has a distinct model described by a set of parameters P_i. Moreover, the image also reveals that each F_i represent a model for a cost function, and all model shares the same training point set.

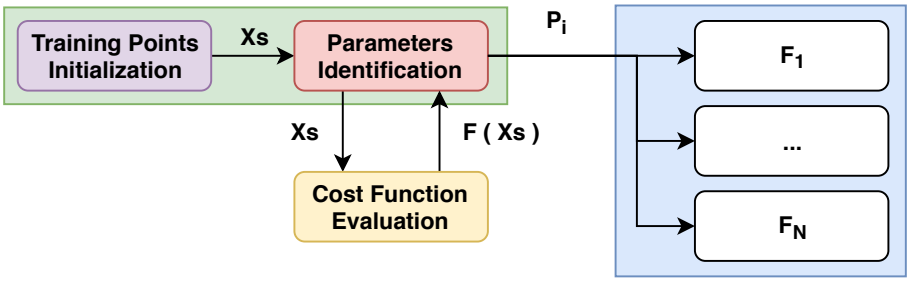


Figure 4.3 – Surrogate Initialization

The model strategy has a peculiar characteristic: there always be an error compared to the source cost functions, no mattering the number of training points in the models. Furthermore, as the surrogate population growths, the improvements in the models' accuracy start to become insignificant. Likewise, the model over-fitting is another critical issue, i.e., condition at which the

generalization performance ceases to improve and subsequently begins to decline (113).

Given that the number of training points directly correlates with the number of cost functions evaluations and the overall execution time of the optimization process, the fewer training points are used to build the model, the higher will be the overall performance increment. Therefore, defining a suitable number of training points is a challenge between accuracy and computational performance.

Based on that, arises the concept of *model generation*, in which the model conceived at *Surrogate Initialization* is the first-generation model. Because the critical population size can only be identified by actually doing the process, an arbitrarily small number of training points composes the first-generation model. Subsequent generations improve as spaces in the population becomes smaller, procedure referenced as *Update*.

Figure 4.4 details the surrogates functionality showed in Fig.4.2, describing the integration of this mechanism into the surrogate's operation, where *V* holds the information about the validation of the models, *S* represents a flag that compels the *Optimization Algorithm* to reevaluate its current iteration solutions over the models, and *P* the set of parameters that defines the cost function models.

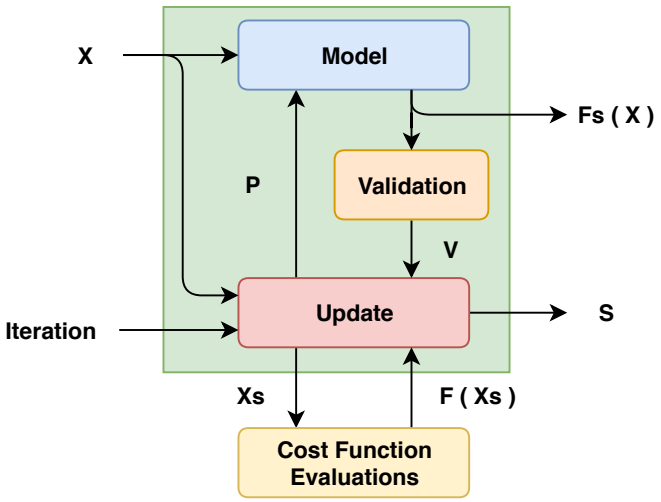


Figure 4.4 – Surrogate's Internal Operation

However, this concepts does not still explain when and how many times should the Surrogate be updated, and neither indicates the number of training points in the final population. Instead, it raises another consideration altogether: "How to judge the necessity of an update given that the process essentially runs blindly regarding the actual cost functions?"

Imparting more information into the system can help to solve this problem. This can be achieved by: (a) executing additional evaluations of the cost functions or (b) teaching insights about its behavior. The first proposal should provide better results, granted the willingness to pay its price. However, the second proposal regarding the transference of intelligent into the process could also yield results associated with a lower cost. Nonetheless, most of works use the first

approach, given the lack of knowledge of the cost function in a black-box problem (4).

In this work none additional cost function evaluations were used to identify the necessity of surrogate update, even taking in considering the possibility of using them as additional training points for the next generation model. That decision is based in the fact that strategically chosen training points are better capable to improve the surrogate model in a global scale.

Initially, in order to execute the second proposal, it was taken into consideration the behavior of the models regarding its number of training points. As expected, smaller populations cannot reproduce the behavior of the cost function as well as a bigger one. As such, models build with fewer training points are more prone to be erroneous than one with a larger population.

Therefore, our system schedules the execution of several periodic updates for the model, triggered by *Optimization Algorithm*'s iteration number. Figure 4.5 reveals that our schedule was built in a way that most of the updates happen at the beginning of the iterative process, then consecutively growths sparser given time.

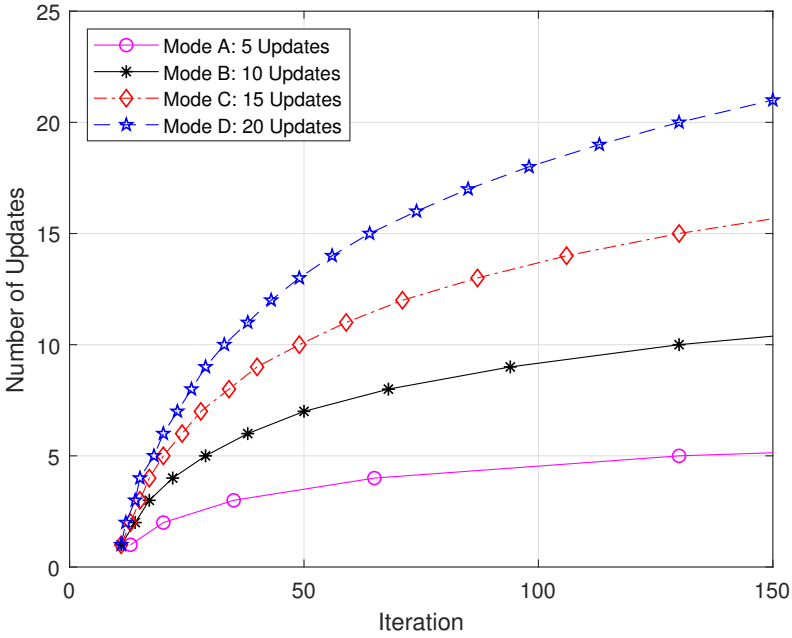


Figure 4.5 – Sequential Updates

Figure 4.5 depicts three suggestion of schedules for a 150-iteration optimization process. All of them share two characteristics, the majority of the updates happen in a time-frame of 50 iterations, and there is 50-iteration padding after their last update.

The updates' concentration at the beginning, as well as the padding at the tail, are per design and represent an updating model strategy based on the experience acquired during the development of this research. Equations 4.1, 4.2, 4.3, and 4.4 details the mathematical equations that formulates the strategy's modus operandi: a) *Mode A*, b) *Mode B*, c) *Mode C*, and d) *Mode D*.

$$\textbf{Mode A: } Iteration = round((10 + Update) \times (e^{\frac{Update^{1.5}}{7.75}})^{1.5}) \tag{4.1}$$

$$\textbf{Mode B: } Iteration = round((10 + Update) \times (e^{\frac{Update^{1.5}}{25.30}})^{1.5}) \tag{4.2}$$

$$\textbf{Mode C: } Iteration = round((10 + Update) \times (e^{\frac{Update^{1.5}}{52.70}})^{1.5}) \tag{4.3}$$

$$\textbf{Mode D: } Iteration = round((10 + Update) \times (e^{\frac{Update^{1.5}}{91.20}})^{1.5}) \tag{4.4}$$

Figure 4.6 depicts the stages that encompasses the model update strategy: a) *Initialization*, b) *Global Improvement*, c) *Local Improvement*, and d) *Refinement*.

At the first stage, *Initialization* builds the first generation model with a predefined number of training points, conventionally a relatively small population. The second stage describes a scenario in which the model update strategy uses the *Global Improvement* approach, during this stage the selection training points aims to improve the models' representability of the cost function in the entire domain. At the third stage, the model update strategy adopts the *Local Improvement* approach. In here, the selection of training points is biased to introduce them in critical locations of the search space, because prioritizing regions closer to the non-dominated solutions yields better results than improving the overall model further (47). At last, at the fourth stage, *Refinement*, the remaining iterations allows the search process to refine the results over the most complex model.

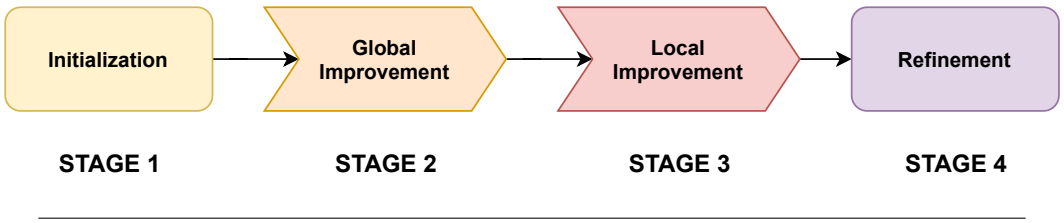


Figure 4.6 – Model Update Strategy

Thus, the main difference between the improvement stages is that the first seeks to improve the model in the entire domain, while the second limits to improve the region close to the *Pareto Set*, i.e., the non-dominated solutions of the current iteration. Nonetheless, at the beginning of the optimization process, the model update strategy' *Global Improvement* approach aims to ensure a minimum representation throughout the space search domain.

However, this strategy is only useful in the long run, as it does not take into consideration punctual errors in the models. This problem was solved by analyzing the behavior of the cost function itself. As for most cases, it is generally possible to estimate its boundaries, i.e., maximum and minimum values.

That way, once a boundary violation on any the models occurs, an update on the models is

them made necessary, a procedure indicated as *Validation* at Fig. 4.4. Moreover, since the cost functions are a byproduct of experiment simulations, acquiring data to update a single model also enables the update of all the other models without any additional cost. However, the non-occurrence of violations does not guarantee that the models are simulating the cost functions well either.

Figure 4.7 detail the internal operation of the entire update system, depicted in Fig 4.4.

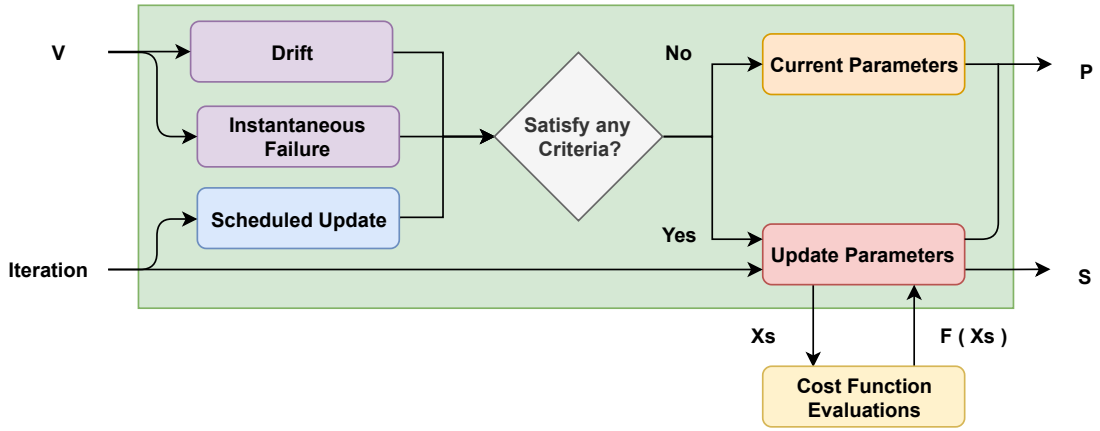


Figure 4.7 – Update's Internal Operation

Figure 4.7 shows that three criteria could trigger an update, of which, two are regarding the number of violations, and the current iteration of the *Optimization Algorithm*. In this work the violation-triggered updates are indented to handle two distinct problems: a) *Drift* and b) *Instantaneous Failure*. The first occurs when a few violations happen sequentially, an issue that could lead to a biased search. The second happens when there are just too many erroneous solutions in the population to let it proceed unattended.

The fulfillment of any criteria triggers the procedure for updating the models, indicated as *Update Parameters* in Fig. 4.7 and further explained in Fig. 4.8.

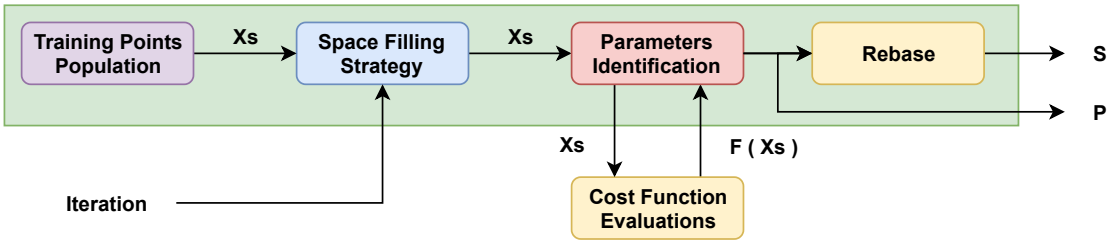


Figure 4.8 – Update Parameters' Internal Operation

The distance between the training points and the evaluation point's dimension ponders the models' output value, in which closer points have a higher weight. As such, during the updates, as indicated in Fig. 4.8, the *Space-filling Strategy* inserts new training points in the models to effectively shorten the distance towards the evaluation population.

Similarly to the population initialization, the procedure that identifies those additional training points is not trivial as their position must be chosen in a way that best complement the current training points population, in which the theory behind the procedure will be fully explaining at Chapter 5. Nevertheless, the general idea is that the improved population of training points has better coverage of the space-search domain.

However, the concept of a model generation makes tracking the convergence of the optimization in real-time problematic. That is because objective-based metrics come from distinct model generations, in which the models of the first generation hold lower trustworthy. For the same reason, previously discard solutions could be part of the *Pareto Front*.

Figure 4.9 depicts the internal operation of the *Rebase*, first apparition at Fig. 4.8. This process is responsible for refreshing the optimization algorithm's objective-based metrics once occurs an update, which requires archiving the iteration's population throughout the experiment. A byproduct of this process is the information regarding the *Pareto Set* and *Pareto Front* from all previous iterations. Thus, as an attempt to recovery possible discarded good solutions, they are then added as candidate solutions in the next iteration of the *Optimization Algorithm*, indicated as *S* in Fig. 4.9.

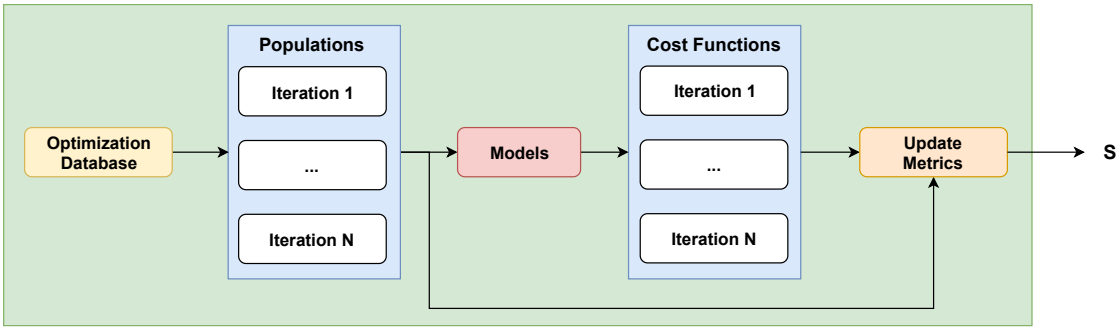


Figure 4.9 – Rebase' Internal Operation

Even then, *Cost Function Evaluation* still cannot promptly access the actual cost function, because before any evaluation, a system first checks their existence in a solution database. The decision of implementing this system is due to the possibility of the duplicity of solutions in the same population, area re-visitation, and experiment's repetition. Thus, this mechanism could save a considerable amount of cost function evaluations as well as speeding up the execution of the overall optimization, given the same experiment scenario. Figure 4.10 depicts the implemented system, where Z represents an arbitrary solution amidst the search space, and the implicit existence of *Solution Database*.

Once the optimization process reaches its end, i.e., finishes to executes its 150 iterations, the process it is not over. In view that *Optimization Algorithm*'s output is composed of approximated data, there may be possible mismatches with the actual problem. Therefore, the output's *Pareto Set* feeds the real cost function to generate the official *Pareto Front* of the entire process.

Due to the procedure used to identify the *Pareto Front*, it would not be strange that the official

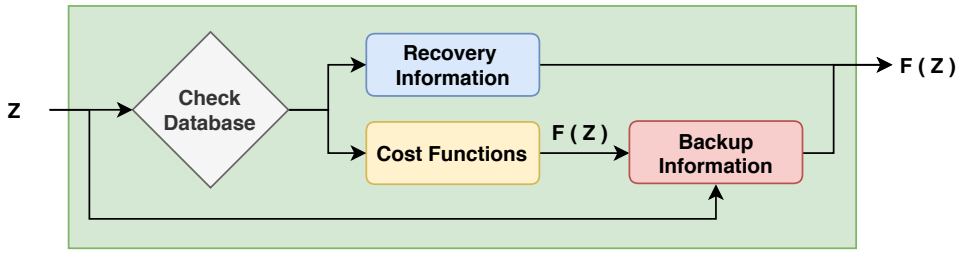


Figure 4.10 – Cost Function Evaluations' Internal Operation

Pareto Front hold fewer solutions than its model-based counterparty. This population reduction effect happens because the optimization process identifies a non-dominant *Pareto Set* from the surrogate models.

However, whatsoever accurately are these models, they are still an approximation. Therefore, some of these solutions may not fulfill the criteria of non-dominance on the image of the real cost function (114). The difference in the size of the solutions' populations is a metric of performance of this model-based optimization process as it measures the fitness of the models in front of the actual cost functions.

So with this ends the explanation of this implementation of model-based optimization. That so far is regulated by the following parameters:

- Iteration number
- Population size
- Optimization algorithm parameters
- Surrogate initial population size
- Surrogate accumulative validation threshold
- Surrogate moving average validation threshold
- Surrogate population increment on updates

4.2 OPTIMIZATION ALGORITHM INTEGRATION

In this work, the Multi-Objective Differential Evolution (MODE) was the base optimization algorithm to the implementation of its model-based version. Besides adding the Surrogate related modules and redirecting the cost functions calls to the models, the optimization remains practically unchanged. Figure 4.11 and 4.12 holds the flowchart of the source algorithm and its model-based version.

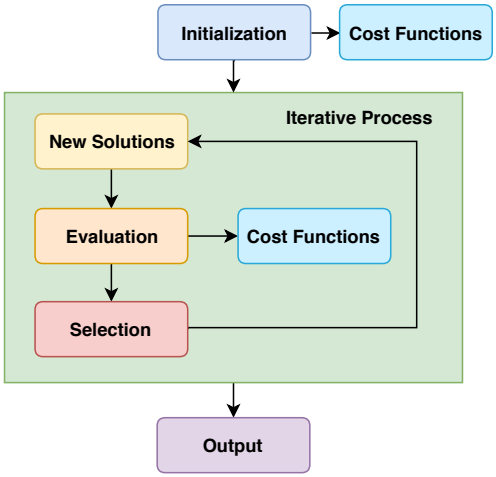


Figure 4.11 – Multi-Objective Differential Evolution

Figure 4.11 depicts the working principles of the four procedures of algorithm MODE: *Initialization*, *Generation*, *Evaluation*, and *Selection*. In which, the initialization procedure chooses points from the search space domain as the initial population of solutions. The generation procedure creates new solutions via a process of *mutation* and *crossover* of the existing solutions. Subsequently, the evaluation procedure handles the cost function calls of the set of new solutions. At last, at the *selection* procedure, the previous generation solution and the new solutions are combined as one population of solutions. Following the current iteration's population is chosen from this combined population. That way, the optimization process' result is the algorithm's last iteration solution population.

Figure 4.12 describes the model-based version of the algorithm MODE. This implementation follows the same working principle as its source algorithm. However, there is a redirection of the cost functions calls to the model, likewise the inclusion of the model-based related procedures.

Another notable difference is that after a model's update, the previous iterations solutions participate in the selection procedure of the next iteration. This mechanism attempts to recover previously misjudge Pareto Set's solutions due to models limitations. In this implementation, the last iteration solution is not the algorithm's results, but a set known as *Candidate Solutions*. The results of the optimization process are the Pareto Set of these *Candidate Solutions* evaluated utilizing the actual cost functions (instead of the model).

