National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

Use of Decision Tree-Based Models in the Evolutionary Algorithm Optimization – Applications in Aerodynamics

Diploma Thesis

Konstantinos Chondros

Supervisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2023

Acknowledgements

I would like to express my deepest gratitude to my supervisor, professor Kyriakos C. Giannakoglou, for his invaluable guidance and support throughout my Diploma Thesis journey. I am truly fortunate to have had the opportunity to work under his supervision and I am grateful for his mentorship and the knowledge I have gained.

Secondly, I would like to extend my sincere appreciation to the entire PCOpt/NTUA team for the support I have received. I would like to express my gratitude to Marina Kontou for the collaboration, assistance and valuable discussions, and to Dr. Varvara Asouti for her support.

Lastly, I would like to thank my family for always supporting me and believing in me throughout my studies. I am deeply grateful to Marios, Thodoris, and all my friends who have accompanied me on this journey, creating unforgettable memories along the way.

National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Parallel CFD & Optimization Unit

Use of Decision Tree-Based Models in the Evolutionary Algorithm Optimization – Applications in Aerodynamics

Diploma Thesis

Konstantinos Chondros

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2023

Abstract

This Diploma Thesis explores the integration of Decision Tree-based models into the shape optimization process, as used in fluid mechanics, supported by Evolutionary Algorithms, in order to either get better solutions with the same cost or reduce the cost for the same solution quality. Three different approaches are investigated: Tree-Based Regression Models as off-line trained metamodels, constraint-based classification Tree-Based Models and design space exploration with Decision Trees.

Two shape optimization cases are used to assess the performance of the proposed methods. The first case is an isolated airfoil with the objective to minimize the drag coefficient, with the constraint on the lift coefficient. The second case is an S-Bend duct designed for minimum total pressure losses, with a constraint on the volume of the duct. Both cases are parameterized using NURBS lattices. Latin Hypercube and Random Sampling were combined to generate a number of different geometries within specific limits. The flow is solved numerically using the PUMA software for each new geometry, yielding the Database used for the training.

In the off-line metamodel approach, Tree-Based regression models are trained on the Database, with the aim to predict objectives and constraints at low cost. These are used, instead of other off-line trained metamodels, within an Evolutionary Algorithm.

In the second approach, class labels are assigned to each Database entry depending on its constraint value. Thus, entries are classified as feasible/infeasible, with two gray zones in between them. Tree-Based classification models are trained on the Database, with the aim to predict the class label of an unseen individual. The models are employed in each generation of the optimization, classifying individuals prior to their evaluation on the CFD software. In this way, individuals assigned certain class labels, which do not fulfill the constraint, are not evaluated on the

CFD tool, avoiding expensive evaluations for non-feasible solutions.

In the third approach, a Decision Tree is used to partition the design space into different subregions. By selecting the subregion containing the optimal solution and initiating the optimization procedure into this subregion, the optimal solution could be found faster. Various approaches for region selection are tested.

The second and third approaches are, also, combined in order to assess the performance of the combination. All the results are compared with the EA and MAEA (assisted by on-line trained RBF networks) optimization methods implemented in the EASY software.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

Χρήση Μοντέλων Βασισμένων στα Δέντρα Αποφάσεων στη Βελτιστοποίηση μέσω Εξελικτικών Αλγορίθμων – Εφαρμογές στην Αεροδυναμική

Διπλωματική Εργασία

Κωνσταντίνος Χονδρός

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2023

Η Διπλωματική αυτή Εργασία διερευνά την ενσωμάτωση μοντέλων βασισμένων στα Δέντρα Αποφάσεων στη διαδικασία βελτιστοποίησης σχήματος μέσω Εξελικτικών Αλγορίθμων, όπως χρησιμοποιείται στη ρευστομηχανική, προκειμένου είτε να βρεθούν καλύτερες λύσεις με το ίδιο υπολογιστικό κόστος είτε να μειωθεί το κόστος για την ίδια ποιότητα λύσης. Εξετάζονται τρεις προσεγγίσεις: μοντέλα παλινδρόμησης βασισμένα στα Δέντρα Αποφάσεων ως off-line μεταμοντέλα, χρήση μοντέλων βασισμένων στα Δέντρα Αποφάσεων για ταξινόμηση με κριτήριο την ικανοποίηση ή όχι των περιορισμών και εξερεύνηση του χώρου σχεδιασμού με Δέντρα Αποφάσεων.

Χρησιμοποιούνται δύο περιπτώσεις βελτιστοποίησης σχήματος για την αξιολόγηση της απόδοσης των προτεινόμενων μεθόδων. Η πρώτη περίπτωση είναι μια αεροτομή με στόχο την ελαχιστοποίηση του συντελεστή οπισθέλκουσας, με περιορισμό στον συντελεστή άνωσης. Η δεύτερη περίπτωση είναι ένας αγωγός σχήματος S με στόχο τις ελάχιστες απώλειες ολικής πίεσης, με περιορισμό στον όγκο του αγωγού. Και οι δύο περιπτώσεις παραμετρικοποιούνται με τεχνικές NURBS. Συνδυάστηκαν οι τεχνικές δειγματοληψίας Latin Hypercube και Random Sampling για τη δημιουργία ενός συνόλου νέων διαφοροποιημένων γεωμετριών εντός συγκεκριμένων ορίων. Η ροή επιλύεται αριθμητικά χρησιμοποιώντας το λογισμικό PUMA για κάθε νέα γεωμετρία, ολοκληρώνοντας έτσι τη Βάση Δεδομένων που χρησιμοποιήθηκε για την εκπαίδευση.

Στην πρώτη προσέγγιση, μοντέλα παλινδρόμησης βασισμένα στα Δέντρα Αποφάσεων εκπαιδεύονται στη Βάση Δεδομένων, με σκοπό την πρόβλεψη των τιμών των συναρτήσεων στόχου και περιορισμών με χαμηλό κόστος. Αυτά τα μοντέλα χρησιμοποιούνται, έναντι άλλων off-line εκπαιδευόμενων μεταμοντέλων, κατά τη βελτιστοποίηση με Εξελικτικό Αλγόριθμο.

Στη δεύτερη προσέγγιση, σε κάθε στοιχείο της Βάσης Δεδομένων, ανάλογα με την τιμή του περιορισμού, αποδίδεται και μία τιμή κλάσης. Έτσι, τα στοιχεία ταξινομούνται ως

αποδεκτά/μη-αποδεκτά, με δύο γκρίζες ζώνες ανάμεσα τους. Εκπαιδεύονται μοντέλα ταξινόμησης βασισμένα στα Δέντρα Αποφάσεων στη Βάση Δεδομένων, με στόχο την πρόβλεψη της κλάσης νέων ατόμων, ως προς το αν σέβονται ή όχι τους περιορισμούς. Τα μοντέλα χρησιμοποιούνται σε κάθε γενιά της βελτιστοποίησης, ταξινομώντας τα άτομα πριν την αξιολόγηση τους από το λογισμικό CFD. Με αυτόν τον τρόπο, τα άτομα στα οποία έχουν αποδοθεί ορισμένες κλάσεις, που δεν πληρούν τον περιορισμό, δεν αξιολογούνται από το λογισμικό CFD, αποφεύγοντας ακριβές αξιολογήσεις για μη αποδεκτές λύσεις.

Στην τρίτη προσέγγιση, χρησιμοποιείται ένα Δέντρο Αποφάσεων για την κατάτμηση του χωρίου σχεδιασμού σε διαφορετικές υποπεριοχές. Επιλέγοντας την υποπεριοχή που περιέχει τη βέλτιστη λύση και ξεκινώντας τη διαδικασία της βελτιστοποίησης σε αυτή την υποπεριοχή, η βέλτιστη λύση μπορεί να βρεθεί ταχύτερα. Δοκιμάστηκαν διάφορες προσεγγίσεις για την επιλογή της κατάλληλης υποπεριοχής.

Η δεύτερη και η τρίτη προσέγγιση συνδυάστηκαν προκειμένου να αξιολογηθεί η απόδοση του συνδυασμού αυτών. Όλα τα αποτελέσματα συγκρίνονται με αυτά των EA και MAEA (υποβοηθούμενο από on-line εκπαιδευμένα νευρωνικά δίκτυα τύπου RBF) που εφαρμόζονται στο λογισμικό EASY.

Nomenclature

NTUA	National Technical University of Athens
PCOpt	Parallel CFD & Optimization Unit
CFD	Computational Fluid Dynamics
AI	Artificial Intelligence
ML	Machine Learning
EA	Evolutionary Algorithm
MAEA	Metamodel Assisted Evolutionary Algorithm
DB	Database
DNN	Deep Neural Networks
RBF	Radial Basis Function
DT	Decision Tree
CART	Classification And Regression Tree
MSE	Mean Squared Error
MAE	Mean Absolute Error
GA	Genetic Algorithm
ES	Evolution Strategies
GP	Genetic Programming
DoE	Design of Experiment
SOO	Single-Objective Optimization

MOO	Multi-Objective Optimization
ShpO	Shape Optimization
NURBS	Non-Uniform Rational B-Splines
LHS	Latin Hypercube Sampling

Contents

Chapter 1

Artificial Intelligence

1.1 Introduction to Artificial Intelligence

Artificial Intelligence (AI) refers to the intelligence displayed by machines, enabling them to perform tasks that typically require human intelligence [1]. AI systems employ algorithms and models that allow machines to analyze large amounts of data and extract patterns. This enables AI systems to make predictions, solve complex problems and recognize objects with accuracy and efficiency surpassing that of humans. AI encompasses a wide range of techniques such as machine learning, natural language processing and computer vision.

In recent years, Artificial Intelligence has seen a rapid increase in popularity, leading to the creation of many applications. Some of the most common AI applications include recommendation systems (used by social media and online advertising) [2], virtual assistant technologies, self-driving cars and language models [3], among others.

Many sectors have been heavily impacted by AI, from healthcare and finance to engineering [4] [5]. In the field of engineering, AI has penetrated as a powerful tool for enhancing tasks and overcoming challenges [6].

1.2 Machine Learning

Machine Learning (ML) is a core component of AI, as illustrated in Figure 1.1, and focuses on the development of algorithms that can learn from data and improve their performance on tasks through experience. The learning process of these algorithms

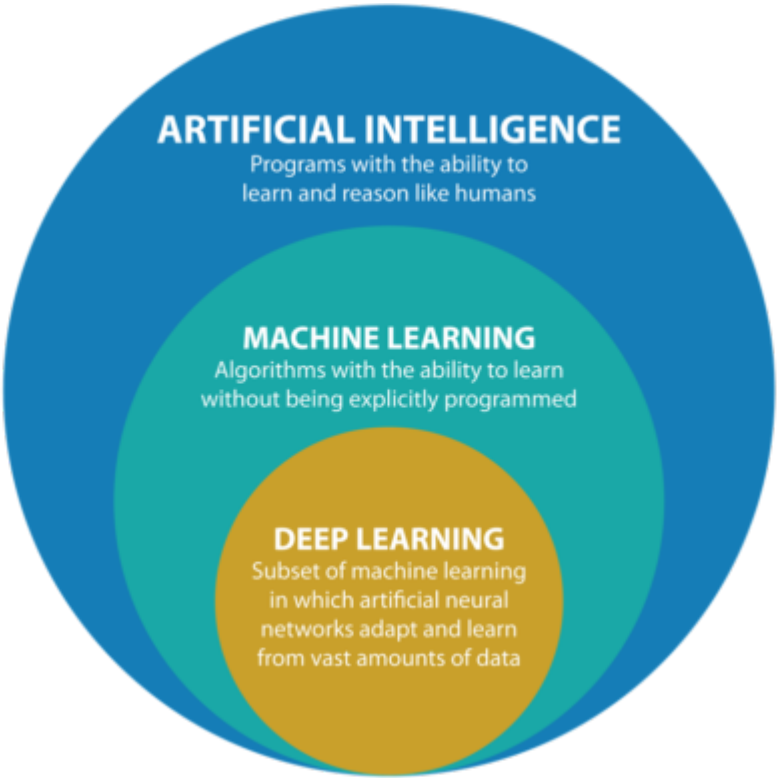


Figure 1.1: *Machine Learning as an Artificial Intelligence subfield. From: [8]*

involves identifying patterns and relationships in data, which can then be used as knowledge by the model to make predictions or classify new data [7].

There are several types of Machine Learning models, for different tasks. The main types of Machine Learning are [1], [9]:

- Supervised Learning: In supervised learning, models are trained on a set of labeled data, where each data point consists of the input and the corresponding output. The goal of these models is to utilize the provided training data, to identify patterns and learn a function that can predict the output for unseen data. These algorithms, when provided with an input, are able to either predict a numerical value (regression) or classify data (classification).
- Unsupervised Learning: Unsupervised learning algorithms are trained on a set of data containing only inputs, without the corresponding outputs. These algorithms analyze data that have not been labeled to identify patterns and similarities within the data.
- Reinforcement Learning: Reinforcement learning algorithms are similar to supervised learning, but the algorithms are not trained on a specific dataset. Instead, the training of the model is done through trial and error. The model is either punished or rewarded based on its decisions. The goal of the model is to maximize its reward, thereby improving its accuracy over time.

1.3 Machine Learning in Shape Optimization

In the case of shape optimization, the purpose of the optimization software is to find the shape of a structure, that minimizes or maximizes an objective function, such as the shape of an airfoil that minimizes Drag. The shape of a structure is parameterized and controlled by some design variables. For example, the shape of an airfoil is parameterized through NURBS curves. Then, these design variables are provided to the Evolutionary Algorithm to seek for the optimal design. The Evolutionary Algorithm needs to evaluate a large number of different geometries, in order to find the optimal design. To evaluate the candidate solutions an evaluation tool is being used.

In computational mechanics, the evaluation of a specific geometry requires the use of computationally expensive simulation codes. Specifically, in fluid applications, Computational Fluid Dynamics software is used to solve the flow numerically. Considering the high number of evaluations required by the Evolutionary Algorithm and the computational cost of CFD simulations, finding the optimal design can be really computationally expensive.

To address this challenge, Machine Learning models are employed in Shape Optimization to reduce the computational expense [10]. By training these models on a Database (DB) with the results of the CFD codes in a number of different geometries, these models are able to predict the desired output for unseen data. The computational cost of running these models is much smaller compared to the CFD simulations.

For this reason, these surrogate models, often referred to as metamodels, can be used instead of the expensive evaluation tools during optimization. If the optimization algorithm that is used is an Evolutionary Algorithm the combination is called Metamodel Assisted Evolutionary Algorithm (MAEA). These metamodels are trained on a dataset of different CFD simulations, each representing a different shape. The number of the CFD simulations is far less than the one needed in the optimization process, so the MAEA optimization greatly reduces the cost.

In the past, different metamodels have been used by the PCOpt/NTUA team, like Radial Basis Function (RBF) networks [11] and Deep Neural Networks (DNN) [10], [12].

1.4 Thesis Scope

The scope of this Diploma Thesis is to research ways in which Machine Learning models based on Decision Trees can enhance the optimization procedure in shape optimization applications. The aim is to explore different approaches that leverage

the unique characteristics of these methods to improve the performance of the optimization procedure. This performance improvement is evaluated either by faster convergence to the optimal solution by the Evolutionary Algorithm or the discovery of superior solutions.

These Tree-Based Machine Learning Algorithms are utilized as off-line trained metamodels, constraint-based classifiers or as a way to explore the design space and guide the optimization procedure. All of these proposed applications are discussed within the context of this Diploma Thesis and then tested in shape optimization problems.

In these applications, the EASY (Evolutionary Algorithms SYstem) software [13] is being used for the optimization and the GPU-accelerated software PUMA [14] for the CFD analyses and the NURBS parameterization, both developed by the PCOpt/NTUA team.

1.5 Thesis Outline

The chapters that comprise this Diploma Thesis are presented:

- **Chapter 2**: A presentation of the Tree-Based algorithms and the way they operate, is made. The Decision Tree model, the building block of the Tree-Based algorithms, is explained, along, with a complete explanation of the way multiple Decision Trees are combined to create the Random Forest and Gradient Boosting algorithms.
- **Chapter 3**: The core optimization method, i.e. Evolutionary Algorithms (EA) are described in brief. The Metamodel-Assisted Evolutionary Algorithm (MAEA) framework is also described. Lastly, the CFD tools utilized in this study are presented.
- **Chapter 4**: The proposed ways in which Tree-Based Models can enhance the performance of shape optimization in fluid mechanics are discussed. These are tested in the S8052 airfoil case and are compared with the outcomes of EA and MAEA (RBF on-line).
- **Chapter 5**: The proposed approaches are tested again in the shape optimization problem of an S-Bend duct, i.e. an internal fluid mechanics problem.

Chapter 2

Tree-Based Machine Learning Algorithms

This chapter introduces the concepts of *Random Forest* and *Gradient Boosting*, two popular algorithms in the field of Machine Learning. These algorithms create prediction models and in this Diploma Thesis they are been tested as metamodels in shape optimization applications. Additionally, this study explores alternative strategies for implementing these methods. The aforementioned methods appear to be really powerful, outperforming mainly used Machine Learning algorithms, like Artificial Neural Networks [15]. Because both algorithms are *Ensemble Models* of *CART Decision Trees*, Decision Trees and Ensembling will be explained first. Subsequently, each algorithm's specifics are described in the corresponding sections.

2.1 Decision Trees

2.1.1 General

Decision Trees (DTs) are a supervised learning method, used for classification and regression problems. In a classification problem, the algorithm predicts the category to which the observation belongs, from a limited set of classes. In contrast, in a regression problem, the algorithm predicts a numerical value, taking as input the observation.

In supervised learning, the model is trained on a training dataset, where the correct output is provided for each input. The model learns from these data, so it can make predictions on new unseen data. The training dataset is $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, where

the input data or features are $\boldsymbol{x}_i \in X = \mathbb{R}^p$ and the output data are $\boldsymbol{y}_i \in Y = \mathbb{R}^k$. In the context of the shape optimization problem that has been described earlier, the input data $\boldsymbol{x}_i$ is a vector of the points of the Morphing Box that controls the NURBS curve which describes the shape of the airfoil. The output data $\boldsymbol{y}_i$ is either a single value or a vector of the objective function depending on whether the problem is Single or Multi-Objective. The training dataset D is a database of the input with the corresponding output data that have been already evaluated by the proper evaluation tool, in the case of this Diploma Thesis, airfoils with different shapes that their flows have been already solved numerically from the CFD software. The trained model based on these training data is expected to predict the objective function of a new airfoil without solving the flow.

As a supervised learning method, DTs are provided with a training dataset. During training, they partition the design space into several regions and assign a value to each region. A design space, in machine learning, is the set of all possible values for a selected set of features from a given dataset. To partition the design space into subregions, a DT performs a process of dividing the sample (the training dataset), a *Split*, based on some criteria that the algorithm selects. In each subregion, a number of datapoints from the original training dataset are included. Based on the output data of these datapoints the algorithm assigns a value to each subregion, most often the mean value of the output data that belong to this subregion. After this procedure is completed, the model is trained on this training dataset. When the user provides some observation to a trained DT, the model determines the subregion to which the specific observation belongs and returns the assigned value of the subregion as the prediction. The way the best split and the value of its subregion are chosen depends on the specific algorithm.

The point at which a split is executed is called, a *Decision Node*. As a result, after each Decision Node, two or more *Child Nodes* are created, depending on the specific algorithm. When a node does not split any further, it is called *Leaf or Terminal Node* and depicts a subregion. In addition, the first node which represents the entire sample and is the beginning of every DT is the *Root Node*. Lastly, the *Depth* of a DT is the number of levels from the Root Node to the last Leaf Node.

A Regression DT model with Depth of 3 is shown in Figure 6.1. The trained model has divided the design space into three subregions R_1, R_2 and R_3, based on two criteria: if $X1 \leq t_1$ and $X2 \leq t_2$, with $X1$, $X2$ the two features of the input data and t_1, t_2 numerical values, often referred as thresholds. When the model will be provided with some unknown input it will check these criteria and will find the Leaf Node (the subregion) that this unknown input belongs. Then, the prediction is expected to be equal to the assigned value of the corresponding subregion.

Regarding the history of DTs, the first regression tree algorithm published was the *Automatic Interaction Detection (AID)* [16], followed by many others. The most important algorithms are [17]: *Classification and Regression Trees (CART)* [18], *ID3* [19], *C4.5* [20], *C5.0* [21], *CHi-squared Automatic Interaction Detector*

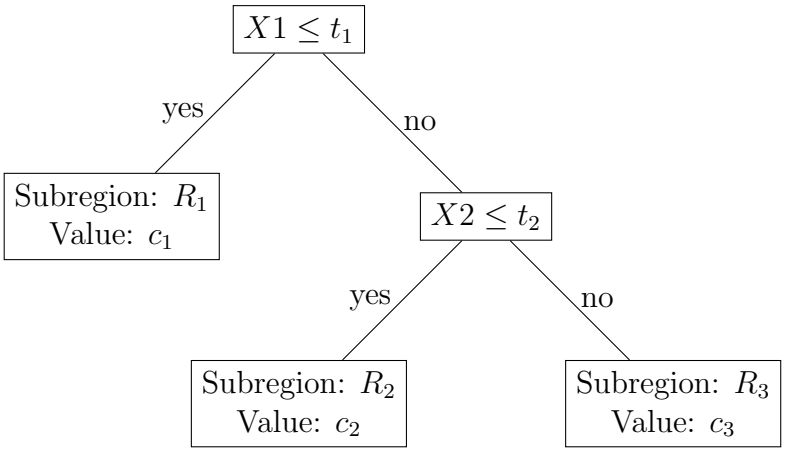


Figure 2.1: *Tree Visualization of a trained DT.*

(CHAID) [22]. In addition, there are many more algorithms optimizing previous ones or implementing different ideas [23].

CART DTs are the foundation of *Random Forest* and *Gradient Boosting* algorithms, which will be assessed as metamodels in this Diploma Thesis. For this reason, the CART DTs will be explained thoroughly.

2.1.2 CART Regression Trees

The Classification And Regression Tree (CART) is a popular tree-based method, first proposed by Breiman et al., 1984 [18]. The basis for the CART model is the DT, but it has certain differences in the way trees are constructed. CART always performs binary splits, which means that every decision node has two child nodes. In addition, CART can handle both continuous and categorical data and automatically identify the significance of a variable [24]. In this section, the details of a CART Regression Tree algorithm will be presented.

Assume, the training data are $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, where the input data $\boldsymbol{x}_i \in X = \mathbb{R}^p$ and the output data $\boldsymbol{y}_i \in Y = \mathbb{R}^k$. The regression model is, in fact, a function $f : X \to Y$. As previously discussed, a Regression DT splits the design space into M subregions, denoted as R_m, with each region assigned a predicted value $\hat{c}_m$. The number of subregions M can be controlled by the user directly and indirectly. The process of Hyperparameter tuning is demonstrated later in this chapter. To predict the output value for a given observation, the model identifies the subregion R_m to which the observation belongs and returns the assigned value $\hat{c}_m$, associated with that subregion. The final prediction model, $\hat{f}(X)$, is:

$$\hat{f}(X) = \sum_{m=1}^{M} \hat{c}_m \mathbb{I}\{X \in R_m\} \tag{2.1}$$

To compute the prediction in each subregion, the CART algorithm seeks for the prediction $\hat{c}_j$ that minimizes the loss function in its subregion,

$$\hat{c}_m = \arg\min_{\hat{c}} \sum_{\boldsymbol{x}_i \in R_m} \mathcal{L}(\boldsymbol{y}_i, \hat{c}_m) \tag{2.2}$$

where $\mathcal{L}(\boldsymbol{y}_i, \hat{c}_m)$ is the loss function, convex and differentiable, that is used to quantify the difference between the true output, $\boldsymbol{y}_i$, and the model's predicted output, $\hat{c}_m$, that corresponds to the subregion R_m.

The most commonly used loss function $\mathcal{L}$ is the *Mean Squared Error (MSE)*, where $\mathcal{L}_{MSE} = (\boldsymbol{y}_i - \hat{y}_m)^2$, but also *Mean Absolute Error (MAE)* and *Poisson* are commonly used.

If the loss function is the mean squared error, equation 2.2:

$$\hat{c}_m = \arg\min_{\hat{c}} \sum_{\boldsymbol{x}_i \in R_m} (\boldsymbol{y}_i - \hat{c}_m)^2 \tag{2.3}$$

The minimum is computed by finding the $\hat{c}_m$ that makes the derivative of the summation equal to zero, so from Equation 2.3:

$$\hat{c}_m = \frac{1}{n_m} \sum_{\boldsymbol{x}_i \in R_m} \boldsymbol{y}_i = mean(\boldsymbol{y}_i)$$

Therefore, $\hat{c}_m$ equals the average of $\boldsymbol{y}_i$'s, at terminal node m, when the loss function is the MSE. Depending on the selected loss function, a wide variety of summary statistics, like the median, could be computed [25].

As mentioned earlier, the algorithm partitions the subregion R_m into two subregions R_1 and R_2, so from a Decision Node two Child Nodes are created. This procedure is called a split, $s_0(k,t)$, where k the number of feature x^k, $\boldsymbol{x} = [x^1, \ldots, x^k, \ldots, x^p]$, and t the selected threshold, a numerical value:

$$s_0(k,t) = \big(\{\boldsymbol{x} | x^k < t, \boldsymbol{x} \in R_1\}, \{\boldsymbol{x} | x_k \geq t, \boldsymbol{x} \in R_2\}\big) \tag{2.4}$$

To select the split, the algorithm selects the feature number k and threshold t, such as:

$$\arg\max_{(k,t)} [\mathcal{L}_{R_m} - (\mathcal{L}_{R_1} + \mathcal{L}_{R_2})] \textbf{ or } \arg\min_{(k,t)} [\mathcal{L}_{R_1} + \mathcal{L}_{R_2}] \tag{2.5}$$

By scanning through the combinations of each variable and the corresponding thresholds, the algorithm selects the best pair (k,t) [9]. Finding the optimal split at each node in a DT can be a computationally complex task, and it has been shown to be NP-hard (a problem for which no efficient solution is known) in the general case

[26].

As a result, at every decision node, the algorithm proceeds to find the best split each time, without considering past or future splits [25]. This behavior is the result of the algorithm being called *"greedy"*, in literature. Furthermore, before and after the selection of a split, the algorithm checks the Hyperparameters that define the stopping criteria, like the *Minimum samples per leaf*, to determine if the split is feasible.

A pseudocode of the CART Regression Algorithm is presented in Algorithm 1 and three different visualizations of a trained CART DT are demonstrated in Figure 2.2.

Algorithm 1 CART Algorithm

Input: Training Dataset: $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ with $\boldsymbol{x}_i \in X = \mathbb{R}^p$, Hyperparameters

Build the root node of the tree using the entire dataset D
repeat
  for each feature x^k where k = 1, ..., p **do**
    for each possible value t of x^k **do**
      $s(p,t) = \big(\{\boldsymbol{x}|x^k < t, \boldsymbol{x} \in R_1\}, \{\boldsymbol{x}|x_k \geq t, \boldsymbol{x} \in R_2\}\big)$: Split the dataset D into two subsets R_1 and R_2 based on the value t of feature x^k
      Compute the loss function for both subsets.
    end for
    $\arg\max_{(k,t)}[\mathcal{L}_{R_m} - (\mathcal{L}_{R_1} + \mathcal{L}_{R_2})]$ **or** $\arg\min_{(k,t)}[\mathcal{L}_{R_1} + \mathcal{L}_{R_2}]$: Select the feature-value pair (k,t) that results in the lowest error
  end for
  Create a new Decision Node for the selected feature-value pair
  Repeat for both subregions R_1 and R_2
until stopping criteria is met
In each Leaf Node compute the assigned value $\hat{c}$, as the mean value of the output data, for regression, or as the majority vote of the output data that belong to each subregion, for classification.
return The Decision Tree

Output: The final Decision Tree: $\hat{f}(X) = \sum_{m=1}^{M} \hat{c}_m \mathbb{I}\{X \in R_m\}$

2.1.3 CART Classification Trees

The CART Classification Tree Algorithm, which is used to solve classification problems, is presented in this section. In classification, a model's output is a set of distinct classes, which are typically denoted as $y_i \in 1, 2, \ldots, C$, where C is the number of classes. While the Regression and Classification CART algorithms share a similar concept of splitting the design space into subregions to minimize the loss function, there are important differences between them.

In the CART Classification Tree Algorithm, specific loss functions are employed, and the outcome of each node is determined through a majority vote of the observations within each leaf node. This means that, for a given leaf node, the class label is determined based on the most frequently occurring class among the samples assigned to that node.

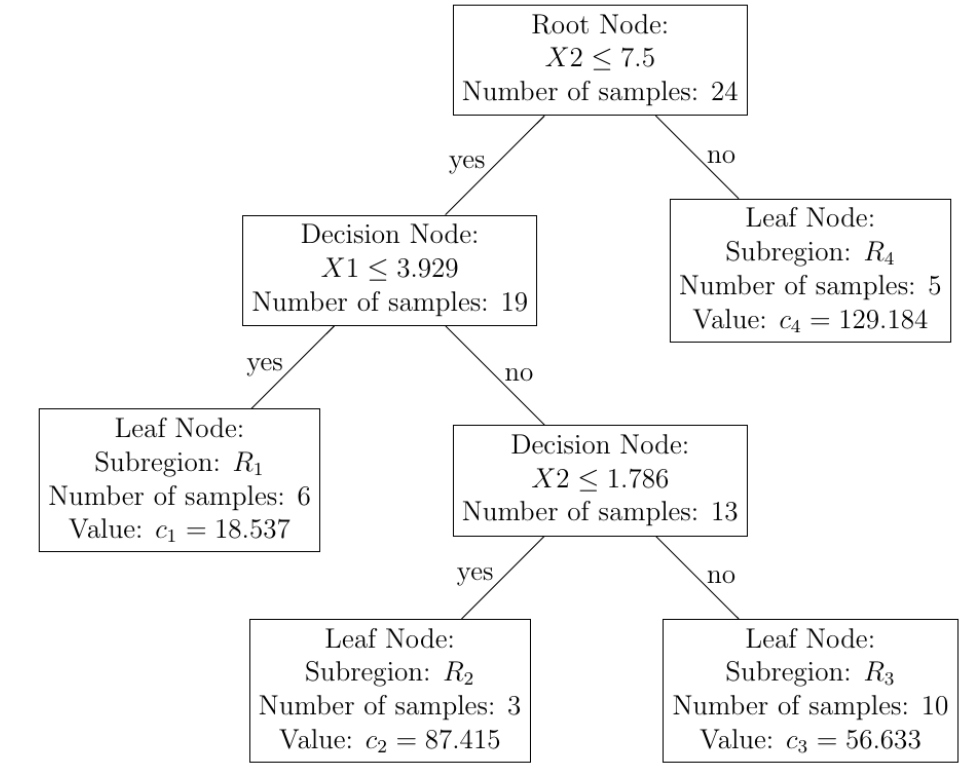
If the classification problem has C number of classes, $\hat{p}_c$ represents the proportion of observations in a region R_m that belong to class c. The three most commonly used loss functions are:

- Misclassification Error: $\mathcal{L}_{misclassification} = 1 - \max_c \hat{p}_c$
- Gini Impurity: $\mathcal{L}_{gini} = \sum_c \hat{p}_c(1 - \hat{p}_c)$
- Cross-Entropy: $\mathcal{L}_{cross} = -\sum_c \hat{p}_c log_2 \hat{p}_c$

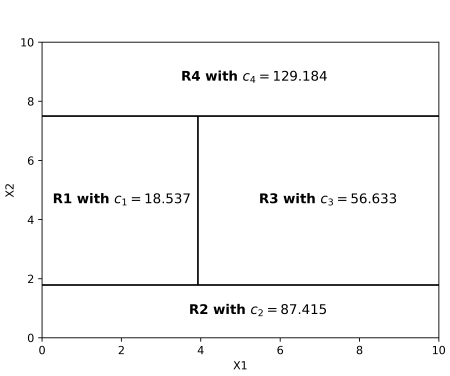
A pseudocode of the CART Classification Tree algorithm is presented in Algorithm 1.

2.1.4 Hyperparameters

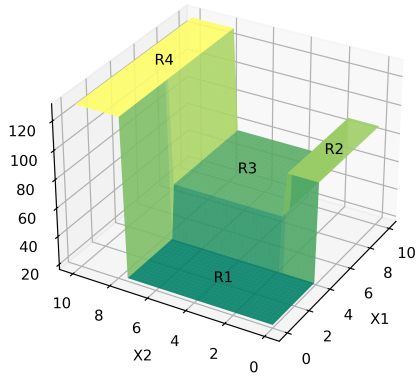
Hyperparameters are parameters that are set before the training process by the user. The selection of these parameters is done empirically through trial and error and is unique to each problem. Some of the hyperparameters for CART DTs include [18]: *Maximum depth*, *Minimum samples per split*, *Minimum samples per leaf*, *Maximum features*, *Minimum improvement in the splitting criterion*, *Maximum number of leaf nodes* and *Splitting criterion*. Some of the aforementioned hyperparameters like Maximum depth, Minimum samples per leaf, Minimum improvement in the splitting criterion and Maximum number of leaf nodes determine when to stop growing a DT and impact the architecture of the final DT. It is important to carefully choose the stopping criteria for growing a tree, as it can have a significant impact on the performance of the model and whether overfitting occurs. Lastly, the user can choose if the DT should be Pruned. *Pruning* involves removing nodes from the trained DT to simplify it and potentially improve generalization.



(a)



(b)



(c)

Figure 2.2: *A trained CART DT model in different representations, applied to a problem with two features. The regression DT was trained on a Database with 30 entries, where* $0 < x_1, x_2 < 10$ *and* $y = x_1^2 + x_2^2$*. The DT's Hyperparameters were tuned, and the model was trained, resulting in splitting the design space into 4 subregions and assigning a corresponding value to each one of them. (a) The tree visualization of the DT: All the splits, including the feature and the value in which each split takes place, can be seen. Also, the Leaf Nodes are shown, each one representing a subregion of the design space with its assigned value, the prediction. (b) Partition of the design space: It demonstrated the way the design space separated into 4 subregions after the training process. (c) The plot of the prediction surface: Including the output value as the third dimension, the surface of the prediction can be seen, with each subregion standing to the corresponding value.*

2.2 Ensemble Methods

Although there are some benefits to DTs [18], they are not a powerful predictive model, as they are unstable, prone to overfitting and they are high variance estimators, meaning their predictions can be very different depending on the data used [18]. However, by combining the predictions of different DTs, new *Ensemble Models* can be created, that tend to perform really well. As mentioned earlier *Random Forests* and *Gradient Boosting*, the algorithms that will be used as metamodels in this Diploma Thesis, are *Ensembles* of *CART DTs*. In this section, *Ensemble Methods* are explained, along with their benefits and the different types.

2.2.1 Ensemble Theory

In Machine Learning, *Ensemble Methods* are meta-algorithms, that train multiple individual models to a specific problem and then derive the final result by combining them, into one predictive model [27]. These individual models are usually called *base learners*, *base models* or *weak learners* and can be DTs, Neural Networks, or any other Machine Learning algorithm. Ensemble methods perform better than the weak learners themselves [28]. Ensembling can be used in a variety of contexts, including classification, regression, and even reinforcement learning. Moreover, they can combine the same weak learners, then called, *homogeneous ensembles*, or different ones, *heterogeneous ensembles*.

Numerous types of Ensembling exist, while the most common ones are [29], *Bayes Optimal Classifier*, *Bootstrap Aggregating (Bagging)*, *Boosting*, *Stacking*.

Random Forest is an extension of the idea of Bagging, while Gradient Boosting is an implementation of Boosting. Consequently, the next sections of this chapter will thoroughly review both these Ensemble techniques.

2.2.2 Reasons to Ensemble

In general, Ensemble Models outperform the weak learners which constitute them and there are different reasons for this. First of all, one reason is Statistical [30]. An individual model is trained on a training dataset and may achieve adequate performance when it predicts the output data of this specific dataset, often referred to as training error. However, there is the risk that the model will not provide sufficient results in real data. In this case, by combining the predictions of individual models, an Ensemble Model can reduce the risk of an inaccurate prediction. To illustrate this, an example using basic probability theory is constructed [31], [32]. Considering the random variable z_i to be the error of a specific base model, for

example, a DT, and the Ensemble Model to be the average of the base models, where $\overline{z}$ is the error of the ensemble. If z_i's are random variables, independent identically distributed (iid), for $0 \leq i < n$ $\text{Var}(z_i) = \sigma^2$.

On the other hand, if $\overline{z}$ represents the average of z's, the variance of the average ($\overline{z}$) is:

$$\text{Var}(\overline{z}) = \text{Var}\left(\frac{1}{n}\sum_i z_i\right) = \frac{\sigma^2}{n} \tag{2.6}$$

If, these variables are not independent and are instead correlated by a factor ρ:

$$\begin{aligned}\text{Var}(\overline{z}) = \text{Var}\left(\frac{1}{n}\sum_i z_i\right) = \frac{1}{n^2}\sum_{i,j}\text{Cov}(z_i, z_j) = \frac{n\sigma^2}{n^2} + \frac{n(n-1)\rho\sigma^2}{n^2} \\ = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2\end{aligned} \tag{2.7}$$

where $\text{Cov}(z_i, z_j)$ symbolizes the covariance between z_i's and z_j's.

From Equation 2.7 it is clear that the variance of the ensemble is decreased when the number, n, of base models used, is increased or when the correlation between the models ρ is decreased. Of course, this implies the Ensemble to be the average of the base models, which is not always the case. However, this example demonstrates the benefits that can be extracted through Ensembling.

Another reason that Ensemble models perform better than weak learners is that Machine Learning algorithms often perform some form of optimization, such as gradient descent in Neural Networks or the greedy splitting rule in a DT. The optimization algorithms may struggle to find the global optimum and instead get trapped in local optima. By implementing Ensemble Methods, the result no longer relies on one local search, but on a combination of them with different starting points, resulting in a better approximation of the function it tries to predict [30].

Lastly, it may be impossible for one model to accurately capture the complexity of the true function. By combining predictions from multiple weak learners, it becomes possible to explore a wider range of predictions and potentially achieve a better understanding of the data [30].

Therefore, by combining weak learners, such as DTs, through Ensembling a new, more robust model can be created providing more accurate predictions.

2.3 Random Forest

Random Forest utilizes an Ensemble Method, called Bagging with DTs. In this section, Bagging and the Random Forest algorithms are explained.

2.3.1 Bootstrap Aggregation - Bagging

The *Bagging or Bootstrap Aggregation algorithm*, was first introduced by Breiman [33] and is an ensembling technique for reducing the variance or increasing the accuracy of a prediction model [9] [34]. To achieve that the algorithm combines several individual models that have been trained by different bootstrap samples of the training data and then averages their predictions.

Bootstrap is a method from statistics that is used to measure the uncertainty of an estimator or learning method [35] [36]. Consider a population P and a training dataset $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, sampled from P. The idea is to generate B number of training sets, Z_b, $b = 1, \ldots, B$ from the initial training dataset D. Each of the new datasets contains N random instances, the same as the initial, with replacement, which means that an instance can be repeated in the same dataset. The result of this procedure is B different datasets, all sampled from the initial training dataset [37].

Each bootstrap sample contains about 63.2% of the instances of the training dataset. This is because, in a bootstrap sample, each instance is chosen with probability $\frac{1}{B}$. As a result, after B draws, with B large enough, the likelihood that a certain instance was not chosen is:

$$(1 - \frac{1}{B})^B \approx e^{-1} \approx 0.368 \tag{2.8}$$

This is true even with a relatively big B. Accordingly, it can be inferred that each sample comprises about 63.2% of the cases [37].

Bagging uses Bootstrap to generate B training sets, denoted as Z_b, $b = 1, \ldots, B$, from the original dataset. Then, a model, like a DT, $\hat{f}_b$, is built and trained in each bootstrap sample Z_b. In regression tasks, the ensemble model is created by aggregating the individual models through a process of averaging their predictions. On the other hand, in classification tasks, a majority voting scheme is used to combine the predictions of the individual models.

- Regression: $f^*(\boldsymbol{x}) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(\boldsymbol{x})$
- Classification: $f^*(\boldsymbol{x}) = majority\ vote\ [\hat{f}_b(\boldsymbol{x})]_1^B$

The Bagging algorithm is shown in Algorithm 2.

The main advantages of Bagging are that it reduces the model's variance, so it works well when it is combined with high-variance individual models [37]. Also, Bagging increases the stability and accuracy of the model. Decision Trees are known to exhibit high variance, which means they have a tendency to overfit the training data [9], so this makes them well-suited as individual models in Bagging.

An additional advantage of bagging is called *out-of-bag error*. Out-of-bag (OOB) error is a measure of the accuracy of a bagging ensemble, calculated by comparing the

prediction of each base model to the actual output of the 33% of the observations not included in the training set for that base model. As demonstrated earlier a bootstrap sample contains about $\frac{2}{3}B$. This allows for a more unbiased estimate of the ensemble's accuracy, as it is not influenced by the selection of the training set for each base model. According to Breiman [33], OOB error can be used as an alternative to cross-validation for estimating the generalization error of a bagging ensemble, as it does not require the partitioning of the data into different subsets for evaluation.

Algorithm 2 Bagging

Input: Training Dataset D, Base learner $\mathcal{T}$, Integer B (number of bootstrap samples).

for $b = 1, \ldots, B$ **do**
  $Z_b \subseteq D$: Bootstrap sample from D of size N (with replacement)
  $\hat{f}_b \leftarrow \mathcal{T}(Z_b)$: Train the base learner to each of the bootstrapped samples
end for
Regression: $f^*(\boldsymbol{x}) = \frac{1}{B}\sum_{j=1}^{B} \hat{f}_j(\boldsymbol{x})$
Classification: $f^*(\boldsymbol{x}) = majority\ vote\ [\hat{f}_j(\boldsymbol{x})]_1^B$

Output: Model f^*.

2.3.2 The Random Forest Algorithm

Random Forests were introduced by Breiman [38] and are an extension to the idea of Bagging [33]. Random Forests can be used either for classification or regression. Considering our applications, as metamodels in a shape optimization problem, there is a need for a model with high accuracy. The predictions of the metamodel are the objective function of the optimization software, so in order to evaluate the candidate solutions properly the accuracy of the metamodel should be adequate. In addition, a high number of points are required in order to accurately parameterize the airfoil shape through NURBS curves, so the model should have the ability to handle a high number of features. Random Forest has been shown to be effective in both of these needs [38]. However, Random forests are regarded as expensive to train, particularly when the number of features is large, but in contrast to the CFD software their computational complexity is negligible.

Consider the Bagging algorithm, with DTs as the individual models. When DTs are generated through Bagging, they are identically distributed (i.d.). In this case, the expectation of the average of B DTs is the same as the expectation of any one of them, indicating that the bias remains the same in both cases [9]. As a result, further improvement can only be achieved through the reduction of variance.

From Equation 2.7, which applies for i.d. variables, in our case the Bagged DTs, it is clear that as B (the number of the bagged Trees) increases the second term tends to zero, meaning that variance is reduced. However, the first term remains unaffected. Thus, the correlation, ρ, between the bagged DTs limits the reduction in variance. The main idea in Random Forests is to further improve the variance reduction, that comes through Bagging, by de-correlating the individual bagged DTs [9].

As mentioned, Bagging is the core idea of Random Forests. In the Random Forest algorithm, a user-defined number of Decision Trees are trained in Bootstrap samples of the provided training dataset. So, in regression tasks, the Random Forest algorithm is trained on a dataset $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, where the input data are $\boldsymbol{x}_i \in X = \mathbb{R}^p$ and the corresponding output data are $\boldsymbol{y}_i \in Y = \mathbb{R}^k$. The algorithm proceeds by generating B number of datasets Z_b, as Bootstrap samples extracted from D all with the same size N. Each of these datasets is used to train a CART DT.

One characteristic that distinguishes the Random Forest algorithm from Bagging and contributes to decorrelating the trees is the selective consideration of only K out of the total p input elements for the selection of the best splitting points during the construction of each DT. The value of K is user-defined or set to a default value, where $0 < K \leq p$. In this way, every tree is significantly different from the rest making the trees less correlated.

The algorithm repeats this procedure B times, resulting in B DTs and the training of the Random Forest is completed. There are several different suggestions regarding the size of the individual trees that constitute the Random Forest [38], [39], but in this Diploma Thesis, it is assumed that the individual trees are grown sufficiently (∞).

Lastly, when the trained model is provided with some unknown input, it generates B predictions from the individual DTs and averages them to provide the final prediction of the Random Forest model. Random Forests are generally parallel algorithms since each tree's development is independent of prior trees [40].

In classification tasks, the Random Forest algorithm employs CART Classification Trees as base learners. The final prediction of the model is determined by aggregating the predictions from the B DTs through a majority vote scheme.

This process is described in Algorithm 3. Also, a plot of the prediction of a Random Forest model in a classification problem, along with the individual trees, that constitute it, can be seen in Figure 2.3.

2.3.3 Hyperparameters

Random Forests perform very well, without the need for extreme parameter tuning [41]. The main parameters of Random Forest are:

Algorithm 3 Random Forest (Regression)

Input: Training Dataset: $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ with $\boldsymbol{x}_i \in X = \mathbb{R}^p$, Base learner $\mathcal{T}$: the Decision Tree

Parameters: Integer B (number of DTs), K: $0 < K \leq p$, where p the dimension of the feature vector $\boldsymbol{x}_i$.

for $b = 1, \ldots, B$ **do**

$Z_b \subseteq D$: Bootstrap sample from D of size N (with replacement)

$\hat{h}_b \leftarrow \mathcal{T}(Z_b, K, \infty)$: Fully build the DT ($\infty$) to each of the bootstrapped samples, with $K \leq p$ randomly chosen features at each split

end for

Regression: $f^*(\boldsymbol{x}) = \frac{1}{B}\sum_{j=1}^{B} \hat{h}_j(\boldsymbol{x})$

Classification: $f^*(\boldsymbol{x}) = majority\ vote\ [\hat{h}_j(\boldsymbol{x})]_1^B$

Output: Model f^*.

- B, the number of trees generated.
- K, the number of randomly chosen feature variables chosen at each node that are considered for the split on the specific node.
- *leaf size* or *maximum depth*, or any parameter that constrains the size of the individual trees.

As shown in [38], as B increases, the generalization error converges to a limit. In small values of B the model may be unstable, so it is recommended to choose a value of B that is as high as computationally possible. Regarding K, the default value for classification is $K = \lfloor\sqrt{B}\rfloor$, while for regression is $K = \lfloor\frac{B}{3}\rfloor$. In practice, the best values are problem dependent and should be tuned in every model explicitly [9]. In modern implementations of the Random Forest algorithm, the user can tune many more parameters, but the model is not very sensitive to them.

2.3.4 Implementation

For the implementation of the Random Forest algorithm in the context of this Diploma Thesis, an open source ML library, called *scikit-learn*, was utilized in Python. The aforementioned library allows the user more than adequate hyper-parameter tuning. The user can tune all the parameters mentioned in this chapter regarding the Random Forest. In addition, the user can control the growth of the individual DTs that are part of the Forest, by selecting their maximum depth and/or the minimum samples per leaf, among others.

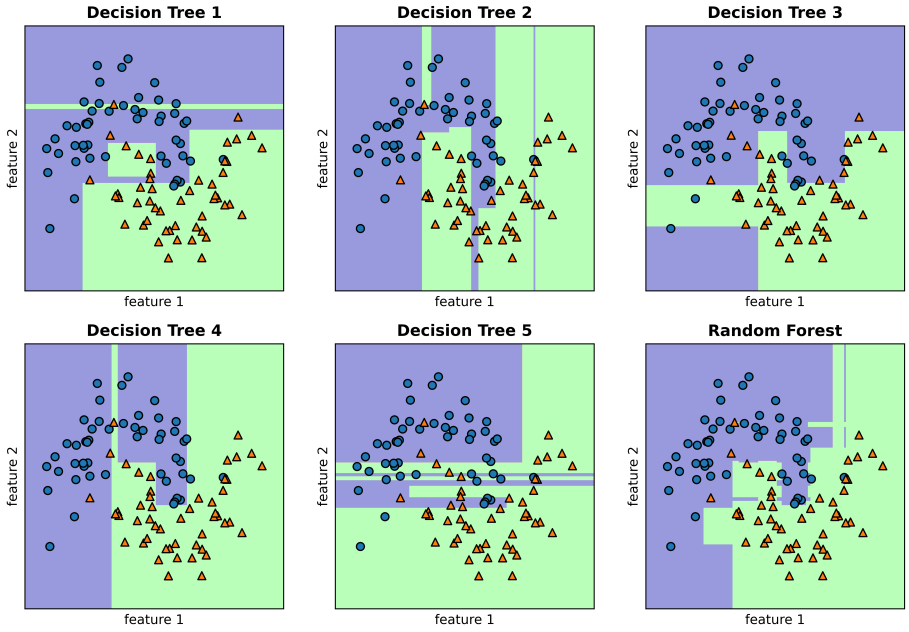


Figure 2.3: *Random Forest and the individual DTs, which comprise it, in a classification problem with two features and two possible outputs represented by different shapes. The dataset used for training consists of two interleaving half circles, where each circle represents one of the classes. The blue color corresponds to the prediction of the blue circles, while the green color represents the prediction of the orange triangles. The Random Forest, by combining the individual trees, results in a better approximation, as its prediction closely mimics the shape of the two interleaving half circles, in contrast to the predictions of the individual DTs.*

2.4 Gradient Boosting

Gradient Boosting is a powerful Machine Learning Algorithm that applies an Ensemble Method called *Boosting* to a number of base models [9]. When these base models are DTs, the algorithm is called *Gradient Boosted Trees* and this algorithm is used as one of the metamodels that are tested in the context of this Diploma Thesis [42].

2.4.1 Boosting

Boosting is an Ensemble technique that combines the predictions of multiple weak models to create a stronger model that can make highly accurate predictions [43]. Boosting algorithms work by training a sequence of weak models, with each model

learning from the mistakes of the previous models [44]. The final model is a weighted combination of the weak models, with the weights of the weak models determined by their accuracy [44]. Boosting algorithms are iterative, meaning that they train the weak models one at a time, and the training process is guided by a loss function that measures the difference between the predicted output and the actual output [42]. By iteratively training weak models and adjusting the weights of the models based on their accuracy, boosting algorithms are able to improve the overall accuracy of the model and achieve strong performance on a variety of tasks [45]. One of the key benefits of Boosting is that it can help to reduce bias in the final model and while Boosting does increase the complexity of the model, it does not typically result in increased variance.

If $H(\boldsymbol{x}_i;\theta)$ is the ensemble model, that predicts the $\boldsymbol{y}_i$ to be approximated, the boosting procedure fits an additive model of the form:

$$H_B(\boldsymbol{x}_i;\theta) = \sum_{b=1}^{B} \alpha_b f_b(\boldsymbol{x}_i;\theta_b) \tag{2.9}$$

where, $f_b(\boldsymbol{x}_i;\theta_b)$ is an individual *weak learner*, θ_b the set of parameters of the weak learner f_b, b the boosting step, α_b is the step-size at step b weighting the b'th weak learner and B the total number of weak learners.

To fit each weak learner in the Ensemble, Boosting does this in a greedy manner, searching for the weak learner that minimizes the selected loss function the most:

$$f_b(x) = \arg\min_{f} \mathcal{L}(\boldsymbol{y}, H_{b-1} + \alpha_b f_b) \tag{2.10}$$

As stated earlier, $\mathcal{L}(\boldsymbol{y}, H_{b-1}+\alpha_b f_b)$ is a loss function, convex and differentiable, that takes as input the real values $\boldsymbol{y}$ and the prediction from $H_{b-1} + \alpha_b f_b$, H_{b-1} is the ensemble model in iteration $b-1$.
After finding f_b, the algorithm adds it to the ensemble and repeats the process:

$$H_b := H_{b-1} + \alpha_b f_b \tag{2.11}$$

Boosting was first developed in the binary classification-focused computational learning theory literature [45][44]. There is not a specific algorithm that defines boosting, but a variety of Boosting techniques fit to the *AnyBoost* framework [46]. There are several types of Boosting algorithms, including *AdaBoost* (Adaptive Boosting) [44], which is an iterative algorithm that trains a sequence of weak models and assigns higher weights to instances that are misclassified by previous models. The final model is a combination of the weak models, with the weights of the weak models determined by their accuracy. Other popular boosting algorithms include *Gradient*

Boosting [42], *XGBoost* [47], and *LightGBM* [48]. These algorithms are similar to AdaBoost in that they train a sequence of weak models and combine them to create a stronger model. However, they differ in the way that the weights of the weak models are computed and the loss function that they use to guide the training process. This Diploma Thesis focuses on the *Gradient Boosting* algorithm.

2.4.2 The Gradient Boosting Algorithm

Gradient Boosting was introduced by [42] and is a generic version of Boosting for every loss function [49]. During training, Gradient Boosting works by fitting weak models to the residuals of the previous weak model, where the residual is the difference between the predicted value and the actual value. The algorithm then adjusts the predicted value by adding a fraction of the predicted value from the new weak model, where the fraction is controlled by the learning rate or step size.

The goal of the algorithm is to find a function $H(\boldsymbol{x})$, such that [50]:

$$H^* = \arg\min_F \mathbb{E}_{y,x}\mathcal{L}(\boldsymbol{y}, H(\boldsymbol{x})) \tag{2.12}$$

where $\mathbb{E}_{y,x}\mathcal{L}(\boldsymbol{y}, H(\boldsymbol{x}))$ is the expected value of $\mathcal{L}(\boldsymbol{y}, H(\boldsymbol{x}))$.

In step b of the training, define g_m to be the gradient of the loss function, $\mathcal{L}(H)$, computed at $H = H_{b-1}$:

$$g_b(x_i) = \left[\frac{\partial \mathcal{L}(\boldsymbol{y}_i, H(\boldsymbol{x}_i))}{\partial H(\boldsymbol{x}_i)}\right]_{H(\boldsymbol{x}_i)=H_{b-1}(\boldsymbol{x}_i)}, i = 1, 2, \ldots, N \tag{2.13}$$

Then, the ensemble:

$$H_b = H_{b-1} - \rho_b g_b \tag{2.14}$$

is updated, where ρ_b is the step-size or learning rate, chosen as:

$$\rho_b = \arg\min_\rho \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{y}_i, H_{b-1} - \rho g_b) \tag{2.15}$$

So, in each iteration, the ensemble is updated by moving in the direction of the negative gradient by a small step-size. This process is similar to gradient descent in function space and is called *functional gradient descent* [49].

This formulation cannot be used to compute the gradient and step-size at any set of points outside of training dataset, D, which is a crucial aspect to keep in mind. The gradient vector g_b is only specified at a very specific set D of N points. The algorithm is modified to fit a parametric function (a weak model) $h(\boldsymbol{x}; \boldsymbol{\theta}_b)$ to the

negative gradient signal, with $\boldsymbol{\theta}_b$ the parameter vector [51], with its predictions being as close to the negative gradient as possible,

$$(\boldsymbol{\theta}_b, \beta_b) = \arg\min_{(\boldsymbol{\theta},\beta)} \sum_{i=1}^{N} (r_i - \beta h(\boldsymbol{x}_i; \boldsymbol{\theta})^2 \tag{2.16}$$

where, $\{r_i = -g_b(\boldsymbol{x}_i)\}_{i=1}^N$, the "pseudoresponses".

ρ_b is computed by solving the one-dimensional optimization problem (the way is dependent on the implementation) using:

$$\rho_b = \arg\min_{\rho} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{y}_i, H_{b-1} + \rho h(\boldsymbol{x}_i; \boldsymbol{\theta}_b)) \tag{2.17}$$

and the updated formula:

$$H_b = H_{b-1} + \rho_b h(\boldsymbol{x}; \boldsymbol{\theta}_b) \tag{2.18}$$

The Gradient Boosting Algorithm is summarized in Algorithm 4 [42] [51].

Algorithm 4 Gradient Boosting Algorithm

Input: Training Dataset: $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ with $\boldsymbol{x}_i \in X = \mathbb{R}^p$
Parameters: Integer B the number of iterations.
Initialize: $H_0(\boldsymbol{x}) = \arg\min_\gamma \sum_{i=1}^N \mathcal{L}(\boldsymbol{y}_i, \gamma)$

for $b = 1, \ldots, B$ **do**
Compute the pseudo-residuals: $r_i = -g_b(\boldsymbol{x}_i) = -\left[\frac{\partial \mathcal{L}(\boldsymbol{y}_i, H(\boldsymbol{x}_i))}{\partial H(\boldsymbol{x}_i)}\right]_{H(\boldsymbol{x}_i)=H_{b-1}(\boldsymbol{x}_i)}, i = 1, 2, \ldots, N$
Fit weak model, h, to the pseudo-residuals: $(\boldsymbol{\theta}_b, \beta_b) = \arg\min_{(\boldsymbol{\theta},\beta)} \sum_{i=1}^N (r_i - \beta h(\boldsymbol{x}_i; \boldsymbol{\theta})^2$
Compute the learning rate: $\rho_b = \arg\min_\rho \sum_{i=1}^n \mathcal{L}(\boldsymbol{y}_i, H_{b-1} + \rho h(\boldsymbol{x}_i; \boldsymbol{\theta}_b))$
Update the model: $H_b(\boldsymbol{x}) = H_{b-1}(\boldsymbol{x}) + \rho_b h(\boldsymbol{x}; \boldsymbol{\theta}_b)$
end for

Output: Model $H^*(\boldsymbol{x}) = H_B(\boldsymbol{x}) = H_0(\boldsymbol{x}) + \sum_{b=1}^B \rho_b h(\boldsymbol{x}_i; \boldsymbol{\theta}_b)$**.**

2.4.3 The Gradient Boosted Regression Trees Algorithm

The algorithm presented, can be used with any weak learner and loss function and it is presented as Friedman proposed in [42]. If the weak learner is DTs, the algorithm

is usually referred to as *Gradient Boosted Regression Trees (GBRT)*. If the loss function is the squared loss, $\mathcal{L}(\boldsymbol{y}_i, H) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{y}_i - H(\boldsymbol{x}_i))^2$, it is easily shown that Equation 2.13 becomes:

$$r_i = -g_b(\boldsymbol{x}_i) = \boldsymbol{y}_i - H_{b-1}(\boldsymbol{x}_i) \tag{2.19}$$

During training, Gradient Boosted Regression Trees compute the residuals of the actual versus the predicted output and train a DT on these residuals. Then, the trained model is added to the Ensemble, after being multiplied by the learning rate and the new residuals are computed. This procedure is completed after one of the stopping criteria is met and the model is trained on the provided dataset. The described process is depicted in Figure 2.4.

The algorithm for GBRT with squared loss as the loss function is shown in Algorithm 5.

Algorithm 5 Gradient Boosted Regression Trees Algorithm with Loss Function the Squared Loss

Input: Training Dataset: $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ with $\boldsymbol{x}_i \in X = \mathbb{R}^p$
Parameters: Integer B the number of iterations.
Initialize: $H_0(\boldsymbol{x}) = \bar{\boldsymbol{y}}$

for $b = 1, \ldots, B$ **do**
Compute the pseudo-residuals: $r_i = -g_b(\boldsymbol{x}_i) = \boldsymbol{y}_i - H_{b-1}(\boldsymbol{x}_i), i = 1, 2, \ldots, N$
Fit a regression tree to the pseudoresiduals, giving terminal regions $R_{jb}, j = 1, 2, \ldots, J_b$ and compute the learning rate: $(\rho_b, \boldsymbol{\theta}_b) = \arg\min_{\boldsymbol{\theta},\rho} \sum_{i=1}^{N}[r_i - \rho h(\boldsymbol{x}_i; \boldsymbol{\theta})]^2$
Update the model: $H_b(\boldsymbol{x}) = H_{b-1}(\boldsymbol{x}) + \rho_b h(\boldsymbol{x}; \boldsymbol{\theta}_b) = H_{b-1}(x) + \sum_{j=1}^{J_b} \rho_{jb}\mathbb{I}(x \in R_{jm})$
end for

Output: Model $H^*(\boldsymbol{x}) = H_B(\boldsymbol{x}) = H_0(\boldsymbol{x}) + \sum_{b=1}^{B} \rho_b h(\boldsymbol{x}; \boldsymbol{\theta}_b)$**.**

2.4.4 Hyperparameters

Some of the main hyperparameters that can be tuned in GBRT models include:

- *Learning rate or Step-size*: Regulates how much each tree contributes to the final prediction. A model with a lower learning rate will be more accurate, but it will also require more trees, which could slow down training and prediction.
- *Number of trees*: Relates to how many trees have been trained for the model. A model with more trees may be more accurate, but it may also take longer

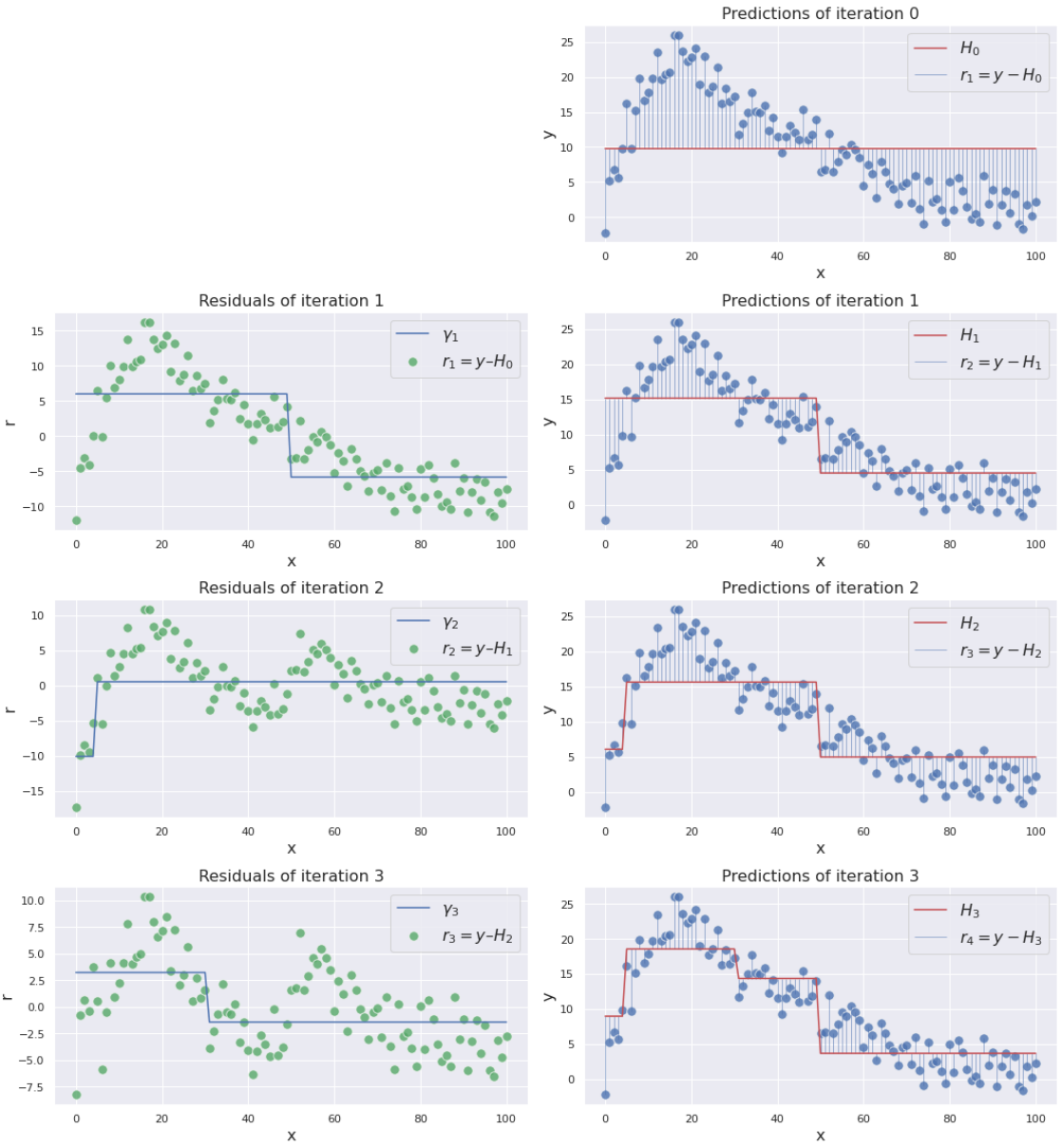


Figure 2.4: *The training procedure of the Gradient Boosted Regression Trees Algorithm. At first, the algorithm initializes the prediction to the mean. Then, the residuals are computed, and a DT is trained on the residuals. Next, this DT is added to the Ensemble multiplied by the learning rate. The residuals are computed again, and another DT is trained on them. This procedure continues until the stopping criteria are met. From [52]*

to train and make predictions.

- *Maximum tree depth*: The maximum depth of each tree in the model. Deeper trees can capture more complex patterns in the data, but they can also lead to overfitting.
- *Minimum samples per leaf*: The minimum number of samples that are required to be at a leaf node in a tree. This can help prevent overfitting by requiring a tree to have more samples at each leaf.
- *Subsample ratio*: The fraction of the training set that is used to train each tree. Using a smaller subsample can enhance the model's stability, but it can also decrease the overall accuracy of the model.
- *Loss function*: The loss function that is used to optimize the model. Different loss functions are suitable for different types of problems and choosing the right one can have a big impact on the performance of the model.

2.4.5 Implementations

In the context of this Diploma Thesis, the Gradient Boosted Regression Trees model is used as metamodel in MAEA-based shape optimization problems. The development and training of the GBRT model is carried out in an open-source machine learning library, called *scikit-learn* using Python. In addition, *XGBoost, or eXtreme Gradient Boosting* was assessed as metamodel in this Diploma Thesis and implemented in Python.

XGBoost, or eXtreme Gradient Boosting, was introduced by [47] and is a powerful and widely used machine learning algorithm for both classification and regression tasks. It is an implementation of Gradient Boosting that has been optimized for efficiency, flexibility, and portability [47] and is one of the most popular supervised learning algorithms.

Chapter 3

Evolutionary Algorithms - EA and CFD tools

3.1 Introduction to Evolutionary Algorithms

Evolutionary Algorithms (EAs) belong to a class of stochastic optimization algorithms that employ a population-based search strategy. EAs are inspired by Charles Darwin's theory of natural evolution. In an EA, the population is randomly initialized, and all individuals are subjected to evaluation using a costly evaluation software. Following the principle of survival of the fittest in nature, the algorithm identifies the fittest individuals based on the value of a selected objective function, and these individuals are selected for reproduction. This reproduction process generates the offspring population for the next generation. This procedure is repeated until the algorithm reaches convergence and the best individuals of the population represent the solution of the algorithm [53, 54].

EAs have many advantages over other optimization algorithms [55]. First of all, they can easily be used for every type of problem, just by using the appropriate evaluation software. Also, EA's have the ability to escape local minima as a stochastic optimization method. Lastly, the implementation of an EA in a new problem is direct as it does not require information about the fitness gradient. Their main disadvantage is that they require a large number of evaluations. This becomes a huge barrier in the case of aerodynamic shape optimization, like the applications on this Diploma Thesis, where the software that solves Navier-Stokes equations and predicts the flow is costly. Additionally, EAs require careful selection of parameters such as population size, reproduction function, mutation, and crossover rates.

The EAs can be categorized into three main types: Genetic Algorithms (GAs) [56], Evolution Strategies (ES) [57] and Genetic Programming (GP) [58]. Although each category has distinct characteristics, they all share common traits.

3.2 Description of an Evolutionary Algorithm

In this section, a generic EA will be described sharing traits from both Genetic Algorithms and Evolutionary Strategies [53]. In each generation, g, the algorithm manages three distinct populations, the population of λ offspring $S^{g,\lambda}$, the population of μ parents $S^{g,\mu}$ and the population of e elites $S^{g,e}$). In the elite population the most fit solutions, that have been found during the evolution process, are stored and carried over to the next generation to maintain the genetic information. During evolution, the elite population serves as a mechanism to promote the preservation of desirable genetic material that has been found in previous generations.

In Algorithm 6, the iterative process of an EA is described, with different operators denoted as T [53]. These operators represent distinct procedures and the most important of them will be explained later in this chapter.

Algorithm 6 Evolutionary Algorithm

Parameters: $S^{g,\mu}$ the size of parents population and $S^{g,\lambda}$ the size of offspring population

Initialize: $S^{0,\lambda}$ population with random candidate solutions, set generations $g = 0$

Evaluate, through the objective function, each offspring of the $S^{0,\lambda}$

repeat

Renew the elite population $S^{g,e}$ with the best candidates of $S^{g,\lambda}$: $S^{g+1,e} = T_e(S^{g,\lambda} \cup S^{g,e})$

Replace candidates of $S^{g,\lambda}$, based on the fitness value, from the new elite population $S^{g,e}$: $S^{g,\lambda} = T_{e2}(S^{g,\lambda} \cup S^{g+1,e})$

Select the parent population for the next generation: $S^{g+1,\mu} = T_\mu(S^{g,\mu} \cup S^{g,\lambda})$

Create the new generation of offspring through recombination and mutation of the parent population: $S^{g+1,\lambda} = T_m(T_r(S^{g+1,\mu} \cup S^{g+1,e}))$

Evaluate the fitness value for the new offspring population $S^{g+1,\lambda}$

until stopping criteria are met

Output: the best solution

3.2.1 Encoding of Individuals

The representation, or encoding, of individuals, is a crucial aspect of EAs as it determines how candidate solutions are encoded as a set of variables (genotype) and manipulated during the evolution process [53]. Encoding of individuals takes place at the beginning of the optimization process and it determines the way the algorithm will read the optimization variables.