Regardless of the implementations, the algorithm MODE requires the configurations of the following parameters:

- Population size
- Scaling factor
- Crossover probability

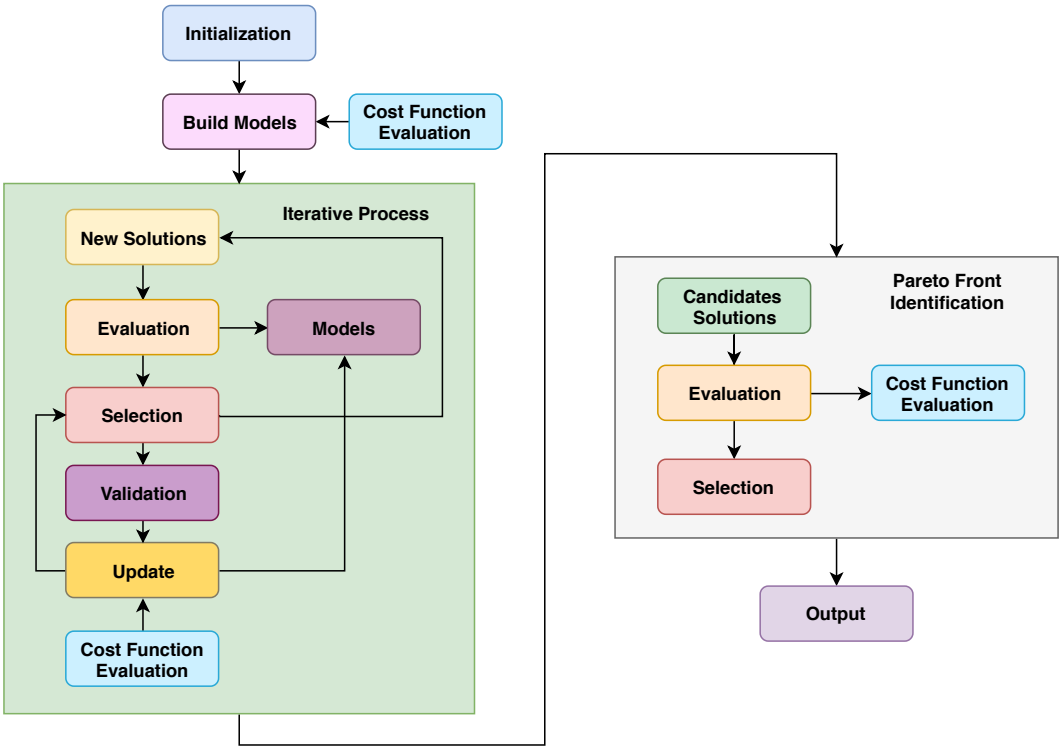


Figure 4.12 – Surrogate-assisted Multi-Objective Differential Evolution

4.3 COST FUNCTION

A core principle of the surrogate methodology is the extraction of as much information as possible from its available data, here characterized by the outputs of the cost function. However, a multi-objective optimization process demands that its criteria have a conflicting behavior, i.e., the enhancement of one deteriorates at least one of the others.

Still, the problems benefited from a model-based procedure generally outputs large amounts of information on each execution. Conventionally while aiming to fulfill the project requirements, a few of them would be chosen as cost functions while the remaining would be left unattended. Be that because they do not comply with the conflict criteria or distinguished as of less importance in the global context.

As previously explained, there are independent models for each of those cost function. While they share the same parameters inputs, their behavior differs from one to another. The same applies to the fitness of these models, as some of them are more easily mimicked then others. However, an extensive set of training points yields a more reliable model.

The comparison between the model's value and the actual cost function accurately assess the model trustworthiness. However, estimating an individual solution's trustworthy only requires the knowledge of its position in the population of training points. As the closest, an evaluation point is to a training point, higher it is his trustworthy. Of course, this is only valid if the image of this cost

function has some degree of smoothness, i.e., it must be continuous and without sharp derivatives. So while respecting those restrictions, the smaller distance towards the model's population can be used as a metric of individual solution's trustworthy.

A model-based optimization has access to three groups of information: the cost functions, the simulations' discarded data, and the individual solution trustworthy. However, only uses the first group as gauges to identify the solutions that compose the Pareto Set, i.e., its best known non-dominated solutions.

Ideally, the utilization of all gathered information would be the optimal scenario. The trustworthy measure and the simulations' discarded data acts as cost function. Moreover, the second could assist the *Validation System* in the identification of modeling failure. Nonetheless, any of those alterations changes how the optimization algorithm operates typically.

The utilization of the trustworthy measure as a cost function leads to a biased search towards high-density regions of the training population, because a uniform distribution of training points is generally not observed. Nonetheless, enforcing trustworthy as a cost function filter ineligible solutions from the Pareto Set.

Besides, once all models share the training points population, a cost function that possesses a high sensitivity to its inputs could facilitate the identification of low-fidelity regions in the models or complex regions amidst the search space. However, feeding the additional data do the *Validation System* is the same as affirming that there is a behavior correlation between the different cost functions.

A model-based optimization is a procedure that simultaneously tries to identify non-dominated solutions from an ample search space while enhances the models that serves as its guide. Essentially, a resource-constrained procedure that learns on-the-fly. Therefore, the more strict are the constraints, higher should be the relaxation impose in the methodology's procedures.

4.4 OPTIMIZATION POPULATION AND TRAINING POINTS

The performance of a search algorithm directly correlates with its population size and initialization. The first reflects the amount of available information to the optimization algorithm on each iteration, while the second helps to identify possibles regions of interest. Overall, a more significant population tend to yield better results because it provides a more in-depth search. Likewise, a balanced initialization generally speeds up the convergence to the optimal solution's region (115). While dealing with time-consuming problems, those are very looked after characteristics, as they could significantly diminish the number of iterations.

On the conventional approach, increasing the population size of a search algorithm yields more evaluations of the cost function. However, this concept loses its meaning in a model-based optimization, due to the comparatively low computational cost provided by the utilization of

models. So that depending on the complexity of the problem, it is feasible the utilization of an unseen optimization population size. Nonetheless, studies have shown that excessively large population are also not helpful (116, 117).

However, the same strategy does not apply to the training points because a quality population is imperative to build an accurate model. Therefore, the best scenario occurs when the least amount of solutions composes the models' training points. Thus each of them must be essential to its constitution.

4.4.1 Search Space Comprehension

In this context, the search space comprehension becomes a critical concept while dealing with surrogate-assisted optimization, because utilizing a single model to mimic a function's behavior at vast search space is exceedingly harder than in a bounded region.

Nonetheless, deploying a restricted region requires the understanding of the concepts of *precision* and *boundary*. The first is responsible for making the number of possible solutions finite, while the second is responsible for diminishing the number of possibilities.

The concept *precision* indicates the number of digits in the right of the decimal point of values of dimensions, i.e., optimized parameters, which in turn may have different precision. The precision number can be an external project restriction, i.e., machinery limitation or design fitting, or a limitation of the application, i.e., insensibility of the cost function. In contrast, the concept of *boundary* is an innate characteristic of an optimization algorithm, because it defines the acceptable range of values for the dimensions.

Therefore, the concept *precision* makes the number of possible configurations finite, whereas the concept of *boundary* restricts the number of possible combinations. That way, it is highly recommended the utilization of a well-fitted boundary while dealing with the surrogate-assisted optimization process.

Conventionally, the problem's boundary is arbitrarily chosen, principally paying attention in the individual requirements of dimensions than in its repercussion in the overall number of possible configurations.

The effects of this negligence are commonly imperceptible in the direct approach, as the algorithm has unrestricted access to the cost function. However, in a surrogate-assisted optimization, a vast boundary implies that each training point is responsible for a higher number of neighbor solutions, which significantly deteriorates the representation of the behavior cost function.

Figure 4.13 depicts the number of possible solutions for a four-dimension problem, where throughout the experiment, the lower boundary and precision of the dimensions respectively equal to zero and one. Nonetheless, the upper boundary ranges from one up to three according to the simulation scenarios: A, B, and C.

This experiment exemplifies the growth of the number of possible configurations due to the

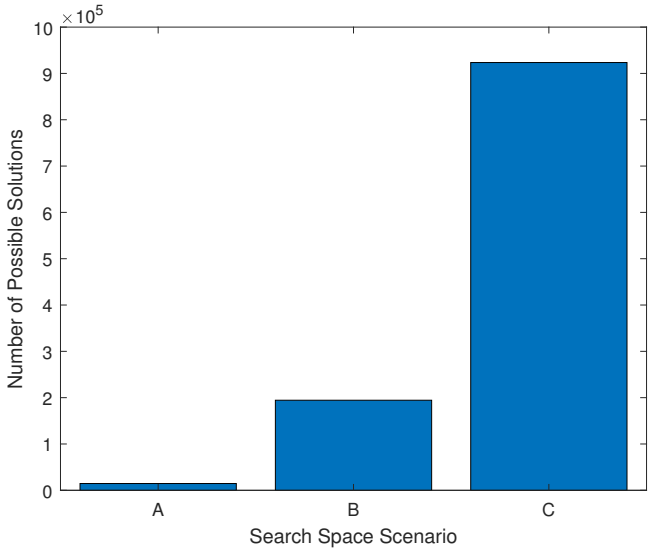


Figure 4.13 – Number of Possible Solutions

enlarging of the search space, which a small boundary's increment results in a massive growth on the number possible solutions, given higher dimensions and precision numbers this effect is even more pronounced.

Altogether, without the definition of a well-fitted search domain, the models' representability becomes questionable. Another point of consideration is the applicability of the database of solutions because the possibility of a solution's reoccurrence would be almost null at an incredibly vast search space.

4.5 INITIALIZATION

4.5.1 Generation Techniques

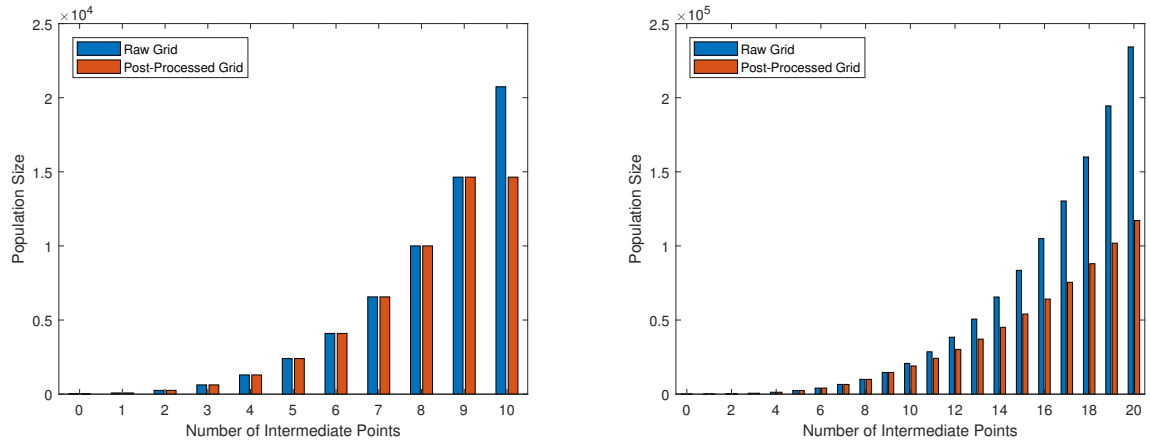
The most straightforward initialization would be utilizing a grid of values that sparsely cover the search domain. Among the several possible implementations, this work adopts the technique that inserts intermediary points between the upper and lower boundary of the search space.

Nonetheless, this procedure could generate intermediaries points that do not attend the precision criteria. There are two ways to deal with this problem. The first method consists of discovering the appropriated number of intermediary points for each dimension, then generating the grid values with that information. While the second method adds the same amount of intermediary points throughout all dimensions while creating the grid of values, but subsequently has to deal with the precision criteria.

The first solution corresponds to a pre-processing approach that ensures the equal spacing of the points regarding the dimensions restrictions and avoids reworks, while the second solution

corresponds to a post-processing approach that enforces the precision criteria over the generated grid, leading to uneven spacing and duplication of the points. Regardless of the solution, the size of the grid's population has an exponential growth that is indirectly regulated by the number of intermediary points.

Figure 4.14 depicts the growth of the grid's population due to the increment of the number of intermediate points between the boundary domain of a four-dimension problem with a fixed precision of one digit. The grids were built with a fixed amount of intermediate points, and later post-processed to deal with problems related to its precision. In this way, *Raw Grid* labels the source grid, and *Post-Processing Grid* labels the processed grid. Moreover, Scenario A's domain ranges from zero to one for all dimensions while Scenario B accommodates the results pertinent to a problem with varying upper boundary, also initializing in zero, but ending respectively at one, three, five, and seven.



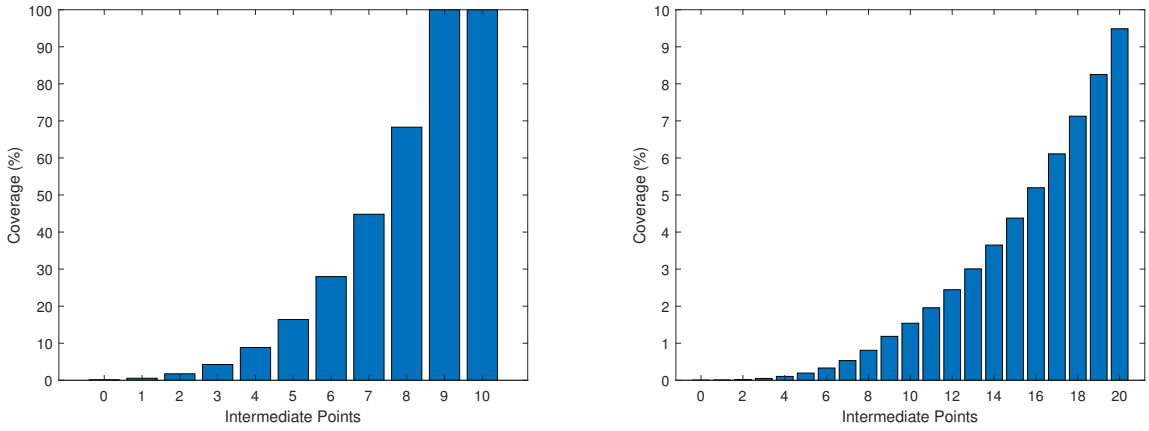
(a) Scenario A

(b) Scenario B

Figure 4.14 – Grid's Population Size Growth

Figure 4.14 demonstrates that Scenario B has to handle for more precision related issues than Scenario A. This happens because the boundary of the later has excellent compatibility with the given precision. Furthermore, its post-processed population only began to differ from its source population at the ninth grid, when all possible acceptable solutions already belonged to the grid, effectively marking the exhaustion of the domain. However, the population of Scenario B still growths even after doubling the number of intermediate points. Figure 4.15 allows an more easy observation of this characteristic.

Figure 4.15 demonstrates the correlation between the domain boundary and the capacity of the grid to cover the entire domain. Those charts illustrate that to achieve the full exhaustion of a copious domain, a higher number of intermediate points is necessary. Thereby reiterates once more the importance of a well-fitted boundary. Nevertheless, the core idea of this procedure is the utilization of a minimal subset of the population to represent the entire group. However, said proposed solution has an exponential population growth given a small increment in the number of intermediates points, an undesired characteristic.



(a) Scenario A (b) Scenario B

Figure 4.15 – Grid's Coverage Growth

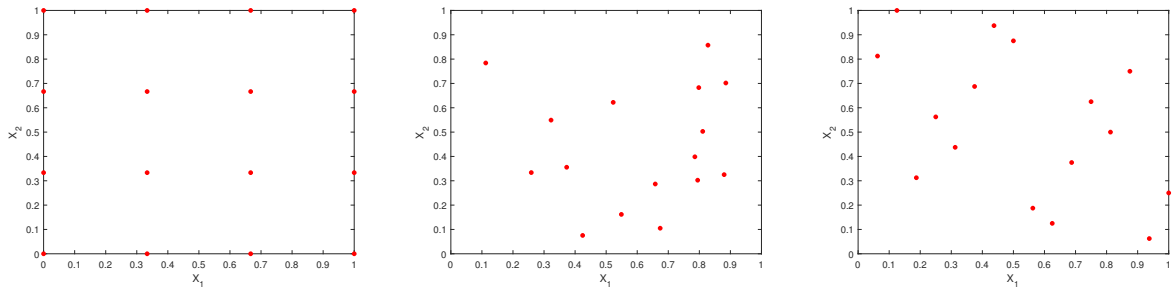
Ideally, the optimum scenario would be a technique that could generate a subset population within the search space domain regardless of its boundary, moreover the size of this subset be a control parameter of this process. This problem can be worked in two stages: distribution and scaling. Initially, occurs the distribution of a set of N elements in a simplified domain. Later, this distribution is then re-scaled to the actual search space domain.

The most straight forward way to generate those distributions would be utilizing a randomizer (RAND). However, such an approach has a fatal issue: there is no correlation between the members of the same population, i.e., the procedure chooses each new member without taking consideration of the existing members. This characteristic leads to non-homogeneous distributions, i.e., there will be regions with a high population density and others without a single member. Likewise, comparisons between subsets yield similar results: there are some outstanding distributions, and others not so much.

However, the problem of selecting a subset from a large population is not new. As such, several methodologies have already been proposed and extensively studied. One of them is a method called *random Symmetric Latin Hypercube Design* (SLHD) (118). In which, distributions created with this method has three beneficial characteristics: *symmetry*, *entropy*, and *intersite distance*. Its symmetry assures that there is a reflection plane within the distribution, i.e., it is only necessary to position half the population. The second ensures a monotonous behavior regarding the entropy of the generated subsets. The last characteristic is related to its space-filling proprieties, wherein minimize problems related to population density in specific regions of the domain.

Should be noted that both methods' inputs are the number of elements of the population and the dimension of the problem, and its output is a distribution of values ranging from zero to one. Figure 4.16 exemplifies the behavior of the three initialization methods in a 2-dimension problem.

In order not to privilege any method, all three distributions contained in Fig. 4.16 contains sixteen members, the population size of a Grid generated with two intermediate points for the



(a) Grid (b) RAND (c) SLHD

Figure 4.16 – Initialization Methods

specified dimension and precision. Figure 4.16 (a) holds the Grid's distribution. In here, the points are equally spaced, however densely concentrated in the boundary limits. This characteristic tends to dimish once the number of intermediate points increases. Nonetheless, from a surrogate modeling perspective, that behavior is very undesired for the distribution of the training points because the points located at the border loses at least haft of its coverage region. A solution to this issue would be down-scaling the distribution into a smaller domain, e.g., values ranging from 0.2 to 0.8.

Figure 4.16 (b) exemplifies the homogeneity's issues of a typical RAND's distribution. Where it is possible to identify two regions of points concentrations divided by an imaginary line connecting the points (0.3, 0) and (0.8, 1). In which the region below this diagonal houses eleven members of the entire population. Moreover, applying the same principle in the upper region outputs a region composed of a single member and another containing the four remaining members.

Similarly, the analyze of the SLHD's distribution, Fig. 4.16 (c), outputs four distinct regions housing respectively two, six, six, and two members. Due to its symmetrical behavior, each group has an equivalent with the same amount of members. However, unlike previously distribution, here the density' discrepancies are less pronounced.

4.5.2 Distinction Criteria

Initially, the distinction of superiority between distributions appears like an easy task for humans. However, that is only true while dealing with problems with graphic representation, up to three dimensions, or problems small populations. Therefore, the formulation of a superiority criterion is necessary to distinguish distributions, preferably one that translates that information into a single comparable metric.

Formally speaking, diversity and uniformity characterize a population's distribution. Although these metrics are related to one another, one does not implies the other. In other words, the presence of satisfactory diversity does not inevitably translate into superior uniformity and vice versa. However, for most cases, a superior uniformity yields consistent dissimilarity between neighbors, whereas a satisfactory diversity holds the highest amount of information (112).

Therefore, unlike the grid's population that heavily prioritize uniformity of its population, the best distribution will be the one that posses the highest diversity.

The distance between the points plays a crucial part in the evaluation of the distribution's dissimilarity. The Euclidean distance (L2 norm) is a commonly used measure for gauging the distance between two points. The leading cause for its popularity is the fact that humans are three-dimension creatures, a dimension where this measure works just fine. However, this distance metric is not well suited for measuring distance on higher dimensions (119, 107, 120, 112). Nonetheless, studies have shown that Lp-norm-based measures are promising alternatives to gauge distances at higher dimensions (121, 122).

Thus among the several methodologies available, two metrics were chosen to represent either approach. In which, the metric *Diversity* (123) ponderates the scale search space and the mean of Euclidean distance between the members of the population, whereas the metric *Pure Diversity* (112) measures the diversity of the whole population by summing the dissimilarity of each point while taking in consideration Lp-norm-based distances.

Figure 4.17 and 4.18 depicts an experiment designed to put into practice the idea of population distinction, where one-thousand distributions were generated utilizing the RAND and SLHD methods for two specific problems: two-dimension with 16 members population, and twenty-dimension with 160 members population.