Two of the most notable encoding methods for the representation of optimization variables are binary and real-valued. In **Binary Encoding** the solutions are represented as binary strings of 0s and 1s and in **Real-valued Encoding** individuals are represented as strings of real values.

The choice of representation can significantly impact the performance and effectiveness of the EA, as it influences the exploration and exploitation of the search space. As a result, the selection of the encoding method is an important decision that can impact the performance of the algorithm.

3.2.2 Recombination Operator

The **Recombination Operator**, also referred to as **Crossover**, is a genetic operator employed in EAs that combines the genotypes of two individuals from the parent population in order to generate one or more offspring. This operator mimics the process of genetic recombination observed in natural reproduction. Through recombination, random segments of each parent's genotypes are combined to create an offspring, which preferably inherits the desirable traits from both parents. The Recombination Operator is a stochastic process, as random choices are made to create the offspring [53].

There are several types of Recombination Operators for each type of encoding [53]. In the case of Binary Encoding, some of the most common are One-Point Crossover, Two-Point Crossover and Uniform Crossover. One-Point Crossover involves selecting a single point between two consecutive binary bits of a genotype. Subsequently, two parent individuals are randomly chosen from the parent population. The genotypes of these parents are split at the preselected point and the resulting segments are exchanged to create two offspring.

In the context of an EA, recombination occurs with a high probability denoted as P_r (usually $P_r = 0.90$). This allows a small probability for parent individuals to be transferred to the new offspring population.

3.2.3 Mutation Operator

The **Mutation Operator** is being applied to the new offspring created after Recombination [53]. The Mutation Operator, when applied, it introduces small, random changes in the genotype of the offspring that is being applied. In the case of Binary Encoding, this random change would be a flip of a random bit of the genotype. There is a probability this operator to be applied, P_m, called mutation rate and it is often very small.

The purpose of the Mutation Operator is to add diversity to the population and lead the algorithm to explore new regions of the search space. In this way, the algorithm is prevented from premature convergence to some local optima, regions in the search space with good, but not optimal candidate solutions. By introducing these small changes exploration is being promoted which is really important for the success of the algorithm.

3.3 Metamodel-Assisted Evolutionary Algorithms (MAEA)

As previously elucidated, EAs are capable to solve complex problems by providing any evaluation software without the need to access its source code. However, a notable limitation is that in complex problems, a significant number of evaluations may be necessary to identify the optimal solution. This is particularly relevant in the context of aerodynamic shape optimization, where the evaluation software entails computationally expensive computational fluid dynamics (CFD) simulations, resulting in substantial costs per evaluation. As a result, the use of EAs in such scenarios can pose challenges due to the associated high computational costs.

In order to tackle this issue, a Metamodel-Assisted Evolutionary Algorithm (MAEA) can be used to reduce the computational cost of the optimization [59]. Metamodels are low-cost surrogates that are trained on a DB of previously evaluated individuals. So, these metamodels can replace the CFD software, or any other evaluation software used and can dramatically reduce the computational cost of the EA. Of course, there is a computational cost associated with the creation of the DB needed to train the metamodel, but it is a fraction of the evaluations needed in an ordinary EA. Also, the cost of each call of the metamodel is considered negligible compared to the CFD software.

It is crucial to mention, that the output of a metamodel may exhibit inaccuracies and should not be considered as a substitute for the evaluation software. Nonetheless, in the context of optimization, it is not necessarily required to obtain the most precise response, but rather to ensure that the best-performing individuals achieve lower

fitness values compared to the worst-performing ones, in the case of a minimization problem.

There are two categories of MAEAs depending on the way the metamodel is connected to the EA, these are off-line trained metamodels and on-line trained metamodels. In EAs that implement off-line trained metamodels, surrogate models are trained a priori to the optimization procedure, covering the entire search space. The construction of the DB, on which the metamodel is trained, is achieved through Design of Experiment (DoE) techniques. Subsequently, during the optimization process, the metamodel is exclusively used for evaluations. After the convergence of the algorithm, the best solutions are reevaluated using the problem-specific evaluation software. The user then determines whether to retrain the metamodel with the addition of the best solutions to the DB, and subsequently restart the EA. This iterative process continues until the algorithm reaches convergence. In the context of this Diploma Thesis, the Tree-Based Models, that have been presented earlier, are utilized as off-line trained metamodels and the flowchart illustrating the off-line metamodel implementation in the optimization process is shown in Figure 3.1.

In EAs that employ on-line trained metamodels, the optimization procedure initially commences as in a conventional EA, where the evaluation software is utilized to create a DB through a number of evaluations over several generations. Then, the EA transitions to utilizing the metamodel instead of the expensive evaluation software. Only a limited number of promising offspring solutions in any generation are evaluated by the evaluation software. In the scope of this Diploma Thesis, Radial Basis Function (RBF) Networks were utilized as an on-line trained metamodel [11].

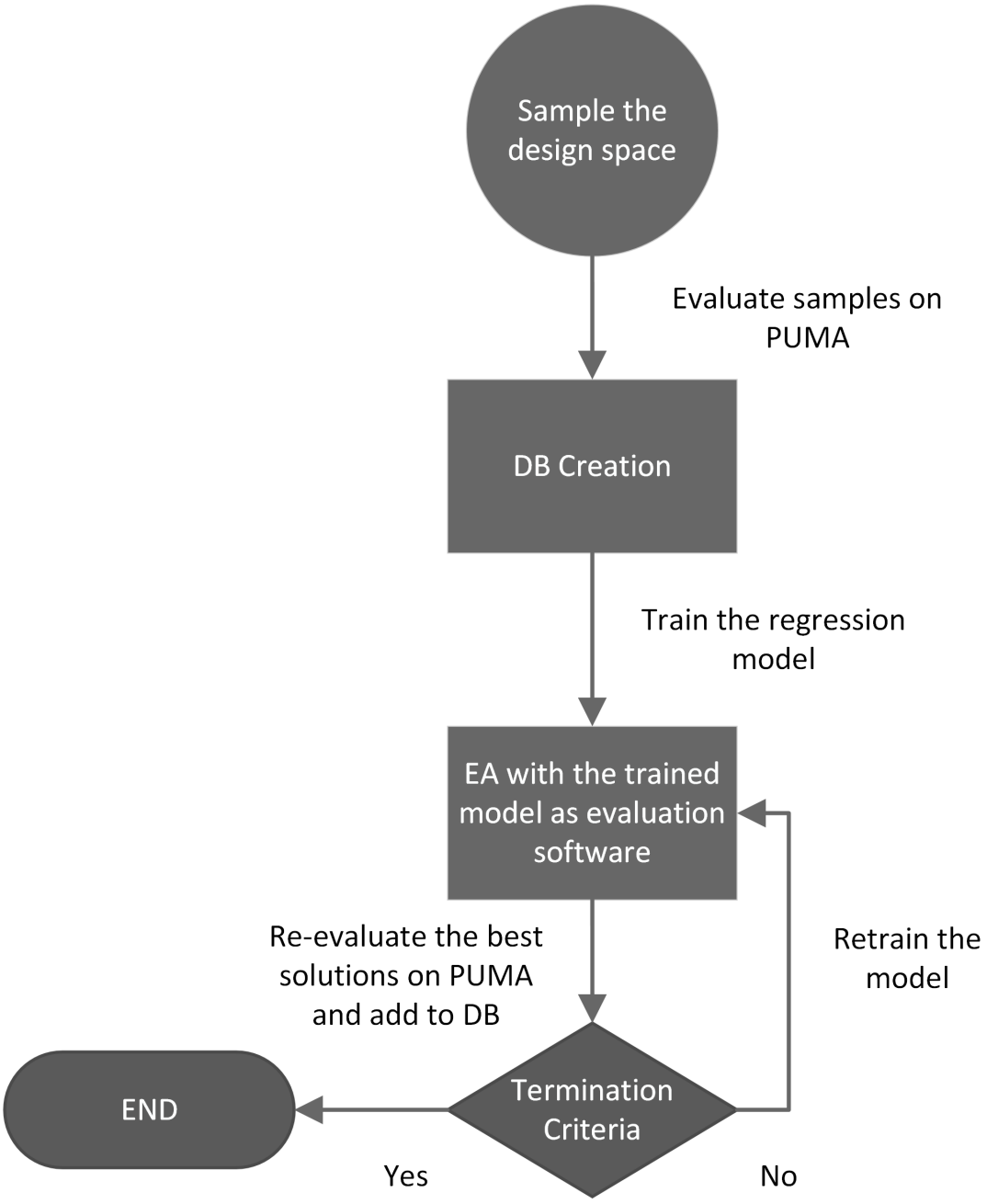


Figure 3.1: *The flowchart of the off-line metamodel implementation.*

3.4 The Evolutionary Algorithms SYstem Software - EASY

The Evolutionary Algorithms SYstem - EASY software, developed by the PCOpt/NTUA, is a generic optimization software, which implements a (μ, λ)EA, that can be used either for Single-Objective (SOO) and Multi-Objective Optimization (MOO) with or without constraints [13]. EASY utilizes stochastic and deterministic optimization methods, as also hybrid methods. The software also supports the use of distributed, asynchronous and hierarchical EAs. Moreover, the numerous parameters that can be tuned provide a high degree of control to the advanced user.

Additionally, another important feature of EASY is the integrated low-cost surrogate evaluation models (metamodels), like the RBF Networks, which are used as on-line trained metamodels and can be easily set for every problem. The software also supports the utilization of (external) off-line trained metamodels, as the Tree-Based Methods that will be tested in the context of this Diploma Thesis, providing the framework for Metamodel-Assisted Evolutionary Algorithm (MAEA) optimization.

3.5 The GPU-enabled CFD Solver PUMA

In this Diploma Thesis, the GPU-enabled PUMA software was utilized to perform all CFD analyses. Developed by the PCOpt/NTUA, PUMA is a software for solving Navier-Stokes equations and turbulence model equations on unstructured/hybrid grids using vertex-centered finite volumes [14]. Additionally, PUMA is equipped with a set of shape parameterization tools and computational grid morphers that can support shape optimization methods, allowing for its integration with the EASY software.

PUMA utilizes the CUDA programming environment and shared on-board memory for data transfer between GPUs in the same computational node, as well as the MPI protocol for communication between GPUs on different nodes.

The implementation of PUMA on GPUs provides a notable improvement in the speed of the CFD analysis compared to CPU-based software, leading to a reduction in the turnaround time.

Chapter 4

Application of Tree-Based Methods in External Aerodynamic ShpO with EAs

This chapter presents and discusses the application of Tree-Based Methods in the context of Shape Optimization (ShpO) with EAs. Specifically, these methods are tested in the ShpO process of the S8052 airfoil case. The problem was set as Single-Objective Optimization (SOO) with the objective of minimizing the drag coefficient c_d, while imposing a constraint on the lift coefficient c_L to be bigger than the baseline value. At first, a thorough presentation of the case study is given, followed by a discussion of the proposed applications and an analysis of the results.

4.1 The S8052 Airfoil Case

The optimization process focuses on the S8052 airfoil case. Figure 4.1 demonstrates the S8052 airfoil's geometric representation, showcasing its shape.

The selected flow conditions for the specific case are presented in Table 4.1.

4.1.1 Computational Mesh

The current study employs a C-type mesh, Figure 4.2. The mesh comprises a total of 47000 nodes. Notably, the mesh exhibits a higher level of node density in proximity

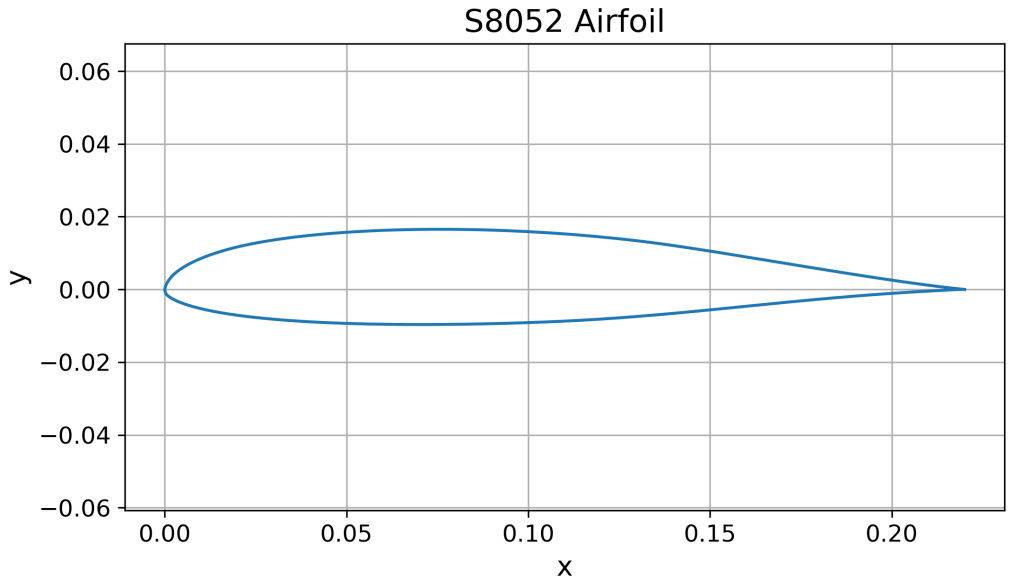


Figure 4.1: *Isolated Airfoil Case: S8052 Airfoil Shape.*

Flow Conditions	
Mach number	0.1
Reynolds number	$5.05 \cdot 10^5$
Angle a_∞ [degrees]	0

Table 4.1: *Isolated Airfoil Case: FarField Flow Conditions.*

to the airfoil, where accuracy is crucial, Figures 4.2, 4.3 and 4.4. Conversely, the node distribution gradually becomes coarser as it moves into the far field region, ensuring computational efficiency while maintaining the necessary level of accuracy.

4.1.2 Verification with Experimental Data

In this Diploma Thesis, CFD analyses are conducted exclusively using the PUMA software, which solves the Reynolds-Averaged Navier-Stokes (RANS) equations. Turbulence effects are taken into account using the one-equation Spalart-Allmaras model [60]. The simulations were conducted at various angles of attack and the corresponding polar plot was generated. The resulting plot was then compared to experimental data revealing significant discrepancies [61].

Consequently, the flow was considered transitional and transition was modeled by solving the two-equation $\gamma - \tilde{Re}_\theta$ model [62]. Then the simulations were rerun again for different angles of attack, with the inclusion of the transition model. Finally, by adjusting the incoming turbulence intensity (unknown data for the case), through trial and error, the resulting plot was closely aligned to the experimental data, as

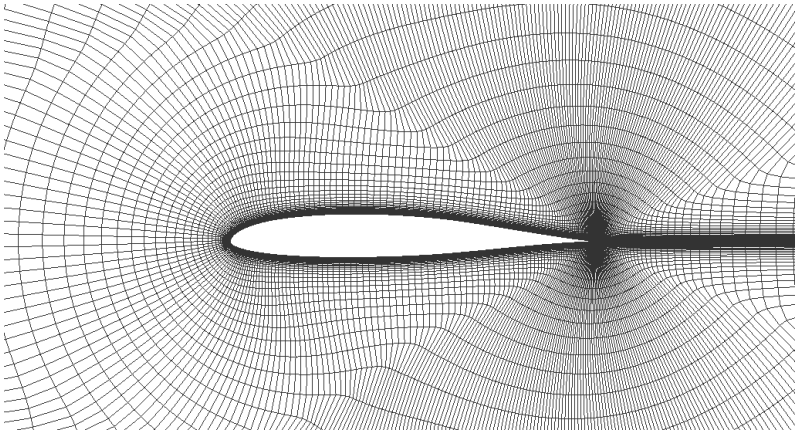


Figure 4.2: *Isolated Airfoil Case: Mesh around the Airfoil.*

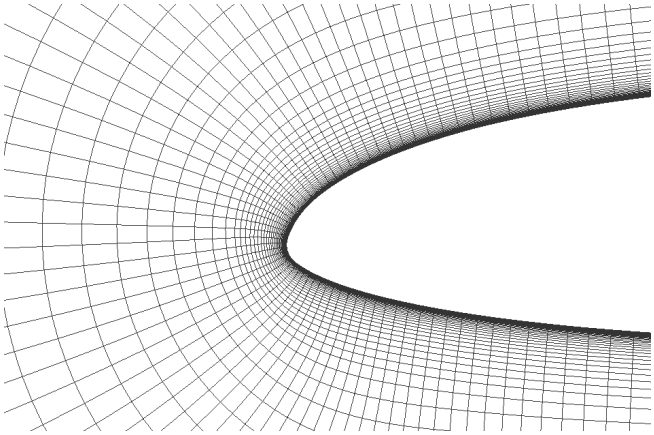


Figure 4.3: *Isolated Airfoil Case: Mesh around the Leading Edge.*

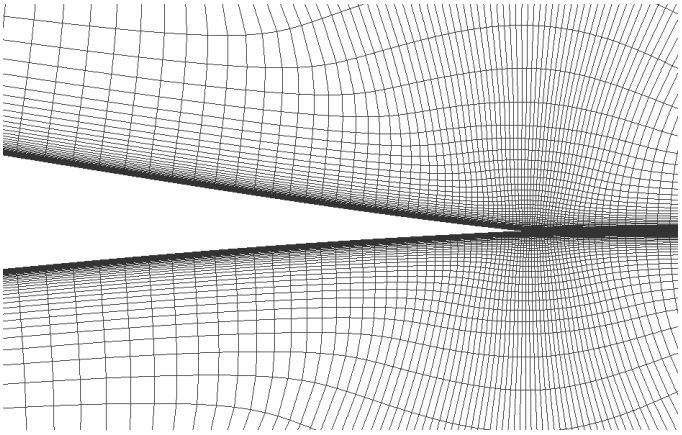


Figure 4.4: *Isolated Airfoil Case: Mesh around the Trailing Edge.*

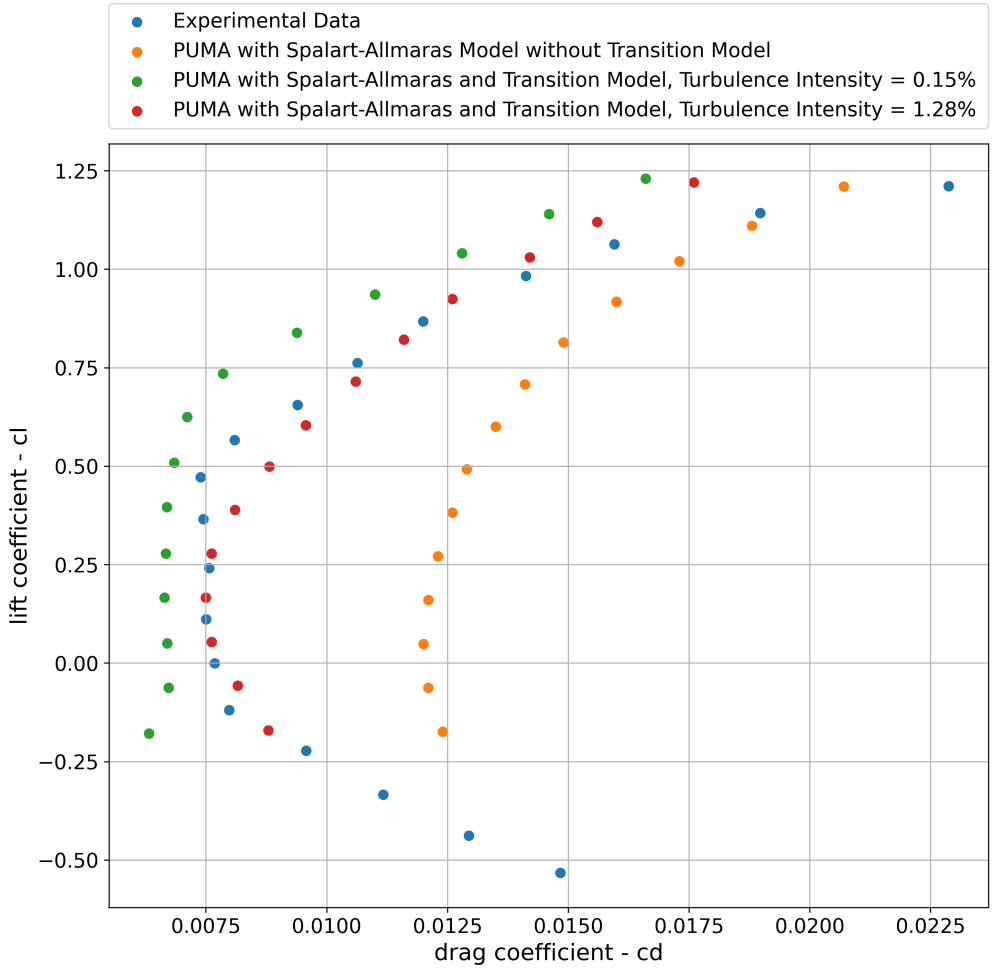


Figure 4.5: *Isolated Airfoil Case: Polar Plots.*

demonstrated in Figure 4.5.

Through this process, the CFD simulations were tuned and validated, establishing a strong alignment with experimental observations.

4.1.3 Parameterization of the Airfoil

In order to control the shape of the airfoil in the optimization process, a parameterization technique is required. Volumetric Non-Uniform Rational B-Splines (NURBS) are used to this end.

The airfoil shape is controlled by a 10×9 NURBS Morphing Box, illustrated in Figure 6.2. Within the morphing box, there are 30 control points highlighted in green which are allowed to move in the vertical (y) direction, i.e. perpendicular to

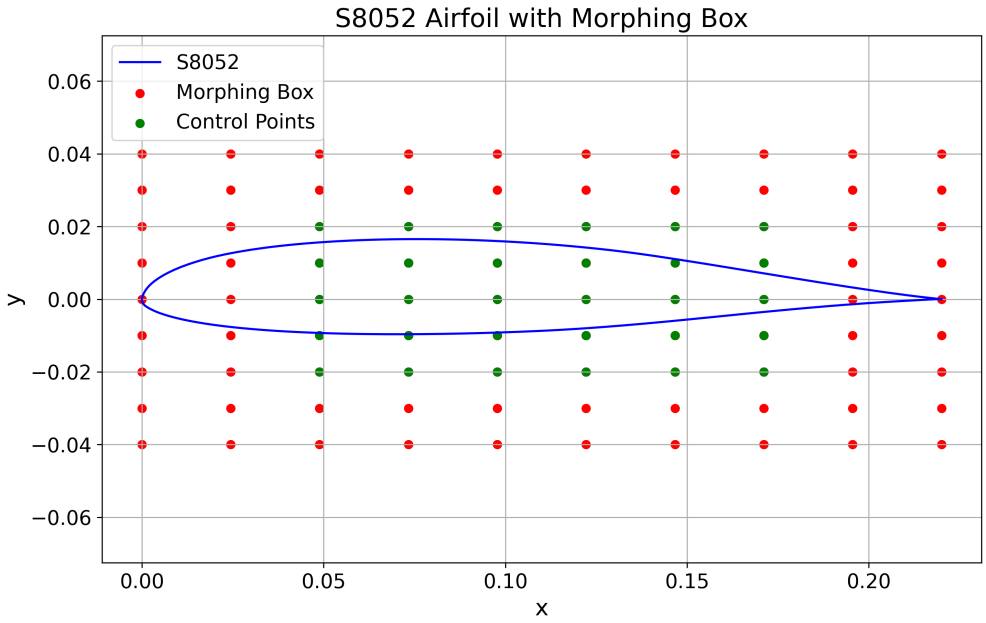


Figure 4.6: *Isolated Airfoil Case: S8052 Airfoil with Morphing Box. Green control points are allowed to move in the y direction only, while the red points are fixed.*

the chord direction. The allowable range of displacement of these control points is limited to the ±20% of the vertical distance between any two adjacent points in the chord direction. The remaining points, depicted in red, are fixed. By optimizing the positions of these 30 control points, new geometries can be generated.

4.1.4 Database Creation

The development of an appropriate DB is an essential component for optimizing the performance of a trained model. In this Diploma Thesis, a DB is created by solving the flow of 200 distinct airfoils using CFD software. To ensure proper coverage of the design space, a Latin Hypercube Sampling (LHS) technique is employed to sample the 30 variables within the pre-defined parameterization range, resulting in 150 distinct geometries. Then, a Random Sampling technique is employed to generate another 50 geometries, while ensuring that each geometry in the DB is unique. The combination of LHS and Random Sampling techniques are used to ensure a representative coverage of the design space in the DB, which can lead to improved performance of the trained model.

Furthermore, to evaluate the performance of the model on unseen data within an acceptable range, an additional test dataset is generated. This test dataset is produced by randomly selecting 30 different geometries and then, solving the flow using

the CFD software. Then, the trained models are tested in this unseen test DB, in order to assess the performance of the model. This extra dataset is used only for checking the quality of the models and is not related to the optimization itself.

4.2 Configuration of the EASY software

In all experimental configurations presented in the following sections, the EASY software is employed as the optimization tool. All attempts of the proposed methods are compared to the results of the EA and MAEA (RBF on-line), implemented in the EASY software, to assess if further improvement is achieved. The EA and MAEA (RBF on-line) have a parent population size of $\mu = 25$ and an offspring population size of $\lambda = 40$

Additionally, the proposed methods are integrated into an EA or MAEA (RBF on-line), depending on the attempt, with the name of the method included in the attempt's name. Both algorithms have the same parent and offspring population size that was mentioned earlier. In the case of MAEA (RBF on-line), the metamodels are activated after the first 80 evaluations.

To ensure fair comparison among the methods, all attempts have been assigned an equal computational budget, set at 500 CFD evaluations. In cases where the utilization of the DB is necessary, the computational cost associated with the DB is considered, resulting in 300 evaluations remaining for the optimization software.

4.3 Implementation of Tree-Based Methods

4.3.1 Tree-Based Methods as Off-line Metamodels in ShpO

Despite the theoretical limitations of Tree-Based Methods, which prevent them from extrapolating into regions of the design space that they are not trained on and limit their predictions to the average values within the DB, these algorithms are tested as off-line trained metamodels in this study.

As previously mentioned, the objective of the problem is to minimize the drag coefficient c_d, with a constraint on the lift coefficient c_L to be greater than the baseline value. Two Random Forest Regression Models and two Gradient Boosting Regression Models are constructed and trained on the aforementioned DB to predict c_D and c_L in the shape optimization process. The hyperparameters of these models are tuned using two Python tools from the scikit-learn library, namely GridSearchCV and RandomizedSearchCV. After selecting some specified parameter values by the user, GridSearchCV makes an exhaustive search creating a model for each combination of these parameters, while RandomizedSearchCV generates random combi-

nations. After comparing the created models, both methods return the model with the highest accuracy.

Two error metrics are used to assess the performance of the trained models on new unseen data, the test DB: the Coefficient of Determination (R^2) and the Mean Absolute Error (MAE) normalized by the maximum difference in c_D or c_L values within the DB ($\Delta c_D^{Database}$ or $\Delta c_L^{Database}$).

The Coefficient of Determination (R^2) is computed using:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.1}$$

where n is the number of the test data points, y_i is the real value, $\hat{y}_i$ is the predicted value and $\bar{y}$ is the mean of the real values. About R^2, a perfect fit of the trained model is indicated by a value of 1, while a value of 0 indicates a poor fit.

The Mean Absolute Error (MAE) is computed using:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.2}$$

The MAE is normalized by the maximum difference in c_D or c_L values within the DB. Smaller values of the normalized MAE indicate a better fit between the predicted and real values. The error metrics of the models evaluated on the test DB are presented in Tables 4.2 and 4.3.

Tables 4.2 and 4.3 indicate that the Gradient Boosting models outperform the Random Forest models. Furthermore, it is evident that the models predicting the c_L demonstrate higher performance compared to the models predicting c_D. In Figures 4.7 and 4.8 the predictions of both models, along with the corresponding real values for the test DB, are presented.

Model	R^2	$MAE/\Delta c_D^{Database}$
Random Forest - Drag Model	0.176	0.156
Gradient Boosting - Drag Model	0.333	0.128

Table 4.2: *Isolated Airfoil Case: Trained regression models' error metrics on their test dataset predictions, for c_D. Values closer to 1 for R^2 and smaller values for $MAE/\Delta c_D^{Database}$ are considered favorable.*

After training the models, the optimization software is initiated, employing the trained models as the evaluation tool. Iterative cycles are used to carry out this process. The EA implemented in the EASY software is initially run for 200 evaluations of the model. It is important to mention that the computational cost per

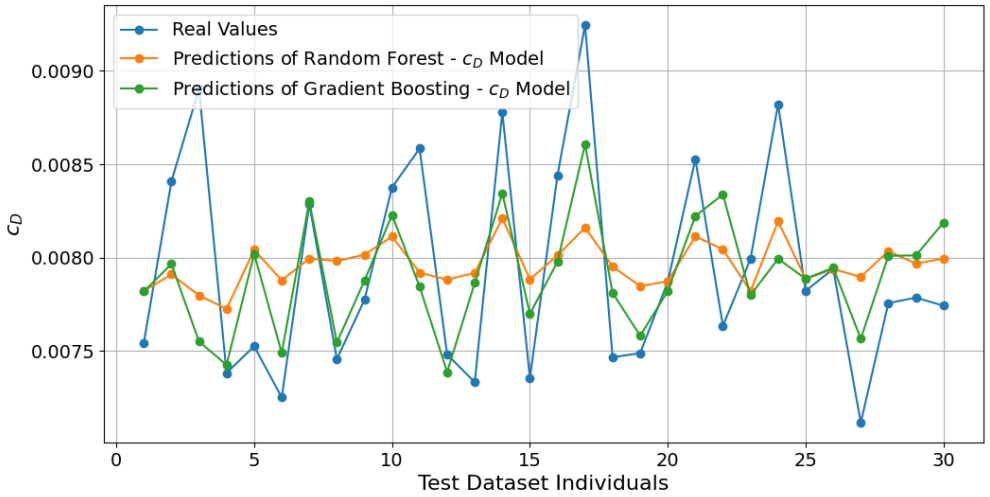


Figure 4.7: *Isolated Airfoil Case: Comparison of Predicted and Actual Values for* c_D.

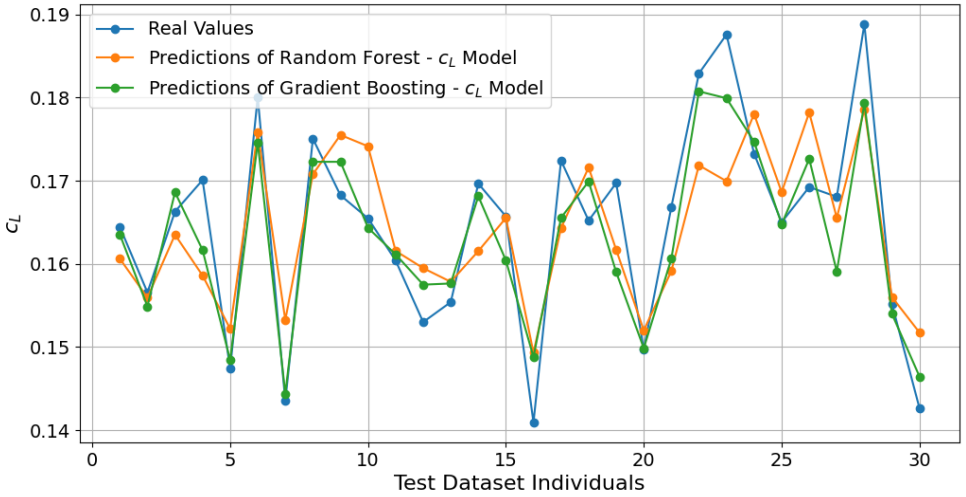


Figure 4.8: *Isolated Airfoil Case: Comparison of Predicted and Actual Values for* c_L.

Model	R^2	$MAE/\Delta c_L^{Database}$
Random Forest - Lift Model	0.642	0.0838
Gradient Boosting - Lift Model	0.834	0.0532

Table 4.3: *Isolated Airfoil Case: Trained regression models' error metrics on their test dataset predictions, for c_L. Values closer to 1 for R^2 and smaller values for $MAE/\Delta c_D^{Database}$ are considered favorable.*

Optimization Type	c_D	c_L
EA	0.00669148	0.193053
MAEA (RBF on-line)	0.00653602	0.176523
MAEA (Random Forest trained off-line)	0.00704201	0.165928
MAEA (Gradient Boosting trained off-line)	0.00667112	0.166800
Baseline	0.00750318	0.165968

Table 4.4: *Isolated Airfoil Case: Results of the optimization targeting minimum c_D using Tree-Based Methods as off-line trained metamodels, compared to the results of the EA and MAEA (RBF on-line). The c_D and c_L values depicted correspond to the best solution in each case computed by the CFD software.*

evaluation using the off-line trained metamodels is significantly lower compared to using the CFD software. Then, the two best solutions are re-evaluated on the CFD software and added to the DB. The models are retrained using the updated DB and the new optimization Cycle begins. This procedure is repeated four times until no better solution can be found. The results are summarized in Table 6.1.

It is noteworthy that in the optimization process for the Random Forest model the baseline is injected into the initial population, whereas in the Gradient Boosting model's optimization process the best individual within the DB is injected into the initial population. That is the reason that no significant improvement is observed in the convergence of this method in Figure 4.10, confirming the theoretical barriers in using these methods as metamodels.

In the convergence plots, the values depicted are the ones that the off-line metamodel generated during optimization. Therefore, at the start of each Cycle, when the model is retrained, although the best solution of the previous Cycle is injected into the initial population of the next Cycle, often the value of the objective function deviates from the model's prediction.

As illustrated in Figures 4.9 4.10, the convergence patterns exhibit negligible improvement in the objective function during the optimization cycles. Notably, even though the convergence plot in Figure 4.10 implies some degree of improvement, in reality, the optimization process fails to find a better solution and returns the individual that it was initially injected with, as the best solution. The improvement

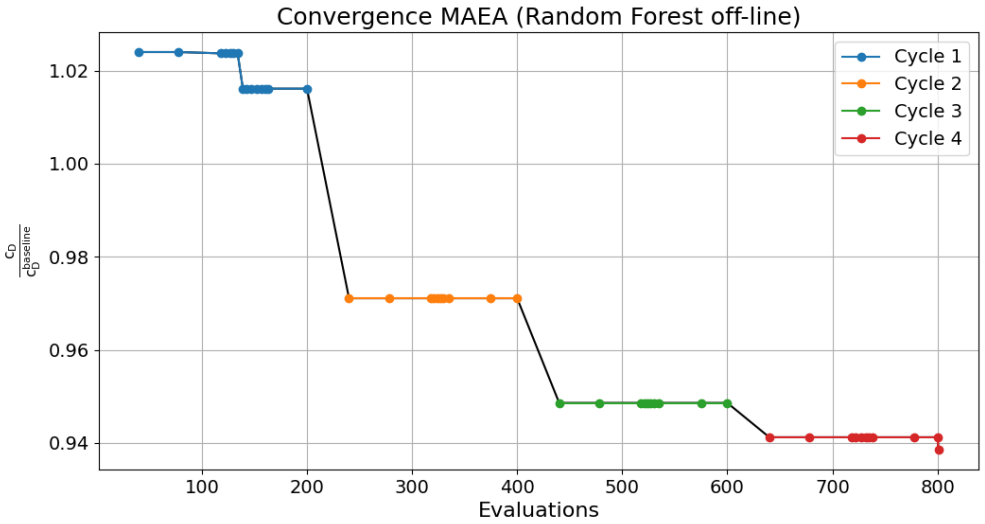


Figure 4.9: *Isolated Airfoil Case: The convergence history of the MAEA (supported by Random Forest trained off-line) optimization. The depicted c_D values are those computed by the metamodel during the optimization process. Similarly, the term "evaluations" refers to evaluations on the metamodel, instead of the CFD code.*

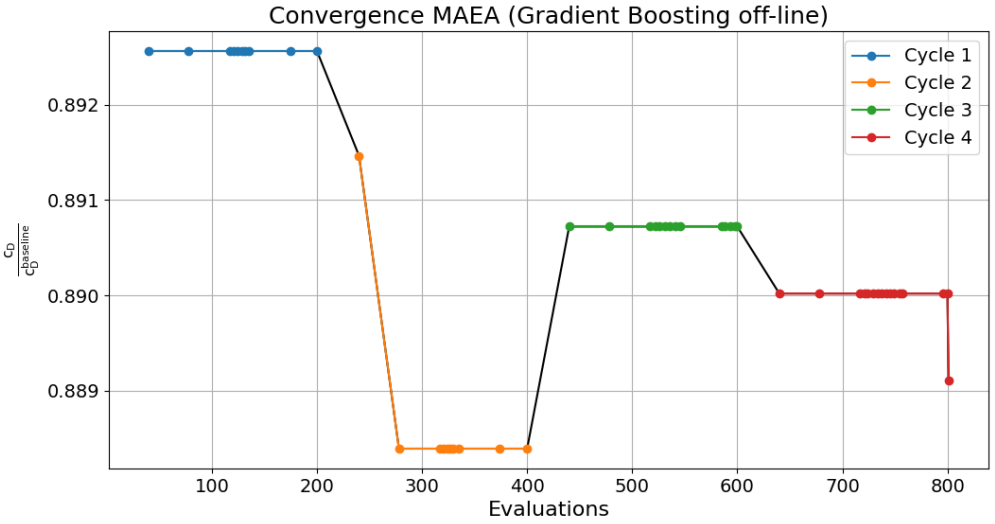


Figure 4.10: *Isolated Airfoil Case: The convergence history of the MAEA (supported by Gradient Boosting trained off-line) optimization. The depicted c_D values are those computed by the metamodel during the optimization process. Similarly, the term "evaluations" refers to evaluations on the metamodel, instead of the CFD code.*

can be attributed to the fact that between the optimization cycles the best solutions are reevaluated and the models are retrained, returning different, hopefully, more accurate predictions for the same individual.

The obtained results unequivocally support our initial assumptions that Tree-Based Methods are inappropriate for use as off-line trained metamodels.

4.3.2 Constraint Based Classification of Candidate Solutions

In this section, the Tree-Based Methods are tested as classifiers to pre-decide about the feasibility of candidate solutions within the optimization process. Feasibility, in this context, refers to the fulfillment of the constraints. The main goal is to incorporate classification models into the optimization procedure in order to classify each solution's feasibility prior to the evaluation software call. Specifically, the aim is to determine whether a given solution satisfies the constraint, giving thus the green light for an evaluation using the computationally expensive CFD software. In this way, the evaluation software is expected to run only for the solutions that appear to be feasible, avoiding expensive evaluations of non-feasible solutions. This procedure is illustrated in Figure 4.11.

For training the classification models, the DB that was described earlier is used. The constraint imposed was that the value of c_L had to exceed the baseline value, where $c_L^{baseline} = 0.166$. Four distinct classes were created based on the c_L values at each point of the DB, as presented in Table 4.5. This way, the airfoils that are unquestionably feasible are given the value of 3, while the non-feasible ones the value of 0, creating a zone of near baseline solutions as shown in Figure 4.12.

c_L Value Range	Class
$c_L < 0.162$	0
$0.162 < c_L < 0.166$	1
$0.166 < c_L < 0.170$	2
$c_L > 0.170$	3

Table 4.5: *Isolated Airfoil Case: The assigned classes for each c_L value range.*

Each point's c_L value is changed in the DB to match its corresponding class value. Three classification models, namely Random Forest, Gradient Boosting and XGBoost, are created and trained using the modified DB. For the hyperparameter tuning of these models, the GridSearchCV and the RandomizedSearchCV tools are used.

The models were evaluated using the Coefficient of Determination (R^2) and the Mean Absolute Error (MAE), as shown in Table 4.6. From this table, it is evident that both the Gradient Boosting and XGBoost models achieved slightly higher accuracy

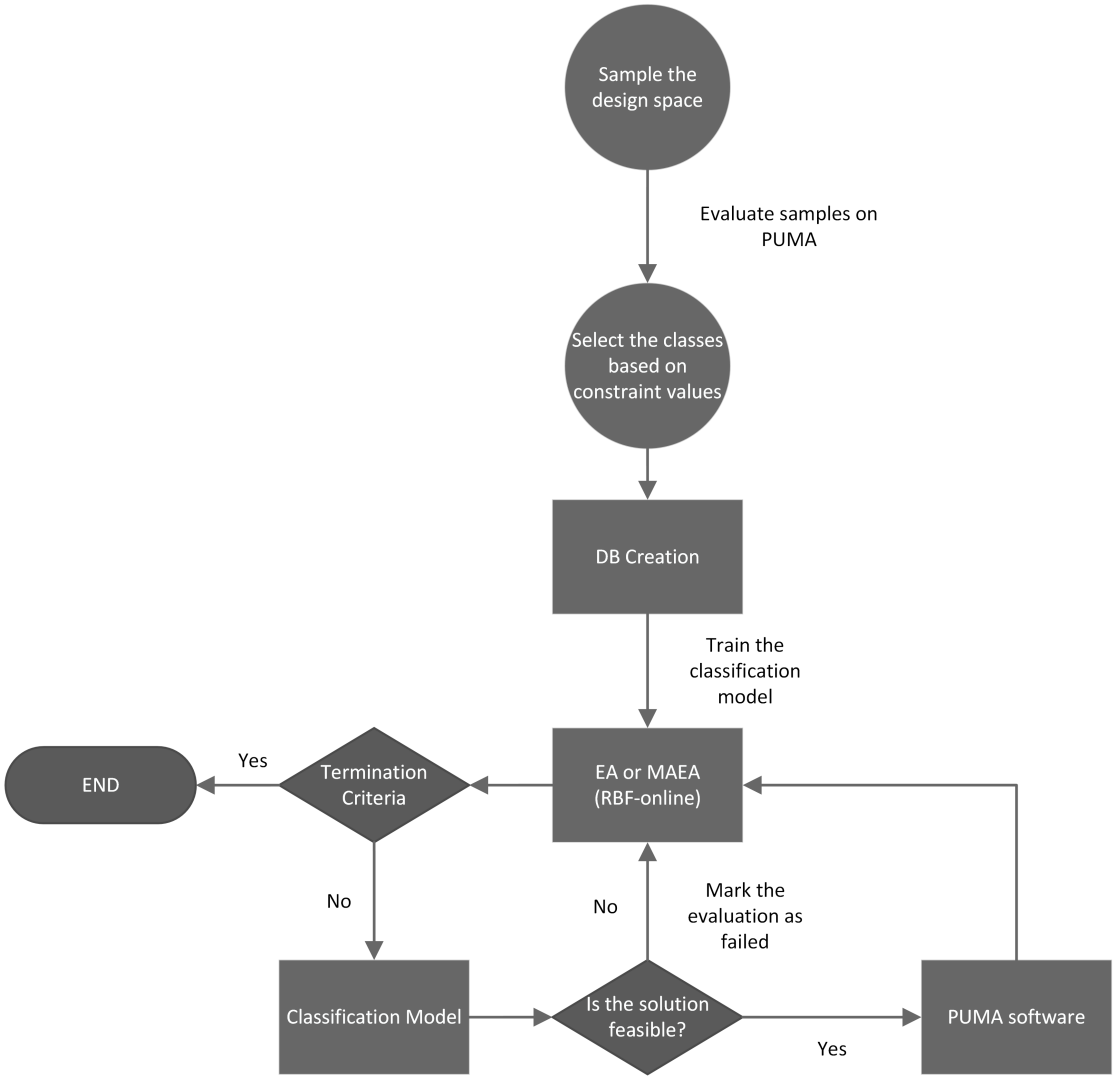


Figure 4.11: *The flowchart of the Constraint Based Classification approach.*

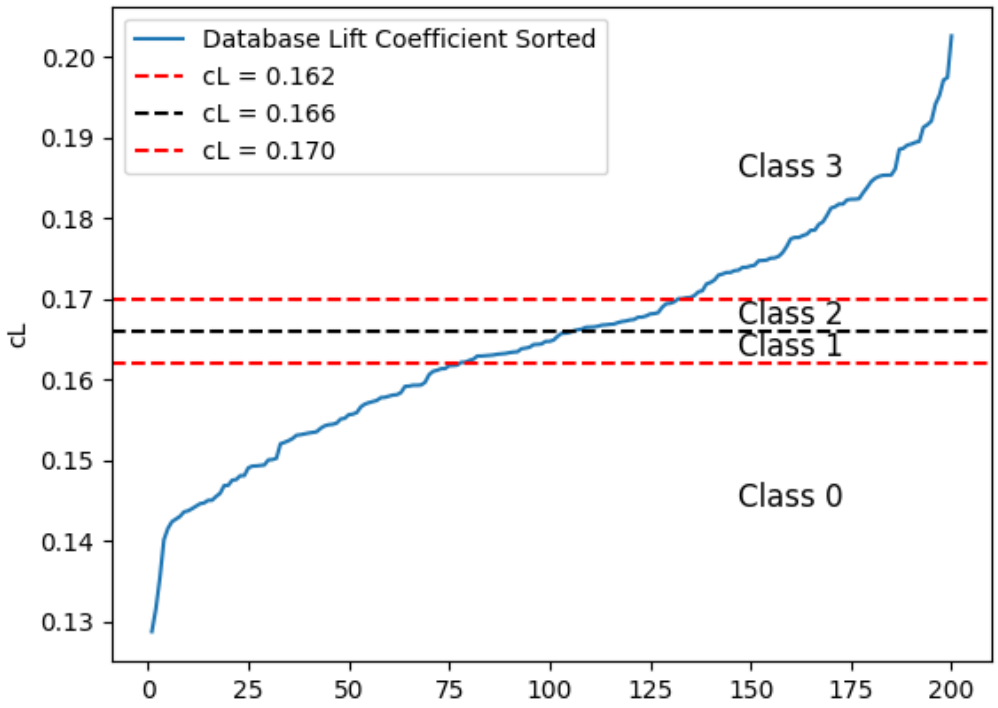


Figure 4.12: *Isolated Airfoil Case: The c_L in the DB sorted, along with the limits between each classification class. For $c_L < 0.162$ class 0 was assigned, class 1 for $0.162 < c_L < 0.166$, class 2 for $0.166 < c_L < 0.170$ and class 3 for $c_L > 0.170$.*

Model	R^2	MAE
Random Forest	-0.107	0.8
Gradient Boosting	0.0776	0.733
XGBoost	0.101	0.767

Table 4.6: *Isolated Airfoil Case: Trained classification models' error metrics on their test DB predictions. Values closer to 1 for R^2 and smaller values for MAE are considered favorable.*

compared to the Random Forest model. Furthermore, Figure 4.13 illustrates the difference between the predicted values of each model and the actual values for each point in the test DB.

After training the classification models, different schemes are implemented. Specifically, in the initial scheme, referred to as *"Test 1"*, the candidate solutions in each generation are classified using the model and if the value of the class is 0 or 1, the solution is not evaluated, during the MAEA-based optimization. Similarly, in *"Test*

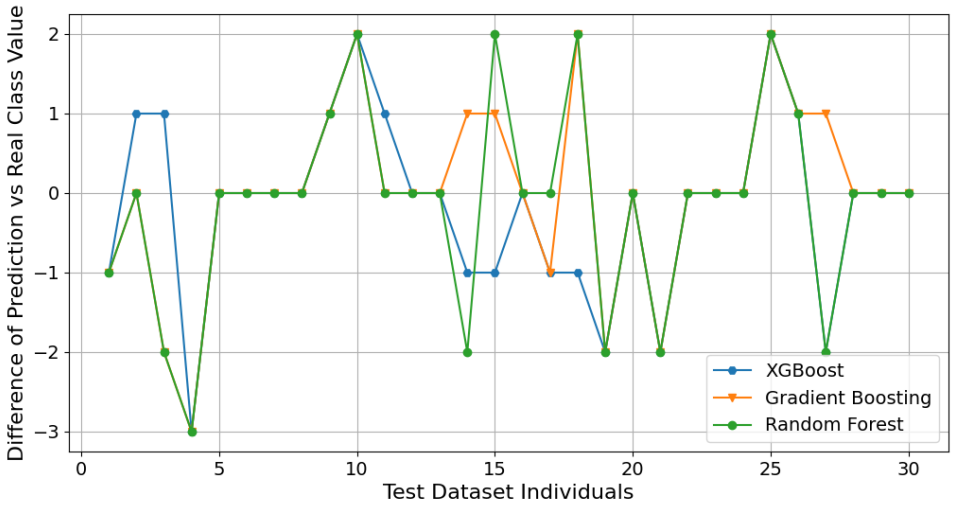


Figure 4.13: *Isolated Airfoil Case: Difference between the Predicted class values and the Real Class value for each point in the test DB, depicted for each model.*

2", if the value of the class is 0, 1 or 2 the solution is not evaluated, while in *"Test 3"* only the class value of 0 is not evaluated. All schemes are summarized in Table 6.4. Each of these schemes is tested with the Random Forest classifier, while Test 2 is tested with Gradient Boosting and XGBoost classifiers also.

Schemes	Classes that are not evaluated
"Test 1"	0, 1
"Test 2"	0, 1, 2
"Test 3"	0

Table 4.7: *The different schemes that are implemented in the constraint-based classification approach. In each scheme, if the predicted class value of a solution belongs to the depicted values, the solution is not evaluated by the CFD software.*

These schemes are evaluated using a Random Forest classifier. The Test scheme that produced the best result with the Random Forest classifier, in Test 2, is used to test the Gradient Boosting and XGBoost models. The goal of this comparison was to determine which model performs better in the constraint-based classification.

All these Tests are compared with the result of an EA and MAEA (RBF on-line) using the EASY software. The size of the DB, which is used to train the classification models, influences the overall cost of the tests. In this case, the size of the DB is 200, giving the optimization software room to carry out 300 evaluation calls, adding up to a total of 500 evaluation calls. As a result, in each Test, the best individual of the DB, that returned an acceptable class value, is injected into the initial population.

Optimization Type	c_D	c_L	% of Objective Improvement
EA	0.00669148	0.193053	10.82%
MAEA (RBF on-line)	0.00653602	0.176523	12.89%
Test 1 (Random Forest) - EA	0.00662657	0.216523	11.68%
Test 2 (Random Forest) - EA	0.00644965	0.172007	14.04%
Test 3 (Random Forest) - EA	0.00662657	0.216523	11.68%
Test 2 (Gradient Boosting) - EA	0.00661844	0.183463	11.79%
Test 2 (XGBoost) - EA	0.0068241	0.184521	9.05%
Baseline	0.00750318	0.165968	

Table 4.8: *Isolated Airfoil Case: Results of the optimization procedure using Tree-Based Methods as Classifiers, compared to the results of the EA and MAEA (RBF networks trained on-line). The c_D and c_L values correspond to the best solution in each case computed by the CFD software.*

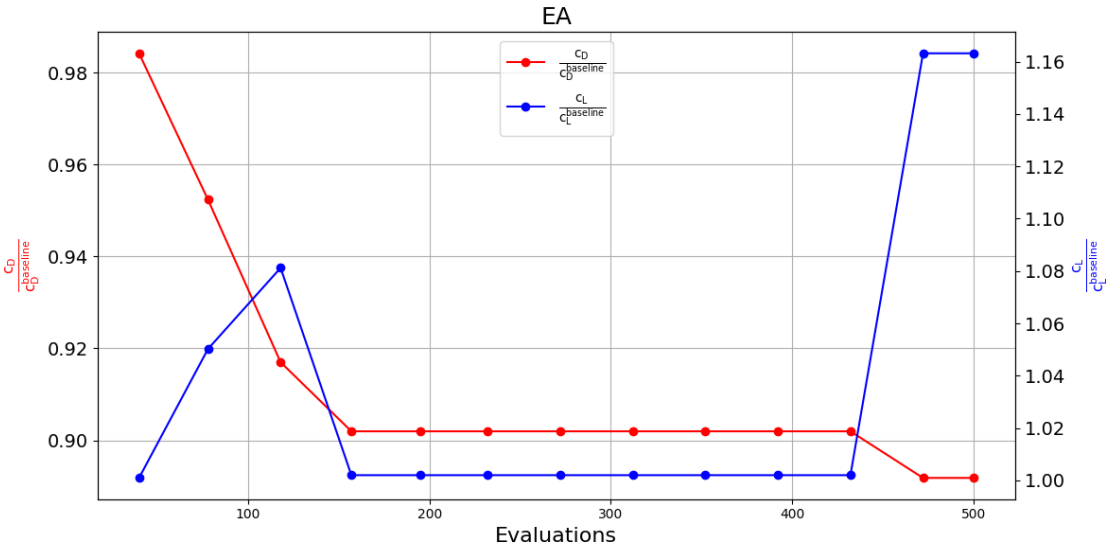


Figure 4.14: *Isolated Airfoil Case: The convergence history of the EA optimization, in which c_D is the target and c_L is the constraint, based exclusively on the CFD code.*

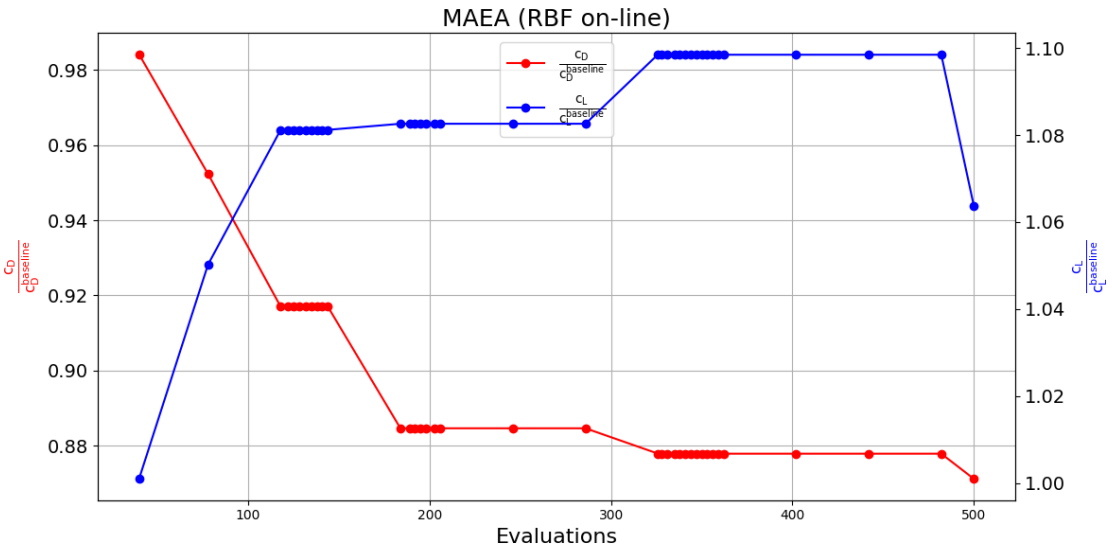


Figure 4.15: *Isolated Airfoil Case: The convergence history of the MAEA (RBF on-line) optimization, in which c_D is the target and c_L is the constraint.*

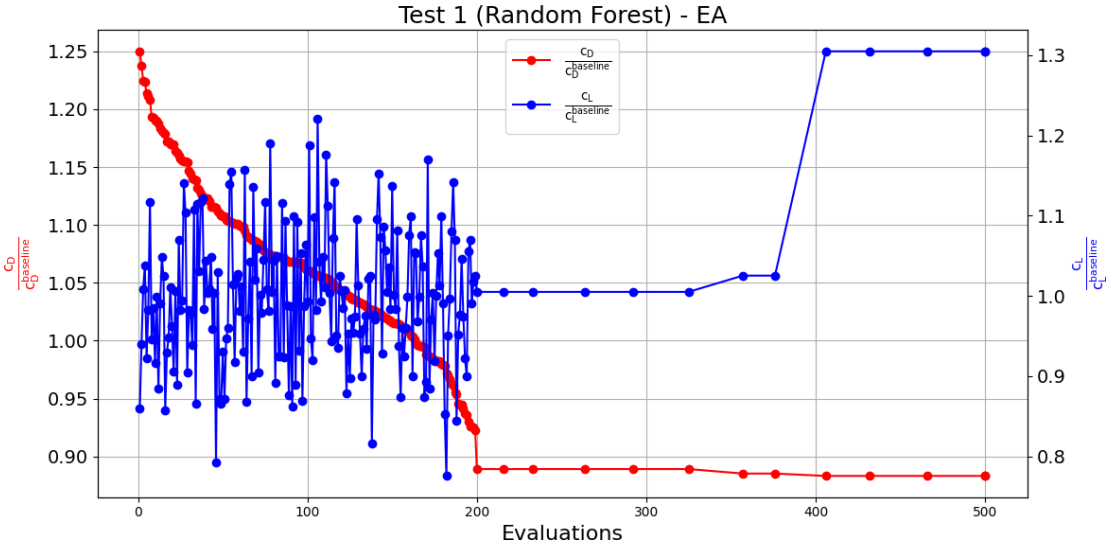


Figure 4.16: *Isolated Airfoil Case: The convergence history of the Random Forest "Test 1" optimization, in which c_D is the target and c_L is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the c_D values.*

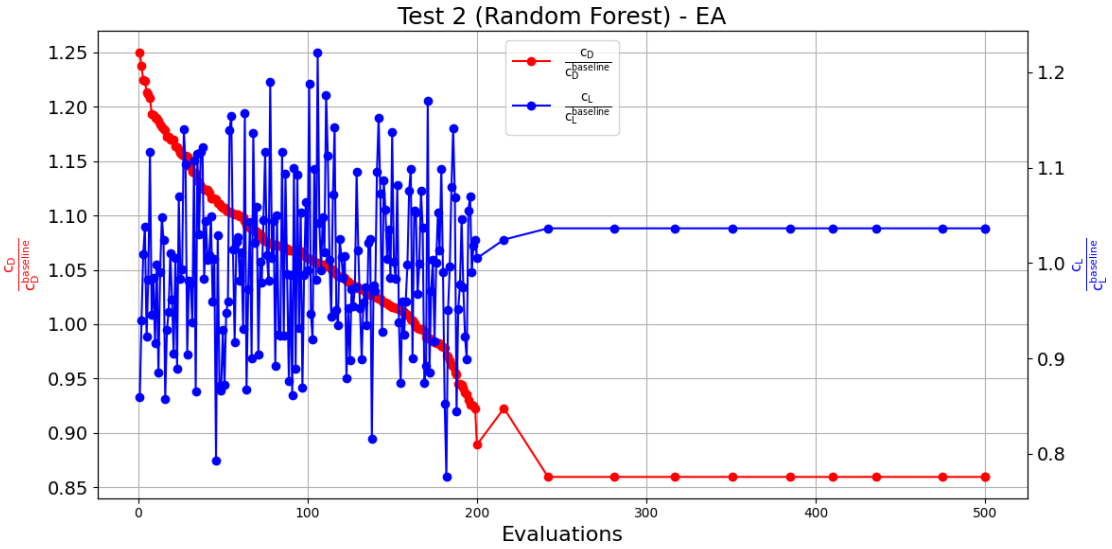


Figure 4.17: *Isolated Airfoil Case: The convergence history of the Random Forest "Test 2" optimization, in which c_D is the target and c_L is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the c_D values.*

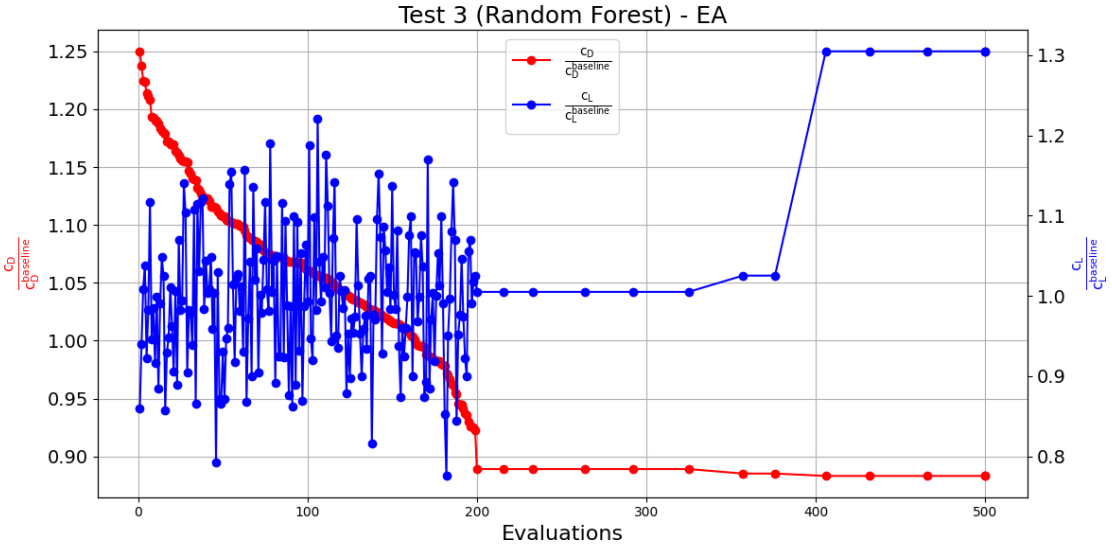


Figure 4.18: *Isolated Airfoil Case: The convergence history of the Random Forest "Test 3" optimization, in which c_D is the target and c_L is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the c_D values.*

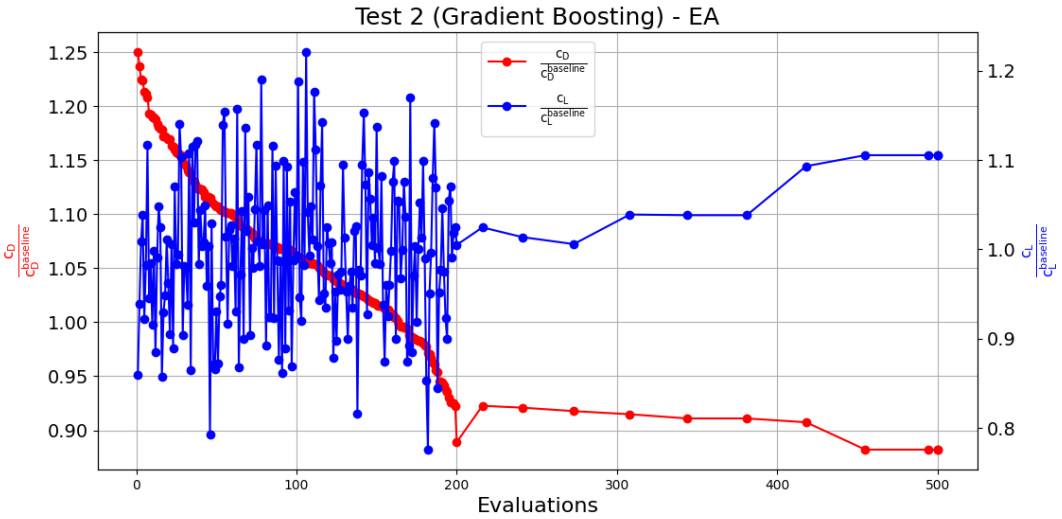


Figure 4.19: *Isolated Airfoil Case: The convergence history of the Gradient Boosting "Test 2" optimization, in which c_D is the target and c_L is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the c_D values.*

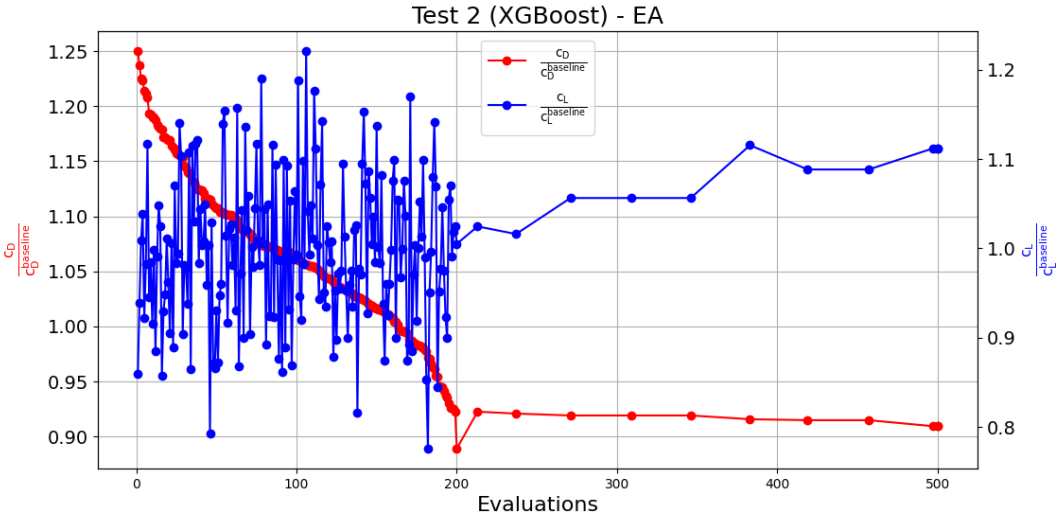


Figure 4.20: *Isolated Airfoil Case: The convergence history of the XGBoost "Test 2" optimization, in which c_D is the target and c_L is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the c_D values.*

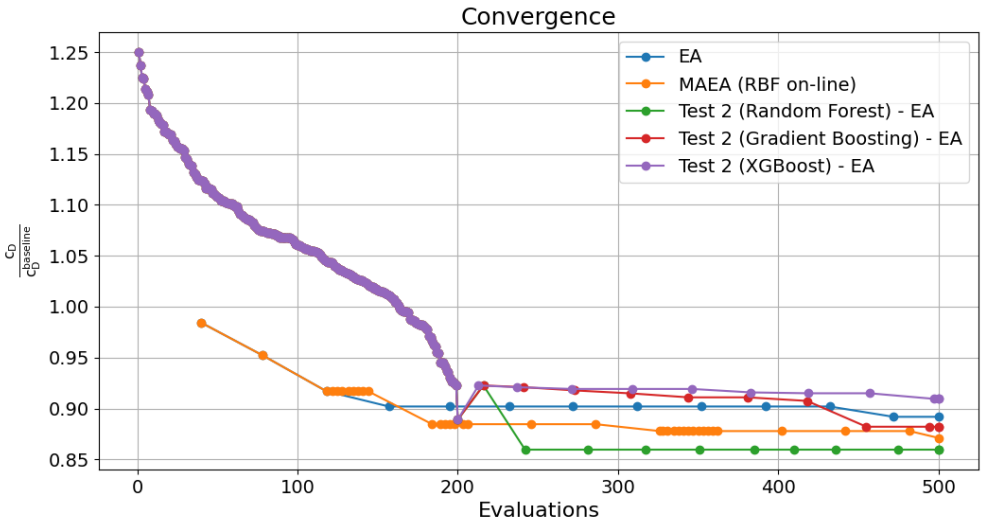


Figure 4.21: *Isolated Airfoil Case: Comparison of the convergence of the objective for different tests.*

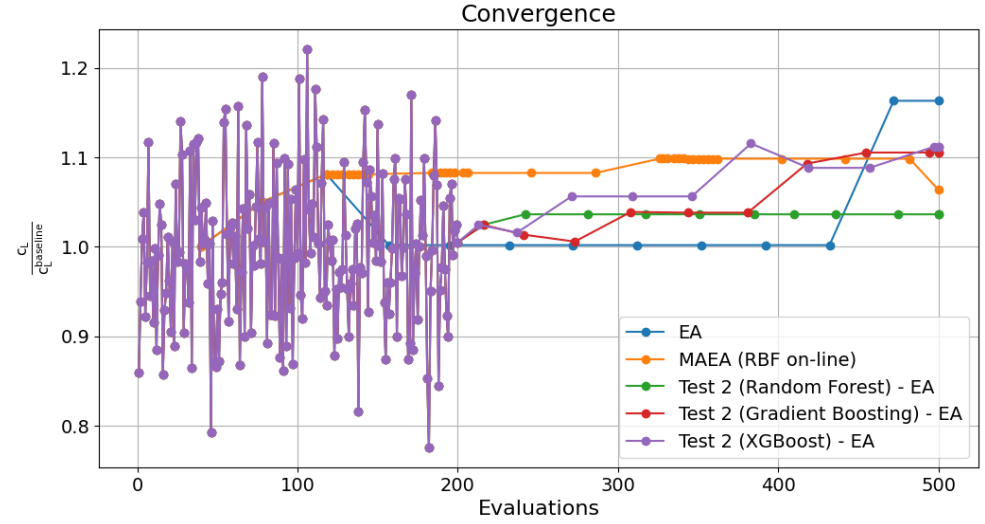


Figure 4.22: *Isolated Airfoil Case: Comparison of the convergence of the constraint for different tests.*

The findings of this study are presented in Table 4.8, while the convergence patterns are depicted in Figures 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 6.4 and 4.22.

As it can be seen from the Table 4.8, all experimental variations manage to achieve sufficient results. However, only the Random Forest model with "Test 2" configuration performs better than the MAEA (RBF on-line) and attains a better solution. In particular, this configuration gives a 14% improvement in the objective function, outperforming the MAEA configuration which achieves an improvement by ∼13%. From Figure 6.4 it is clear that the Random Forest "Test 2" variant manages to find this solution within approximately 250 evaluations, indicating this occurred 50 evaluations after the initiation of the optimization process, considering 200 of those are used for the DB. It is important to acknowledge that this result can be due to the stochastic elements of the whole process, ranging from the selection of the DB to the EA itself.