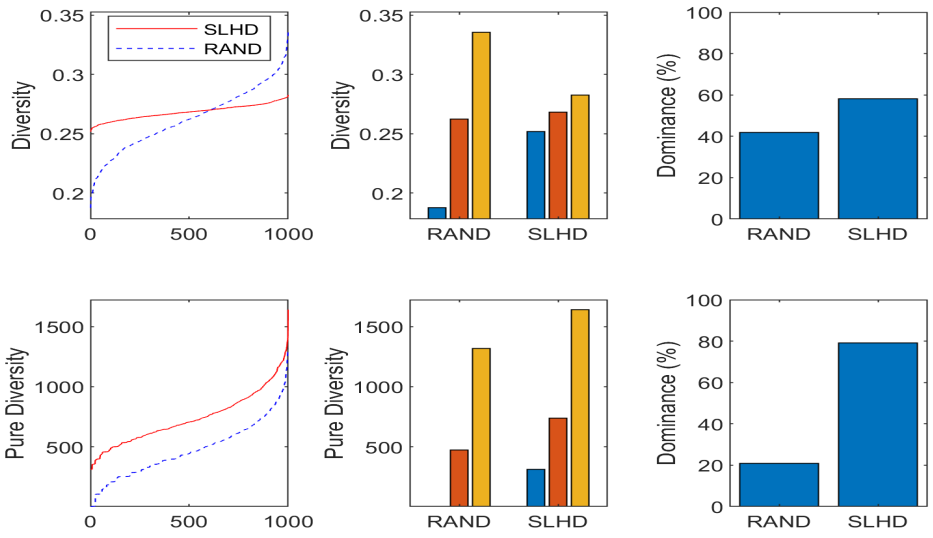


Figure 4.17 – Diversity of a Distribution - 2D

On these experiments, two metrics gauge the diversity of those distributions, respectively Diversity, and Pure Diversity. The graphics aim to compare the diversity of the populations, achieve through dominance criteria and the observation of general behavior of the methods. The first evaluates if given two populations generated by either method is capable of dominating the other, i.e., yield greater diversity. The latter identifies the minimum, mean, and maximum values during the experiment, following the diversity measures are then sorted and plotted.

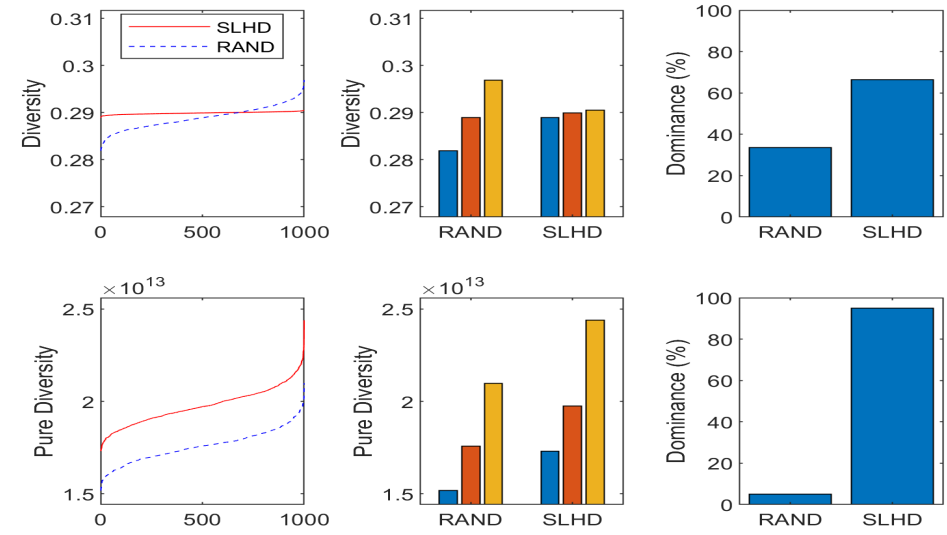


Figure 4.18 – Diversity of a Distribution - 20D

Throughout the experiment of both scenarios, the diversity measure by the metric Diversity of the SLHD's distributions has low variance, whereas the RAND's distributions possess a high variance. Regarding the dominance criterion, the results at a lower dimension are not as conclusive as at higher dimension. However, Fig. 4.18 shows that the method's discerning ability decays as the dimension increases on distributions of similar properties, e.g., SLHD's distributions. Nonetheless, the result indicates that SLHD's distributions provide consistent performance while maintaining a satisfactory distribution of the population.

Regarding the metric Pure Diversity, this experiment reveals its critical issue: lack of scale in the results. Figure 4.18 shows that at higher dimensions, this value grows to unprecedented levels, but that is not an atypical behavior. The metric builds tree-branches starting from each point then sum their length to measure the population dissimilarity. However, due to the spatial distortions caused by the *dimension number* and the *Lp-norm*, these lengths become considerably high. During the experiment, the metric output similar measures pattern regardless of the dimension of the population, in which the SLHD method best the RAND method even at higher dimensions.

4.6 OVERVIEW

Several engineering problems demands the execution of simulations of high complexity, therefore, high computational requirements and lengthy execution times. That only become worse at an optimization procedure, which requires the execution many of such simulations.

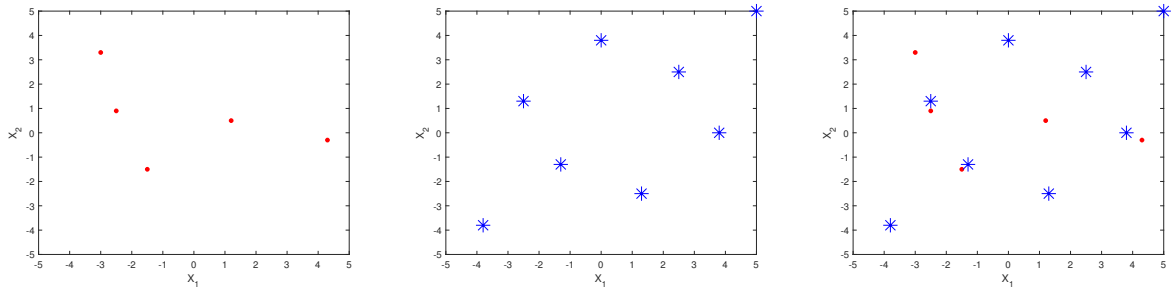
Thus, the use of surrogate modeling techniques has become indispensable, due to their cheap evaluation cost reconciled with a generic representation of the problems, characteristics very desired in at an optimization task.

This chapter developed a new surrogate-assisted optimization algorithm, which it is capable of actively identifying update triggers based on violations on the cost functions' range of values. Moreover, the algorithm implements a mechanism that periodically try to recuperate possible non-dominated solutions that were previous discarded due to model's deficient. Next chapter will be devoted to discussing the space-filling algorithms developed in this work.

5 SPACE-FILLING ALGORITHMS

The study of space-filling algorithms arises from the need to improve the model's population without discarding the predefined training points. That happens, because creating a population with more training points would offshoot in the evaluation of the cost function on its entire population. Thus, this process results in far more cost function calls than just processing a set of complementary training points.

The simplest solution to the space-filling problem would be stacking a new population of training points on top of the existing population. However, this process does not take into consideration the position of either population in the search space. In other words, there is the possibility of adding new training points on highly populated regions, instead of filling the gaps of the target population. Figure 5.1 depicts an example of stacking different populations to improve a model's population, where the variables X_1 and X_2 ranges from -5 to $+5$ with single-digit precision.



(a) Initial Population (b) Stacking Population (c) Improved Population

Figure 5.1 – Stacking Populations

Figure 5.1 (a) portrays an experiment where the *Initial Population* has a five-member population generated through a *RAND* distribution. The *Stacking Population*, Fig. 5.1 (b), has a total of eight members distributed in the domain via the *SLHD* method. Following at 5.1 (c), there is the *Improved Population*, which is the resulting population of the combination of previous populations.

This experiment aims to demonstrate that stacking a population on top of other it is not a very effective method to complement a model's population. Moreover, Fig. 5.1 (c) shows that the *Improved Population* still has regions without any training points, but has highly congested regions. Moreover, the methodology to be used to gauge the improvement of the population is another point of consideration, because both diversity-based metrics introduced at Chapter 4, Diversity (123) and Pure Diversity (112), possess strong dependence on the population' size.

The quality of a surrogate model's population is directly correlated with its coverage, in which smaller gaps indicate a higher degree of coverability of the search space. The problem of locating those **gaps** at a domain for humans is a relatively simple task. However, writing an algorithm that

translates this task into a methodology is quite challenging.

Therefore, the following chapter will be devoted to the development of a set of new space-filling algorithms. Namely, this chapter represent a detailing of the process **"Space Filling Strategy"** at Fig. 4.8 from the Chapter 4.

Section 5.1 develops an algorithm with the capability of identifying additional training points at a *bi*-dimension search space. Section 5.2 extends the first algorithm's operation to a *N*-dimensional problem, while introducing others issues. Section 5.3 develops a space-filling algorithm capable of working on any problem dimension while outputting promising results at a low computational cost. Section 5.4 develops a space-filling metric in order to measure the presence of gaps in populations. Section 5.5 highlights the influence of algorithm' improvements on its internal operation. Section 5.6 discuss the iterative usage of the algorithm. At last, Section 5.7 discourse on the challenges of its utilization at high dimension problems.

5.1 THE FIRST ALGORITHM PROPOSED: FILL TRI

The *Fill Tri* algorithm utilize a triangle-based method to identify regions without points in the space search on a two-dimension problem. A procedure done in three stage: a) *creation*, b) *elimination*, and c) *selection*.

5.1.1 Creation Stage

The *creation* stage cluster the points of the *Initial Population* in groups of size equals to the dimension of the problem plus one to form triangles. For each point in the population, the procedure the identifies the *depth number* (DN) points closest to the initial point then build triangles with remaining points of the population while calculating their area. Subsequently, the algorithm utilizes the triangle's area as a classifier to select DN triangles among them, in which larger areas indicates longer distances between its points.

Equation 5.1 describe the *Shoelace Formula* used to determine the area of these triangles, where A_x, B_x, and C_x describe the x and y coordinates of three arbitrary points A, B, and C respectively.

$$Area = \begin{vmatrix} 1 & 1 & 1 \\ A_x & B_x & C_x \\ A_y & B_y & C_y \end{vmatrix} \tag{5.1}$$

Within the region delimited by a triangle, there is a point called *incenter*, which is located at the intersection of the triangle's three angle bisectors. Moreover, this is also the position of the center of the triangle's *incircle*. Therefore, in this algorithm, the circles' center position composes the population of candidates training points selected to improve the model's coverage

in the region.

Equations 5.2, and 5.3 describe the mathematically used to solve the *incenter* coordinate problem, where A_x, B_x, and C_x describe the x and y coordinates of three arbitrary points A, B, and C respectively. The side lengths opposing the vertex A, B, and C are indicated as a, b, and c. At last, p indicates the perimeter of the triangle ABC, i.e., the sum of the side lengths a, b, and c.

$$O_x = \frac{aA_x + bB_x + cC_x}{p} \tag{5.2}$$

$$O_y = \frac{aA_y + bB_y + cC_y}{p} \tag{5.3}$$

Figure 5.2 exemplify the geometric interpretation of the *creation* stage, where the points in black indicate the triangle vertices, and in blue is the center of the triangle's incircle, i.e., the *candidate training point*.

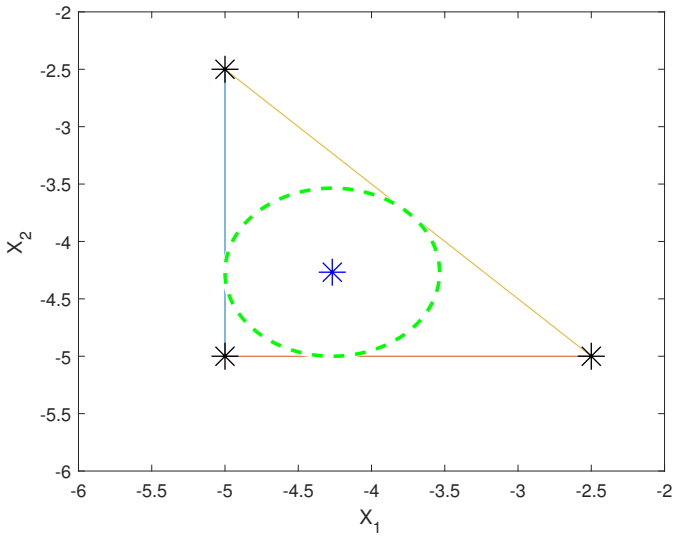


Figure 5.2 – Triangle's Problem - Geometric Interpretation

Moreover, it should be noted that the circle's center coordinates do not represent the final position of the candidate point, because all of them must still respect the precision parameters for each dimension. Therefore, there will be a slight deviation from its geometric interpretation after the application of the corrective measures.

5.1.2 Elimination Stage

The *elimination* stage uses the distance between the candidate points towards the closest point from the *Initial Population*, and the distance between themselves to shrink the candidates' population. This process aims to removes the a) *unwanted* candidates, b) *duplicates* candidates, and c)

overlapping candidates.

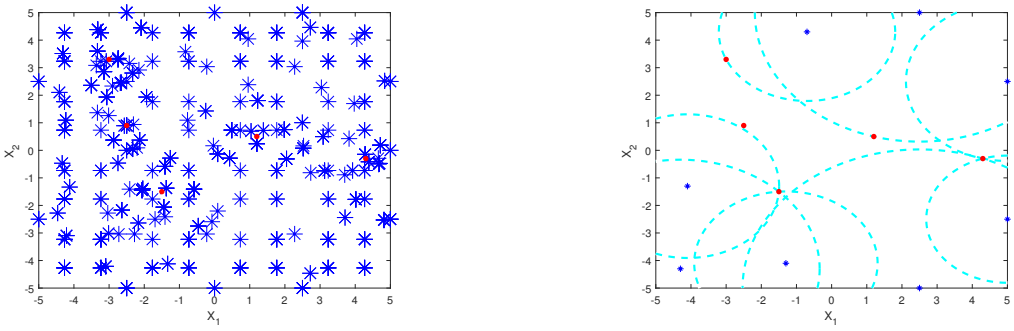
The *unwanted* candidates are the ones that are too close to the *Initial Population* or are not within a desired region of the search space. Therefore, they would not improve the coverage of the model effectively. The *duplicates* candidates describe solutions that already compose the candidates' populations. At last, the *overlapping* candidates is a type of unwanted candidate but regarding the candidate population itself, i.e., points that are too close to another point of the candidate population.

5.1.3 Selection Stage

The *selection* stage utilize the distance towards the *Initial Population* to choose the training candidates that would best improve the coverage of the *Improved Population*. This procedure chooses first the candidates that are further away from the initial population then checks whether the subsequent ones are located within their respective coverage region.

5.1.4 Results

Figure 5.3 exemplifies the behavior of the algorithm *Fill Tri* utilizing the same initial population as in Fig. 5.1, where Fig. 5.3 (a) holds all identified candidate training points, and Fig. 5.3 (b) shows the improved population, in which the cyan circles represent the coverage area of each new training point.



(a) Candidate Training Points (b) Improved Population

Figure 5.3 – Space-filling Algorithm: Fill Tri

However, the algorithm does not guarantee the matching between the required number of additional training points with the infill population's size. Therefore, should none of the candidates meets the pre-requisites, the procedure is *inconclusive*.

5.1.5 Auxiliary Population

However, the position of the population's points restricts the candidate training point's position, i.e., the candidate's coordinates are within the region described by three training points.

Thus, this procedure is incapable of creating candidate training points in unexplored regions of the search space domain. Therefore, an *auxiliary population* was used to supplement the coverage deficiency of the initial population and aid to identify additional training points in the entire search space (see line 2 of Algorithm 1).

The utilization of grids as the auxiliary population has the advantage of evenly covering the entire search domain while making the procedure *non-stochastic*. However, due to its population growth rate, the auxiliary population's size becomes an issue, because of the number of three-member grouping would be overwhelming. Therefore, the overall computational requirements and processing time of the algorithm becomes prohibitive. Therefore, the parameters *depth number*, and the *grid size* are fundamental to control the overall computational requirements and processing time of the Algorithm 1.

5.1.6 Algorithm Implementation

Therefore, Algorithm 1 details the implementation of this concept in the form of an algorithm, in which reveals that several parameters were used to regulate the operation of the space-filling algorithm's three stages, respectively: a) *Creation* (lines 1 to 20), b) *Elimination* (line 21), and c) *Selection* (line 22).

Algorithm 1 Fill Tril

INPUT: Initial Population (IP)
Number of Points (N)
Initial Bounds (ID)
Final Bounds (FD)
Precision (P)
Lower Boundary (LB)
Upper Boundary (FD)
Behavior (B)
Grid Size (GS)
Depth Number (DN)
D Factor (DF)
R Factor (RF)
OUTPUT: Filling Points (F)

1: **procedure** FILLTRI
2: AP = *AuxPopulation(*ID, FD, GS*)*
3: m = *count(*AP*)* + *count(*IP*)*
4: D = *DistanceMatrix(*AP, IP*)*
5: **for** $i = 1$ to m **do**
6: $P1$ = *SelectP1(*i, D*)*
7: **for** $j = 1$ to DN **do**
8: $P2$ = *SelectP2(*$P1$, D*)*
9: **for** $k = 1$ to $m - 2$ **do**
10: $P3$ = *SelectP3(*$P1$, $P2$, D*)*
11: T_k = *Triangle(*$P1$, $P2$, $P3$*)*
12: A_k = *Area(*T_k*)* ▷ Eq. 5.1
13: **end for**
14: **for** $k = 1$ to DN **do**
15: S_k = *SelectTriangle(*A_k, T_k*)*
16: C_k = *Center(*S_k*)* ▷ Eq. 5.2, 5.3
17: D_k = *Distance(*S, P, AP*)*
18: **end for**
19: **end for**
20: **end for**
21: E = *Elimination(*LB, UB, DF, RF, D, C*)*
22: F = *Selection(*N, B, E*)*
23: **return** F
24: **end procedure**

The first stage is responsible to generate the candidate training points, procedure ruled by five parameters: *Grid Number*, *Depth Number*, *Precision*, *Initial Domain*, and *Final Domain*. In the second stage, four eliminatory procedures filter invalid or undesired solutions, in which four

parameters regulates its operation: *R Factor*, *D Factor*, a *Lower Boundary*, and an *Upper Boundary*. At last, the third stage regulates the selection of the training points among the remaining candidates, a process controlled by a pair of parameters: *Behavior* and the *Number of Points*.

The *Grid Number* regulates the number of intermediate points in the grid that composes the auxiliary population, while the *Depth Number* specifies the number of neighbors points accept for each pairing while forming the groups. In this algorithm, the input parameter *Precision* regulates the number of digits in the right of the decimal point of the coordinates. Likewise, the parameters *Initial Domain* and *Final Domain* define the grid's covered region within the boundary search space.

Meanwhile, the *R Factor* and *D Factor* are both proximity parameters, respectively related to the minimum distance to any member of the mesh and the minimum distance between candidates. In this context, the pair *Lower Boundary* and *Upper Boundary* define the allowed space search domain while the parameter *Behavior* manages the occurrence of overlapping coverage of the additional training points. At last, the *Number of Points* stipulates the desired number of additional training points.

At first, the functionality of the parameters *boundary* and *domain* may be seen very similar. Nonetheless, the parameter *domain* specifies the region where the auxiliary population resides, whereas the parameter *boundary* defines the acceptable region for the candidates training points. Thus, since the positions of candidate training points are generally within the region delimited by its group point, the skillful manipulation of those parameters allows the conditioning of the candidate training points' coordinates.

The parameters *behavior* and *number of points* are the principal factors to regulate the number of additional candidates, in which the first controls the overlapping of candidates' coverage, i.e., the distance between the candidates, and the second influence the minimum distance between the training points and the initial population.

Moreover, the behavior control has two distinct modes of operation: a) aggressive control, and b) mild control. The first has an aggressive control over the overlapping of coverage of the additional training points, which restricts the overlapping of coverage's regions by additional training points, i.e., it aims to use a single point to supplement the gaps in the population. In contrast, the second mode of operation uses a mild approach that combines several candidates training points in order to improve the coverage of the improved population. Therefore, the former hinder the addition of many points but also enhances less appealing regions, whereas the second, on the other hand, places a high priority on the most disadvantaged region, and thus may not be able to enhance the other regions.

Thereby, the list below records all the configurable parameters necessary to operate the developed algorithm:

- Input population
- Number of points
- Initial domain

- Final domain
- Upper boundary
- Grid number
- Precision
- Depth number
- D factor
- Lower boundary
- Behavior
- R factor

5.1.7 Limitation of this algorithm

Nonetheless, the algorithm has some known limitations: a) *fixed problem dimension* (2D), and b) *complexity*. The first issue is an inheritance of its geometric foundation, the mathematical equations of triangles prohibit its escalation to higher dimensions. The second issue is due to the identification of the candidates training points, which demands the solution of a system of equations for each new training points evaluated.