The Gradient Boosting and XGBoost variations of "Test 2", do not achieve a better result than the Random Forest model. These variations also fail to surpass the MAEA solution, indicating that the Random Forest model with the "Test 2" scheme exhibits the best performance among the three classification models.

4.3.3 Design Space Exploration with Decision Trees

As previously discussed, the DT algorithm partitions the design space into different regions and assigns a value to each region, which is determined as the average value of the points in the DB that belong to this region. In this way, it is possible to locate the best region and initiate the optimization software there by tweaking the design variable ranges, thereby guiding the optimization process toward the optimal solution with a reduced number of evaluation software calls. The flowchart of this process is shown in Figure 4.23.

However, it is critical to recognize the inherent risk that lies in this approach. While focusing on a specific region increases the chance of identifying better solutions, there is the possibility that the optimal solution belongs to another region. As a result, there is always the risk that this method could potentially restrict the EA in a suboptimal space.

In this study, a CART DT is trained using the aforementioned DB. The optimal hyperparameters for the DT model are determined using GridSearchCV and RandomizedSearchCV. Specifically, the DT is chosen to have 10 leaf nodes, resulting in the division of the design space into 10 different regions, as depicted in Figure 4.24.

It is important to note that various DT structures are tested with different hyperparameters, training DB sizes and number of leaf nodes. This is used to examine the way different models split the design space and to determine how the DT training process affects the design space division.

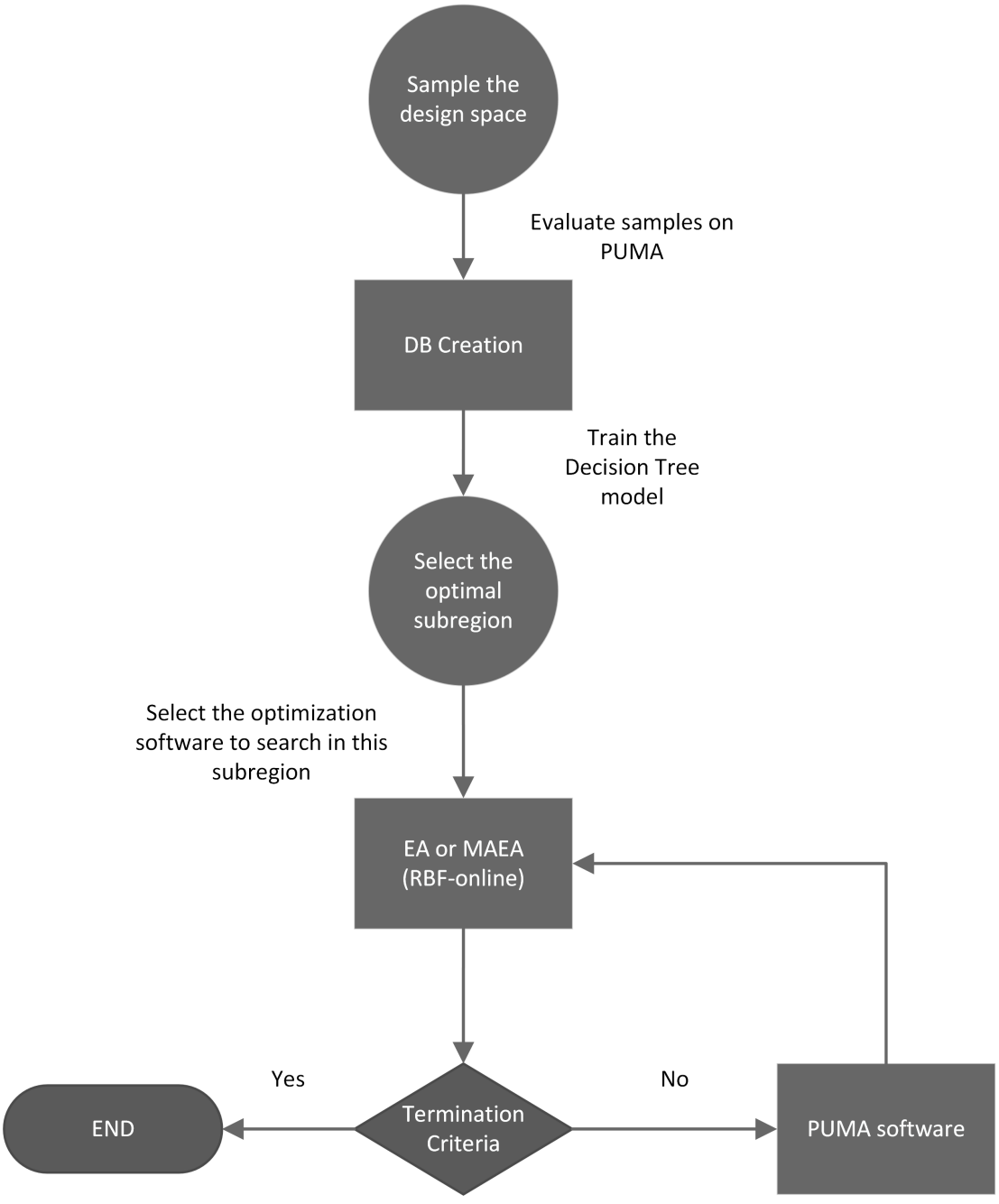


Figure 4.23: *The flowchart of the Design Space Exploration approach with Decision Trees.*

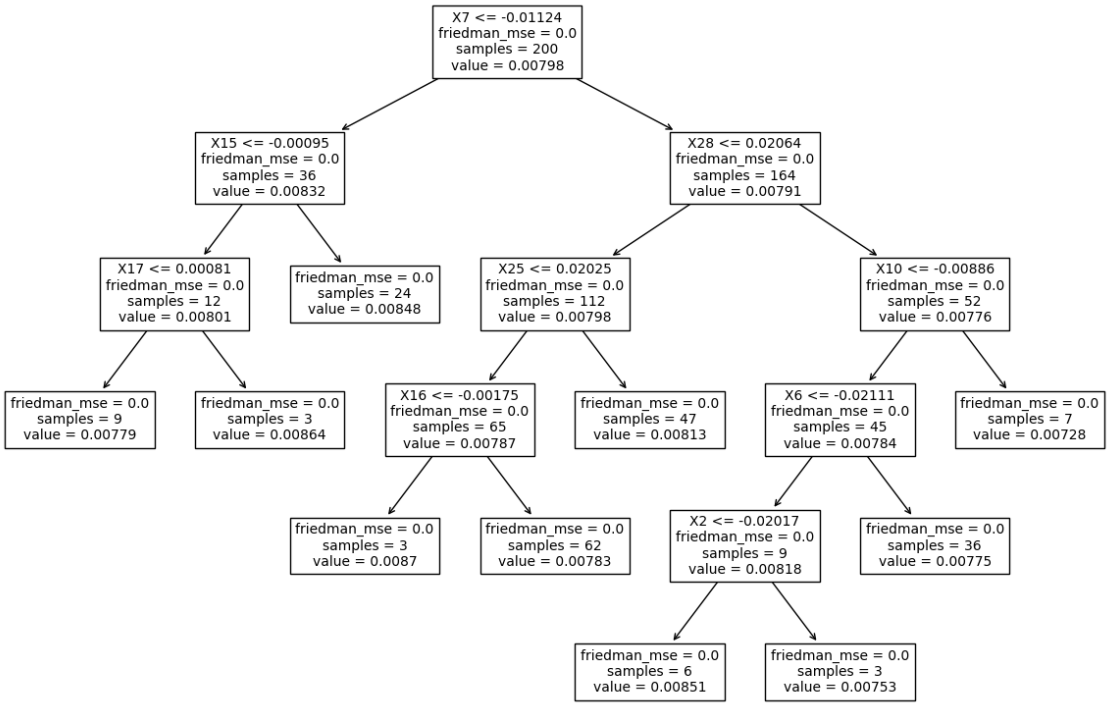


Figure 4.24: *Isolated Airfoil Case: The trained DT that was used to partition the design space.*

A crucial aspect is to identify the optimal region within these partitions. Several approaches are explored through various attempts. Initially, in Try 1, the region with the best value, specifically the minimum, among the leaf nodes is selected. This approach aims to prioritize the region that the DT model identifies as the best, based on the available data. Subsequently, in Try 2, the region in which the best individual of the DB belongs was chosen. In this Try, the knowledge accumulated by the DB is leveraged, allowing the optimization process to focus on the region that has previously shown promising results.

The goal of these attempts is to identify the region within the DT partitions with the highest potential for optimal solutions, identified either through the DT model's evaluation or the insights provided by the DB.

Since the computational cost of generating the DB has already been charged, in each of these tries, the best individual of the DB that belongs to the corresponding region is injected into the initial population. By starting the optimization process with the best individuals from the DB, it is anticipated to further improve the efficiency and effectiveness of the optimization process while capitalizing on the computational investment made during the creation of the DB.

The results are summarized in Table 4.9, while the convergences are shown in Figures 6.6 and 4.26. It is evident that all attempted approaches produce sufficient results.

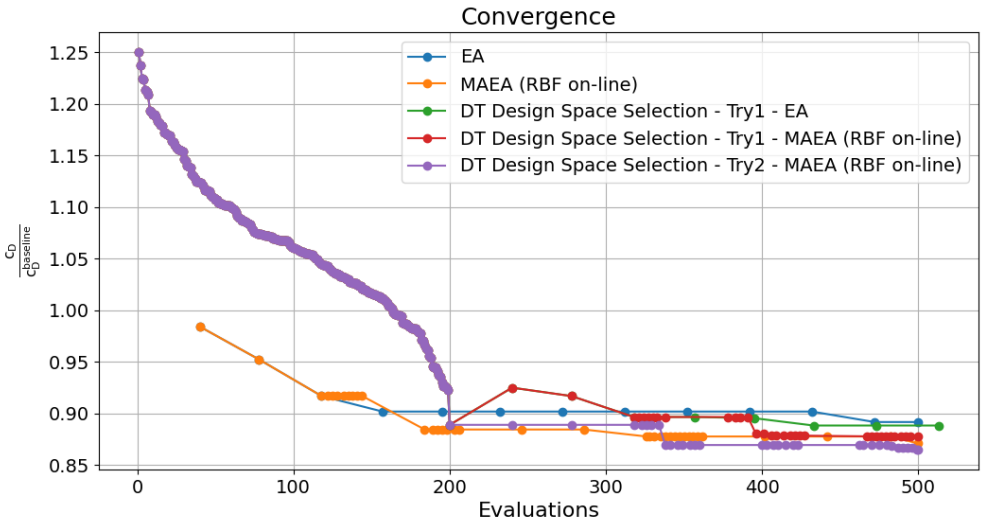


Figure 4.25: *Isolated Airfoil Case: Evaluating the Convergence of the objective for Various Design Space Exploration Techniques.*

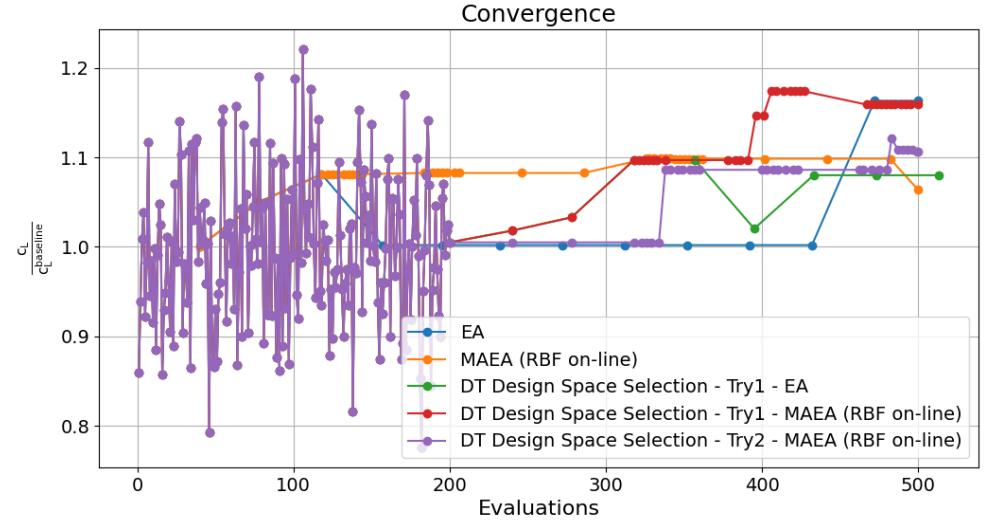


Figure 4.26: *Isolated Airfoil Case: Evaluating the Convergence of the constraint for Various Design Space Exploration Techniques.*

Optimization Type	c_D	c_L	% of Objective Improvement
EA	0.00669148	0.193053	10.82%
MAEA (RBF on-line)	0.00653602	0.176523	12.89%
DT Design Space Selection - Try1 - EA	0.00666616	0.179254	11.16%
DT Design Space Selection - Try1 - MAEA (RBF on-line)	0.00658729	0.192346	12.21%
DT Design Space Selection - Try2 - MAEA (RBF on-line)	0.0064909	0.183642	13.49%
Baseline	0.00750318	0.165968	

Table 4.9: *Isolated Airfoil Case: Results of the optimization procedure using the Design Space Exploration approach, compared to the results of the EA and MAEA (RBF on-line). The c_D and c_L values correspond to the best solution in each case computed by the CFD software.*

As a DB is needed for the DT, all tries are made over a period of 300 evaluations, including an additional 200 evaluations for the DB, for a total of 500 evaluations.

In Try1 using the EA, a slightly better result than the EA is obtained, though it does not outperform the MAEA's (RBF on-line) performance. In Try1 with the MAEA (RBF on-line) the result is comparable to that of the MAEA (RBF on-line) but still falling short. Finally, the Try2 with the MAEA (RBF on-line) achieves slightly better result than the MAEA (RBF on-line), exhibiting a 13.5% improvement in the objective function, shortly after the optimization software is initiated.

4.3.4 Combined Application of Methods

In order to implement any of the presented methods, the creation of a DB is necessary. Therefore, it is possible to combine the best-performing approaches without incurring any extra computational cost. In this section, the combination of the best-performing attempts from each approach is explored: "Test 2 (Random Forest) - EA", which has demonstrated the best performance out of the classification attempts, with the "DT Design Space Selection - Try2 - MAEA (RBF on-line)" which is the best performer in the design space exploration approach.

Specifically, the Random Forest classification model with the "Test 2" scheme is implemented in the subregion of the design space that is used in the "DT Design Space Selection - Try2 - MAEA (RBF on-line)" attempt. The best individual contained into the DB, which was classified as class 3 from the classification model, is injected into the initial population. This is the second-best individual in the DB. Two attempts are carried out, namely "Combination Test - EA" and "Combination Test - MAEA (RBF on-line)", implementing the ordinary EA and the MAEA (RBF on-line), respectively.

The results are presented in Table 4.10, while Figures 6.8 and 4.28 demonstrate the convergence patterns for the objective and constraint, respectively. As shown in Table 4.10, both attempts outperform any other test that is presented in this chapter. Particularly, "Combination Test - EA" demonstrates the best solution in

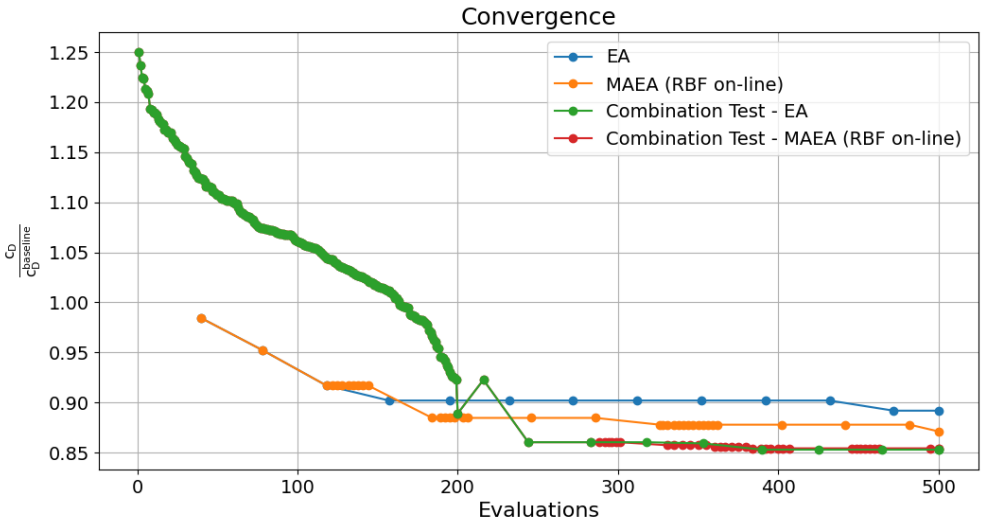


Figure 4.27: *Isolated Airfoil Case: Evaluating the Convergence of the objective for the Combination Tests.*

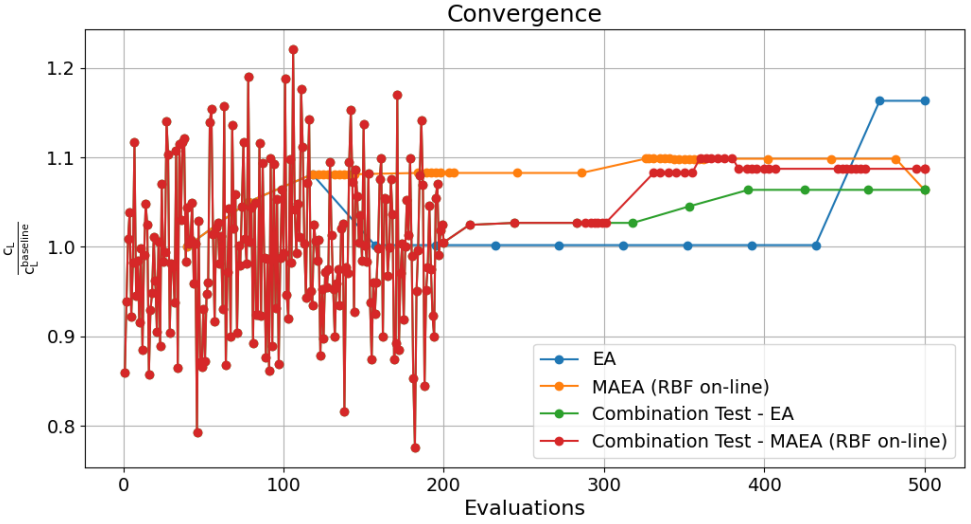


Figure 4.28: *Isolated Airfoil Case: Evaluating the Convergence of the constraint for the Combination Tests.*

Optimization Type	c_D	c_L	% of Objective Improvement
EA	0.00669148	0.193053	10.82%
MAEA (RBF on-line)	0.00653602	0.176523	12.89%
Combination Test - EA	0.00639942	0.176541	14.71%
Combination Test - MAEA (RBF on-line)	0.00640940	0.180446	14.58%
Baseline	0.00750318	0.165968	

Table 4.10: *Isolated Airfoil Case: Results of the optimization procedure using the Combination of Methods approach, compared to the results of the EA and MAEA (RBF on-line). The c_D and c_L values correspond to the best solution in each case computed by the CFD software.*

terms of objective function improvement, achieving an improvement by ∼15%. It is worth noting that both attempts converge to the solutions within approximately 190 evaluations after the optimization procedure is initiated. Furthermore, satisfactory solutions are found even 50 evaluations after the initialization of the optimization software, as depicted in Figure 6.8.

4.4 DNN as Off-line Metamodel

For comparison of the results that were obtained with methods that are already employed by the PCOpt/NTUA, the airfoil case is subjected to optimization using an off-line trained metamodel based on DNN, in the off-line metamodel framework that has already been described.

Initially, two DNN models are constructed using the TensorFlow library in Python, one for predicting the drag coefficient c_D and another for the lift coefficient c_L. The hyperparameters of these DNN models are determined through trial and error. Table 4.11 summarizes the DNN configuration for each model.

The training progress of the c_D model is illustrated in Figure 4.29, depicting that after 3500 epochs the losses cannot decrease any further. As a result, the training process is terminated, and the same applies to the c_L model.

During the training of the DNN models, the design variables are scaled within the range of (0,1) using their parameterization ranges. Similarly, the outputs, c_D and

Model	Layers	Neurons/Layer	Act. Functions
c_D	6	(256, 512, 512 ,512, 256, 1)	ReLu/tanh
c_L	7	(256, 256, 512 ,512, 256, 256, 1)	ReLu/tanh

Table 4.11: *Isolated Airfoil Case: The DNN configurations obtained through trial-and-error.*

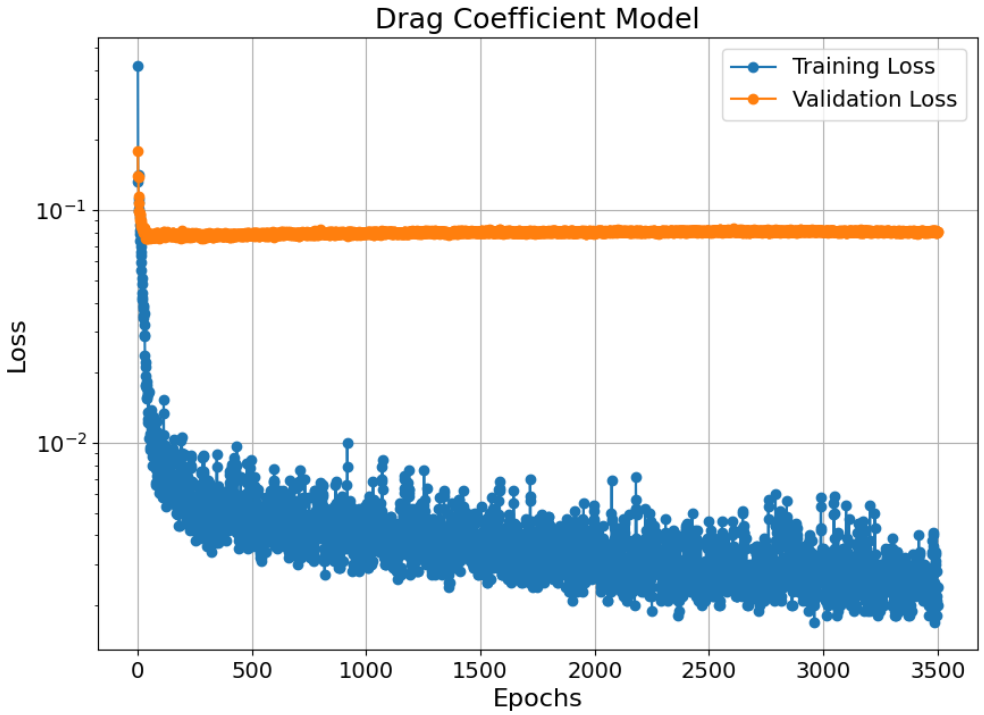


Figure 4.29: *Isolated Airfoil Case: Training and Validation loss during the training of the DNN model for c_D. The y-axis is presented on a logarithmic scale.*

c_L, are scaled and then transformed back to the original values.

The optimization is carried out in 3 Cycles. After each Cycle, the solution obtained is re-evaluated and added to the DB. The DNN model is then retrained on the updated DB and the next Cycle utilizes the re-trained model.

At first, the best solution stored in the DB is injected into the initial population, because the computational cost of the DB has already been charged. After each Cycle, the best solution obtained is injected into the initial population of the next Cycle.

The convergence history of the MAEA (DNN trained off-line) optimization is depicted in Figure 4.30. From the Figure, it is evident that this attempt does not manage to obtain a better solution than the one that was injected with, similar to the off-line optimization with the Tree-Based methods. Notably, because no improvement is made in the first Cycles, Cycle 3 is configured with the double number of evaluations, specifically 400, and no change is observed.

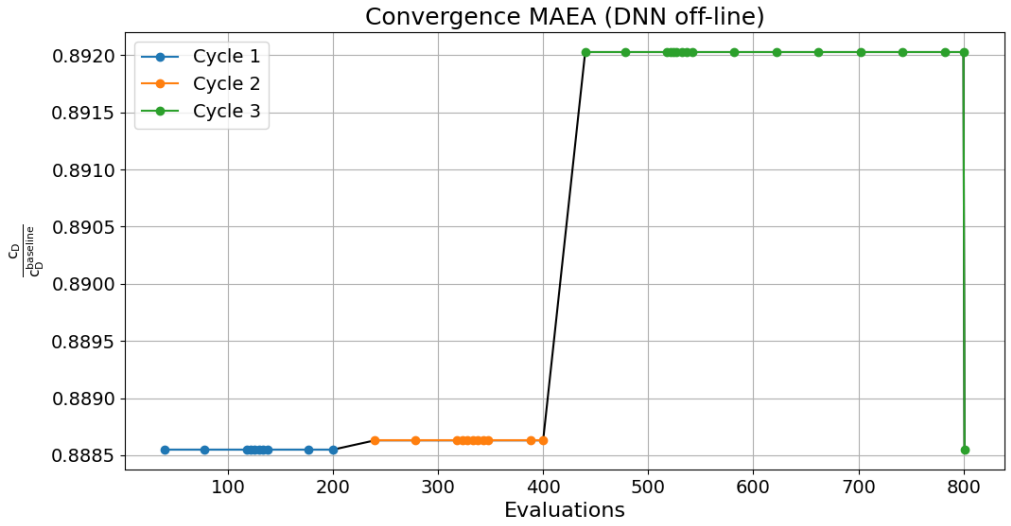


Figure 4.30: *Isolated Airfoil Case: The convergence history of the MAEA (supported by DNN trained off-line) optimization. The depicted c_D values are those computed by the metamodel during the optimization process. Similarly, term "evaluations" refers to evaluations on the metamodel, instead of the CFD code.*

4.5 Aggregated Results

First of all, the Tree-Based Methods were implemented as off-line trained metamodels. Nonetheless, as expected, these attempts failed, because of the theoretical limitations of these methods. It is worth mentioning that the same framework was applied using a DNN model as the off-line trained metamodel. However, similar to the Tree-Based Methods, this approach did not yield any improvement.

Then, the constraint-based classification approach was tested. The main objectives were to assess the performance of this approach and to identify the best classification scheme and the best Tree-Based Model. While, sufficient results were achieved by all the attempts, only "Test 2 (Random Forest) - EA" outperformed the MAEA (RBF on-line).

Next, the design space exploration approach using DTs was examined. In this approach, the EA or MAEA was initiated within a subregion of the design space determined by the DT. The main objectives were to evaluate the performance of this approach and identify the subregion that contains the optimal solution. Various attempts were made, achieving sufficient results, but only "DT Design Space Selection - Try2 - MAEA (RBF on-line)" found a superior solution to the MAEA (RBF on-line). This proves the capability of this approach, although the selection of the subregion remains uncertain.

Lastly, the best approaches from the constraint-based classification and the design

space exploration approaches were combined to assess whether the combination could bring an improved solution or faster convergence. Both attempts achieved superior solutions compared to all the previous ones with fast convergence. This demonstrates the potential of this combined approach, only if the appropriate classification scheme and the best subregion are identified.

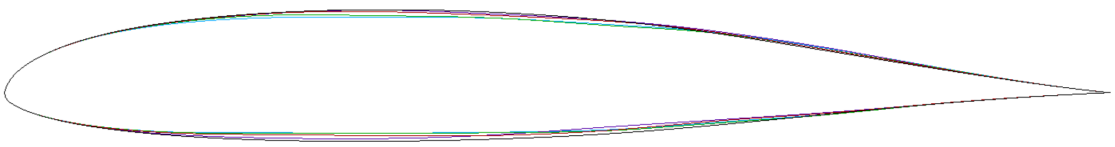
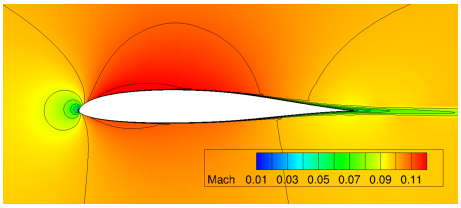
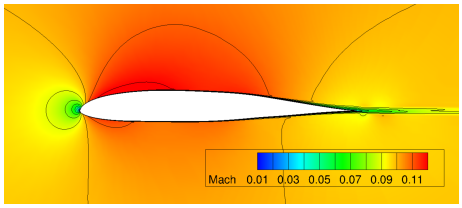


Figure 4.31: *Isolated Airfoil Case: The airfoil shape of the baseline (black) and the shape of the best solutions of the: MAEA (RBF on-line) (red), "Test 2 (Random Forest) - EA" (green), "DT Design Space Selection - Try2 - MAEA (RBF on-line)" (purple) and "Combination Test - EA" (light blue).*

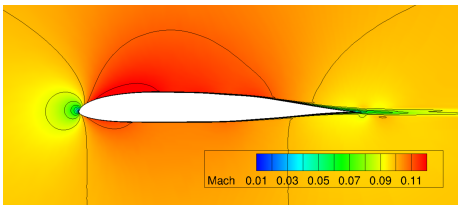
The baseline shape and the shapes of the best solutions from the best-performing attempts from each approach are depicted in Figure 4.31. Also, in Figures 4.32, the Mach number iso-areas computed by PUMA for the same attempts, are shown.



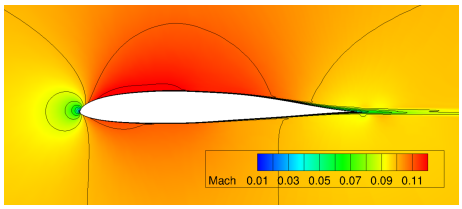
(a) Baseline



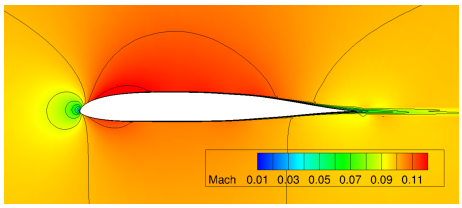
(b) MAEA (RBF on-line)



(c) Test 2 (Random Forest) - EA



(d) DT Design Space Selection - Try2 - MAEA (RBF on-line)



(e) Combination Test - EA

Figure 4.32: *Isolated Airfoil Case: Mach number iso-areas computed by PUMA for the best solution of the shape optimization by different attempts.*

Chapter 5

Application of Tree-Based Methods in Internal Aerodynamic ShpO with EAs

In this chapter, the applications Tree-Based Methods in the context of ShpO with EAs are tested in an S-Bend Duct case. The problem was configured as Single-Objective Optimization (SOO) with the objective of minimizing the total pressure losses, while imposing a constraint on the volume of the duct to be smaller than the baseline value.

5.1 The S-Bend Duct Case

The optimization process focuses on the S-Bend Duct case. Figure 5.1 shows the S-Bend Duct's geometry.

The flow is laminar, with a Reynolds number of $Re = 1000$, based on its inlet width. The PUMA software was used for all the CFD analyses.

5.1.1 Computational Mesh

The mesh, Figure 5.1, consists of 9000 nodes. This is a structured mesh which will be treated as an unstructured by the flow solver. The mesh has variable node density, which is enhanced near the walls of the duct, ensuring accurate representation of the flow features in these areas where viscous effects are important.

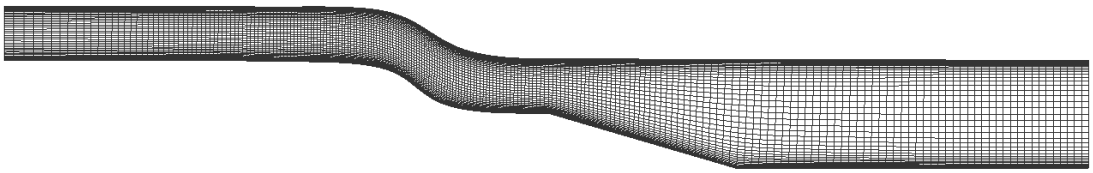


Figure 5.1: *Duct Case: The Mesh of the S-Bend Duct.*

5.1.2 Parameterization of the Duct

In order to control the shape of the duct in the optimization process, this is parameterized using the Volumetric Non-Uniform Rational B-Splines (NURBS).

The duct shape is controlled by a 8×9 NURBS Morphing Box, Figure 6.3. Within the morphing box, a total of 20 control points, highlighted in green can be displaced vertically (in the y direction). These control points are only allowed to move up to $\pm 20\%$ of the vertical distance between neighboring points (in the y direction). The remaining points, shown in red, are fixed and do not change during the optimization process. These 20 control points can be moved around to create new geometries that can be used during the optimization process.

5.1.3 Database Creation

As mentioned earlier, metamodels are trained on a DB formed by different geometries. As in the Airfoil Case, Latin Hypercube Sampling (LHS) and Random Sampling methods are combined in order to ensure complete coverage of the design space. Through LHS, a total of 150 different geometries are obtained by sampling the 20 variables within the predetermined parameterization range. Also, 50 additional geometries are generated using Random Sampling to increase the variability of the DB. Then, the CFD software is employed to simulate the flow within the duct for each case. During these simulations, the software computes both the total pressure losses and the volume of the duct, thus getting the responses of the members of the DB.

An additional test dataset is created to evaluate the model's performance on unseen data. This test dataset is created by randomly sampling 40 different geometries and using the CFD software to solve the flow and compute the same responses.

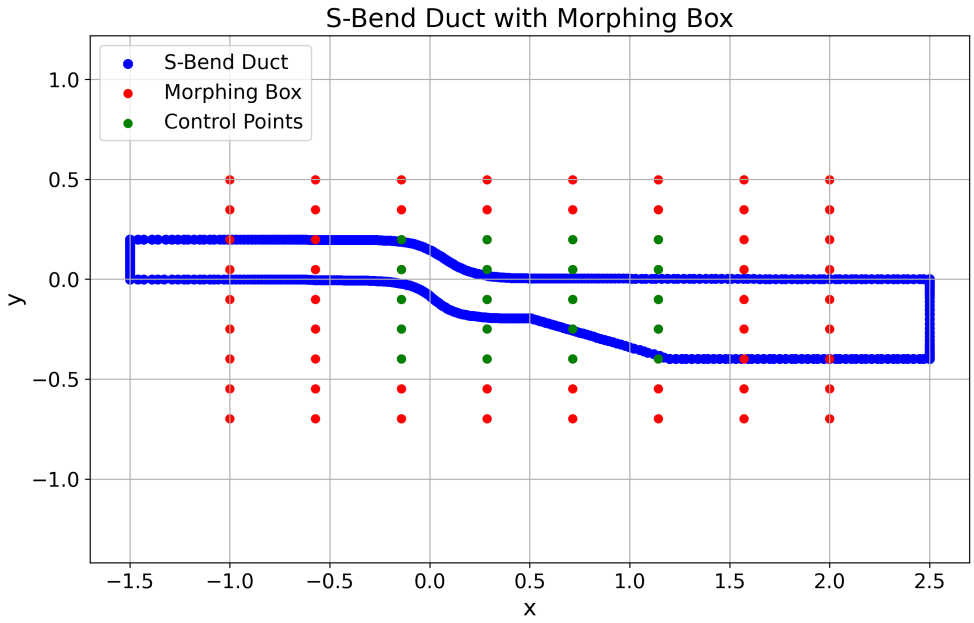


Figure 5.2: *Duct Case: S-Bend Duct with Morphing Box. Green control points are allowed to move in the y direction only, while the red points are fixed.*

5.2 Configuration of the EASY software

Similarly to the airfoil case, the EASY software is used as the optimization tool for each of the experimental setups described. The proposed methods are compared to the results obtained using the EA and MAEA (RBF on-line), implemented in the EASY software, to assess if further improvement can be achieved. The EA and MAEA (RBF on-line) have a parent population size of $\mu = 25$ and an offspring population size of $\lambda = 40$.