Thus, the restrictions of the algorithm *Fill-Tri* lies in the lack of generalization of its mathematical foundation, which limits its operation to only 2D problems. Nonetheless, beyond the geometric topological space, extends the manifold topological space, which already has a well-defined n-dimensional problem that is very similar to the first algorithm proposition. Consequently, in the next section, a new algorithm is then proposed to solve the fixed dimension restriction.

5.2 SECOND ALGORITHM PROPOSITION: FILL N-SPHERE

The *N*-dimensional version deviates from the bi-dimensional version of the algorithm through their interpretation of the process of locating the candidate training points, in which *N* is linked to the problem's dimension, i.e., the number of design variables. The cornerstone of this algorithm is the concept of *N-Sphere*, which is a generalization of an ordinary sphere to spaces of arbitrary dimension.

5.2.1 Creation Stage

Given that according to the *Delaunay Triangulation*'s theory, the coordinates of N-Sphere's center point lies on the bisectors of line segments connecting the $N + 1$ points on its perimeter (124). In here, the idea is finding an N-Sphere in which a set of points resides in its surface, where similarly to the algorithm *Fill Tri*, the center of those N-Sphere are the coordinates of the candidate training points. For that reason, *Fill N-Sphere* labels the n-dimensional version of the space-filling algorithm.

The process consists in letting one of these points acts as an anchor point, $\overrightarrow{x_o}$, and then determining N vectors $\overrightarrow{v_1}, ..., \overrightarrow{v_N}$ pointing towards the remaining points, as indicated in Eq. 5.4.

$$\overrightarrow{v_i} = \overrightarrow{x_i} - \overrightarrow{x_0} \tag{5.4}$$

Later, as described in Eq. 5.5, the normalization of these vector results in the unit-length vector $\overrightarrow{u_1}, ..., \overrightarrow{u_N}$.

$$\overrightarrow{u_i} = \frac{\overrightarrow{v_i}}{2} \tag{5.5}$$

The identification of a vector $\overrightarrow{r}$ pointing from $\overrightarrow{x_o}$ towards the center of the N-Sphere is the final step of this procedure, which takes advantage of the fact that the projection of the vector $\overrightarrow{r}$ on each unit-length vector $\overrightarrow{u_i}$ has its endpoint at the midpoint of the line segment from $\overrightarrow{x_o}$ to $\overrightarrow{x_i}$ (the projection of $\overrightarrow{r}$ on $\overrightarrow{v_i}$ equals $\frac{\overrightarrow{v_i}}{2}$) to build system of linear equation $AX = B$ in order to determine R (124). Moreover, should the $\overrightarrow{x_0}, ..., \overrightarrow{x_N}$ be points on a sphere, Eq. 5.6 is non-singular and always solvable (see line 26 of algorithm).

$$\overrightarrow{r} = \begin{bmatrix} \overrightarrow{u_1}^T \\ \vdots \\ \overrightarrow{u_N}^T \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{2} \left\| \overrightarrow{v_1} \right\| \\ \vdots \\ \frac{1}{2} \left\| \overrightarrow{v_N} \right\| \end{bmatrix} \tag{5.6}$$

Thus, Equation 5.7 describe that the center of the N-Sphere $\overrightarrow{c}$ is the sum of the anchor point $\overrightarrow{x_o}$ and vector $\overrightarrow{r}$ (see line 27 of algorithm).

$$\overrightarrow{c_i} = \overrightarrow{x_0} + \overrightarrow{r_i} \tag{5.7}$$

Figure 5.4 exemplify a bi-dimensional geometric interpretation of the concepts behind the algorithm *Fill N-Sphere*, where the points in black indicate the points over the N-Sphere's perimeter, and in blue is the center position, i.e., the *candidate training point*.

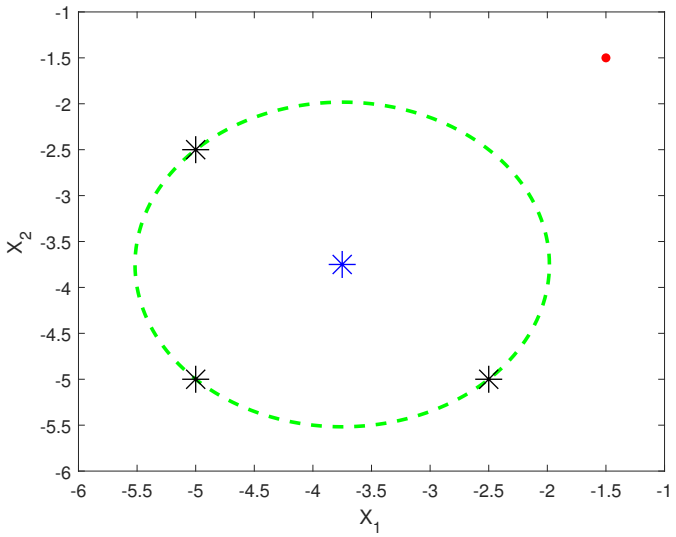


Figure 5.4 – N-Sphere's Problem - Geometric Interpretation

Algorithm 2 Grouping

INPUT: Dimension (N)
Number of Points (NP)
Depth Number (DN)
Distance Matrix (D)
OUTPUT: Points Groups (PG)

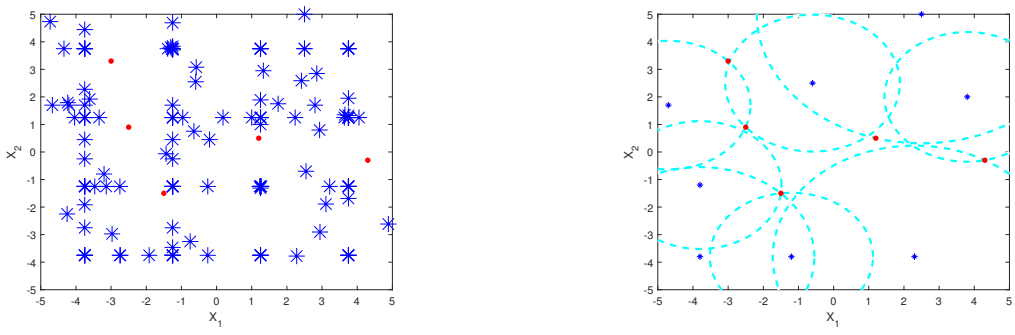
procedure GROUPING
  PG = []
  for $i = 1$ to NP **do**
    G_0 = i
    for $j = 1$ to N **do**
      G_j = []
      for $i = 1$ to *count*(G_{j-1}) **do**
        P = G_{j-1}(*k, :*)
        Dj = *sum*($D(P,:)$)
        [Dj, $Index$] = *sort*(Dj, *2*)
        F = *min*(*NP-j*, DN, *count*(Dj))
        for $k = 1$ to F **do**
          G_j = [G_j P, $Index(k)$]
        end for
      end for
    end for
    PG = [PG; G_N]
  end for
  Out F
end procedure

5.2.2 Elimination Stage & Selection Stage

Nonetheless, at the satge *elimination* and *selection* the algorithm *Fill N-Sphere* utilizes the same process as the *Fill Tri*.

5.2.3 Results

Figure 5.5 exemplifies the behavior of the algorithm *Fill N-Sphere* utilizing the same initial population as Fig. 5.1, where the blue asterisks at Fig. 5.5 (a) represent position of the *candidate training points*, and Fig. 5.5 (b) shows the improved population, in which the cyan circles represent the coverage area of each new training point.



(a) Candidate Training Points (b) Improved Population

Figure 5.5 – Space-filling Algorithm: Fill N-Sphere

5.2.4 Algorithm Implementation

The grouping process was another procedure of the space-filling algorithm that had a fixed dimension implementation. Algorithm 2 implements a new grouping process, which essentially

recursively pair the points in order to build groups with $N + 1$ points from the initial population and the auxiliary population. This paring process, thou still selects the points that are the closest to its anchor point.

Therefore, Algorithm 3 holds the pseudo-code of the implementation of the algorithm *Fill N-Sphere*.

Algorithm 3 Fill N-Sphere

INPUT:	Initial Population (IP)
	Number of Points (N)
	Initial Bounds (ID)
	Final Bounds (FD)
	Precision (P)
	Lower Boundary (LB)
	Upper Boundary (FD)
	Behavior (B)
	Grid Size (GS)
	Depth Number (DN)
	D Factor (DF)
	R Factor (RF)
OUTPUT:	Filling Points (F)

```
1: procedure FILLNSPHERE
2:     AP = AuxPopulation(ID, FD, GS)
3:     D = DistanceMatrix(AP, IP)
4:     G = Grouping(D, AP, DN, IP)
5:     for i = 1 to count(G) do
6:         R_i = DetermineR(G_i)            ▷ Eq. 5.6
7:         C_i = G_i(0) + R_i               ▷ Eq. 5.7
8:     end for
9:     E = Elimination(LB, UB, DF, RF, D, C, R)
10:    F = Selection(N, B, E, R)
11:    return F
12: end procedure
```

5.2.5 Limitation of this algorithm

However, even though the algorithm *Fill N-Sphere* removes the *fixed problem dimension* present in the algorithm *Fill Tri*, it still remains as a relatively complex process that has to solve a system of linear equations for each candidate training point. Moreover, should a solution exist, the center position may not even be within the space search domain. This problem is due to bad conditioning of the matrices of the system of linear equations, which means that the points that compose the group are not within the perimeter of any *N-Sphere*.

Thus, the physical interpretation restricts the operation of the algorithm. Nonetheless, the aim of the algorithm is just adding points in a skillfully matter in a pre-established population. As such, a physical correlation of this methodology is entirely unnecessary. Therefore, in the next section develops a single yet effective strategy to identify the coordinates of the candidate training point.

5.3 THIRD ALGORITHM PROPOSITION: FILL-AVG

This version took reference from the algorithm *Fill N-Sphere*, while following a single guideline: keep it simple. Its inspiration arises from the analyses of the *Geometric Interpretation* of the algorithm *Fill N-Sphere* (see Fig. 5.4), which essentially describes a process that the candidate

training points are somewhat distant from a set of points. In other words, a strategy that uses $N + 1$ points to determine the location of a *candidate training point* within the search space.

Nonetheless, the creation process' strategies of the algorithms *Fill Tri* and *Fill N-Sphere* have issues regarding their scalability and dimension operation.

5.3.1 Creation Stage

Hence, a new strategy was developed in order to identify the new candidate training points, which limits to locating a point within the search space at a different position than its $N+1$ anchor points. Moreover, this strategy has to possess a low computational cost and must not demand the resolution of any system of equation.

Equation 5.8 describes the new strategy, in which the mean location of the $N + 1$ anchor points that composes group replaces all the previous complex procedures, where $\overrightarrow{x_{AP}}$ indicates the set of anchor points. The main advantage of this strategy is that all generated candidate training points respect the search space restrictions while reconciling easy implementation and low computational cost. Therefore, due to its average behavior, this algorithm as named *Fill AVG*.

$$\overrightarrow{x_c} = \frac{1}{N+1} \sum_{i=1}^{N+1} \overrightarrow{x_{APi}} \tag{5.8}$$

Subsequently, Eq. 5.9 describes the criterion utilized in the *Elimination* and *Selection* stages of the algorithm, in which adopts the the shortest distance towards any point that make up the initial population in order to evaluate the training candidates points, where $\overrightarrow{x} = [\overrightarrow{x_1}, ..., \overrightarrow{x_m}]^T$ indicates the set of points that composes the *Initial Population*.

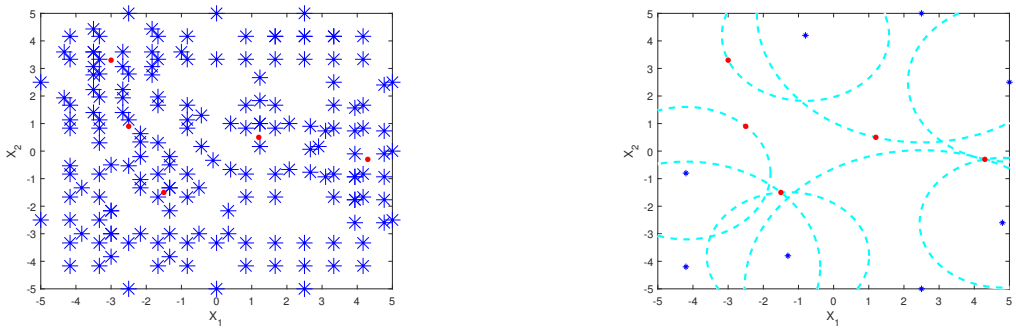
$$R_c = min(\| \overrightarrow{x} - \overrightarrow{x_c} \|) \tag{5.9}$$

5.3.2 Elimination Stage & Selection Stage

Nonetheless, at the satge *elimination* and *selection* the algorithm *Fill N-Sphere* utilizes the same process as the *Fill Tri*.

5.3.3 Results

Figure 5.6 exemplifies its utilization given the same initial population as Fig. 5.1, where the dotted green circle marks the coverage of the candidate training point to the nearest point in the model's population, and the blue asterisks represent the location of the candidate training points.



(a) Candidate Training Points (b) Improved Population

Figure 5.6 – Space-filling Algorithm: Fill AVG

5.3.4 Algorithm Implementation

Therefore, Algorithm 4 holds the pseudo-code of the implementation of the algorithm *Fill AVG*.

Algorithm 4 Fill AVG

INPUT:	Initial Population (IP)
	Number of Points (N)
	Initial Bounds (ID)
	Final Bounds (FD)
	Precision (P)
	Lower Boundary (LB)
	Upper Boundary (FD)
	Behavior (B)
	Grid Size (GS)
	Depth Number (DN)
	D Factor (DF)
	R Factor (RF)
OUTPUT:	Filling Points (F)

```
1:  procedure FILLAVG
2:      AP = AuxPopulation(ID, FD, GS)
3:      D = DistanceMatrix(AP, IP)
4:      G = Grouping(D, AP, DN, IP)
5:      for i = 1 to count(G) do
6:          C_i = mean(G_i)                    ▷ Eq. 5.8
7:          R_i = MinDistance(C_i, IP)         ▷ Eq. 5.9
8:      end for
9:      E = Elimination(LB, UB, DF, RF, D, C,
    R)
10:     F = Selection(N, B, E, R)
11:     return F
12: end procedure
```

5.3.5 Advantages of this algorithm

This algorithm adopts an identification strategy for the candidate training point's coordinates that do not relies on the solution of a linear system of equation. Moreover, it is both computationally cheap and guarantees that the identified candidate training points are within the predefined search space.

5.4 SPACE-FILLING METRIC

The aim of the space-filling algorithms is the identification of a set of training points located at the gaps of the initial population. Nonetheless, the distances between these points and the inputs points are a byproduct of this process, where large values indicate the existence of large gaps in the population. Therefore, the information regarding the additional training points could be used to assess the initial population's coverage of the search space.

Algorithm 5 consolidates the information about the candidate training points and space search bounds into a *gap-based metric*, which a population without gaps receives a score equals to zero. Highlights that, the metric is biased to heavy ponderer the training points with the largest distances from the population because these would be the location of the highest probability to be a low-efficient region within the surrogate model. Moreover, this metric is independent of population size, given the utilization of the space-filling algorithm with similar input parameters.

Algorithm 5 Fill Score

INPUT: Distance R (R)
Filling Points (F)
Initial Bounds (ID)
Final Bounds (FD)
OUTPUT: Score ($Score$)

1: **procedure** FILLSCORE
2: $Delta$ = *mean(*$ID - FD$*)*
3: N = *count(*F*)*
4: $Score = -N$
5: **for** $i = 1$ to N **do**
6: $Score$ += $\left(1 + \frac{R_i}{Delta}\right)^{N-i+1}$
7: **end for**
8: **return** $Score$
9: **end procedure**

Nonetheless, its accuracy depends on the gathering of quality information about gaps in the model, i.e., the identification of regions that demands additional training points. However, skillfully adding several additional points is many times harder than inserting a few of them, due to poor positioning of the candidate training points or restrictions imposed over the space-filling algorithm.

Table 5.1 holds the metrics of the *Initial Population* and the four *Improved Population* comprising the ones generated via the *Stacking Population* method and the algorithms *Fill Tri*, *Fill N-Sphere* and *Fill AVG*, where the measures were generated utilizing the algorithm *Fill AVG* with twenty-five additional points.

Experiment	Metric	Percentage
Initial Population	8.1515	100.00 %
Stacking Population	6.1268	75.16 %
Fill Tri	5.3026	65.05 %
Fill N-Sphere	5.3766	65.96 %
Fill AVG	5.2519	64.43 %

Table 5.1 – Space-filling Metric: Fill Score

These results attest the advantages of the utilization of the developed space-filling algorithms,

in which either of the methodologies is capable of locating critical regions otherwise previously neglected. Nonetheless, it is undeniable that adding an SLHD-based population over the Initial Population does help to reduce its coverage's gaps.

5.5 AN EFFICIENCY ANALYSIS ABOUT THE ALGORITHM'S INTERNAL OPERATION

It can be observe that the complexity of Fill Tri (see Algorithm 1), Fill Tri (see Algorithm 3), and Fill Tri (see Algorithm 4) are practically the same for a problem of similar dimensionality. Nonetheless, this section adopts an *efficiency* perspective to evaluate the performance of the proposed algorithms, where it reflects the quality of the identified candidate training points.

All three developed space-filling algorithms share the same three-stage structure: a) *Creation*, b) *Elimination*, and c) *Selection*. Moreover, the last two stages have an identical implementation on all of them. Thus, it is correct to say that the only difference between them is the *Creation* stage, which is responsible for generating the set of candidate training points. Nonetheless, its is worth reminding that, one of the major improvements of the algorithms *Fill N-Sphere*, and *Fill AVG* over the *Fill Tri* is their points grouping process, which also composes the *Creation* stage. Therefore, the grouping process and creations strategy mark the major enhancements that occur during the development of the space-filling algorithms.

Figure 5.7 depicts the number of candidates training points during the execution of the *Elimination* stage for a problem of two dimension, where the variables X_1 and X_2 ranges from minus five to five with single digit precision.

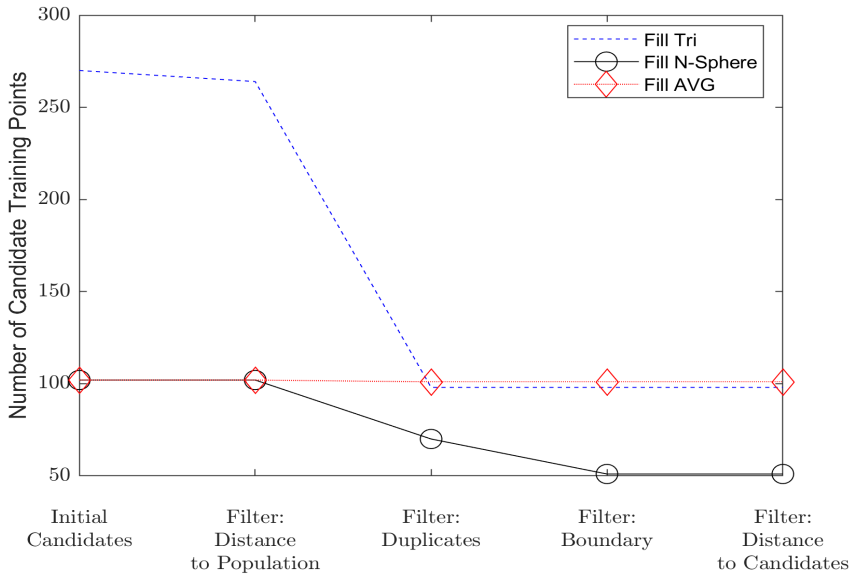


Figure 5.7 – Number of Candidates Training Points

The *Elimination* stage four criterion of elimination analyses the candidates training points in order to filter the unfitted points, respectively taking in consideration the a) *Distance to the Initial Population*, b) *Points Duplicity*, c) *Boundary Violations*, and the d) *Distance to Candidates*.