Additionally, the proposed methods are again integrated into an EA or MAEA (RBF on-line) with the configurations that were mentioned, depending on the specific attempt, and explicitly denoted in each attempt's name. For the MAEA (RBF on-line), the metamodels are activated after the first 80 evaluations.

All attempts have been allocated the same computational budget to ensure fair comparison among the methods, which is set at 500 CFD evaluations. In cases where the DB is required for training the models, the computational cost of the DB is taken into account, resulting in 300 evaluations remaining for the optimization software.

5.3 Implementation of Tree-Based Methods

In this section, the results of the different experimental configurations are presented. It is worth noting that the volume constraint possesses a unique characteristic: unlike other constraints, it does not require the use of the CFD software to compute it. Instead, after adjusting to the new geometry, the duct volume can be determined. However, the volume constraint is treated as all other constraints in order to guarantee the comparability of methods and the legitimacy of the applications presented. As a result, its computation occurs after the CFD software has been converged, making sure that even if the volume constraint was replaced by another flow-depending constraint, this assessment would also be valid.

5.3.1 Constraint Based Classification of Candidate Solutions

In Section 4.3.2, the idea of constraint-based classification of candidate solutions and its integration into the optimization procedure was discussed. On top of that, the purpose of this section is to implement constraint-based classification to the S-Bend duct optimization case. The main objective remains the same - to utilize classification models to assess the feasibility of candidate solutions during the optimization, to reduce the computational cost by avoiding the CFD evaluation of solutions that appear to be non-feasible.

The DB previously mentioned is used to train the classification models. The constraint imposed is that the value of Duct Volume has to be smaller than the baseline value, where $Volume^{baseline} = 0.762$. Four distinct classes are produced based on the $Volume$ values in every point of the DB, as shown in Table 5.1. This results in a zone of close to baseline solutions as shown in Figure 5.3, where the geometries that are unquestionably feasible are given the value of 3, while the non-feasible ones are given the value of 0.

$Volume$ Value Range	Class
$Volume > 0.766$	0
$0.762 < Volume < 0.766$	1
$0.758 < Volume < 0.762$	2
$Volume < 0.758$	3

Table 5.1: *Duct Case: The assigned classes for each $Volume$ value range.*

The DB is modified by replacing the $Volume$ values for each point with the corresponding class value. Subsequently, three classification models, namely Random Forest, Gradient Boosting and XGBoost, are trained on the modified DB. For the hyperparameter tuning of these models, the GridSearchCV and the RandomizedSearchCV tools are used.

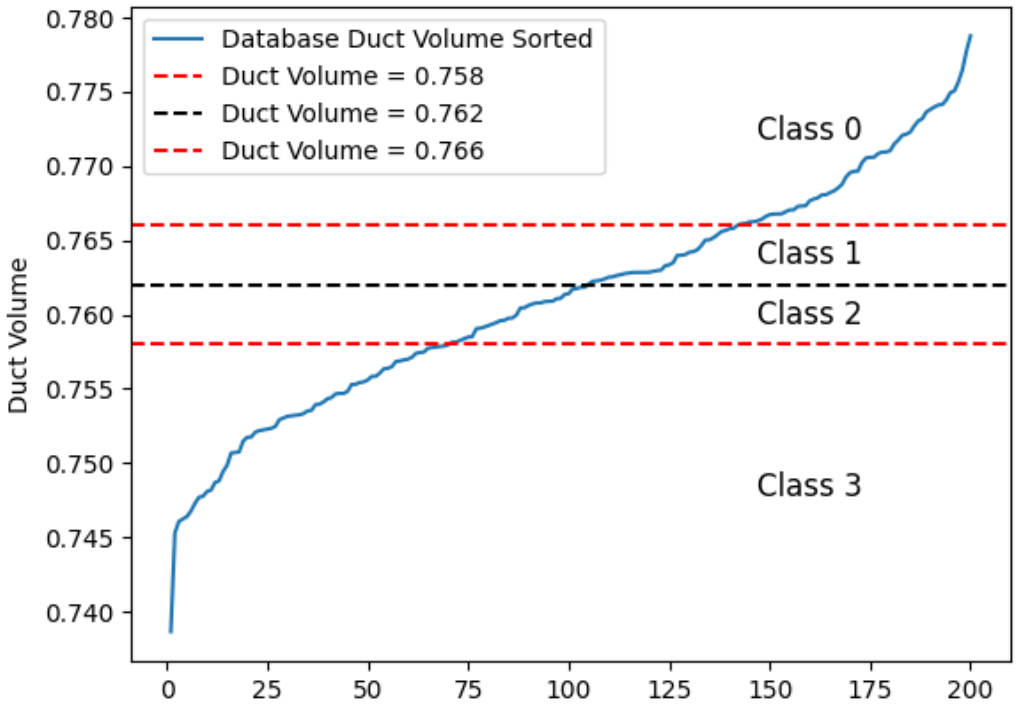


Figure 5.3: *Duct Case: The Duct Volume in the DB sorted, along with the limits between each classification class. For* $Volume > 0.766$ *class 0 was assigned, class 1 for* $0.762 < Volume < 0.766$*, class 2 for* $0.758 < Volume < 0.762$ *and class 3 for* $Volume < 0.758$*.*

Model	R^2	MAE
Random Forest	0.426	0.525
Gradient Boosting	0.481	0.5
XGBoost	0.481	0.5

Table 5.2: *Duct Case: Trained classification models' error metrics on their test dataset predictions. Values closer to 1 for* R^2 *and smaller values for* MAE *are considered favorable.*

The models are evaluated using performance metrics such as the Coefficient of Determination (R^2) and the Mean Absolute Error (MAE), as presented in Table 5.2. This table highlights that both the Gradient Boosting and XGBoost models exhibit slightly better accuracy compared to the Random Forest model. Additionally, Figure 5.4 illustrates the difference between the predicted values of each model and the real values for each individual in the test DB.

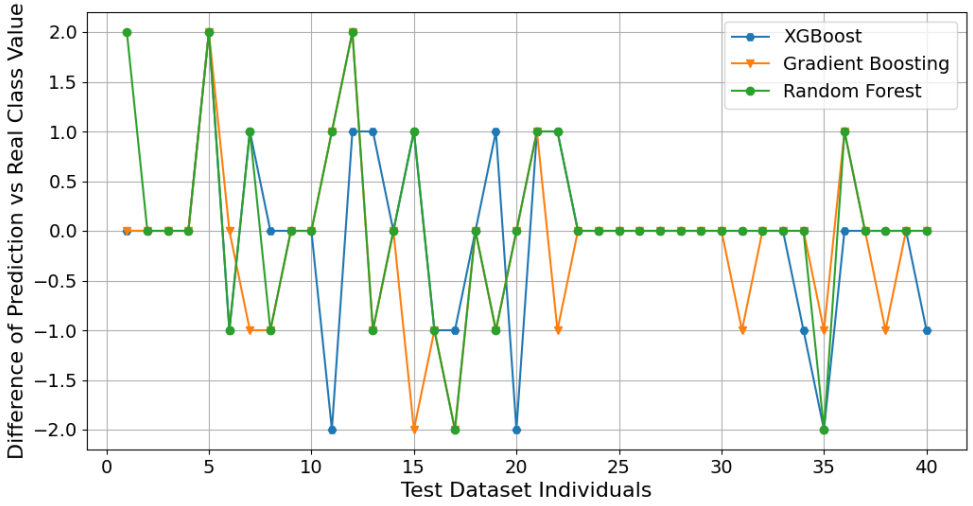


Figure 5.4: *Duct Case: Difference between the Predicted class values and the Real Class value for each point in the test DB, depicted for each model.*

After training the classification models, similar schemes as those implemented in the airfoil case are employed in the duct case, as shown in Table 6.4.

Schemes	Classes that are not evaluated
"Test 1"	0, 1
"Test 2"	0, 1, 2
"Test 3"	0

Table 5.3: *The different schemes that are implemented in the constraint-based classification approach. In each scheme, if the predicted class value of a solution belongs to the depicted values, the solution is not evaluated by the CFD software.*

These schemes are tested using a Random Forest classification model. After the results are collected, the Gradient Boosting and XGBoost models are utilized in the Test case that provided the best result with the Random Forest classifier, in this case, Test 1. This comparison aims to identify which of the models has the best performance in the constraint-based classification.

All these Tests are compared to the result of standard EA and MAEA (RBF on-line) using the EASY software. In each Test, the best individual of the DB, that returns an acceptable class value, is injected into the initial population, as the computational cost of the DB has already been charged to these attempts.

The study results are presented in Table 5.4, while the convergence patterns in Figures 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 6.4 and 5.13.

From Table 5.4, it is evident that all experimental variations, with the exception Test

Optimization Type	Pressure Losses	Duct Volume	% of Objective Improvement
EA	98.0712	0.761158	11.93%
MAEA (RBF on-line)	98.5599	0.759584	11.49%
Test 1 (Random Forest) - EA	77.8402	0.757096	30.10%
Test 2 (Random Forest) - EA	97.5854	0.760115	12.36%
Test 3 (Random Forest) - EA	96.378	0.761182	13.45%
Test 1 (Gradient Boosting) - EA	101.662	0.757831	8.70%
Test 1 (XGBoost) - EA	92.9512	0.760542	16.52%
Baseline	111.3515	0.761546	

Table 5.4: *Duct Case: Results of the optimization procedure using Tree-Based Methods as Classifiers, compared to the results of the EA and MAEA (RBF on-line). The "Pressure Losses" and "Duct Volume" values correspond to the best solution in each case computed by the CFD software.*

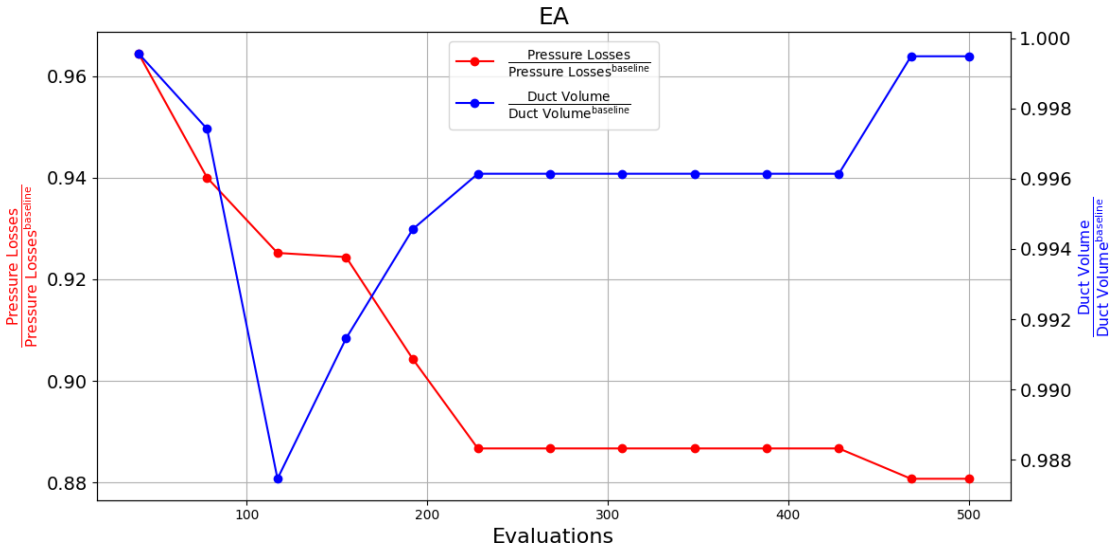


Figure 5.5: *Duct Case: The convergence history of the EA optimization, in which "Pressure Losses" is the target and "Duct Volume" is the constraint, based exclusively on the CFD code.*

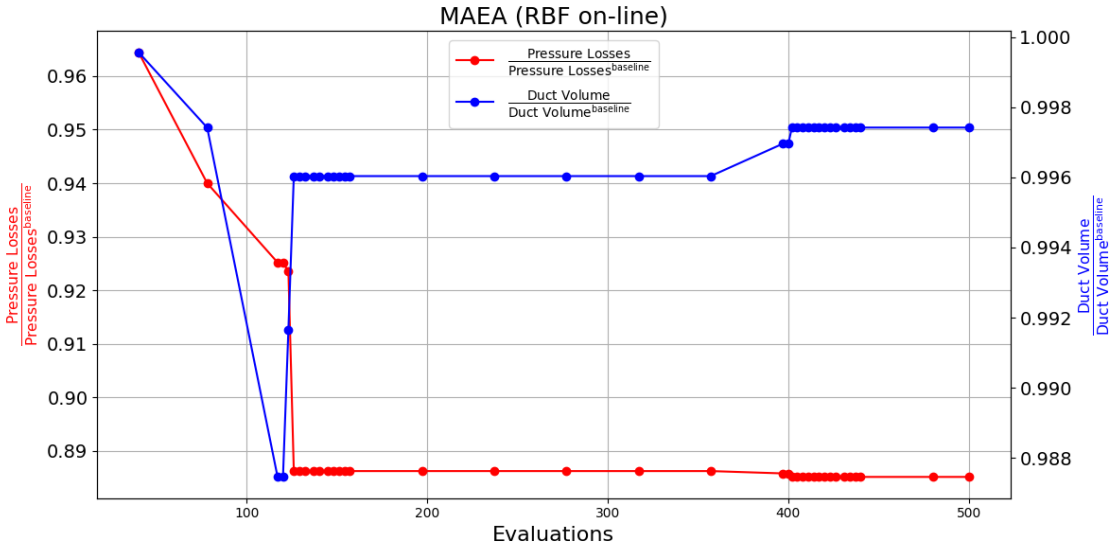


Figure 5.6: *Duct Case: The convergence history of the MAEA (RBF on-line) optimization, in which "Pressure Losses" is the target and "Duct Volume" is the constraint.*

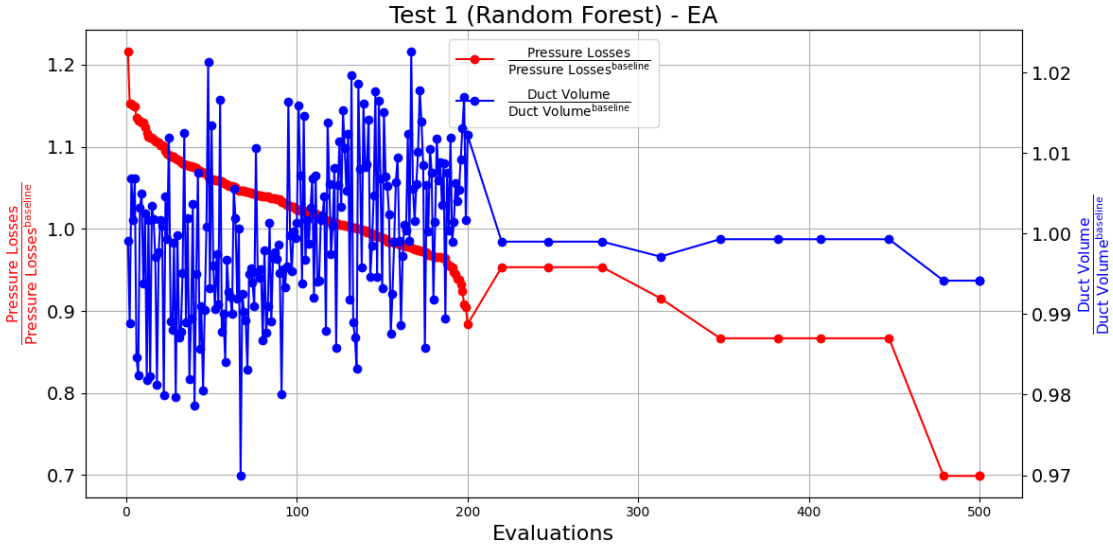


Figure 5.7: *Duct Case: The convergence history of the Random Forest "Test 1" optimization, in which "Pressure Losses" is the target and "Duct Volume" is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the "Pressure Losses" values.*

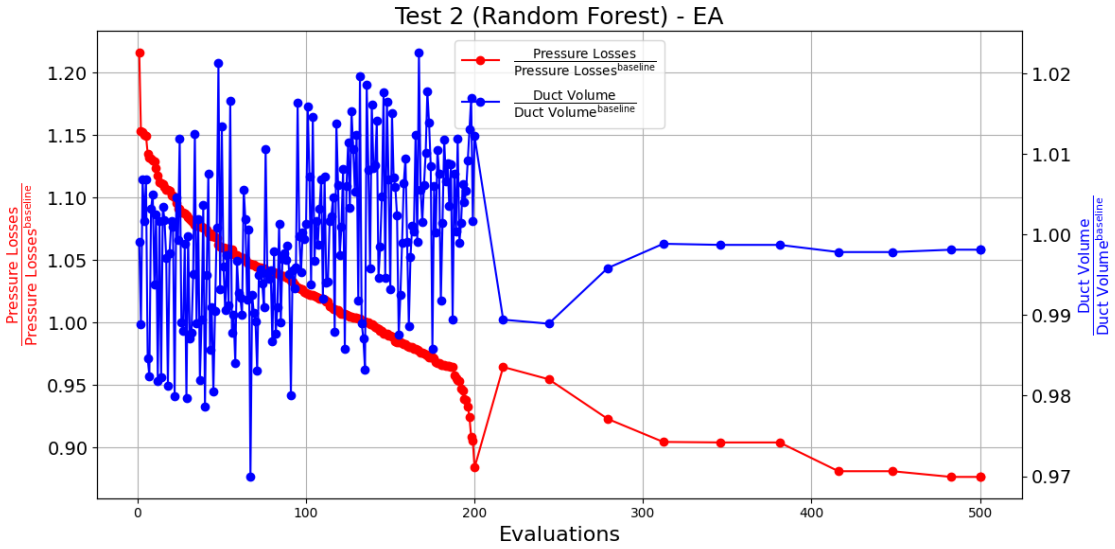


Figure 5.8: *Duct Case: The convergence history of the Random Forest "Test 2" optimization, in which "Pressure Losses" is the target and "Duct Volume" is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the "Pressure Losses" values.*

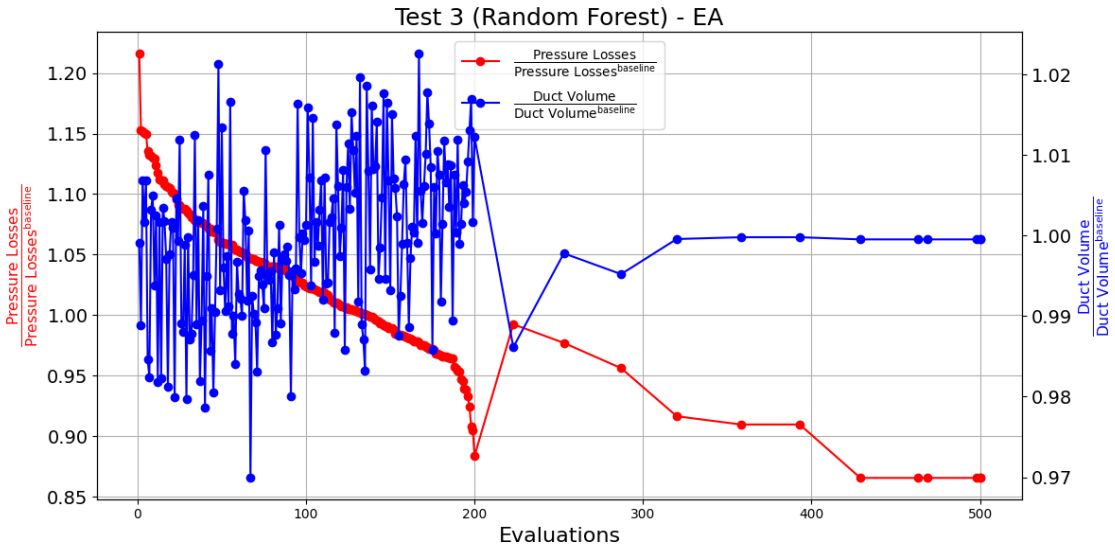


Figure 5.9: *Duct Case: The convergence history of the Random Forest "Test 3" optimization, in which "Pressure Losses" is the target and "Duct Volume" is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the "Pressure Losses" values.*

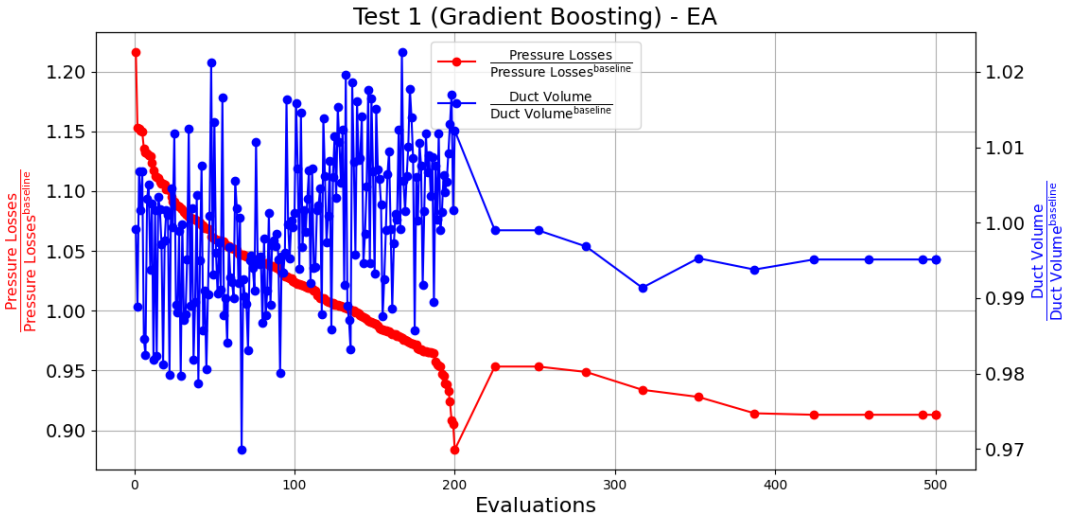


Figure 5.10: *Duct Case: The convergence history of the Gradient Boosting "Test 1" optimization, in which "Pressure Losses" is the target and "Duct Volume" is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the "Pressure Losses" values.*

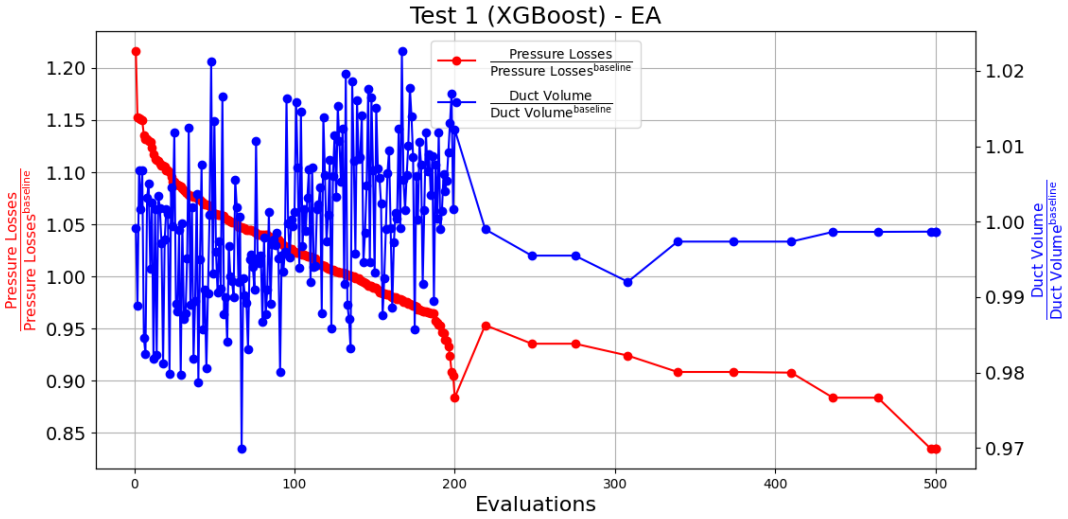


Figure 5.11: *Duct Case: The convergence history of the XGBoost "Test 1" optimization, in which "Pressure Losses" is the target and "Duct Volume" is the constraint. The first 200 evaluations correspond to the DB used for training sorted by the "Pressure Losses" values.*

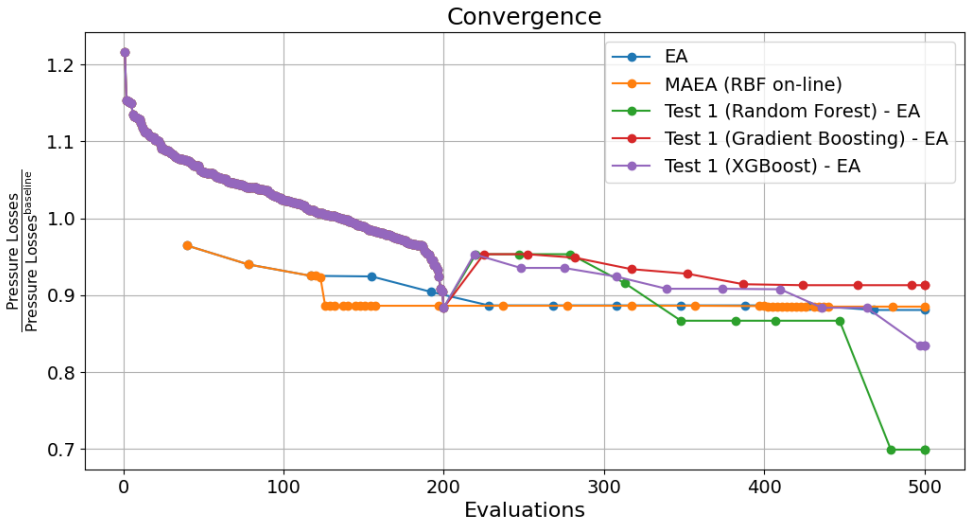


Figure 5.12: *Duct Case: Comparison of the convergence of the objective for different tests.*

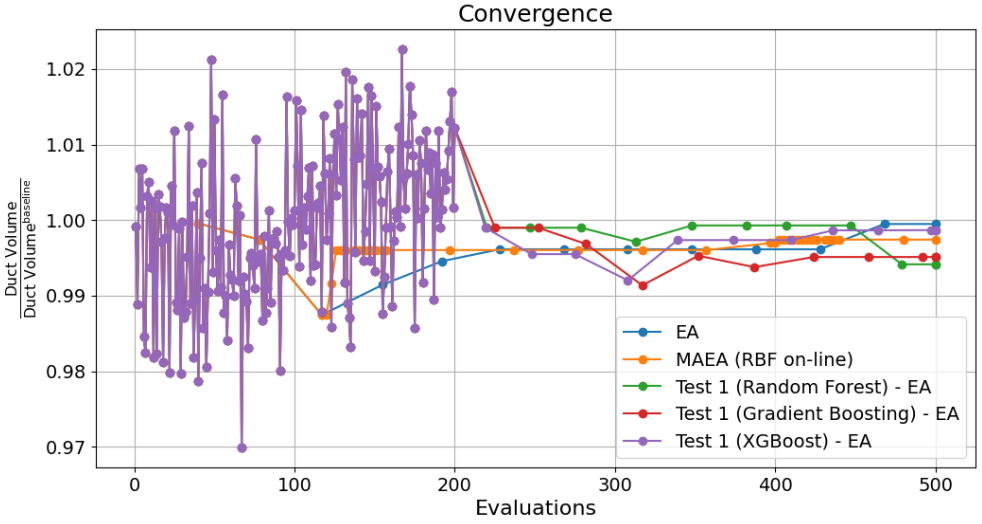


Figure 5.13: *Duct Case: Comparison of the convergence of the constraint for different tests.*

1 (Gradient Boosting) - EA, outperform both the EA and MAEA configurations and attain better solutions. Notably, Test 1 (Random Forest) - EA shows remarkable performance achieving an improvement by ∼30% in the objective function, against an average of ∼11.5% achieved by the EA and MAEA configurations. However, the Gradient Boosting and XGBoost models do not achieve such performance, with the Gradient Boosting variation failing to surpass the EA and MAEA results. On the other hand, XGBoost Test 1 achieves the second-best performance with an improvement by ∼16.5% in the objective function. The comparison of the convergence of the different models in Test 1, alongside the EA and MAEA configurations, is illustrated in Figure 6.4.

5.3.2 Design Space Exploration with Decision Trees

In this section, the design space exploration approach using DTs is applied in the S-Bend duct case. The main idea is to use the DT algorithm to partition the design space into several subregions. Then, initialize the EA into one of the subregions, that potentially carries the optimal solution. This approach, in fact, guides the optimization algorithm towards the optimal solution.

A CART DT is trained using the aforementioned DB. The optimal hyperparameters for the DT model are determined using GridSearchCV and RandomizedSearchCV. Specifically, the DT is chosen to have 10 leaf nodes, resulting in the division of the design space into 10 different regions, Figure 5.14.

Several approaches are explored in order to identify the subregion with the optimal solution. Initially, in Try 1, the region with the best value, i.e. the minimum, among the leaf nodes is selected. This approach aims to prioritize the region that the DT model identified as the best, based on the available data. As has already been discussed, in all these attempts, the best individual of the DB, that belongs to the corresponding region, is injected into the initial population, as the computational cost for the DB has already been charged. The region, in Try 1, contains most of the best solutions that belong to the DB, along with the best feasible solution. As a result, in the duct case, Try 1 encompasses both Try 1 and Try 2 schemes, as defined in the airfoil case.

Subsequently, in Try 3, the region containing the best solution that has already been identified, specifically in Test 1 (Random Forest) - EA, is chosen. In this Try, the knowledge from previous tests is utilized to locate the best region. However, the individual injected into the initial population of this Try involves the best solution from the DB that belonged to this region, rather than using the solution itself.

Finally, in Try 4, the optimization algorithm is limited to the region with the second-best value, that the DT model identified and is injected accordingly.

The results are presented in Table 5.5. Additionally, the convergence patterns of the optimization process are shown in Figures 6.6 and 5.16. It is important to

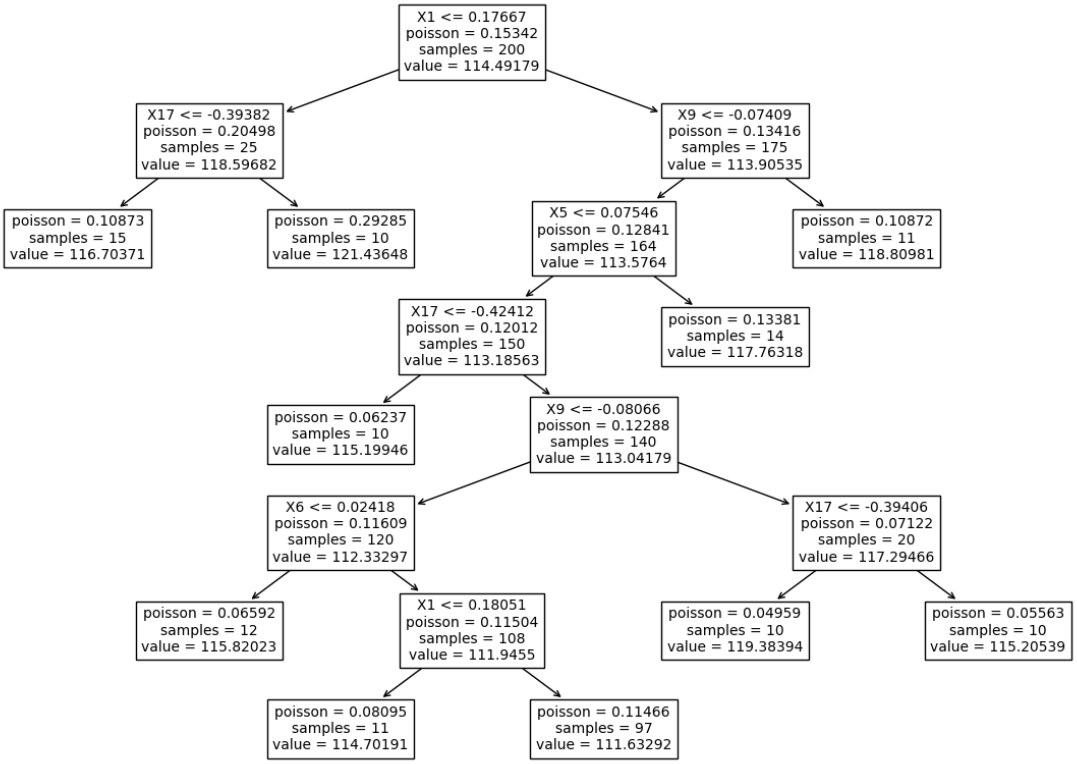


Figure 5.14: *Duct Case: The trained DT that was used to partition the design space.*

Optimization Type	Pressure Losses	Duct Volume	% of Objective Improvement
EA	98.0712	0.761158	11.93%
MAEA (RBF on-line)	98.5599	0.759584	11.49%
DT Design Space Selection - Try1 - MAEA (RBF on-line)	101.06	0.757052	9.24%
DT Design Space Selection - Try3 - MAEA (RBF on-line)	88.1883	0.757564	20.80%
DT Design Space Selection - Try4 - MAEA (RBF on-line)	101.816	0.758155	8.56%
Baseline	111.3515	0.761546	

Table 5.5: *Duct Case: Results of the optimization procedure using the Design Space Exploration approach, compared to the results of the EA and MAEA (RBF on-line). The "Pressure Losses" and "Duct Volume" values correspond to the best solution in each case computed by the CFD software.*

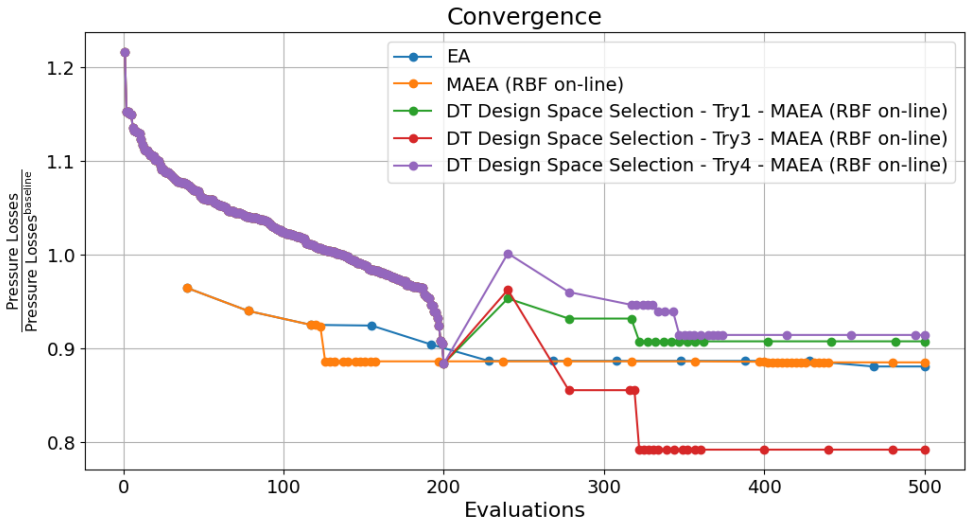


Figure 5.15: *Duct Case: Evaluating the Convergence of the objective for Various Design Space Exploration Techniques.*

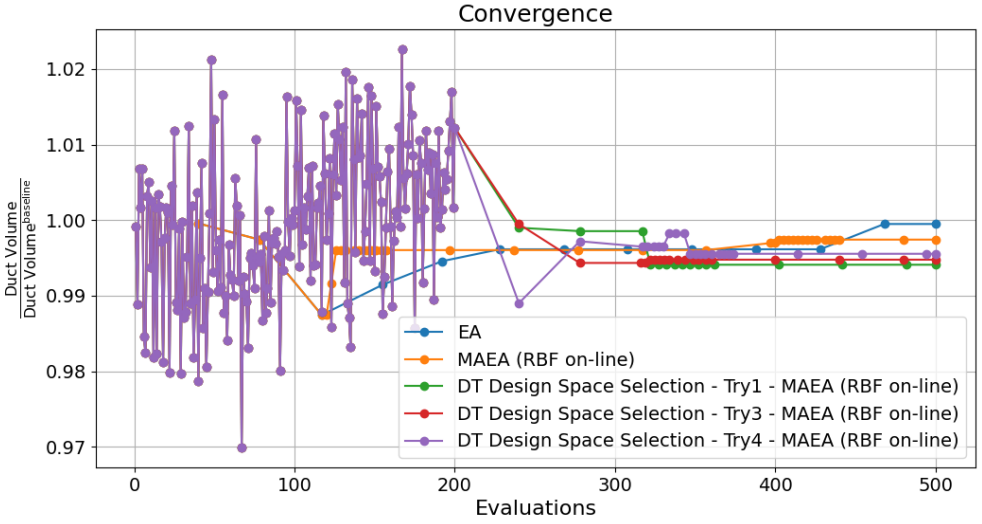


Figure 5.16: *Duct Case: Evaluating the Convergence of the constraint for Various Design Space Exploration Techniques.*

note that, as a DB is needed for the DT, all tries are made over a period of 300 evaluations, including an additional 200 evaluations for the DB, to reach a total of 500 evaluations.