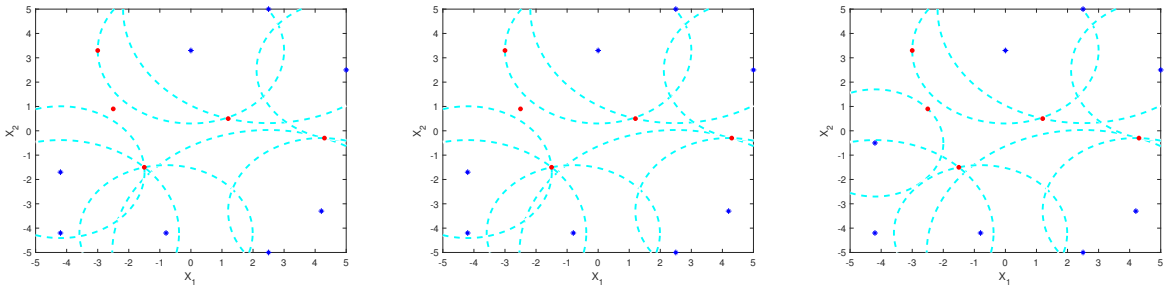
The comparison of the number of *Initial Candidates* at Fig. 5.7 highlights the improvement of the grouping process, which yield a significant decrease in their amount, thereby also significantly reducing the computational cost and execution time. In contrast, the information regarding the *Distance to Population* reveals that generating points too close to the initial population it is not a major issue for all the algorithms.

Nonetheless, the execution of the procedure to correct the precision of the candidate training points triggers the algorithms *Fill Tri*, and *Fill N-Sphere*: points duplicity. Moreover, this experiment also reveals how serious is the issue of boundary violations in algorithm *Fill N-Sphere*. Thus, it is indisputable that the algorithm *Fill AVG* is the most efficient among its peers, regarding the use of resources.

5.6 IN LOOP OPERATION

Regardless of the input parameters configuration, its universal that the firsts candidate training points are the best at improving the coverage of the population. From that notion surges a strategy that bypasses the necessity of fine-tuning algorithms input parameters. The idea is to run the space-filling algorithm in an iterative form, while subsequently adding the first candidates into the population. Nonetheless, this methodology has a high computational cost but yields the best possible results.

Figure 5.8 exemplifies its utilization given the same initial population as Fig. 5.1, where the dotted green circle marks the coverage of the candidate training point to the nearest point in the model's population, and the blue asterisks represent the location of the candidate training points. The number that follows "Loop" indicates the number of additional training points added on each iteration to the Improved Population, which in this example demanded eight, four, and two iterations respectively.



(a) Fill AVG: Loop 1
Metric: 5.1893

(b) Fill AVG: Loop 2
Metric: 5.1893

(c) Fill AVG: Loop 4
Metric: 5.1965

Figure 5.8 – In Loop Space-filling

The metric fill score of each of them reveals that this mode of operation is indeed capable of improving the coverage of the population. However, that advantage does not worth the additional processing time.

5.7 HIGH-DIMENSION PROBLEMS

The exponential growth of the grid marks the algorithm biggest issue, a characteristic that translates in the creation of a massive number of possible candidates, which make its utilization prohibitive regarding the hardware requirements, and its execution time, especially at a high-dimension problem.

In those cases, a population with an *acceptable number of points* could act as the auxiliary population instead of the grid. Nonetheless, the usage of dynamic populations eliminates the algorithm's **non-stochastic behavior**, i.e., the algorithm no longer outputs the same improved population given the same parameters and initial population.

Table 5.2 holds the results of an experiment of a twenty-dimension problem, where the variables $X_1, ..., X_{20}$ ranges from minus five to five with single-digit precision, and the Initial Population has a total of fifty points, in which the space-filling algorithm has to identify twenty-five additional points while utilizing a set of auxiliary populations of different size.

Experiment	Auxiliary Population	Metric
Initial Population	-	171.40
Improved Population with Stacking Population	-	166.37
Improved Population with Fill AVG	250 points	164.48
Improved Population with Fill AVG	500 points	167.50
Improved Population with Fill AVG	1000 points	166.56
Improved Population with Fill AVG	2000 points	166.36
Improved Population with Fill AVG	4000 points	164.86
Improved Population with Fill AVG	8000 points	163.10

Table 5.2 – Space-filling at High Dimensions

In order to ensure fairness in the experiment, the auxiliary population with a higher number of points houses the points that compose the smaller population. Therefore, the auxiliary population remains static throughout the experiment during the different stages of the experiment. Thus, enabling the analysis of the influence of the auxiliary population's size in the performance of the space-filling algorithm.

The construction of the static auxiliary population is an extremely time-consuming procedure, in which its first-generation took an SLHD distribution to set the position of 250 points in its search space. Following, batches of fifth additional points populate the subsequent points, which ensures that the auxiliary population improves as it growths.

These experiments demonstrate that even with a relatively small auxiliary population the algo-

rithm *Fill Tri* outperforms its benchmark, the methodology based on stacking population. Moreover, the number of possible positions at a high-dimension problem is so expressive that utilizing a small set of points to cover this space search naturally yield in its fault coverage, i.e., the population that is just too small to cover the domain effectively. Therefore, these characteristics practically make irrelevant the usage of a vast auxiliary population to improve the algorithm performance, due to the sheer amount of large gaps in the population.

5.8 OVERVIEW

In this chapter, three new space-filling algorithms were developed to handle the problem of efficiently adding new points into an already existing population. This process of creation resulted in an algorithm that adopts an extremely simple strategy to identify the location of these additional points, which also culminated in the creation of a space-filling metric that is independent of the initial population size. Next chapter will be devoted to discussing the results of the surrogate-based optimization methodology in several Mechanical Structural Design's problems.

6 CASE STUDIES

A practical example is an incredible tool for validating and testing the functionalities of methodologies. Nonetheless, the goal of the following case study is not the application itself neither their results, but the capability of the surrogate-assisted technique to model the problem's cost functions. Therefore, although there would be the problem's contextualization, the analysis, interpretation, and discussion of the results will be centered in the capabilities of the surrogate model.

Therefore, the following chapters will be devoted to developing three structural optimization problems. Chapter 6.1 utilizes a heatsink to discuss about the optimization of thermal systems. Chapter 6.2 optimize the structure of a coffee table given the application of external loads. At last, Chapter 6.2 analyze the optimization process of a moving support structure of an exoskeleton.

6.1 HEATSINK

The design of thermal systems is a particular field of research within the engineering design process, in which the thermodynamics, the fluid flow, and the thermal transport composes the project main criterion (125). In particular, the design of microprocessor applications has become increasingly challenging due to the increase in the total power, die shrinkage and other complexities of microprocessor design, which have increase the solution local powers densities (126).

6.1.1 Problem's Description

Therefore, thermal management of microelectronics components has become critical to maintaining the device performance and reliability (127). In that context, a cost-effective solution for removing heat from the components are solid metal *heatsink*.

A heatsink is a passive heat exchanger that transfers the heat generated by an electronic or a mechanical device to a fluid medium, often air or a liquid coolant, where it is dissipated away from the device, thereby allowing regulation of the device's temperature at optimal levels.

Figure 6.1 describe the simulation scenario of a heatsink that was created based on the *Fischer Elektronik heatsink SK47* (128), where there is an $40mm$ height ten-fin heatsink over a microprocessor measuring 100 mm x 75 mm x 10 mm (width x depth x height). This experiment utilizes the software *ANSYS version R19.1* to execute a *Steady-State Thermal* simulation of a microprocessor and its heatsink, wherein the image **A** indicates convective heat transfer, **B** the radiation from surface to surface of the heatsink, **C** the radiation from heatsink to the ambient, **D** the radiation from surface to surface of the microprocessor, **E** the radiation from the microprocessor to the

ambient, **F** a heat flow of $60W$ in the microprocessor, and **G** a negative heat flow of $30W$ in the heatsink.

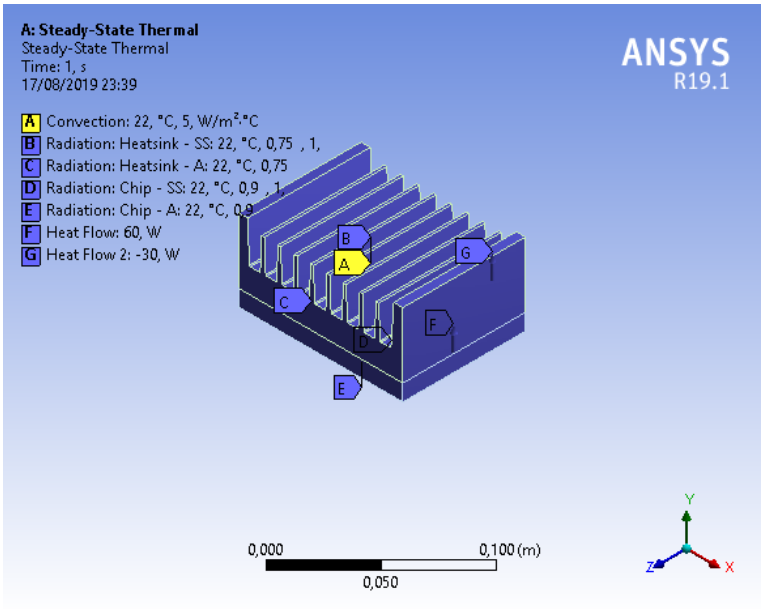


Figure 6.1 – Heatsink - Simulation Scenario

Table 6.1 holds the thermal properties of the selected materials used in the case study, where the heatsink is made of copper alloy and the microprocessor is made of silicon anisotropic.

Material	Thermal Conductivity	Radiation Emissivity
Copper Alloy	$401Wm^{-1}C^{-1}$	0.75
Silicon Anisotropic	$124Wm^{-1}C^{-1}$	0.90

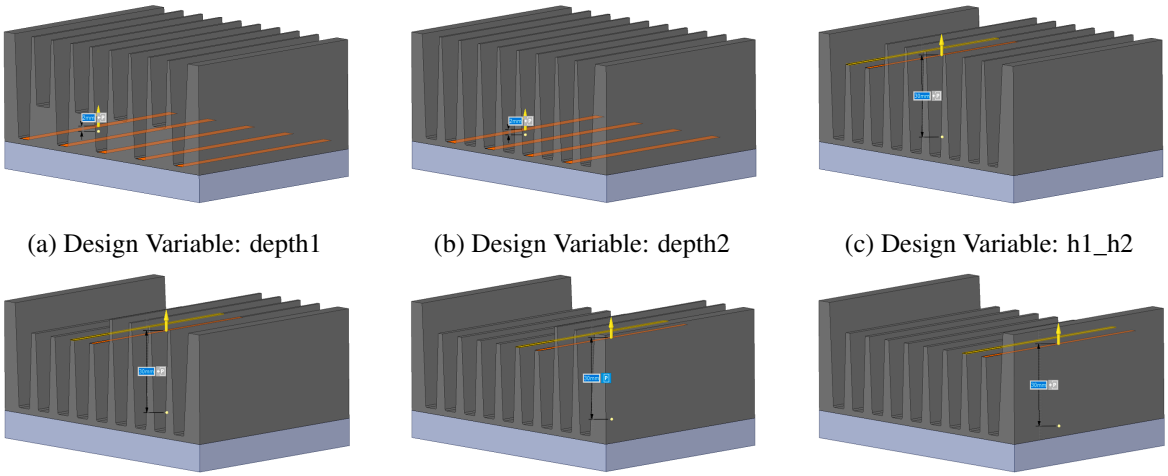
Table 6.1 – Heatsink - Engineering Data

6.1.2 Design Variables

This study case aims to optimize the heatsink while taking into consideration the thermal management of the microprocessor. Therefore, utilizing the approach of structural optimization, the process becomes a problem of shape optimization, where the design variables are the fin's height and the deepness between fin.

Highlights that during optimization, the fins' height could be independent design variables, and so the shape optimization would be a ten-dimension problem. Nonetheless, only four fin's height was specified as design variables in order to reduce four dimensions of the optimization problem.

Figure 6.2 specifies which of the geometric features of the heatsink that were selected as design variables of the optimization, where six design variables composes the list of design variables: a) *depth1*, b) *depth2*, c) *h1_h2*, d) *h3_h4*, e) *h5_h6*, and e) *h7_h8*.



(a) Design Variable: depth1 (b) Design Variable: depth2 (c) Design Variable: h1_h2

(d) Design Variable: h3_h4 (e) Design Variable: h5_h6 (f) Design Variable: h7_h8

Figure 6.2 – Heatsink - Design Features

Table 6.2 describes the characteristics of domain and precision of each of the six design variables, which taking in consideration the precision of each of them results in a search space of **63504** possible solutions.

Design Variable	Inital Domain	Final Domain	Precision
depth1, depth2	2 mm	8 mm	0
h1_h2, h3_h4, h5_h6, h7_h8	30 mm	35 mm	0

Table 6.2 – Heatsink - Design Variables

6.1.3 Cost Functions

The cost functions of the optimization process are: a) the *weight* of the heatsink and b) the *maximum temperature* on the microprocessor measure at its top surface, in which the first depends on the geometry of the heatsink while the second indicates the cooling capacity of the heatsink. Moreover, the information regarding *heat flux* on the microprocessor's top surface feeds the *Validation* system.

In order to configure the surrogate-based optimization process, several simulations manually executed to determine the typical range of values of the cost functions, in which Tab. 6.3 holds the identified the expected cost functions' range of values for this specific heatsink design.

Cost Function	Lower Image's Value	Upper Image's Value
Heatsink's Mass	$890g$	$1200g$
Microprocessor's Top Surface Heat Flux	$11900W/m^{-2}$	$12600W/m^{-2}$
Microprocessor's Top Surface Temperature	$70°C$	$80°C$

Table 6.3 – Heatsink - Cost Functions' Range of Values

6.1.4 Results

In all instances of optimization process execution, the implemented solution run simulations on 1593 out of 63504 possible configurations of the problem, which would take approximately 25 hours to simulate **without occurring errors** in its execution. This measure of time only takes in consideration the execution time of each of the simulations which the minimum, mean, median and maximum execution time are 51.37 seconds, 56.67 seconds, 56.00 seconds, and 77.98 seconds respectively.

Therefore, an optimization process that runs for 150 iterations with a population size equals to 30 individuals would demand the execution of 4500 simulations. Thus, taking in consideration the median execution time of the evaluated simulations, this optimization process would take approximately **70** hours to finish, whereas sweeping the entire domain would take more than **41** days. Moreover, these estimations are for a particularly simple structure, given that the execution time of a single simulation of a complex structure could take from hours to days, the utilization of the conventional approach becomes impracticable.

Figure 6.3 depicts the results of the surrogate-based optimization process, which the first-generation surrogate model was built with 25 training points and 3 additional filing points were added on each update call, and the periodic update curve *Mode A* prescribes 5 model updates.

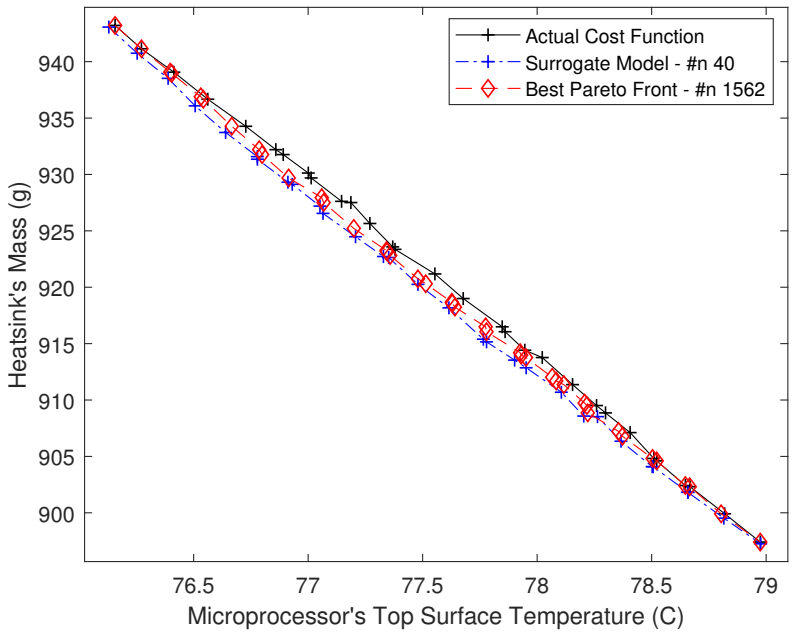


Figure 6.3 – Heatsink - Pareto Front

Figure 6.3 holds three Pareto Front: a) the Surrogate Model's Pareto Front, b) the Actual Cost Function's Pareto Front, and c) the Best Pareto Front. The first corresponds to the set of solutions that composes the last iteration of the optimization process, also known as *candidate solutions*. The second evaluates these *candidate solutions* in the actual cost function to identify the official

Pareto Front of the process. The third indicates the best known Pareto Front, which is identified taking utilizing all previously executed solution in the solution's database.

That way, during the experiment, 40 simulations were executed to build the models of the optimization process, and 30 additional evaluations were also done to evaluate the candidate candidates solution, which indicates that the surrogate-based optimization only had access to the actual cost function 70 times.

Nonetheless, the developed methodology was able to identify an accurate Pareto Front with **less than** 2% **of the number of cost functions evaluations of the conventional method** (70 out of 4500).

Figure 6.4 depicts when these cost functions evaluations happen during the execution of the optimization, and also indicates the amount of them that were already in the solution's database.

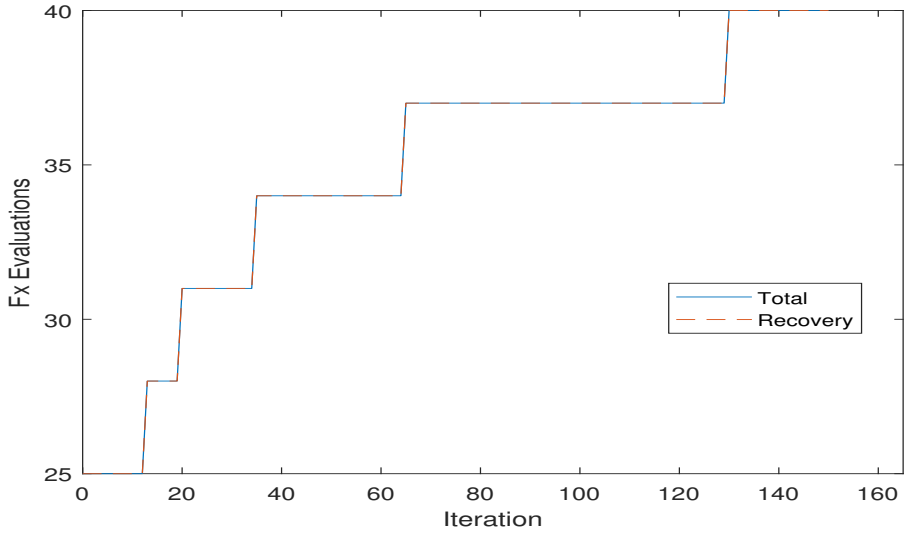
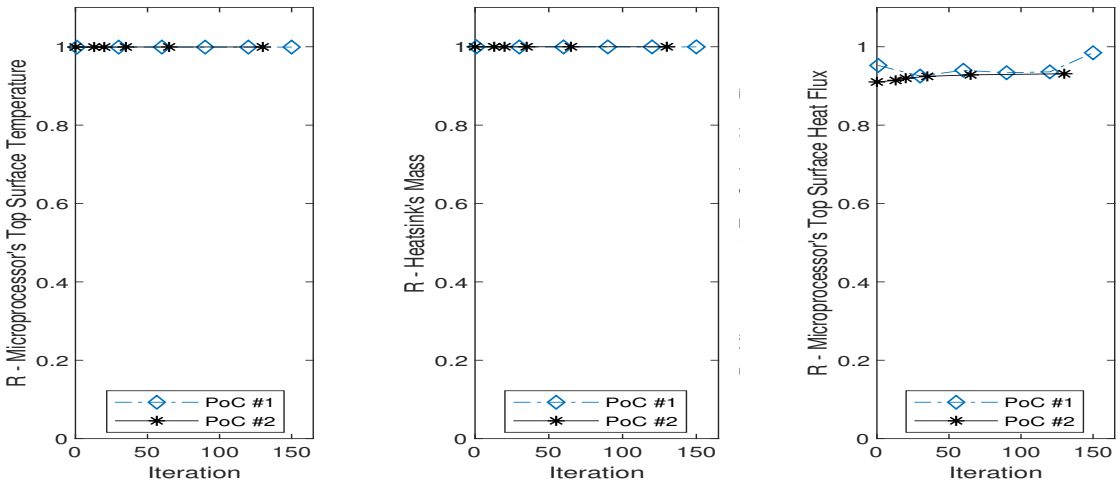


Figure 6.4 – Heatsink - Cost Function Evaluations

As *Proof of Concept*, the accuracy of the surrogate model was tracked utilizing two approaches: a) *Global Accuracy* and b) *Local Accuracy*. The *Global Accuracy* evaluates the values estimated of the cost functions in a population of points that are independent of the optimization process on each model update, whereas the *Local Accuracy* evaluates the iteration's Pareto Set with a frequency of 30-iteration to measure the model accuracy. Highlights that, both pare procedures are **completely independent** of the optimization process.

Figure 6.5 depicts the measures coefficient of correlation (R) of both of these approaches, in which values close to 1 indicates that the models outputs good estimations, where *Local Accuracy* is indicated as *PoC #1*, and *Global Accuracy* is indicated as *PoC #2*. In here, a grid with one intermediate points between the dimensions boundary was used to measure the global accuracy, in which it was composed by 729 points.



(a) Maximum Temperature (b) Mass (c) Heat Flux

Figure 6.5 – Heatsink - Proof Of Concept

As can be seen in Fig. 6.5, throughout the optimization, the model was able to estimate all the cost functions with reasonable accuracy. Nonetheless, the measures of *Global Accuracy* shows that the model build for the Microprocessor's Top Surface Heat Flux decreased its accuracy when additional training points were added to the model, which indicates that these new points improved the accuracy of model locally, but at the same time decrease the aspect of generalization of the model in a global scale.

Figure 6.6 depicts the disposition of the Pareto Set on each Design Variable in the domain, where the curve in red describe the nonparametric kernel-smoothing distribution of the data.

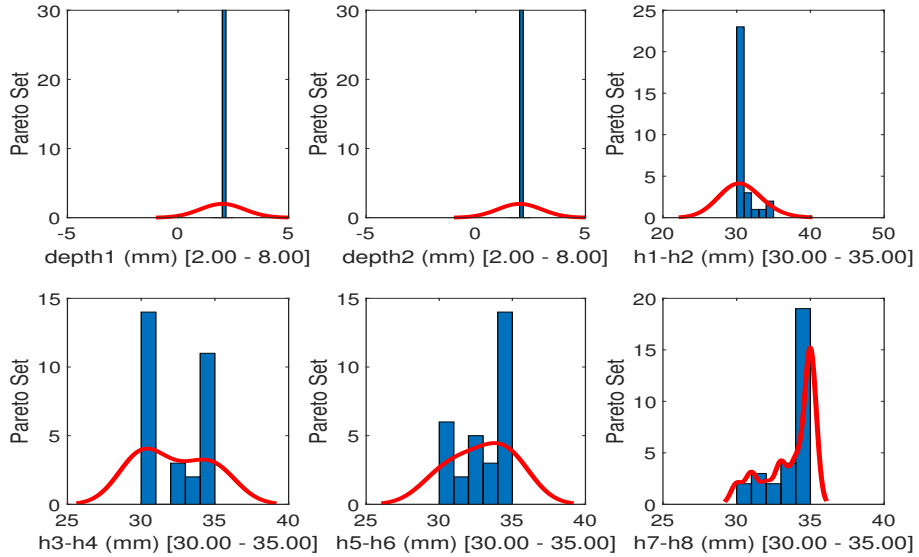


Figure 6.6 – Heatsink - Pareto Set

As can be seen in Fig. 6.6, the design variables *depth1*, and *depth2* remains the same on all solutions, which indicates that the Pareto Set resides in a particular region of the search space. Therefore, the six-dimension problem has only four meaningful design variables, which translates into a four-dimension problem that has only **1296** possible configurations. Thus, the surrogate-based optimization process could be used to measure the meaningfulness of the design variables prior the execution of the optimization process with the intention of assisting the construction of the problem itself.

6.2 COFFEE TABLE

The second study case follows similar approach as the one utilized in the first study case. Nonetheless, it aims to optimize a structure given the application of external loads, in which takes a Coffee Table - Model Sehpa (129) as the optimization problem's target structure.

6.2.1 Problem's Description

Figure 6.7 describes a simulation scenario built at the software *ANSYS version R19.1* to execute a *Static Structural* where a load of $1500N$ (approximately $150Kg$) are applied over the structure's top surface. This experiment aims to simulate the wrongfully utilization of the structure, where it could is been used as a chair or to support a object weighting over $150Kg$.

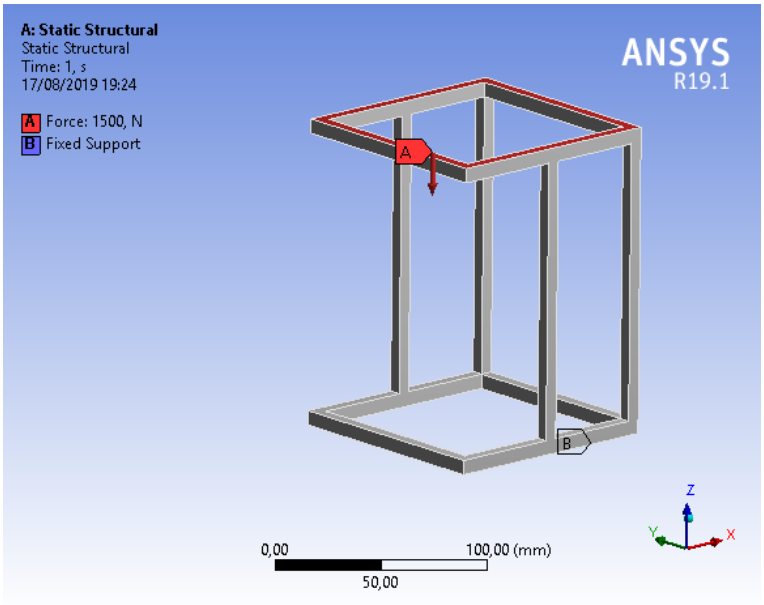


Figure 6.7 – Coffee Table - Simulation Scenario

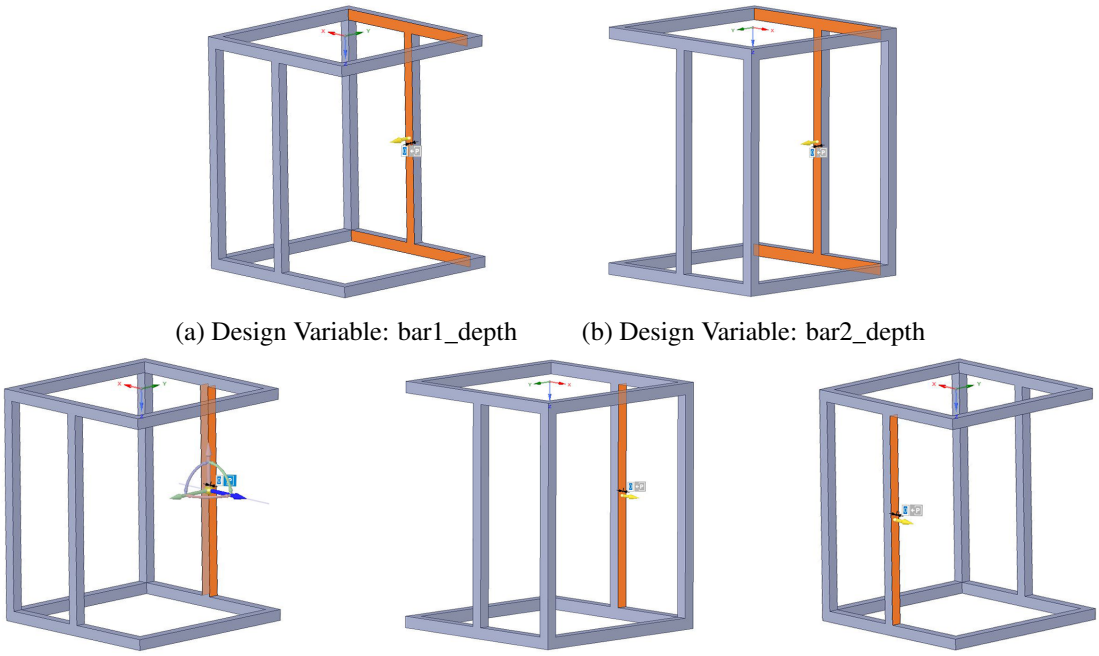
Table 6.4 holds the mechanical properties of the selected materials used in the case study, where the coffee table is made of structural steel.

Material	Density	Tensile Yield Strength	Young's Modulus	Poisson's Ratio
Structural Steel	$7850Kgm^{-3}$	$250MPa$	$2E+05Mpa$	0.3

Table 6.4 – Coffee Table - Engineering Data

6.2.2 Design Variables

This experiment was build into a five-dimension shape optimization problem, where five geometric features of the Coffee Table's structure were select as Design Variables: a) *bar1_depth*, b) *bar2_depth*, c) *bar1_pos*, d) *bar2_right*, and e) *bar2_left*. The *bar1_depth*, *bar2_depth*, *bar2_right* and *bar2_left* indicates the extrusion of the indicated regions of the structure. At last, the *bar1_pos* is responsible to set the position of one the central bar.



(a) Design Variable: bar1_depth (b) Design Variable: bar2_depth

(c) Design Variable: bar1_pos (d) Design Variable: bar2_right (e) Design Variable: bar2_left

Figure 6.8 – Coffee Table - Design Features

Table 6.5 describes the characteristics of domain and precision of each of the five design variables, which taking in consideration the precision of each of them results in a search space of **69696** possible solutions.

Design Variable	Initial Domain	Final Domain	Precision
bar1_pos	20 mm	35 mm	0
bar2_right, bar2_left	0 mm	10 mm	0
bar1_depth, bar2_depth	0 mm	5 mm	0

Table 6.5 – Coffee Table - Design Variables

6.2.3 Cost Functions

The cost functions of the optimization process are the *weight* and the *safety factor* on the coffee table, which the first depends on its geometry while the second indicates the actual load-bearing capacity of a structure or component. Moreover, the information regarding *total deformation* on the coffee table feeds the *Validation* system.

In engineering, safety factor, also known as factor of safety, expresses the structure's ability to sustain loads, which is commonly defined as the ratio between the strength of the material and the maximum stress in the part. Therefore, this measure enables a binary evaluation of the structural reliability, in which values smaller than one indicates a critical failure.

Nonetheless, several engineering projects, such as aviation and medical applications, have to respects strict regulations of minimum safety factor to ensure the structural integrity at adverse situations, e.g., emergencies, unexpected loads, misuse, or degradation.

Highlights that *safety factor* is innately a maximization cost function in a minimization optimization process. Therefore, the optimization process evaluates the negative value of this cost function in order to translate it to the minimization form.

In order to configure the surrogate-based optimization process, several simulations manually executed to determine the typical range of values of the cost functions, in which Tab. 6.6 holds the identified values for the specified structure.

Cost Function	**Lower Image's Value**	**Upper Image's Value**
Safety Factor	2.5	3.8
Mass	$400g$	$850g$
Total Distortion	$0.1mm$	$0.6mm$

Table 6.6 – Coffee Table - Cost Functions' Range of Values

6.2.4 Results

In all instances of optimization process execution, the implemented solution run simulations on 7195 out of 69696 possible configurations of the problem, which would take approximately 135 hours to simulate **without occurring errors** in its execution. This measure of time only takes in consideration the execution time of each of the simulations which the minimum, mean, median and maximum execution time are 44.11 seconds, 67.12 seconds, 72.45 seconds, and 138.35 seconds respectively.

Figure 6.9 depicts the results of the surrogate-based optimization process, which the first-generation surrogate model was built with 15 training points and 3 additional filing points were added on each update call, and the periodic update curve *Mode A* prescribes 5 model updates.

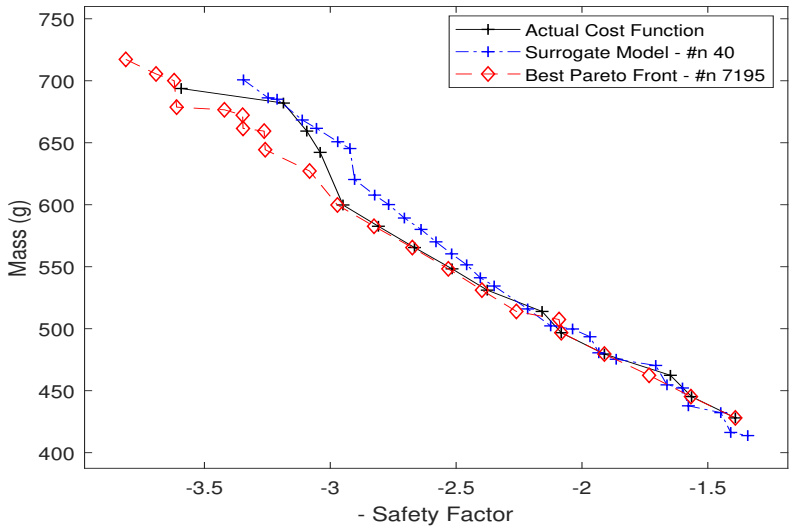


Figure 6.9 – Coffee Table - Pareto Front

Figure 6.10 depicts when these cost functions evaluations happen during the execution of the optimization, and also indicates the amount of them that were already in the solution's database.

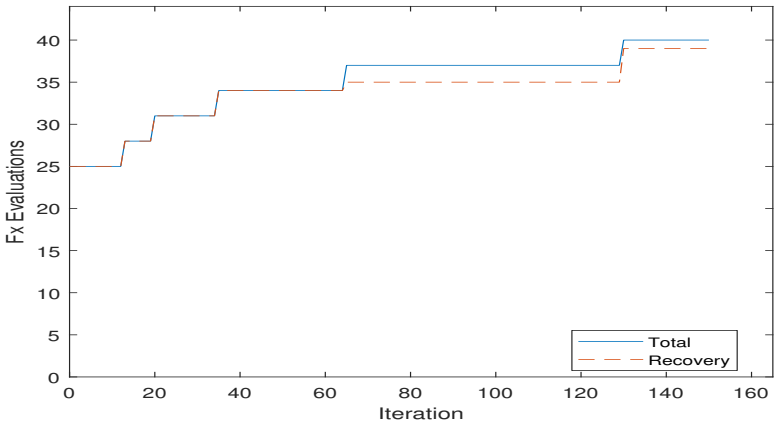
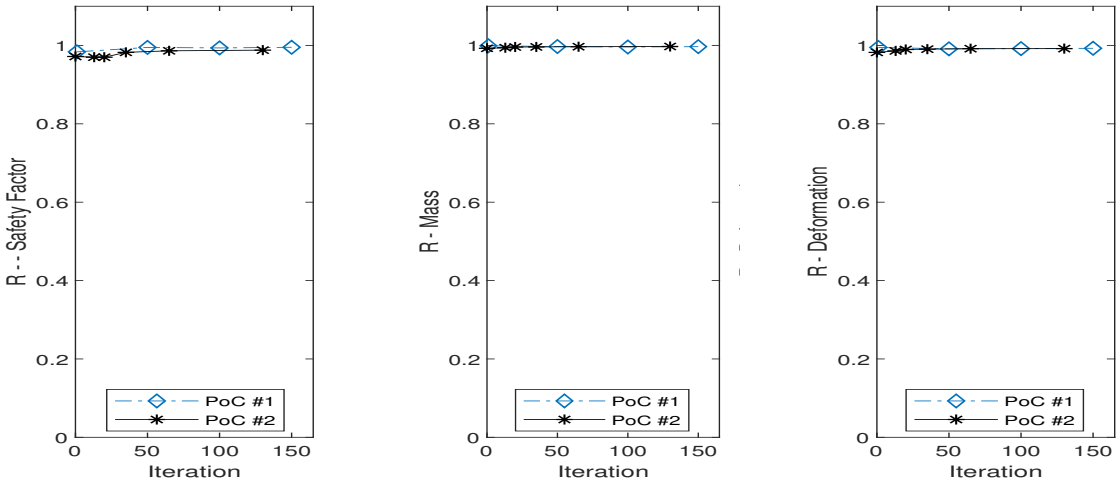


Figure 6.10 – Coffee Table - Cost Function Evaluations

That way, during the experiment, 40 simulations were executed to build the models of the optimization process, and 30 additional evaluations were also done to evaluate the candidate candidates solution, which indicates that the surrogate-based optimization only had access to the actual cost function 70 times. Nonetheless, the developed methodology was able to identify an accurate Pareto Front with less than 2% of the number of cost functions evaluations of the conventional method (70 out of 4500).

Figure 6.11 depicts the results regarding the *Local Accuracy* (*PoC #1*), and *Global Accuracy* (*PoC #2*), which are measures of coefficient of correlation (R) between the values of the actual function and the model's estimations, where values close to 1 indicates that the models outputs

good estimations. In here, a grid with one intermediate points between the dimensions boundary was used to measure the global accuracy, in which it was composed by 243 points.



(a) Safety Factor (b) Mass (c) Deformation

Figure 6.11 – Coffee Table - Proof Of Concept

As can be seen in Figure 6.11, throughout the optimization, the model was able to estimate all the cost functions with exceptional accuracy. Nonetheless, the measures of *Global Accuracy* reveals the effects of the *Global Improvement* of the *Model Update Strategy*, which graphically translates into improvement of the *PoC #2*'s measures.

Figure 6.12 depicts the disposition of the Pareto Set on each Design Variable in the domain, where the curve in red describe the nonparametric kernel-smoothing distribution of the data.

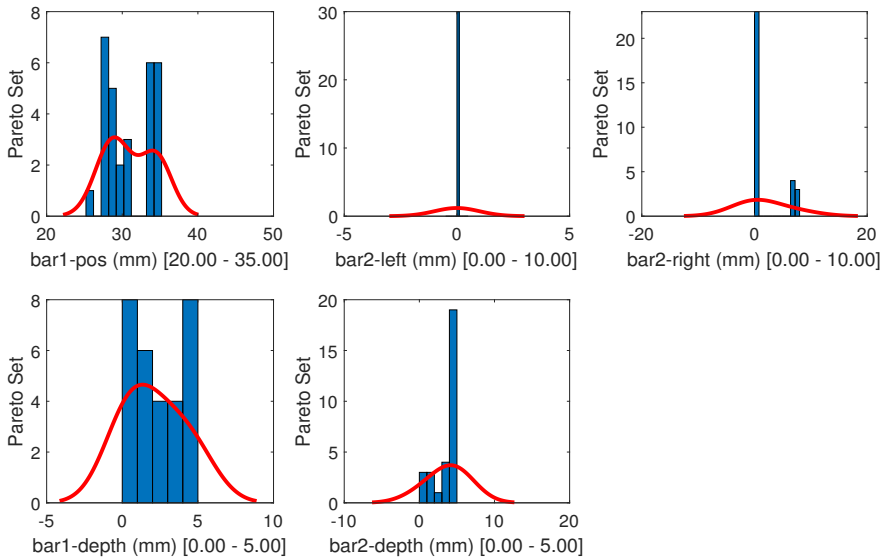


Figure 6.12 – Coffee Table - Pareto Set

As can be seen in Fig. 6.12, the design variables *bar2_left*, and *bar2_right* possess the same values for almost all solutions, which indicates that the Pareto Set resides in a particular region of the search space. Therefore, the five-dimension problem has only three meaningful design variables, which translates into a three-dimension problem that has only **576** possible configurations, which makes even feasible sweeping the entire search space of solutions.

6.3 EXOSKELETON KNEE COUPLING

In this final case study, the target structure is a critical component of an exoskeleton's knee join designed by the author in conjunction with the other researchers of the Laboratory of Embedded Systems and Integrated Circuits Applications (LEIA) from the University of Brasilia - Brazil. This piece in particular is responsible to connects the shaft of the gearbox to the moving parts of the exoskeleton. Therefore, its structure has to support the torque and loads of the coming from the exoskeleton and the its user.

6.3.1 Problem's Description

Figure 6.13 describes a simulation scenario built at the software *ANSYS version R19.1* to execute a *Static Structural*, wherein the image **A** indicates a bearing load of $1000N$ (approximately $100Kg$) at the central hole, **B** represents the actuation of an external supporting bearing, **C** indicates a bearing load of $250N$ (approximately $25Kg$) at each of the small holes, **D** a torque of $35Nm$ at the keyway (square hole), and **E** a torque of $35Nm$ at the small holes.

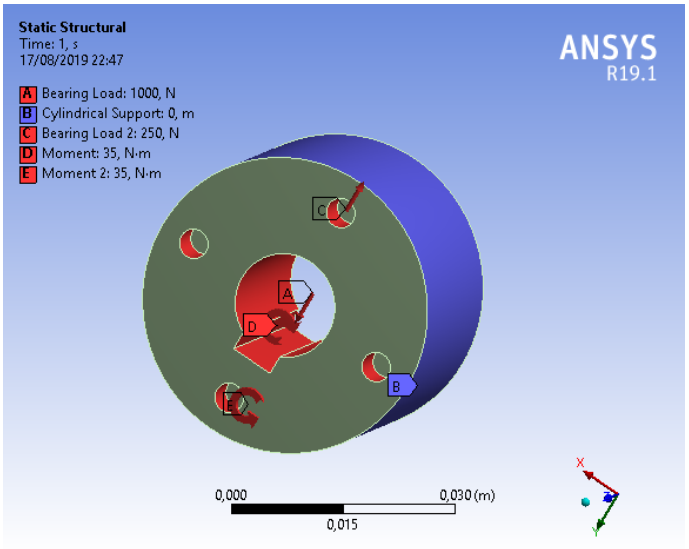


Figure 6.13 – Exoskeleton Knee Coupling - Simulation Scenario

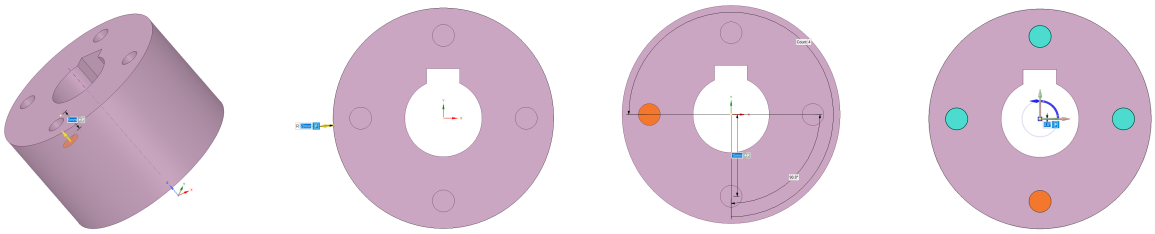
Table 6.7 holds the mechanical properties of the selected materials used in the case study, where the coffee table is made of brass.