As it can be seen from Table 5.5, both Try 1 and Try 4 do not yield superior solutions from those obtained by the EA and MAEA (RBF on-line). However, in Try 3, a better solution is found, that improved the baseline by ∼21%, nearly double the improvement achieved by the MAEA (RBF on-line). Figure 6.6 illustrates that Try 3 attempt identifies this solution in 320 evaluations, only 120 evaluations or 3 generations after the initiation of the EA. It is important to note that in Try 3, the selection of the region relies on knowledge that is obtained from previous attempts, exceeding the computational budget of the specific attempt. Consequently, this results in just a verification that the method is effective, rather than a rigorous approach for the selection of the optimal region.

5.3.3 Combined Application of Methods

Both the constraint-based classification and the design space exploration approaches require the creation of a DB, which is indeed costly. However, since the cost of the DB has already been charged, it is possible to combine both methods. In this section, the best-performing tests from each approach are combined to assess their performance. Two attempts, namely "Combination Test - EA" and "Combination Test - MAEA (RBF on-line)", are carried out using the EA and MAEA (RBF on-line), respectively. These tests implement the Random Forest classification model within the "Test 1" scheme in the "DT Design Space Selection - Try3" subregion, which yielded the best results among other attempts in the same context. In both cases, the best individual from the DB, which belongs to the corresponding subregion, is injected into the initial population.

Optimization Type	Pressure Losses	Duct Volume	% of Objective Improvement
EA	98.0712	0.761158	11.93%
MAEA (RBF on-line)	98.5599	0.759584	11.49%
Combination Test - EA	86.0479	0.759015	22.72%
Combination Test - MAEA (RBF on-line)	87.3704	0.757583	21.54%
Baseline	111.3515	0.761546	

Table 5.6: *Duct Case: Results of the optimization procedure using the Combination of Methods approach, compared to the results of the EA and MAEA (RBF on-line). The "Pressure Losses" and "Duct Volume" values correspond to the best solution in each case computed by the CFD software.*

The results are summarized in Table 5.6 and the convergence patterns are depicted in Figures 6.9 and 5.18. As shown in Table 5.6, both attempts achieve satisfactory results with the "Combination Test - EA", exhibiting a better solution with an improvement by ∼23% in the objective function. It is important to note, that both

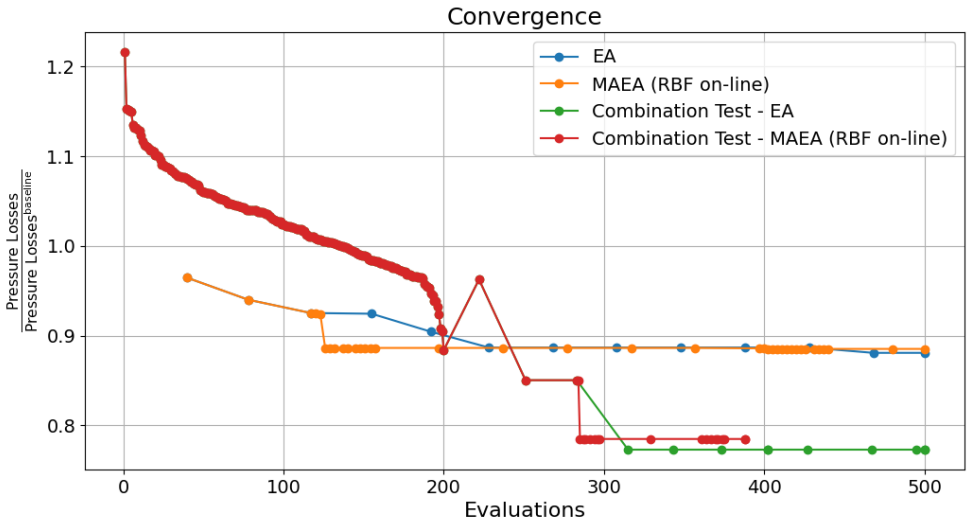


Figure 5.17: *Duct Case: Evaluating the Convergence of the objective for the Combination Tests.*

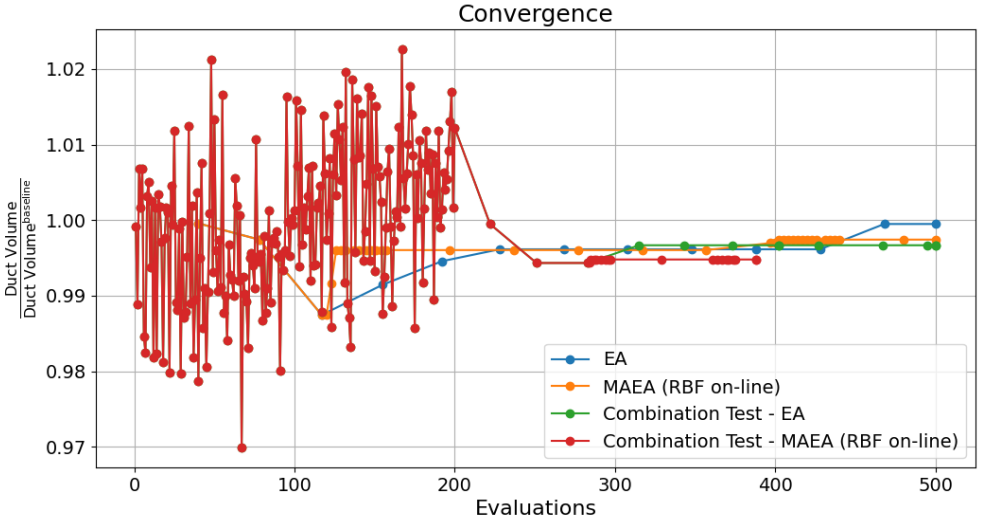


Figure 5.18: *Duct Case: Evaluating the Convergence of the constraint for the Combination Tests.*

attempts outperform the "DT Design Space Selection - Try3 - MAEA (RBF on-line)" configuration, but neither of them surpasses the performance of "Test 1 (Random Forest) - EA". The convergence patterns in Figure 6.9 indicate that both attempts converge to the optimal solution within a small number of evaluations after the initiation of the optimization process, around 120 evaluations. This demonstrates that the combination of the best classification scheme within the best subregion of the design space can quickly converge in good solutions. Lastly, in "Combination Test - MAEA (RBF on-line)" the optimization software terminates early as no better solution can be found in subsequent generations.

5.4 Aggregated Results

First, the constraint-based classification approach was implemented. Similar to the S8052 airfoil case, the primary objectives of the tests are to evaluate the method's performance and to identify the best-performing scheme and model. In contrast to the results of the airfoil case, multiple attempts achieved superior solutions compared to the EA. Particularly, "Test 1 (Random Forest) - EA" solution achieved a remarkable improvement by $\sim$30% in the objective function, compared to the $\sim$12% of the EA solution. These results confirm the previous findings and highlight the effectiveness of this strategy.

Subsequently, the design space exploration approach was assessed. Different subregions were selected in different attempts, with only one test outperforming the EA, namely the "DT Design Space Selection - Try3 - MAEA (RBF on-line)" (improvement by $\sim$21% in the objective). Notably, the subregion, that concluded to the best-performing test, was selected by leveraging knowledge obtained from previous attempts. However, this kind of research may not be possible if the computational budget is limited or the problem is more complex, raising concerns about the best method for selecting the subregion.

Finally, when the best of the two approaches were combined, two tests were conducted: one with EA (improvement by $\sim$23% in the objective) and one with MAEA (RBF on-line) (improvement by $\sim$22% in the objective). Both tests achieved superior solutions compared to the conventional EA, with very fast convergence. However, even though they surpassed the "DT Design Space Selection - Try3 - MAEA (RBF on-line)", they did not outperform "Test 1 (Random Forest) - EA". These results confirm the capability for synergy between the two approaches, as observed in the previous case.

It is important to note that in this case, the improvement achieved by the presented approaches compared to the standard EA was significantly greater than in the S8052 airfoil case. The best-performing attempts exhibited an improvement that was two to three times greater than that of the EA solution.

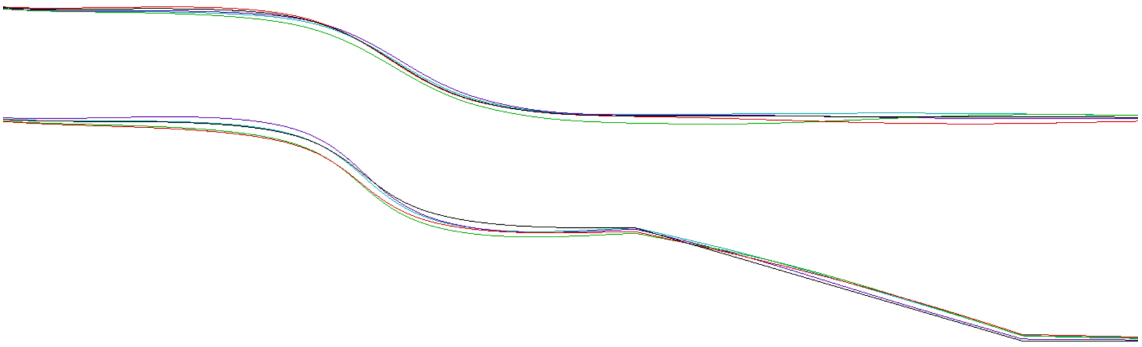
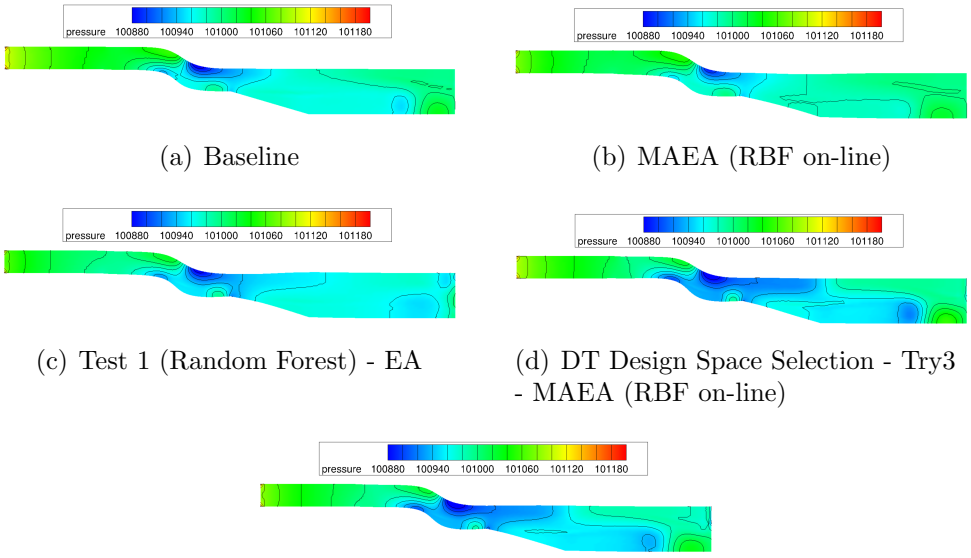


Figure 5.19: *Duct Case: The duct shape of the baseline (black) and the shape of the best solutions of the: MAEA (RBF on-line) (red), "Test 1 (Random Forest) - EA" (green), "DT Design Space Selection - Try3 - MAEA (RBF on-line)" (purple) and "Combination Test - EA" (light blue).*



(a) Baseline

(b) MAEA (RBF on-line)

(c) Test 1 (Random Forest) - EA

(d) DT Design Space Selection - Try3 - MAEA (RBF on-line)

(e) Combination Test - EA

Figure 5.20: *Duct Case: Pressure iso-areas computed by PUMA for the best solution of the shape optimization by different attempts.*

The duct shape of the baseline and the shapes of the best solutions obtained from the top-performing attempts in each approach are shown in Figure 5.19. Also, the pressure iso-areas for the same attempts are depicted in Figure 5.20.

Chapter 6

Conclusions

6.1 Overview

This Diploma Thesis focused on the implementation of Tree-Based Methods in Aerodynamic Shape Optimization. Three different approaches were examined: using a Tree-Based Regression Model as off-line trained metamodel in MAEA optimization, utilizing a constraint-based classification model to determine whether a solution is feasible and continue the optimization procedure only if the solution is classified as feasible, and finally, employing a DT to split the design space into subregions and initiate the optimization software in one of them. These approaches were tested in two cases, specifically the S8052 airfoil and S-Bend duct case. In both cases, the performance of the approaches was compared to the results of the EA and MAEA (RBF on-line) optimization methods implemented in the EASY software. The evaluation criteria included the objective function improvement and convergence patterns within a computational budget of 500 evaluations for each attempt.

For the creation of the training DB that all the models would be trained on, the baseline geometries were parameterized using NURBS lattices. Latin Hypercube and Random Sampling were combined to generate a total number of 200 different geometries for each case with an additional test dataset for validating the models. Then, all designs were evaluated by the PUMA software to create the DB.

For the S8052 case, the off-line trained metamodel approach yielded poor results, confirming the concerns, that pre-existed, about the use of Tree-Based metamodels as off-line trained metamodels. Consequently, this approach was not further explored in the next case study. In the constraint-based classification approach, a Random Forest, a Gradient Boosting and an XGBoost model were trained and implemented in different schemes. The goal was to assess the performance of this approach, the

best scheme, and the best model, for each case. In both cases, most attempts in this approach achieved satisfactory results with some outperforming the EA and MAEA (RBF on-line). Bigger improvements in the objective function were noticed in the S-Bend duct case. Notably, the Random Forest classification model produced the best results in both cases. However, different schemes ("Test 2" and "Test1") were the best performing between the two cases, so selecting the optimal scheme remains unclear.

In the design space exploration approach, a DT model with 10 leaf nodes was employed to split the design space into 10 subregions, for each case. Different attempts were made by initiating the optimization algorithm in different subregions. Attempts in both cases surpassed the performance of the EA and MAEA (RBF on-line) methods. However, the subregions of the best-performing attempts, in each case, were selected with different criteria. For this reason, it is unclear which one of the subregions should be selected. It is worth noting the risk of such an approach. If the selected region does not contain the optimal solution, the optimization algorithm will be restricted in a suboptimal area.

Lastly, the best-performing attempts from the constraint-based classification and design space exploration approaches were combined in each case study. The main objective was to evaluate the performance of the combined approach, considering that no additional computational expense was required to implement both methods together. In the S8052 airfoil case, both the EA and MAEA combined approaches achieved superior solutions compared to all other approaches. Similarly, in the S-Bend duct case, the combined approach outperformed most attempts, except for the best among the constraint-based classification attempts. Importantly, the combined approaches managed to achieve highly competitive solutions within a small number of evaluations.

6.2 Conclusions

After the completion of all the previous studies, the following conclusions are drawn:

- Caveats are expressed on the use of the Tree-Based regression models as off-line trained metamodels in MAEA optimization.
- The constraint-based classification approach, employing Tree-Based models as classifiers, can achieve better solutions than the EA and MAEA (RBF on-line). However, the optimal scheme that these models should be implemented within requires further investigation, particularly between "Test 1" and "Test 2" schemes. The Random Forest model exhibits better performance than the other models.
- The design space exploration approach effectively assists the EA or MAEA to converge to the optimal solution by selecting the appropriate subregion.

Through the previous studies, no systematic way of determining the optimal subregion could be detected. It is up to the user to leverage the previous knowledge, in each problem, to select the best subregion, with the underlying risk that the optimization software could be restricted.

- The combination of the constraint-based classification and design space exploration approaches can outperform other methods with fast convergence, provided that the appropriate classification scheme and the optimal subregion are selected.

6.3 Future Work Proposals

Based on the findings of this research, the following future works are proposed:

- More research can be conducted by testing different classification models, including DNN's, for the constraint-based classification approach.
- The constraint-based classification approach can be implemented on-line within the EASY software, training the model on the evaluated individuals of previous generations. In this way, in future generations, evaluations of non-feasible individuals could be avoided, reducing the computational expense.
- A proposed way for implementing the combined approach is to manage the computational budget in each case and test the best scheme and some promising subregions with a small number of evaluations. Then combine the best out of the initial tests for the remaining computational budget. The fast convergence observed in the combination tests, as noted in the previous studies, implies that better solutions could potentially be found faster than with the EA and MAEA (RBF on-line).

Bibliography

[1] Russell, Stuart J., Norvig, Peter, and Davis, Ernest: *Artificial Intelligence: A Modern Approach*. Pearson Educación, 2022.

[2] Zhang, Qian, Lu, Jie, and Jin, Yaochu: *Artificial intelligence in recommender systems*. Complex & Intelligent Systems, 7:439–457, 2021.

[3] Zhao, Wayne Xin, Zhou, Kun, Li, Junyi, Tang, Tianyi, Wang, Xiaolei, Hou, Yupeng, Min, Yingqian, Zhang, Beichen, Zhang, Junjie, Dong, Zican, *et al.*: *A Survey of Large Language Models*. arXiv preprint arXiv:2303.18223, 2023.

[4] Topol, Eric J: *High-performance medicine: the convergence of human and artificial intelligence*. Nature medicine, 25(1):44–56, 2019.

[5] Fernández, Ana: *Artificial intelligence in financial services*. Banco de Espana Article, 3:19, 2019.

[6] Pham, Duc T. and Pham, PTN: *Artificial intelligence in engineering*. International Journal of Machine Tools and Manufacture, 39(6):937–949, 1999.

[7] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron: *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] Aggarwal, Piero Cinquegrana, Shefali, Cinquegrana, Piero, and Aggarwal, Shefali: *Deep Learning: The Latest Trend in AI and ML*, May 2023. `https://www.qubole.com/blog/deep-learning-the-latest-trend-in-ai-and-ml`.

[9] Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome: *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, NY, 2nd edition, 2009.

[10] Kontou, Marina G, Asouti, Varvara G, and Giannakoglou, Kyriakos C: *DNN surrogates for turbulence closure in CFD-based shape optimization*. Applied Soft Computing, page 110013, 2023.

[11] Karakasis, Marios K and Giannakoglou, Kyriakos C: *On the use of metamodel-assisted, multi-objective evolutionary algorithms*. Engineering Optimization, 38(8):941–957, 2006.

[12] Kapsoulis, Dimitrios: *Low-cost metamodel-assisted evolutionary algorithms with applications in shape optimization in fluid dynamics*. PhD thesis, National Technical University of Athens, School of Mechanical Engineering, 2019.

[13] Giannakoglou, Kyriakos, 2008. `http://velos0.ltt.mech.ntua.gr/EASY/`.

[14] Asouti, Varvara G, Trompoukis, Xenofon S, Kampolis, Ioannis C, and Giannakoglou, Kyriakos C: *Unsteady CFD computations using vertex-centered finite volumes for unstructured grids on Graphics Processing Units*. International Journal for Numerical Methods in Fluids, 67(2):232–246, 2011.

[15] Caruana, Rich and Niculescu-Mizil, Alexandru: *An Empirical Comparison of Supervised Learning Algorithms*. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168, 2006.

[16] Morgan, James N and Sonquist, John A: *Problems in the Analysis of Survey Data, and a Proposal*. Journal of the American Statistical Association, 58(302):415–434, 1963.

[17] Alsagheer, Radhwan HA, Alharan, Abbas FH, and Al-Haboobi, Ali SA: *Popular Decision Tree Algorithms of Data Mining Techniques: A Review*. International Journal of Computer Science and Mobile Computing, 6(6):133–142, 2017.

[18] Breiman, Leo, Friedman, Jerome, Stone, Charles J., and Olshen, R.A.: *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[19] Quinlan, John Ross: *Induction of Decision Trees*. In Shavlik, Jude W. and Dietterich, Thomas G. (editors): *Readings in Machine Learning*. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986.

[20] Quinlan, John Ross: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[21] Quinlan, John Ross: *C5*, 2007. `http://rulequest.com`.

[22] Kass, Gordon V: *An Exploratory Technique for Investigating Large Quantities of Categorical Data*. Journal of the Royal Statistical Society: Series C (Applied Statistics), 29(2):119–127, 1980.

[23] Loh, Wei Yin: *Fifty Years of Classification and Regression Trees*. International Statistical Review, 82(3):329–348, 2014.

[24] Sharma, Himani and Kumar, Sunil: *A Survey on Decision Tree Algorithms of Classification in Data Mining*. International Journal of Science and Research (IJSR), 5(4):2094–2097, 2016.

[25] Berk, Richard A: *Statistical Learning from a Regression Perspective*. Springer Texts in Statistics. Springer International Publishing, Cham, Switzerland, 2nd edition, 2016.

[26] Laurent, Hyafil and Rivest, Ronald L: *Constructing Optimal Binary Decision Trees is NP-Complete.* Information Processing Letters, 5(1):15–17, 1976.

[27] Zhou, Zhi Hua: *Ensemble Methods: Foundations and Algorithms.* Whittles Publishing, Caithness, UK, 2012.

[28] Opitz, David and Maclin, Richard: *Popular Ensemble Methods: An Empirical Study.* Journal of Artificial Intelligence Research, 11:169–198, 1999.

[29] Ying, Xu: *Ensemble Learning.* May 2014.

[30] Dietterich, Thomas G.: *Ensemble Methods in Machine Learning.* In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer, 2000.

[31] Raphael, John and Lamarre, Townshend: *CS229 Lecture Notes - Ensembling Methods*, 2018.

[32] Alpaydin, Ethem: *Introduction to Machine Learning.* Adaptive Computation and Machine Learning Series. MIT Press, London, England, 3rd edition, 2014.

[33] Breiman, Leo: *Bagging Predictors.* Machine learning, 24(2):123–140, 1996.

[34] Zhang, Cha and Ma, Yunqian: *Ensemble Machine Learning: Methods and Applications.* Springer Publishing Company, Incorporated, 2012, ISBN 1441993258.

[35] Tibshirani, Robert J and Efron, Bradley: *An Introduction to the Bootstrap.* Monographs on Statistics and Applied Probability, 57:1–436, 1993.

[36] James, Gareth, Witten, Daniela, Hastie, Trevor, and Tibshirani, Robert: *An Introduction to Statistical Learning: with Applications in R* . Springer, 2013.

[37] Ferreira, Artur J. and Figueiredo, Mário A.T.: *Boosting Algorithms: A Review of Methods, Theory, and Applications.* Ensemble Machine Learning, pages 35–85, 2012.

[38] Breiman, Leo: *Random Forests.* Machine learning, 45(1):5–32, 2001.

[39] Segal, Mark and Xiao, Yuanyuan: *Multivariate Random Forests.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1):80–87, 2011.

[40] Mohan, Ananth, Chen, Zheng, and Weinberger, Kilian: *Web-Search Ranking with Initialized Gradient Boosted Regression Trees.* In *Proceedings of the Learning to Rank Challenge*, pages 77–89. PMLR, 2011.

[41] Cutler, Adele, Cutler, D Richard, and Stevens, John R: *Random Forests.* In *Ensemble Machine Learning*, pages 157–175. Springer, 2012.

[42] Friedman, Jerome H: *Greedy Function Approximation: A Gradient Boosting Machine.* Annals of Statistics, pages 1189–1232, 2001.

[43] Freund, Yoav and Schapire, Robert E.: *Experiments with a New Boosting Algorithm.* In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, page 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc., ISBN 1558604197.

[44] Freund, Yoav and Schapire, Robert E: *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.* Journal of Computer and System Sciences, 55(1):119–139, 1997.

[45] Schapire, Robert E: *The Strength of Weak Learnability.* Machine learning, 5(2):197–227, 1990.

[46] Mason, Llew, Baxter, Jonathan, Bartlett, Peter, and Frean, Marcus: *Boosting Algorithms as Gradient Descent.* Advances in Neural Information Processing Systems, 12, 1999.

[47] Chen, Tong and Guestrin, Carlos: *XGBoost: A Scalable Tree Boosting System.* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794, 2016.

[48] Ke, Guolin, Meng, Qi, Finley, Thomas, Wang, Taifeng, Chen, Wei, Ma, Weidong, Ye, Qiwei, and Liu, Tie Yan: *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.* Advances in Neural Information Processing Systems, 30, 2017.

[49] Murphy, Kevin P: *Machine learning: A Probabilistic Perspective.* MIT press, 2012.

[50] He, Zhiyuan, Lin, Danchen, Lau, Thomas, and Wu, Mike: *Gradient Boosting Machine: A Survey.* arXiv preprint arXiv:1908.06951, 2019.

[51] Izenman, Alan Julian: *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning.* Springer Publishing Company, Incorporated, 1st edition, 2008, ISBN 0387781889.

[52] Tomonori, Masui: *All you need to know about Gradient Boosting algorithm - Part 1. Regression*, 2022. `https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502`.

[53] Giannakoglou, Kyriakos: *Optimization Methods in Aerodynamics.* NTUA, 2006. `http://velos0.ltt.mech.ntua.gr/kgianna/optim/distr/Book-Optim-Giannakoglou.pdf`.

[54] Bäck, Thomas, Fogel, David B., and Michalewicz, Zbigniew: *Handbook of Evolutionary Computation.* Institute of Physics Pub., 2002.

[55] Eiben, Agoston E and Smith, James E: *Introduction to Evolutionary Computing.* Springer, 2015.

[56] Mitchell, Melanie: *An Introduction to Genetic Algorithms*. MIT, 1998.

[57] Beyer, Hans Georg and Schwefel, Hans Paul: *Evolution Strategies-a comprehensive introduction*. Natural computing, 1:3–52, 2002.

[58] Koza, John R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[59] Kapsoulis, Dimitrios, Tsiakas, Konstantinos, Trompoukis, Xenofon, Asouti, Varvara, and Giannakoglou, Kyriakos: *Evolutionary multi-objective optimization assisted by metamodels, kernel PCA and multi-criteria decision making techniques with applications in aerodynamics*. Applied Soft Computing, 64:1–13, 2018.

[60] Spalart, Philippe and Allmaras, Steven: *A one-equation turbulence model for aerodynamic flows*. In *30th aerospace sciences meeting and exhibit*, page 439, 1992.

[61] Lyon, Christopher A., Broeren, Andy P., Giguére, Philippe, Gopalarathnam, Ashok, and Selig, Michael S.: *Summary of Low-Speed Airfoil Data*, volume 3. SoarTech, 1997.

[62] Piotrowski, Michael GH and Zingg, David W: *Smooth Local Correlation-based Transition Model for the Spalart-Allmaras Turbulence Model*. AIAA Journal, 59(2):474–492, 2021.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής
& Βελτιστοποίησης

Χρήση Μοντέλων Βασισμένων στα Δέντρα Αποφάσεων στη Βελτιστοποίηση μέσω Εξελικτικών Αλγορίθμων – Εφαρμογές στην Αεροδυναμική

Διπλωματική εργασία - Εκτενής Περίληψη στα Ελληνικά

Κωνσταντίνος Χονδρός

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2023

Εισαγωγή

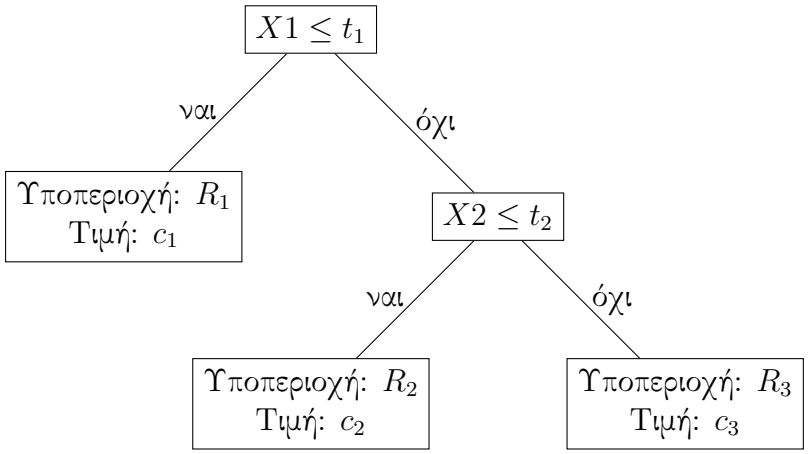
Η Διπλωματική αυτή Εργασία διερευνά την ενσωμάτωση μοντέλων βασισμένων στα Δέντρα Αποφάσεων στην διαδικασία της βελτιστοποίησης σχήματος μέσω Εξελικτικών Αλγορίθμων, όπως χρησιμοποιείται στην ρευστομηχανική, προκειμένου είτε να βρεθούν καλύτερες λύσεις με το ίδιο υπολογιστικό κόστος είτε να μειωθεί το κόστος για την ίδια ποιότητα λύσης. Στο παρελθόν, διαφορετικά μεταμοντέλα έχουν χρησιμοποιηθεί από την ΜΠΥΡΒ του ΕΜΠ, όπως Radial Basis Function (RBF) networks και Βαθιά Νευρωνικά Δίκτυα (ΒΝΔ).

Τρεις προσεγγίσεις εξετάζονται: μοντέλα παλινδρόμησης βασισμένα στα Δέντρα Αποφάσεων ως off-line μεταμοντέλα, χρήση μοντέλων βασισμένων στα Δέντρα Αποφάσεων για ταξινόμηση με κριτήριο την ικανοποίηση ή όχι των περιορισμών και εξερεύνηση του χώρου σχεδιασμού με Δέντρα Αποφάσεων. Οι προσεγγίσεις αυτές αξιολογούνται σε δύο περιπτώσεις βελτιστοποίησης σχήματος στη μηχανική των ρευστών.

Μοντέλα Βασισμένα στα Δέντρα Αποφάσεων

Τα Δέντρα Αποφάσεων (ΔΑ) είναι μοντέλα Μηχανικής Μάθησης και ανήκουν στην κατηγορία της επιτηρούμενης μάθησης (supervised learning), δηλαδή εκπαιδεύονται σε μια Βάση Δεδομένων, η οποία περιέχει άτομα μαζί με τις αποκρίσεις αυτών. Τα ΔΑ μπορούν να χρησιμοποιηθούν για παλινδρόμηση ή ταξινόμηση.

Τα ΔΑ, εν συντομία, δημιουργούν ιεραρχικές δομές που χωρίζουν τον χώρο σχεδιασμού σε υποπεριοχές, βάσει των τιμών των μεταβλητών σχεδιασμού, Σχήμα 6.1. Σε κάθε υποπεριοχή αντιστοιχεί ένας αριθμός, ο μέσος όρος ή η πλειοψηφία των στοιχείων της Βάσης Δεδομένων που αντιστοιχούν στην εκάστοτε υποπεριοχή (ανάλογα αν το πρόβλημα είναι παλινδρόμησης ή ταξινόμησης). Εάν στο μοντέλο δοθεί ένα άγνωστο στοιχείο για πρόβλεψη, ο αλγόριθμος θα εντοπίσει την υποπεριοχή που ανήκει το στοιχείο αυτό και θα αποδώσει την αντίστοιχη τιμή.



Σχήμα 6.1: *Απεικόνιση ενός εκπαιδευμένου ΔΑ.*

Το Random Forest και το Gradient Boosting είναι μοντέλα Μηχανικής Μάθησης που βασίζονται στα ΔΑ.

Το Random Forest αποτελείται από ένα σύνολο από ΔΑ, όπου κάθε ΔΑ εκπαιδεύεται με ένα τυχαίο υποσύνολο της ΒΔ. Κατά την πρόβλεψη, ο αλγόριθμος συνδυάζει τις προβλέψεις όλων των επιμέρους ΔΑ, με μέσο όρο ή πλειοψηφία, για την τελικά πρόβλεψη.