Material	Density	Tensile Yield Strength	Young's Modulus	Poisson's Ratio
Structural Steel	$8150Kgm^{-3}$	$87.7MPa$	$96000Mpa$	0.345

Table 6.7 – Exoskeleton Knee Coupling - Engineering Data

6.3.2 Design Variables

This experiment was build into a four-dimension shape optimization problem, where five geometric features of the Coffee Table's structure were select as Design Variables: a) The *depth* describes, b) *outer_r*, c) *screw_r*, and d) *screw_ang*. The *depth* describes the depth of the fixing holes. The *outer_r* indicates the outer radius of the structure. The *screw_r* indicates the circumference's radius in which the holes are positioned. At last, the *screw_ang* describe the angle of aliment between the holes and the keyway.



(a) Design Variable: depth (b) Design Variable: outer_r (c) Design Variable: screw_r (d) Design Variable: screw_ang

Figure 6.14 – Exoskeleton Knee Coupling - Design Features

Table 6.8 describes the characteristics of domain and precision of each of the five design variables, which taking in consideration the precision of each of them results in a search space of **17920** possible solutions.

Design Variable	Inital Domain	Final Domain	Precision
depth	5 mm	20 mm	0
outer_r	18 mm	20 mm	0
screw_r	12 mm	15 mm	0
screw_ang	0°	89°	0

Table 6.8 – Exoskeleton Knee Coupling - Design Variables

6.3.3 Cost Functions

The cost functions of the optimization process are the *weight* and the *safety factor* on the knee coupling, which the first depends on its geometry while the second indicates the actual load-bearing capacity of a structure or component. Moreover, the information regarding *total deformation* on the coffee table feeds the *Validation* system.

In order to configure the surrogate-based optimization process, several simulations manually executed to determine the typical range of values of the cost functions, in which Tab. 6.9 holds

the identified values for the specified structure.

Cost Function	Lower Image's Value	Upper Image's Value
Safety Factor	3	9
Mass	$150g$	$230g$
Total Distortion	$0mm$	$0.003mm$

Table 6.9 – Exoskeleton Knee Coupling - Cost Functions' Range of Values

6.3.4 Results

In all instances of optimization process execution, the implemented solution run simulations on 5621 out of 17920 possible configurations of the problem, which would take approximately 106 hours to simulate **without occurring errors** in its execution. This measure of time only takes in consideration the execution time of each of the simulations which the minimum, mean, median and maximum execution time are 60.87 seconds, 67.48 seconds, 67.66 seconds, and 189.45 seconds respectively.

Figure 6.15 depicts the results of the surrogate-based optimization process, which the first-generation surrogate model was built with **800** training points and 10 additional filing points were added on each update call, and the periodic update curve *Mode D* prescribes 20 model updates.

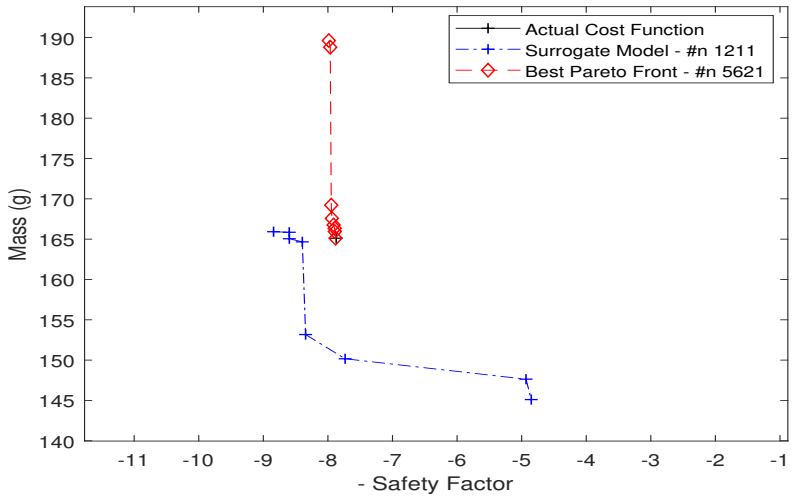


Figure 6.15 – Exoskeleton Knee Coupling - Pareto Front

That way, during the experiment, 1211 simulations were executed to build the models of the optimization process, and 30 additional evaluations were also done to evaluate the candidate candidates solution, which indicates that the surrogate-based optimization only had access to the actual cost function 1241 times. Nonetheless, unlike the previous examples, the developed methodology was not able to identify an accurate Pareto Front, even utilizing almost 7% of the number of possible configurations of the search space.

Figure 6.16 depicts when these cost functions evaluations happen during the execution of the optimization, and also indicates the amount of them that were already in the solution's database.

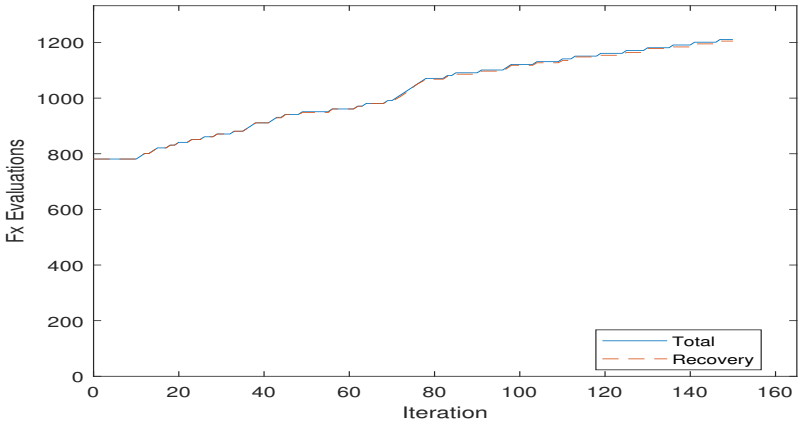


Figure 6.16 – Exoskeleton Knee Coupling - Cost Function Evaluations

Figure 6.17 depicts the results regarding the *Local Accuracy* (*PoC #1*), and *Global Accuracy* (*PoC #2*), which are measures of coefficient of correlation (R) between the values of the actual function and the model's estimations, where values close to 2 indicates that the models outputs good estimations. In here, a grid with one intermediate points between the dimensions boundary was used to measure the global accuracy, in which it was composed by 192 points.

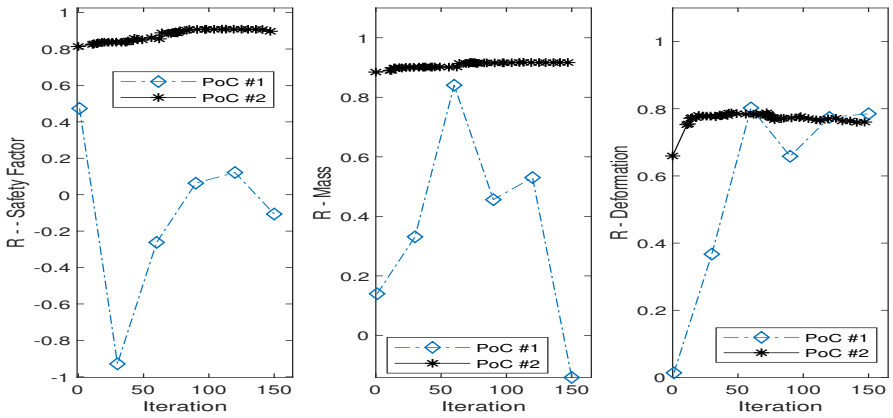


Figure 6.17 – Exoskeleton Knee Coupling - Proof Of Concept

As can be seen in Figure 6.17, throughout the optimization, the *Global Accuracy*'s measures indicate that the model had an acceptable generalization of all the cost functions with exceptional accuracy. Nonetheless, the measures of *Local Accuracy* reveals fails to estimate both cost functions, which resulted in the activation of several updates triggers during its execution.

Figure 6.18 depicts the disposition of the Pareto Set on each Design Variable in the domain, where the curve in red describe the nonparametric kernel-smoothing distribution of the data.

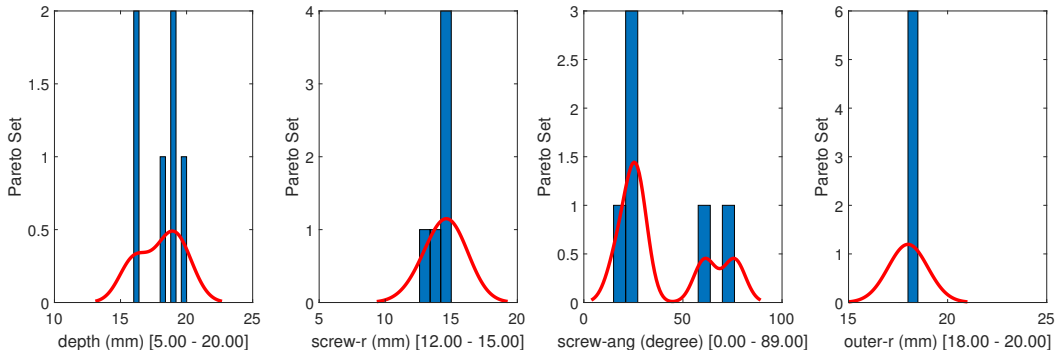


Figure 6.18 – Exoskeleton Knee Coupling - Pareto Set

As can be seen in Fig. 6.18, the optimization process was only able to locate six non-dominated solution. Nonetheless, due to the models' deficiencies, these results can not be used to draw any conclusion. Thus, this case study outputs an inconclusive outcome because the proposed methodology fails to build precise models for the cost function.

6.4 OVERVIEW

The cost functions at a structural optimization are the outputs of a series of simulations, which its simulation scenario changes dynamically. Nonetheless, during this work, it was possible to perceive that even utilizing the automatic setting procedures of the software Ansys, there was still the occurrence of errors that demanded the user interaction, which interrupts the operation flow of the optimization procedure.

Therefore, these experiments reveal that the complete execution of the optimization process is a challenge itself, due to thirty-party issues related to the execution of the simulations at the software Ansys.

Nonetheless, the method's database feature was fundamental to mitigate the effects of these external failures, because it enables an almost seamless recovery of the experiment in the occurrence of reboot or re-execution.

This chapter utilized three case studies to evidence the capabilities and limitations of the new surrogate-assisted optimization technique in conjunction with the space-filling algorithm, in which the case study regarding the Heatsink (see Chapter 6.1) and Coffee Table (see Chapter 6.2) shown that the method is capable of promoting an expressive performance gain reconciled with a small loss in accuracy, whereas the case study of Exoskeleton Knee Coupling (see Chapter 6.3) reveals that it could also fail to output satisfactory results.

Therefore, it is admissible to state that the new surrogate-assisted optimization technique is capable of achieving its proposed aims given reasonable conditions of operation.

7 CONCLUSION

This manuscript focused on the study and development of a new surrogate-assisted optimization technique. Here we have explored the different concepts in the literature and developed a series of procedures and auxiliary algorithms in order to implement our optimization algorithm, thus contributing to strengthening a relatively new field of research.

At the beginning of this work, we set out not only to develop the technique, but also, to understand the source of the problem, namely the *Design Process*, and how it becomes an optimization problem. Our approach was built in a bottom-Up format, describing each of the steps that lead to the main topic of this work: surrogate-assisted optimization algorithm.

Our technique develops a novel management strategy, which it is capable of actively updating the surrogate models based on violations of the expected cost functions' range of values. Indeed we noticed from our case studies that our management strategy was able to improve the accuracy of low-efficiency models, which enables the utilization of a small initial population to build the surrogate models.

One particularity of our implementation is that we have also introduced a mechanism that periodically tries to recuperate suitable solutions that were previously discarded due to the model's deficient. That happens because the early models have lower trustworthy due to their corresponding amount of training points.

Complementary to improvements of the optimization algorithm itself, three new space-filling algorithms were created to assist in the process responsible for updating the surrogate model. These algorithms identify a set of additional training points that would best improve the coverage of the model while taking into consideration the position of the points that already compose an already existing surrogate model. More so, we also were able to develop a space-filling metric to measure the presence of gaps in a population of points within a search space, which enables the tracking of the coverage capabilities of the surrogate model.

Finally, three case studies were used to verify the new surrogate-assisted optimization technique in conjunction with the space-filling algorithm, which has shown that our solution is capable of promoting a significant performance gain at the cost of a small accuracy decrease. Nonetheless, it also reveals that it could also fail to output satisfactory results. Therefore, we believe that our approach is on the right track, which indicates that it is indeed possible to represent the dynamics of cost functions with a small set of points but does not apply to all kind of problems.

So with this notion, we also believe the foundation provided in this manuscript opens up the possibilities of several subsequent research with surrogates model to solve complex problems, no restricting to the context of structural optimization.

7.1 FUTURE WORK

Although we were able to describe a new surrogate-assisted optimization approach to create three space-filling algorithms, and to validate its functionality in three distinct mechanical structural design's problems, there are still several aspects that can be done to improve and extend this research.

As for now, the current implementation still has dependencies of the problem at hand, namely the process responsible for the identification of cost functions' range of values. Therefore, the implementation of an automated, and adaptive process of identification of cost functions' range of values would enable the utilization of the technique regardless of the target problem.

Moreover, improvements would also be possible at update triggers mechanism, as seen in the case study of the Heatsink and Coffee Table there are problems that a relatively small number of training points are sufficient to model the cost functions. Nonetheless, the violations-based trigger mechanism is only capable of indicating the necessity of updates, but not unnecessariness.

Although the initial proposal utilizes the justification of loss of useful information to prevent the utilization of a validation population in order to measure the models' accuracy, nonetheless, such a mechanism could prevent the execution of periodic updates on surrogate models that possess reasonable accuracy. Moreover, these validation population could utilize the evaluations points used to identify the range of values of the cost functions.

At last, as shown in the case study of Exoskeleton Knee Coupling, creating a single model for the entire domain is not always the best-indicated method to build the problem's model. A mechanism that tries different manners to build the surrogate models would complement this deficiency of the algorithm (130, 91, 90).

BIBLIOGRAPHIC REFERENCES

1 CHRISTENSEN, P. W.; KLARBRING, A. *An introduction to structural optimization*. [S.l.]: Springer Science & Business Media, 2008. v. 153.

2 JIN, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, Elsevier, v. 1, n. 2, p. 61–70, 2011.

3 REGIS, R. G. Particle swarm with radial basis function surrogates for expensive black-box optimization. *Journal of Computational Science*, Elsevier, v. 5, n. 1, p. 12–23, 2014.

4 MÜLLER, J.; SHOEMAKER, C. A.; PICHÉ, R. So-mi: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers & Operations Research*, Elsevier, v. 40, n. 5, p. 1383–1400, 2013.

5 WANG, C.; DING, J.; CHENG, R.; LIU, C.; CHAI, T. Data-driven surrogate-assisted multi-objective optimization of complex beneficiation operational process. *IFAC-PapersOnLine*, Elsevier, v. 50, n. 1, p. 14982–14987, 2017.

6 HO-HUU, V.; NGUYEN-THOI, T.; VO-DUY, T.; NGUYEN-TRANG, T. An adaptive elitist differential evolution for optimization of truss structures with discrete design variables. *Computers & Structures*, Elsevier, v. 165, p. 59–75, 2016.

7 CAI, X.; GAO, L.; LI, X.; QIU, H. Surrogate-guided differential evolution algorithm for high dimensional expensive problems. *Swarm and Evolutionary Computation*, Elsevier, v. 48, p. 288–311, 2019.

8 BHOSEKAR, A.; IERAPETRITOU, M. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, Elsevier, v. 108, p. 250–267, 2018.

9 PASSOS, J. C.; SANTOS, C. E.; SAMPAIO, R. C.; COELHO, L. S.; LLANOS, C. H. Sizing optimization of an exoskeleton structure utilizing finite element analysis and multi-objective search. *CBA - Congresso Brasileiro de Automática, 2018*, v. 1, p. 1–8, 2018.

10 SIMPSON, T.; TOROPOV, V.; BALABANOV, V.; VIANA, F. Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come-or not. In: *12th AIAA/ISSMO multidisciplinary analysis and optimization conference*. [S.l.: s.n.], 2008. p. 5802.

11 QUEIPO, N. V.; HAFTKA, R. T.; SHYY, W.; GOEL, T.; VAIDYANATHAN, R.; TUCKER, P. K. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, Elsevier, v. 41, n. 1, p. 1–28, 2005.

12 FORRESTER, A. I.; KEANE, A. J. Recent advances in surrogate-based optimization. *Progress in aerospace sciences*, Elsevier, v. 45, n. 1-3, p. 50–79, 2009.

13 SER, J. D.; OSABA, E.; MOLINA, D.; YANG, X.-S.; SALCEDO-SANZ, S.; CAMACHO, D.; DAS, S.; SUGANTHAN, P. N.; COELLO, C. A. C.; HERRERA, F. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*, Elsevier, v. 48, p. 220–250, 2019.

14 ALEXANDROV, N.; LEWIS, R.; GUMBERT, C.; GREEN, L.; NEWMAN, P. Optimization with variable-fidelity models applied to wing design. In: *38th aerospace sciences meeting and exhibit*. [S.l.: s.n.], 2000. p. 841.

15 SIMPSON, T. W.; MAUERY, T. M.; KORTE, J. J.; MISTREE, F. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA journal*, v. 39, n. 12, p. 2233–2241, 2001.

16 SINGH, G.; GRANDHI, R. V. Mixed-variable optimization strategy employing multifidelity simulation and surrogate models. *AIAA journal*, v. 48, n. 1, p. 215–223, 2010.

17 WESSING, S.; RUDOLPH, G.; TURCK, S.; KLIMMEK, C.; SCHÄFER, S. C.; SCHNEIDER, M.; LEHMANN, U. Replacing fea for sheet metal forming by surrogate modeling. *Cogent Engineering*, Taylor & Francis, v. 1, n. 1, p. 950853, 2014.

18 GONG, W.; DUAN, Q. An adaptive surrogate modeling-based sampling strategy for parameter optimization and distribution estimation (asmo-pode). *Environmental modelling & software*, Elsevier, v. 95, p. 61–75, 2017.

19 VOUTCHKOV, I.; KEANE, A. Multi-objective optimization using surrogates. In: *Computational Intelligence in Optimization*. [S.l.]: Springer, 2010. p. 155–175.

20 ULLMAN, D. G. *The mechanical design process*. [S.l.]: McGraw-Hill New York, 1992. v. 2.

21 BUDDIES, S. *The Engineering Design Process*. 2018. <https://www.nasa.gov/audience/foreducators/best/edp.html>. [Online; accessed 19-July-2019].

22 LIU, L. *Failure in the Design Process*. 2018. <https://uxdesign.cc/failure-in-the-design-process-baed825f1f48>. [Online; accessed 19-July-2019].

23 DUNBAR, B.; MAY, S. *NASA BEST - Engineering Design Process*. 2018. <https://www.nasa.gov/audience/foreducators/best/edp.html>. [Online; accessed 19-July-2019].

24 DUAN, Q.; SOROOSHIAN, S.; GUPTA, V. K. Optimal use of the sce-ua global optimization method for calibrating watershed models. *Journal of hydrology*, Elsevier, v. 158, n. 3-4, p. 265–284, 1994.

25 COYETTE, A.; KIEFFER, S.; VANDERDONCKT, J. Multi-fidelity prototyping of user interfaces. In: SPRINGER. *IFIP Conference on Human-Computer Interaction*. [S.l.], 2007. p. 150–164.

26 PHAM, D. T.; GAULT, R. S. A comparison of rapid prototyping technologies. *International Journal of machine tools and manufacture*, Elsevier, v. 38, n. 10-11, p. 1257–1287, 1998.

27 VIRZI, R. A. What can you learn from a low-fidelity prototype? In: SAGE PUBLICATIONS SAGE CA: LOS ANGELES, CA. *Proceedings of the Human Factors Society Annual Meeting*. [S.l.], 1989. v. 33, n. 4, p. 224–228.

28 SAUER, J.; SEIBEL, K.; RÜTTINGER, B. The influence of user expertise and prototype fidelity in usability tests. *Applied ergonomics*, Elsevier, v. 41, n. 1, p. 130–140, 2010.

29 MCCURDY, M.; CONNORS, C.; PYRZAK, G.; KANEFSKY, B.; VERA, A. Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success. In: ACM. *Proceedings of the SIGCHI conference on Human Factors in computing systems*. [S.l.], 2006. p. 1233–1242.

30 SCHINDLER, M.; EPPLER, M. J. Harvesting project knowledge: a review of project learning methods and success factors. *International journal of project management*, Elsevier, v. 21, n. 3, p. 219–228, 2003.

31 CHARVAT, J. *Project management methodologies: selecting, implementing, and supporting methodologies and processes for projects*. [S.l.]: John Wiley & Sons, 2003.

32 WEEBER, K.; HOOLE, S. R. H. Geometric parametrization and constrained optimization techniques in the design of salient pole synchronous machines. *IEEE transactions on magnetics*, IEEE, v. 28, n. 4, p. 1948–1960, 1992.

33 COTTRELL, J. A.; HUGHES, T. J.; BAZILEVS, Y. *Isogeometric analysis: toward integration of CAD and FEA*. [S.l.]: John Wiley & Sons, 2009.

34 KRULL, F. N. The origin of computer graphics within general motors. *IEEE Annals of the History of Computing*, IEEE, n. 3, p. 40–56, 1994.

35 FIELD, D. A. Education and training for cad in the auto industry. *Computer-Aided Design*, Elsevier, v. 36, n. 14, p. 1431–1437, 2004.

36 KASIK, D. J. Viewing the future of cad. *IEEE Computer Graphics and Applications*, IEEE, v. 20, n. 1, p. 34–35, 2000.

37 LLEWELYN, A. Review of cad/cam. *Computer-Aided Design*, Elsevier, v. 21, n. 5, p. 297–302, 1989.

38 GOODACRE, C. J.; GARBACEA, A.; NAYLOR, W. P.; DAHER, T.; MARCHACK, C. B.; LOWRY, J. Cad/cam fabricated complete dentures: concepts and clinical methods of obtaining required morphological data. *The Journal of prosthetic dentistry*, Elsevier, v. 107, n. 1, p. 34–46, 2012.

39 AOUAD, G.; WU, S.; LEE, A.; ONYENOBI, T. *Computer aided design guide for architecture, engineering and construction*. [S.l.]: Routledge, 2013.

40 MÜLLER, T. E.; KLASHORST, E. v. d. A quantitative comparison between size, shape, topology and simultaneous optimization for truss structures. *Latin American Journal of Solids and Structures*, SciELO Brasil, v. 14, n. 12, p. 2221–2242, 2017.

41 COELLO, C. A. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. et al. *Evolutionary algorithms for solving multi-objective problems*. [S.l.]: Springer, 2007. v. 5.

42 KHUNKITTI, S.; WATSON, N. R.; CHATTHAWORN, R.; PREMRUDEEPREECHACHARN, S.; SIRITARATIWAT, A. An improved da-pso optimization approach for unit commitment problem. *Energies*, Multidisciplinary Digital Publishing Institute, v. 12, n. 12, p. 2335, 2019.

43 MATLOCK, K.; BERLOW, N.; KELLER, C.; PAL, R. Combination therapy design for maximizing sensitivity and minimizing toxicity. *BMC bioinformatics*, BioMed Central, v. 18, n. 4, p. 116, 2017.

44 DOERNER, K.; GUTJAHR, W. J.; HARTL, R. F.; STRAUSS, C.; STUMMER, C. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of operations research*, Springer, v. 131, n. 1-4, p. 79–99, 2004.

45 BAUMGARTNER, U.; MAGELE, C.; RENHART, W. Pareto optimality and particle swarm optimization. *IEEE Transactions on magnetics*, IEEE, v. 40, n. 2, p. 1172–1175, 2004.

46 BENSON, H. P. *Multicriteria Optimization, Matthias Ehrgott, Springer (2005), 323 pages, ISBN: 3-540-21398-8*. [S.l.]: North-Holland, 2007.

47 VOUTCHKOV, I.; KEANE, A. Multiobjective optimization using surrogates. 2006.

48 KOLAR, J. W.; BIELA, J.; MINIBOCK, J. Exploring the pareto front of multi-objective single-phase pfc rectifier design optimization-99.2% efficiency vs. 7kw/din 3 power density. In: IEEE. *2009 IEEE 6th International Power Electronics and Motion Control Conference*. [S.l.], 2009. p. 1–21.

49 WALL, W. A.; FRENZEL, M. A.; CYRON, C. Isogeometric structural shape optimization. *Computer methods in applied mechanics and engineering*, Elsevier, v. 197, n. 33-40, p. 2976–2988, 2008.

50 BENDSØE, M. P.; SIGMUND, O. *Optimization of structural topology, shape, and material*. [S.l.]: Springer, 1995. v. 414.

51 PAINCHAUD-OUELLET, S.; TRIBES, C.; TRÉPANIER, J.-Y.; PELLETIER, D. Airfoil shape optimization using a nonuniform rational b-splines parametrization under thickness constraint. *AIAA journal*, v. 44, n. 10, p. 2170–2178, 2006.

52 MANH, N. D.; EVGRAFOV, A.; GERSBORG, A. R.; GRAVESEN, J. Isogeometric shape optimization of vibrating membranes. *Computer Methods in Applied Mechanics and Engineering*, Elsevier, v. 200, n. 13-16, p. 1343–1353, 2011.

53 LIU, G.-R.; QUEK, S. S. *The finite element method: a practical course*. [S.l.]: Butterworth-Heinemann, 2013.

54 DING, Y. Shape optimization of structures: a literature survey. *Computers & Structures*, Elsevier, v. 24, n. 6, p. 985–1004, 1986.

55 XIE, Y. M.; STEVEN, G. P. Basic evolutionary structural optimization. In: *Evolutionary structural optimization*. [S.l.]: Springer, 1997. p. 12–29.

56 FOURIE, P.; GROENWOLD, A. A. The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization*, Springer, v. 23, n. 4, p. 259–267, 2002.

57 SHOJAEE, S.; MOHAMMADIAN, M. Structural topology optimization using an enhanced level set method. *Scientia Iranica*, Elsevier, v. 19, n. 5, p. 1157–1167, 2012.

58 PICELLI, R.; TOWNSEND, S.; BRAMPTON, C.; NORATO, J.; KIM, H. Stress-based shape and topology optimization with the level set method. *Computer Methods in Applied Mechanics and Engineering*, Elsevier, v. 329, p. 1–23, 2018.

59 SUN, S.; YU, T.; NGUYEN, T. T.; ATROSHCHENKO, E.; BUI, T. Structural shape optimization by igabem and particle swarm optimization algorithm. *Engineering Analysis with Boundary Elements*, Elsevier, v. 88, p. 26–40, 2018.

60 HUNTER, P.; PULLAN, A. Fem/bem notes. *Department of Engineering Science, The University of Auckland, New Zeland*, 2001.

61 TOHAMY, A. S. *Critical Shear Stresses for Webs of Plate Girder Subjected to Shear Loading and Contained Cut-outs*. 172 p. Dissertação (Mestrado) — Minia University, 06 2015.

62 FARMAGA, I.; SHMIGELSKYI, P.; SPIEWAK, P.; CIUPINSKI, L. Evaluation of computational complexity of finite element analysis. In: IEEE. *2011 11th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*. [S.l.], 2011. p. 213–214.

63 KURZYDŁOWSKI, K.; LOBUR, M.; FARMAGA, I.; MATVIYKIV, O. Data-processing method for determination thermophysical parameters of composite materials. In: IEEE. *2010 Proceedings of VIth International Conference on Perspective Technologies and Methods in MEMS Design*. [S.l.], 2010. p. 264–266.

64 ZHOU, Z.; ONG, Y. S.; LIM, M. H.; LEE, B. S. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Computing*, Springer, v. 11, n. 10, p. 957–971, 2007.

65 LEIFSSON, L.; KOZIEL, S.; TESFAHUNEGN, Y. A. Multiobjective aerodynamic optimization by variable-fidelity models and response surface surrogates. *AIAA Journal*, American Institute of Aeronautics and Astronautics, v. 54, n. 2, p. 531–541, 2015.

66 HAFTKA, R. T.; GRANDHI, R. V. Structural shape optimization—a survey. *Computer methods in applied mechanics and engineering*, Elsevier, v. 57, n. 1, p. 91–106, 1986.

67 DAOUD, F.; FIRL, M.; BLETZINGER, K. Filter techniques in shape optimization with cad-free parametrization. In: *Proceedings of 6th World Congress of Structural and Multidisciplinary Optimization*. [S.l.: s.n.], 2005.

68 FIRL, M.; WÜCHNER, R.; BLETZINGER, K.-U. Regularization of shape optimization problems using fe-based parametrization. *Structural and Multidisciplinary Optimization*, Springer, v. 47, n. 4, p. 507–521, 2013.

69 CHEN, J.; SHAPIRO, V.; SURESH, K.; TSUKANOV, I. Parametric and topological control in shape optimization. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. [S.l.], 2006. p. 575–586.

70 IMAM, M. H. Three-dimensional shape optimization. *International Journal for Numerical Methods in Engineering*, Wiley Online Library, v. 18, n. 5, p. 661–673, 1982.

71 HAN, X.; ZINGG, D. W. An adaptive geometry parametrization for aerodynamic shape optimization. *Optimization and Engineering*, Springer, v. 15, n. 1, p. 69–91, 2014.

72 BENDSØE, M. P. Optimal shape design as a material distribution problem. *Structural optimization*, Springer, v. 1, n. 4, p. 193–202, 1989.

73 ALLAIRE, G.; JOUVE, F.; TOADER, A.-M. A level-set method for shape optimization. *Comptes Rendus Mathematique*, Elsevier, v. 334, n. 12, p. 1125–1130, 2002.

74 LI, H.; LI, P.; GAO, L.; ZHANG, L.; WU, T. A level set method for topological shape optimization of 3d structures with extrusion constraints. *Computer Methods in Applied Mechanics and Engineering*, Elsevier, v. 283, p. 615–635, 2015.

75 LARSSON, R. Methodology for topology and shape optimization: Application to a rear lower control arm. *Department of Applied Mechanics CHALMERS UNIVERSITY OF TECHNOLOGY, Goteborg*, 2016.

76 FRANS, R.; ARFIADI, Y. Sizing, shape, and topology optimizations of roof trusses using hybrid genetic algorithms. *Procedia Engineering*, Elsevier, v. 95, p. 185–195, 2014.

77 SAVSANI, V. J.; TEJANI, G. G.; PATEL, V. K.; SAVSANI, P. Modified meta-heuristics using random mutation for truss topology optimization with static and dynamic constraints. *Journal of Computational Design and Engineering*, Elsevier, v. 4, n. 2, p. 106–130, 2017.

78 ROZVANY, G. I.; ZHOU, M.; BIRKER, T. Generalized shape optimization without homogenization. *Structural optimization*, Springer, v. 4, n. 3-4, p. 250–252, 1992.

79 MAUTE, K.; RAMM, E. Adaptive topology optimization. *Structural optimization*, Springer, v. 10, n. 2, p. 100–112, 1995.

80 OLHOFF, N.; BENDSØE, M. P.; RASMUSSEN, J. On cad-integrated structural topology and design optimization. *Computer Methods in Applied Mechanics and Engineering*, Elsevier, v. 89, n. 1-3, p. 259–279, 1991.

81 HINTON, E.; SIENZ, J. *Aspects of adaptive finite element analysis and structural optimization*. [S.l.]: University College of Swansea, Department of Civil Engineering, 1994.

82 LIANG, Q. Q.; STEVEN, G. P. A performance-based optimization method for topology design of continuum structures with mean compliance constraints. *Computer methods in applied mechanics and engineering*, Elsevier, v. 191, n. 13-14, p. 1471–1489, 2002.

83 OSIO, I. G.; AMON, C. H. An engineering design methodology with multistage bayesian surrogates and optimal sampling. *Research in Engineering Design*, Springer, v. 8, n. 4, p. 189–206, 1996.

84 YONDO, R.; ANDRES, E.; VALERO, E. A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses. *Progress in Aerospace Sciences*, Elsevier, v. 96, p. 23–61, 2018.

85 XU, Q.; WEHRLE, E.; BAIER, H. Knowledge-based surrogate modeling in engineering design optimization. In: *Surrogate-Based Modeling and Optimization*. [S.l.]: Springer, 2013. p. 313–336.

86 EDWARDS, A. L. *An introduction to linear regression and correlation*. [S.l.], 1984.

87 ELDRED, M.; DUNLAVY, D. Formulations for surrogate-based optimization with data fit, multifidelity, and reduced-order models. In: *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. [S.l.: s.n.], 2006. p. 7117.

88 LEIFSSON, L.; KOZIEL, S. *Simulation-driven aerodynamic design using variable-fidelity models*. [S.l.]: World Scientific, 2015.

89 BENNER, P.; FREUND, R. W.; SORENSEN, D. C.; VARGA, A. *Special issue on "order reduction of large-scale systems"*. [S.l.]: North-Holland, 2006.

90 SHAN, S.; WANG, G. G. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, Springer, v. 41, n. 2, p. 219–241, 2010.

91 VIANA, F. A.; HAFTKA, R. T.; STEFFEN, V. Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. *Structural and Multidisciplinary Optimization*, Springer, v. 39, n. 4, p. 439–457, 2009.

92 GOEL, T.; HAFTKA, R. T.; SHYY, W.; QUEIPO, N. V. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, Springer, v. 33, n. 3, p. 199–216, 2007.

93 JIN, R.; CHEN, W.; SIMPSON, T. W. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and multidisciplinary optimization*, Springer, v. 23, n. 1, p. 1–13, 2001.

94 MULLUR, A. A.; MESSAC, A. Metamodeling using extended radial basis functions: a comparative approach. *Engineering with Computers*, Springer, v. 21, n. 3, p. 203, 2006.

95 DÍAZ-MANRÍQUEZ, A.; TOSCANO, G.; COELLO, C. A. C. Comparison of metamodeling techniques in evolutionary algorithms. *Soft Computing*, Springer, v. 21, n. 19, p. 5647–5663, 2017.

96 POWELL, M. J. The theory of radial basis function approximation in 1990. *Advances in numerical analysis*, Clarendon Press, p. 105–210, 1992.

97 BOUHLEL, M. A.; BARTOLI, N.; OTSMANE, A.; MORLIER, J. Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, Springer, v. 53, n. 5, p. 935–952, 2016.

98 ACOSTA, F. M. A. Radial basis function and related models: an overview. *Signal Processing*, Elsevier, v. 45, n. 1, p. 37–58, 1995.

99 BJÖRKMAN, M.; HOLMSTRÖM, K. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, Springer, v. 1, n. 4, p. 373–397, 2000.

100 REGIS, R. G. Multi-objective constrained black-box optimization using radial basis function surrogates. *Journal of computational science*, Elsevier, v. 16, p. 140–155, 2016.

101 HOU, H.; ANDREWS, H. Cubic splines for image interpolation and digital filtering. *IEEE Transactions on acoustics, speech, and signal processing*, IEEE, v. 26, n. 6, p. 508–517, 1978.

102 HARDER, R. L.; DESMARAIS, R. N. Interpolation using surface splines. *Journal of aircraft*, v. 9, n. 2, p. 189–191, 1972.

103 LOWE, D. Multi-variable functional interpolation and adaptive networks. *Complex Systems*, v. 2, p. 321–355.

104 HARDY, R. L. Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research*, Wiley Online Library, v. 76, n. 8, p. 1905–1915, 1971.

105 BAGHERI, S.; KONEN, W.; EMMERICH, M.; BÄCK, T. Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing*, Elsevier, v. 61, p. 377–393, 2017.

106 SCHUËLLER, G.; PRADLWARTER, H.; KOUTSOURELAKIS, P.-S. A critical appraisal of reliability estimation procedures for high dimensions. *Probabilistic engineering mechanics*, Elsevier, v. 19, n. 4, p. 463–474, 2004.

107 BOURINET, J.-M.; DEHEEGER, F.; LEMAIRE, M. Assessing small failure probabilities by combined subset simulation and support vector machines. *Structural Safety*, Elsevier, v. 33, n. 6, p. 343–353, 2011.

108 KOZIEL, S.; LEIFSSON, L. Scaling properties of multi-fidelity shape optimization algorithms. *Procedia Computer Science*, Elsevier, v. 9, p. 832–841, 2012.

109 PAPALAMBROS, P. Y.; MICHELENA, N. F. Trends and challenges in system design optimization. In: *Proceedings of the International Workshop on Multidisciplinary Optimization*. [S.l.: s.n.], 2000. p. 1–15.

110 PANOSSIAN, H. Richard bellman and stochastic control systems. *Computers & Mathematics with Applications*, Pergamon, v. 12, n. 6, p. 825–829, 1986.

111 GASPAR, B.; TEIXEIRA, A.; SOARES, C. G. Assessment of the efficiency of kriging surrogate models for structural reliability analysis. *Probabilistic Engineering Mechanics*, Elsevier, v. 37, p. 24–34, 2014.

112 WANG, H.; JIN, Y.; YAO, X. Diversity assessment in many-objective optimization. *IEEE transactions on cybernetics*, IEEE, v. 47, n. 6, p. 1510–1522, 2017.

113 CAWLEY, G. C.; TALBOT, N. L. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, v. 11, n. Jul, p. 2079–2107, 2010.

114 REGIS, R. G.; SHOEMAKER, C. A. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE transactions on evolutionary computation*, IEEE, v. 8, n. 5, p. 490–505, 2004.

115 ZHAO, Z.; YANG, J.; HU, Z.; CHE, H. A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric latin hypercube design for unconstrained optimization problems. *European Journal of Operational Research*, Elsevier, v. 250, n. 1, p. 30–45, 2016.

116 HE, J.; YAO, X. From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 6, n. 5, p. 495–511, 2002.

117 CHEN, T.; TANG, K.; CHEN, G.; YAO, X. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, Elsevier, v. 436, p. 54–70, 2012.

118 KENNY, Q. Y.; LI, W.; SUDJIANTO, A. Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of statistical planning and inference*, Elsevier, v. 90, n. 1, p. 145–159, 2000.

119 AGGARWAL, C. C.; HINNEBURG, A.; KEIM, D. A. On the surprising behavior of distance metrics in high dimensional space. In: SPRINGER. *International conference on database theory*. [S.l.], 2001. p. 420–434.

120 MORGAN, R.; GALLAGHER, M. Sampling techniques and distance metrics in high dimensional continuous landscape analysis: Limitations and improvements. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 18, n. 3, p. 456–461, 2013.

121 JARA, E. C. Multi-objective optimization by using evolutionary algorithms: The p-optimality criteria. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 18, n. 2, p. 167–179, 2013.

122 WANG, H.; JIAO, L.; YAO, X. Two_arch2: An improved two-archive algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 19, n. 4, p. 524–541, 2014.

123 RIGET, J.; VESTERSTRØM, J. S. A diversity-guided particle swarm optimizer-the arpso. *Dept. Comput. Sci., Univ. of Aarhus, Aarhus, Denmark, Tech. Rep*, v. 2, p. 2002, 2002.

124 COTTER, N. E. *DELAUNAY TRIANGULATION - Sphere center point*. 2010. <https://www.ece.utah.edu/eceCTools/Triangulation/TriangulationSphereCntr.htm>. [Online; accessed 19-July-2019].

125 JALURIA, Y. *Design and optimization of thermal systems*. [S.l.]: CRC press, 2007.

126 SAUCIUC, L.; CHRYSLER, G.; MAHAJAN, R.; SZLEPER, M. Air-cooling extension-performance limits for processor cooling applications. In: IEEE. *Ninteenth Annual IEEE Semiconductor Thermal Measurement and Management Symposium, 2003.* [S.l.], 2003. p. 74–81.

127 MAHAJAN, R.; CHIU, C.-p.; CHRYSLER, G. Cooling a microprocessor chip. *Proceedings of the IEEE*, IEEE, v. 94, n. 8, p. 1476–1486, 2006.

128 GODEFROY, D. *Fischer Elektronik heatsink SK47*. 2018. <https://grabcad.com/library/fischer-elektronik-heatsink-sk47-1>. [Online; accessed 19-July-2019].

129 YELER, Y. *Coffee Table - Sehpa*. 2018. <https://grabcad.com/library/coffee-table-sehpa-1>. [Online; accessed 12-August-2019].

130 LEARY, S. J.; BHASKAR, A.; KEANE, A. J. A constraint mapping approach to the structural optimization of an expensive model using surrogates. *Optimization and Engineering*, Springer, v. 2, n. 4, p. 385–398, 2001.