Στην περίπτωση του Gradient Boosting, ο αλγόριθμος βασίζεται στην εκπαίδευση ενός συνόλου από ΔΑ, εκπαιδεύοντας το κάθε νέο ΔΑ στοχεύοντας την μείωση του σφάλματος της πρόβλεψης από το προηγούμενο ΔΑ. Η τελική πρόβλεψη προκύπτει από τον συνδυασμό όλων των ΔΑ. Τέλος, το XGBoost αποτελεί μία υλοποίηση του Gradient Boosting

Εξελικτικοί Αλγόριθμοι

Οι εξελικτικοί αλγόριθμοι (ΕΑ) εμπνέονται από τη θεωρία του Δαρβίνου περί της φυσικής εξέλιξης και ανήκουν σε μία κατηγορία στοχαστικών αλγορίθμων βελτιστοποίησης που χειρίζονται πληθυσμούς υποψήφιων λύσεων (population-based methods) σε κάθε επανάληψη. Ο πληθυσμός αρχικοποιείται τυχαία και όλα τα άτομα αξιολογούνται από ένα λογισμικό αξιολόγησης, βάσει της εκάστοτε εφαρμογής. Έτσι, με βάση την αρχή της επιβίωσης του καλύτερου στην φύση, ο αλγόριθμος προσδιορίζει τα καλύτερα άτομα με βάση την τιμή μιας συνάρτησης στόχου, που ανάλογα με το πρόβλημα καλείται να ελαχιστοποιηθεί ή να μεγιστοποιηθεί. Τα καλύτερα άτομα της προηγούμενης γενιάς (γονείς) συνδυάζονται/αναπαράγονται για να παράξουν τον νέο πληθυσμό της επόμενης γενιάς (απόγονοι). Αυτή η διαδικασία επαναλαμβάνεται έως τη σύγκλιση του αλγορίθμου και τα καλύτερα άτομα του πληθυσμού αποτελούν την λύση του αλγορίθμου.

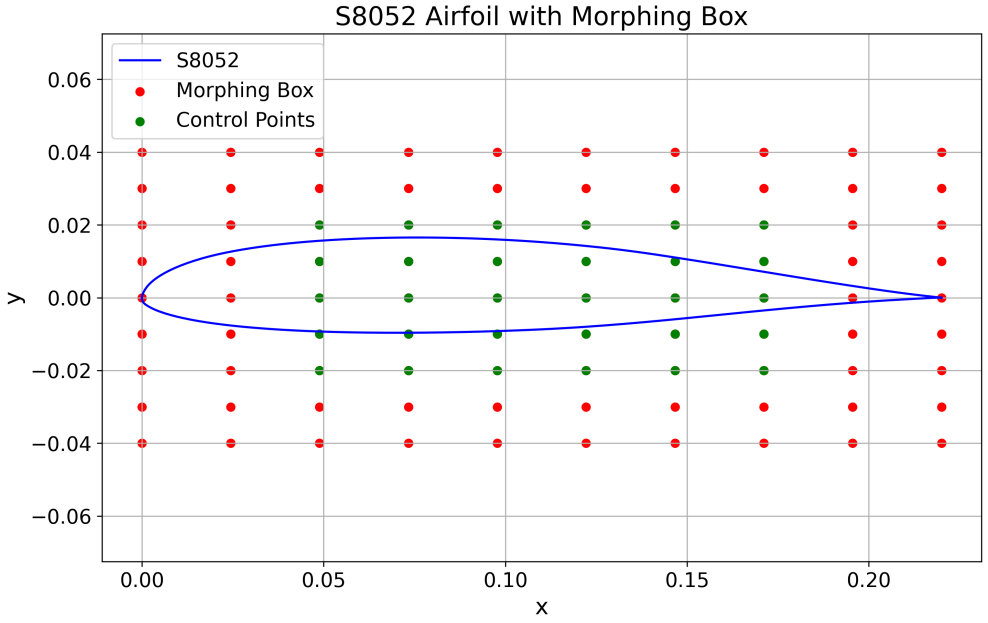
Τα βασικά πλεονεκτήματα των ΕΑ, είναι η δυνατότητα χρήσης σε κάθε είδους πρόβλημα με την προϋπόθεση ότι υπάρχει το κατάλληλο λογισμικό αξιολόγησης, η δυνατότητα μη-εγκλωβισμού σε τοπικά ακρότατα (ως στοχαστική μέθοδος) και η άμεση εφαρμογή τους σε νέα προβλήματα, καθώς δεν απαιτούνται παρεμβάσεις στην διαδικασία της βελτιστοποίησης, όπως στις αιτιοκρατικές μεθόδους. Το κύριο τους μειονέκτημα είναι ότι απαιτούν μεγάλο αριθμό αξιολογήσεων. Αυτό αποτελεί μεγάλο εμπόδιο στην περίπτωση της αεροδυναμικής βελτιστοποίησης σχήματος, όπως στις εφαρμογές της Διπλωματικής αυτής Εργασίας, όπου το λογισμικό που λύνει τις εξισώσεις Navier-Stokes και προβλέπει τη ροή είναι χρονοβόρο.

Εφαρμογή των Μεθόδων Βασισμένων στα Δέντρα Αποφάσεων στην Βελτιστοποίηση Σχήματος

Η πρώτη περίπτωση αφορά της βελτιστοποίησης σχήματος της μεμονωμένης αεροτομής S8052, με στόχο την ελαχιστοποίηση του συντελεστή οπισθέλκουσας, c_D και με περιορισμό στον συντελεστή άνωσης, c_L, της νέας γεωμετρίας να είναι μεγαλύτερος από αυτόν της αρχικής. Για την επίλυση της ροής χρησιμοποιήθηκε το λογισμικό PUMA, επιλύοντας τις Reynolds-Averaged Navier-Stokes (RANS) εξισώσεις, χρησιμοποιώντας το μοντέλο τύρβης μίας εξίσωσης Spalart-Allmaras και το μοντέλο μετάβασης δύο εξισώσεων $\gamma - \tilde{Re}_\theta$.

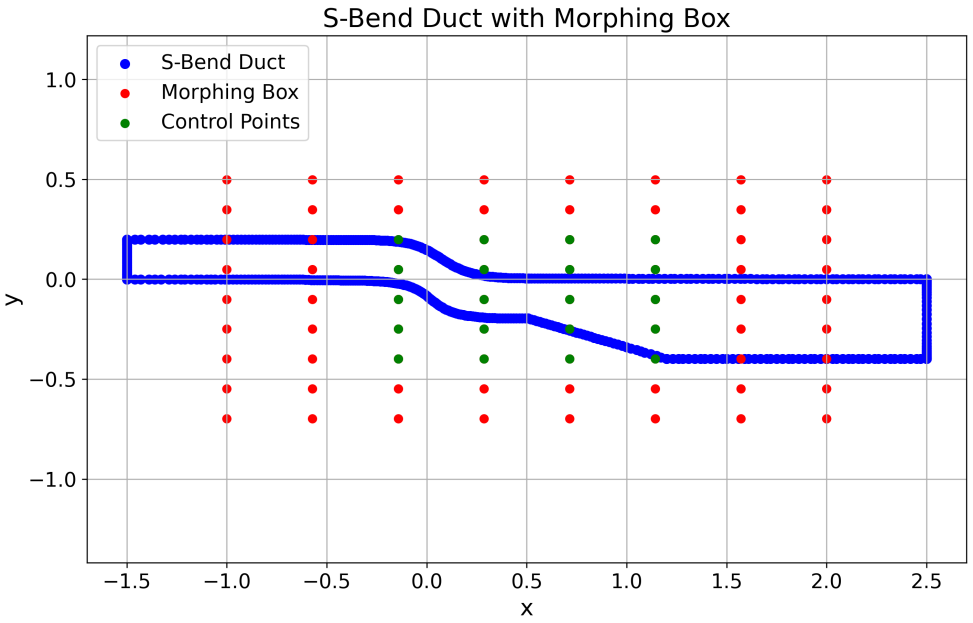
Η δεύτερη περίπτωση είναι ένας αγωγός σχήματος S με στόχο τις ελάχιστες απώλειες ολικής πίεσης, με περιορισμό στον όγκο του αγωγού να είναι μικρότερος από αυτόν της αρχικής γεωμετρίας. Η ροή μοντελοποιήθηκε ως στρωτή.

Η γεωμετρία της αεροτομής παραμετρικοποιείται με χρήση ενός πλέγματος ελέγχου βασισμένου σε ογκομετρικές NURBS 10×9, Σχήμα 6.2, ενώ του αγωγού με χρήση αντίστοιχου πλέγματος ελέγχου 8×9, Σχήμα 6.3. Τα 30 σημεία ελέγχου της αεροτομής και τα 20 σημεία ελέγχου του αγωγού φαίνονται με πράσινο χρώμα και επιτρέπεται να κινηθούν κατά την κάθετη κατεύθυνση, σε εύρος $\pm 20\%$ της κάθετης απόστασης μεταξύ δύο γειτονικών (κατά την κάθετη κατεύθυνση) σημείων, ενώ τα σημεία με κόκκινο χρώμα κρατιούνται σταθερά.



Σχήμα 6.2: *Περίπτωση Αεροτομής: Η Αεροτομή S8052 με το πλέγμα ελέγχου βασισμένου σε ογκομετρικές NURBS. Τα πράσινα σημεία ελέγχου μπορούν να κινηθούν μόνο στην y κατεύθυνση, ενώ τα κόκκινα σημεία μένουν σταθερά.*

Για τη δημιουργία της Βάσης Δεδομένων (ΒΔ), με την οποία θα εκπαιδευτούν τα μοντέλα, και για τις δύο εφαρμογές, δημιουργούνται 200 δείγματα διαφορετικών γεωμετριών, εκ των οποίων 150 με την τεχνική Latin Hypercube Sampling (LHS) και 50 με την τεχνική Random Sampling, έτσι ώστε να γίνει όσο το δυνατόν καλύτερη κάλυψη του χώρου σχεδιασμού από την ΒΔ. Στη συνέχεια, για κάθε γεωμετρία επιλύθηκε η ροή με το λογισμικό CFD, δημιουργώντας έτσι τις ΒΔ. Επιπλέον, για την αξιολόγηση την απόδοσης των εκπαιδευμένων μοντέλων, δημιουργήθηκε ακόμη μία ΒΔ με δεδομένα για δοκιμή. Αυτή η ΒΔ χρησιμοποιείται μόνο για τον έλεγχο της ποιότητας των προβλέψεων των μοντέλων και δεν σχετίζεται ούτε χρεώνεται στη βελτιστοποίηση.



Σχήμα 6.3: *Περίπτωση Αγωγού: Αγωγός Σχήματος S με το πλέγμα ελέγχου βασισμένου σε ογκομετρικές NURBS. Τα πράσινα σημεία ελέγχου μπορούν να κινηθούν μόνο στην y κατεύθυνση, ενώ τα κόκκινα σημεία μένουν σταθερά.*

Σε όλες τις δοκιμές, και για τις δύο εφαρμογές, το λογισμικό EASY χρησιμοποιείται ως το εργαλείο βελτιστοποίησης. Όλες οι δοκιμές των προτεινόμενων μεθόδων συγκρίνονται με τα αποτελέσματα ενός EA και MAEA (RBF on-line), στο ίδιο πρόβλημα, με πληθυσμό γονέων $\mu = 25$ και πληθυσμό απογόνων $\lambda = 40$. Ακόμη, οι προτεινόμενες μέθοδοι ενσωματώνονται σε έναν EA ή MAEA (RBF on-line), με τις ρυθμίσεις που αναφέρθηκαν, και αναγράφεται ανάλογα στο όνομα της εκάστοτε δοκιμής. Τέλος, όλες οι δοκιμές είχαν το ίδιο υπολογιστικό κόστος, τις 500 αξιολογήσεις από το λογισμικό CFD. Στις δοκιμές που απαιτείται η χρήση της ΒΔ για την εκπαίδευση των μοντέλων, το υπολογιστικό κόστος της ΒΔ αφαιρείται από τον προϋπολογισμό, αφήνοντας 300 αξιολογήσεις για το λογισμικό βελτιστοποίησης.

Μοντέλα Παλινδρόμησης βασισμένα στα Δέντρα Αποφάσεων ως off-line Μεταμοντέλα

Από τη θεωρία των μοντέλων βασισμένων στα ΔΑ είναι γνωστό, ότι καθώς οι προβλέψεις των ΔΑ περιορίζονται σε μέσους όρους των αποκρίσεων των ατόμων που ανήκουν στην ΒΔ, τα μοντέλα δεν μπορούν να κάνουν ακριβείς προβλέψεις σε χωρία του χώρου σχεδιασμού για τα οποία δεν έχουν εκπαιδευτεί. Παρ' όλα αυτά, στην πρώτη προσέγγιση δοκιμάζονται μοντέλα παλινδρόμησης βασισμένα στα ΔΑ ως off-line εκπαιδευμένα μεταμοντέλα στη βελτιστοποίηση.

Στην περίπτωση της αεροτομής, εκπαιδεύτηκαν δύο μοντέλα παλινδρόμησης Random

Δοκιμή	c_D	c_L
EA	0.00669148	0.193053
MAEA (RBF on-line)	0.00653602	0.176523
MAEA (Random Forest trained off-line)	0.00704201	0.165928
MAEA (Gradient Boosting trained off-line)	0.00667112	0.166800
Αρχική Γεωμετρία	0.00750318	0.165968

Πίνακας 6.1: *Περίπτωση Αεροτομής: Τα αποτελέσματα της βελτιστοποίησης με στόχο το ελάχιστο c_D χρησιμοποιώντας Μοντέλα Παλινδρόμησης βασισμένα στα ΔΑ ως off-line Μεταμοντέλα, σε σύγκριση με τα αποτελέσματα από τον ΕΑ και MAEA (RBF on-line). Οι τιμές των c_D και c_L αντιστοιχούν στην καλύτερη λύση της κάθε δοκιμής, υπολογισμένα από το λογισμικό CFD.*

Forest και δύο Gradient Boosting στη ΒΔ, με σκοπό να προβλέπουν το c_D και το c_L. Οι υπερπαράμετροι αυτών των μοντέλων ρυθμίστηκαν χρησιμοποιώντας δύο εργαλεία της Python από τη βιβλιοθήκη scikit-learn, το GridSearchCV και το RandomizedSearchCV. Τα μοντέλα αξιολογήθηκαν στη ΒΔ των δοκιμών.

Έπειτα, πραγματοποιείται η βελτιστοποίηση με ΕΑ, στο λογισμικό EASY, χρησιμοποιώντας τα εκπαιδευμένα μοντέλα ως λογισμικό αξιολόγησης. Η διαδικασία της βελτιστοποίησης γίνεται σε κύκλους, όπου σε κάθε κύκλο ο ΕΑ τρέχει για 200 αξιολογήσεις. Στο τέλος κάθε κύκλου, οι καλύτερες δύο λύσεις επαναξιολογούνται από το λογισμικό CFD και προστίθενται στη ΒΔ. Τα μοντέλα επανεκπαιδεύονται στη νέα ΒΔ και ξεκινά ο νέος κύκλος. Αυτή η διαδικασία έγινε 4 φορές, έως να μην υπάρχει περαιτέρω βελτίωση. Τα αποτελέσματα παρουσιάζονται στον Πίνακα 6.1.

Σημειώνεται, ότι στην περίπτωση του μοντέλου Random Forest στον αρχικό πληθυσμό τοποθετήθηκε η αρχική γεωμετρία, ενώ στην περίπτωση του μοντέλου Gradient Boosting η καλύτερη λύση από την ΒΔ. Αυτός είναι και ο λόγος που δεν παρατηρείται βελτίωση στην περίπτωση του μοντέλου Gradient Boosting, επιβεβαιώνοντας τα εμπόδια για χρήση αυτών των μοντέλων ως μεταμοντέλα.

Παρατηρείται ότι παρουσιάζεται αμελητέα βελτίωση στη συνάρτηση κόστους κατά τη διάρκεια της βελτιστοποίησης. Στην περίπτωση του Gradient Boosting δεν επιτεύχθηκε καμία βελτίωση και η λύση που τοποθετήθηκε στον αρχικό πληθυσμό επιστρέφεται ως βέλτιστη λύση. Οι βελτιώσεις που παρουσιάζονται μεταξύ των κύκλων, οφείλονται στο ότι το μοντέλο επανεκπαιδεύεται και για τις ίδιες λύσεις δίνει πιο ακριβείς προβλέψεις. Τα αποτελέσματα που προέκυψαν υποστηρίζουν τις αρχικές επιφυλάξεις για τη χρήση των μοντέλων βασισμένων στα ΔΑ ως off-line trained μεταμοντέλα και, για αυτόν τον λόγο, δεν δοκιμάστηκε αυτή η προσέγγιση στην περίπτωση του αγωγού.

Χρήση Μοντέλων βασισμένων στα Δέντρα Αποφάσεων για Διαχείριση Περιορισμών

Στη δεύτερη προσέγγιση, χρησιμοποιούνται μοντέλα ταξινόμησης βασισμένα στα ΔΑ για να προ-αποφανθεί εάν μία υποψήφια λύση είναι αποδεκτή ή όχι, κατά τη διάρκεια

της βελτιστοποίησης. Ο βασικός στόχος είναι, κατά τη διάρκεια της βελτιστοποίησης, τα μοντέλα αυτά να ταξινομούν κάθε υποψήφια λύση βάσει του αν ικανοποιεί τον περιορισμό ή όχι. Στην περίπτωση που μία λύση δεν ταξινομείται ως αποδεκτή, δεν αξιολογείται από το ακριβό λογισμικό CFD, αποφεύγοντας με αυτόν τον τρόπο δαπανηρές αξιολογήσεις για μη εφικτές λύσεις.

Σε κάθε στοιχείο της Βάσης Δεδομένων, ανάλογα με την τιμή του περιορισμού της, αποδίδεται μία κλάση. Με αυτόν τον τρόπο, όλα τα στοιχεία ταξινομούνται στο εάν είναι αποδεκτά ή όχι, με δύο γκρι ζώνες, πάνω και κάτω από την τιμή της αρχικής γεωμετρίας, την οποία οι αποδεκτές λύσεις πρέπει να ξεπερνούν. Τα όρια των τεσσάρων κλάσεων και για τις δύο εφαρμογές παρουσιάζονται στους Πίνακες 6.2 και 6.3.

Πίνακας 6.2: *Περίπτωση Αεροτομής: Οι τιμές κλάσης ανά εύρος τιμών του c_L.*

Εύρος Τιμών c_L	Class
$c_L < 0.162$	0
$0.162 < c_L < 0.166$	1
$0.166 < c_L < 0.170$	2
$c_L > 0.170$	3

Πίνακας 6.3: *Περίπτωση Αγωγού: Οι τιμές κλάσης ανά εύρος τιμών του $Volume$.*

Εύρος Τιμών $Volume$	Class
$Volume > 0.766$	0
$0.762 < Volume < 0.766$	1
$0.758 < Volume < 0.762$	2
$Volume < 0.758$	3

Τρία μοντέλα ταξινόμησης, Random Forest, Gradient Boosting και XGBoost, δημιουργήθηκαν και εκπαιδεύτηκαν στην ανανεωμένη ΒΔ, για κάθε εφαρμογή. Οι υπερπαράμετροι αυτών ρυθμίστηκαν χρησιμοποιώντας τα GridSearchCV και RandomizedSearchCV και τα εκπαιδευμένα μοντέλα αξιολογήθηκαν στη ΒΔ των δοκιμών.

Επιπλέον, εφαρμόζονται διαφορετικά σχήματα, όσον αφορά τα άτομα ποιών κλάσεων θα προχωρούν σε αξιολόγηση. Τα τρία σχήματα που δοκιμάζονται παρουσιάζονται στον Πίνακα 6.4.

Σχήμα	Κλάσεις που δεν Αξιολογούνται από το CFD
"Test 1"	0, 1
"Test 2"	0, 1, 2
"Test 3"	0

Πίνακας 6.4: *Τα διαφορετικά σχήματα που εφαρμόστηκαν στην δεύτερη προσέγγιση.*

Όλα τα σχήματα δοκιμάζονται με το μοντέλο Random Forest. Έπειτα, το σχήμα που παρουσίασε την καλύτερη απόδοση με το μοντέλο Random Forest, δοκιμάζεται και με τα μοντέλα Gradient Boosting και XGBoost. Σκοπός είναι να προσδιοριστεί το καλύτερο μοντέλο και σχήμα για αυτήν την προσέγγιση.

Δοκιμή	c_D	c_L	% Βελτίωσης της Αντικειμενικής
EA	0.00669148	0.193053	10.82%
MAEA (RBF on-line)	0.00653602	0.176523	12.89%
Test 1 (Random Forest) - EA	0.00662657	0.216523	11.68%
Test 2 (Random Forest) - EA	0.00644965	0.172007	14.04%
Test 3 (Random Forest) - EA	0.00662657	0.216523	11.68%
Test 2 (Gradient Boosting) - EA	0.00661844	0.183463	11.79%
Test 2 (XGBoost) - EA	0.0068241	0.184521	9.05%
DT Design Space Selection - Try1 - EA	0.00666616	0.179254	11.16%
DT Design Space Selection - Try1 - MAEA (RBF on-line)	0.00658729	0.192346	12.21%
DT Design Space Selection - Try2 - MAEA (RBF on-line)	0.0064909	0.183642	13.49%
Combination Test - EA	0.00639942	0.176541	14.71%
Combination Test - MAEA (RBF on-line)	0.00640940	0.180446	14.58%
Αρχική Γεωμετρία	0.00750318	0.165968	

Πίνακας 6.5: *Περίπτωση Αεροτομής: Τα αποτελέσματα της βελτιστοποίησης όλων των δοκιμών, με στόχο το ελάχιστο c_D, σε σύγκριση με τα αποτελέσματα από τον EA και MAEA (RBF on-line). Οι τιμές των c_D και c_L αντιστοιχούν στην καλύτερη λύση της κάθε δοκιμής, υπολογισμένα από το λογισμικό CFD.*

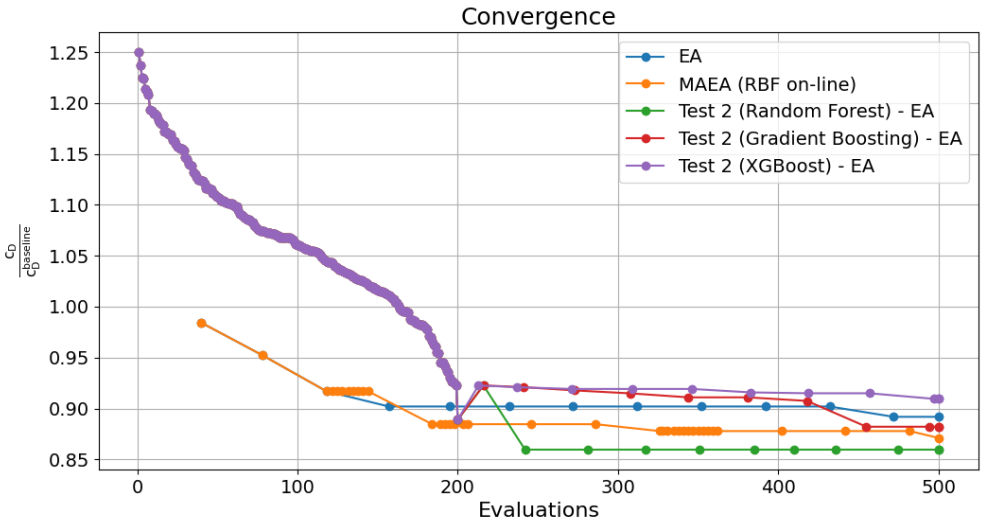
Καθώς για την εκπαίδευση των μοντέλων χρειάζεται η ΒΔ που χρησιμοποιεί 200 αξιολογήσεις, το καλύτερο άτομο αυτής τροφοδοτείται στον αρχικό πληθυσμό, καθώς το υπολογιστικό κόστος έχει ήδη χρεωθεί.

Για την περίπτωση της αεροτομής, τα αποτελέσματα για όλες τις δοκιμές παρουσιάζονται στον Πίνακα 4.8 και οι συγκλίσεις των δοκιμών με το σχήμα "Test 2" σε σύγκριση με αυτές των EA και MAEA (RBF on-line) στο Σχήμα 6.4. Όπως φαίνεται από τον Πίνακα, όλες οι δοκιμές κατάφεραν ικανοποιητικά αποτελέσματα, όμως μόνο η δοκιμή του μοντέλου Random Forest με το σχήμα "Test 2" κατάφερε να βρει καλύτερη λύση (βελτίωση της συνάρτησης στόχου κατά 14%) από το MAEA (RBF on-line) (βελτίωση της συνάρτησης στόχου κατά ∼13%). Ακόμη, από το Σχήμα 6.4, φαίνεται ότι αυτή η δοκιμή κατάφερε να βρει την καλύτερη λύση σε μόλις 50 αξιολογήσεις, από την έναρξη του EA.

Για την περίπτωση του αγωγού, τα αποτελέσματα φαίνονται στον Πίνακα 6.6 και οι συγκλίσεις των δοκιμών με το καλύτερο σχήμα, "Test 1", στο Σχήμα 6.5. Σε αντίθεση με την περίπτωση της αεροτομής, πολλές δοκιμές ξεπέρασαν τον EA και MAEA (RBF on-line), με τη δοκιμή με το μοντέλο Random Forest με το σχήμα "Test 1" να πετυχαίνει ∼30% βελτίωση της συνάρτησης στόχου, σε αντίθεση με τον απλό EA που πέτυχε μόλις ∼12%.

Εξερεύνηση του Χώρου Σχεδιασμού με Δέντρα Αποφάσεων

Όπως έχει αναφερθεί, ο αλγόριθμος των ΔΑ χωρίζει τον χώρο σχεδιασμού σε διαφορετικές υποπεριοχές και δίνει μία τιμή σε κάθε μία από αυτές, την μέση τιμή των ατόμων της ΒΔ που ανήκουν σε αυτή την περιοχή. Έτσι, με την επιλογή της κατάλληλης περιοχής και την εφαρμογή του λογισμικού βελτιστοποίησης (EA ή MAEA) σε αυτήν, είναι δυνατό η βελτιστοποίηση να οδηγηθεί στη βέλτιστη λύση με μειωμένο αριθμό αξιολογήσεων. Φυσικά, σε αυτήν την προσέγγιση υπάρχει πάντοτε το ρίσκο, εάν δεν επιλεχθεί η υποπεριοχή που περιέχει την καλύτερη λύση, ο EA να περιοριστεί



Σχήμα 6.4: *Περίπτωση Αεροτομής: Σύγκριση των συγκλίσεων της συνάρτησης στόχου για διαφορετικές δοκιμές στην δεύτερη προσέγγιση.*

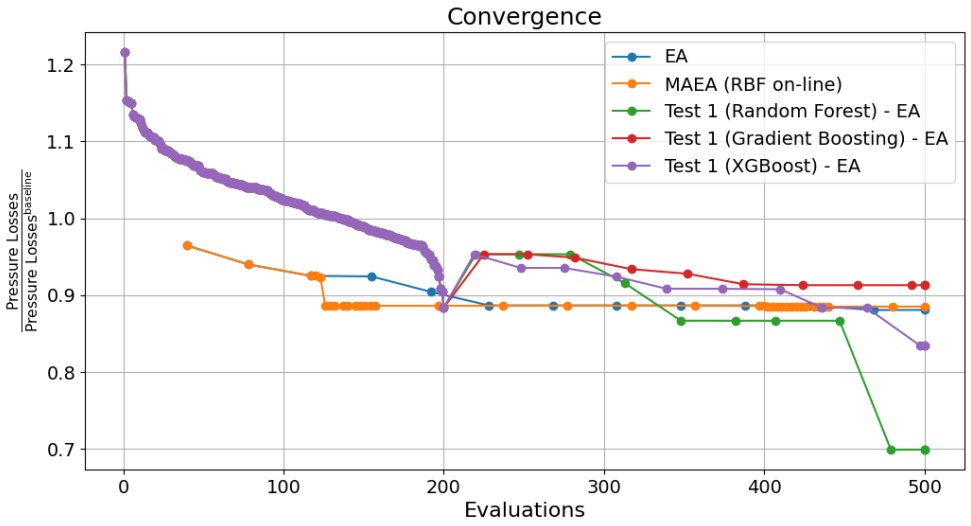
Δοκιμή	Pressure Losses	Duct Volume	% Βελτίωσης της Αντικειμενικής
EA	98.0712	0.761158	11.93%
MAEA (RBF on-line)	98.5599	0.759584	11.49%
Test 1 (Random Forest) - EA	77.8402	0.757096	30.10%
Test 2 (Random Forest) - EA	97.5854	0.760115	12.36%
Test 3 (Random Forest) - EA	96.378	0.761182	13.45%
Test 1 (Gradient Boosting) - EA	101.662	0.757831	8.70%
Test 1 (XGBoost) - EA	92.9512	0.760542	16.52%
DT Design Space Selection - Try1 - MAEA (RBF on-line)	101.06	0.757052	9.24%
DT Design Space Selection - Try3 - MAEA (RBF on-line)	88.1883	0.757564	20.80%
DT Design Space Selection - Try4 - MAEA (RBF on-line)	101.816	0.758155	8.56%
Combination Test - EA	86.0479	0.759015	22.72%
Combination Test - MAEA (RBF on-line)	87.3704	0.757583	21.54%
Αρχική Γεωμετρία	111.3515	0.761546	

Πίνακας 6.6: *Περίπτωση Αγωγού: Τα αποτελέσματα της βελτιστοποίησης όλων των δοκιμών, με στόχο τις ελάχιστες απώλειες ολικής πίεσης, σε σύγκριση με τα αποτελέσματα από τον EA και MAEA (RBF on-line). Οι τιμές των "Pressure Losses" και "Duct Volume" αντιστοιχούν στην καλύτερη λύση της κάθε δοκιμής, υπολογισμένα από το λογισμικό CFD.*

σε μία μη βέλτιστη περιοχή.

Και για τις δύο εφαρμογές, εκπαιδεύεται ένα CART ΔΑ στη ΒΔ, ρυθμίζοντας τις υπερπαραμέτρους του χρησιμοποιώντας τα GridSearchCV και RandomizedSearchCV, ενώ επιλέγεται να έχει 10 κόμβους φύλλα, επομένως να χωρίζει τον χώρο σχεδιασμού σε 10 υποπεριοχές.

Για την επιλογή της βέλτιστης υποπεριοχής γίνονται διάφορες δοκιμές και παρουσιάζονται στον Πίνακα 6.7. Καθώς το κόστος της ΒΔ έχει χρεωθεί για κάθε μία από αυτές τις προσπάθειες, το καλύτερο άτομο της ΒΔ που ανήκει στην εκάστοτε περιοχή



Σχήμα 6.5: *Περίπτωση Αγωγού: Σύγκριση των συγκλίσεων της συνάρτησης στόχου για διαφορετικές δοκιμές στην δεύτερη προσέγγιση.*

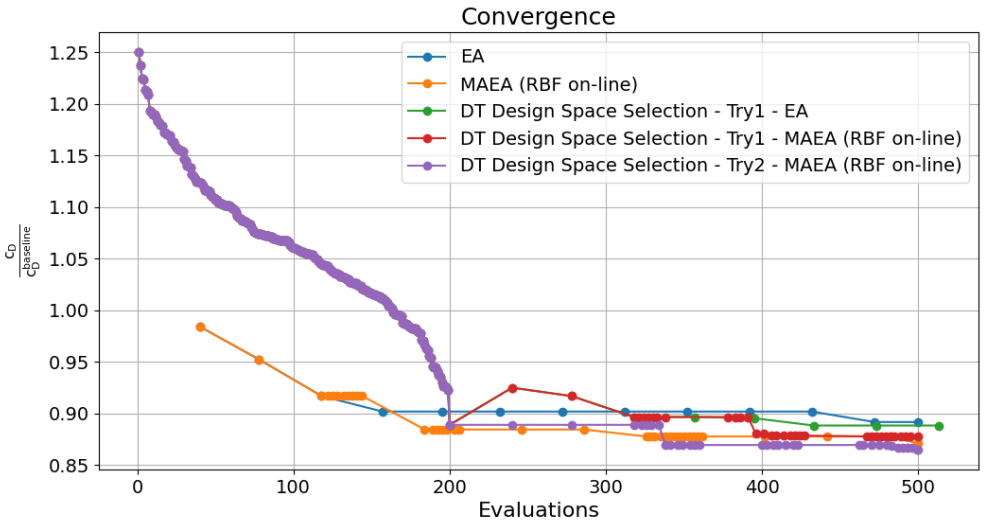
τροφοδοτείται στον αρχικό πληθυσμό.

Δοκιμές	Υποπεριοχή που αντιστοιχούν
"Try1"	Η υποπεριοχή με την καλύτερη τιμή από τους κόμβους-φύλλα του εκπαιδευμένου ΔΑ.
"Try2"	Η υποπεριοχή στην οποία ανήκει το καλύτερο άτομο της ΒΔ.
"Try3"	Η υποπεριοχή στην οποία ανήκει η καλύτερη λύση που έχει βρεθεί από όλες τις προηγούμενες δοκιμές.
"Try4"	Η υποπεριοχή με την δεύτερη καλύτερη τιμή από τους κόμβους-φύλλα του εκπαιδευμένου ΔΑ.

Πίνακας 6.7: *Οι διαφορετικές δοκιμές που δοκιμάστηκαν στην τρίτη προσέγγιση.*

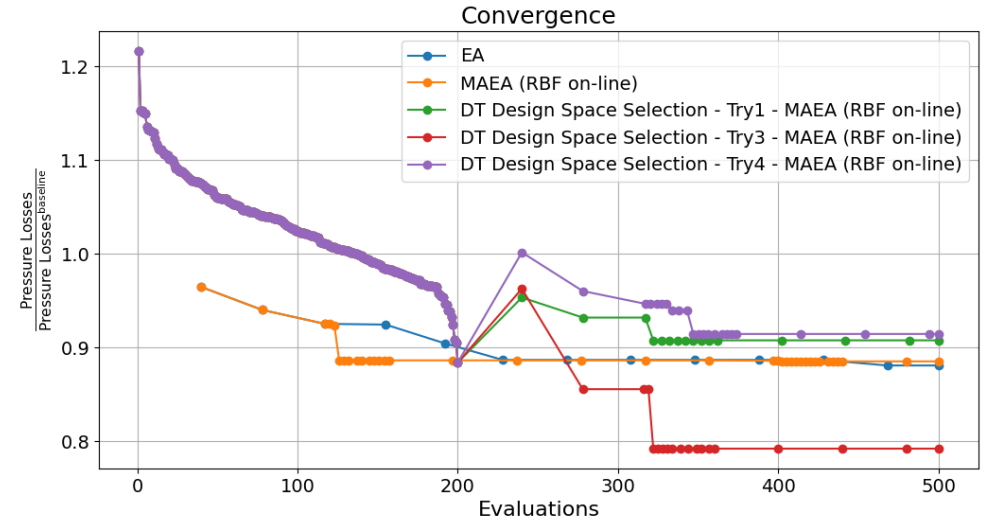
Τα αποτελέσματα, για την περίπτωση της αεροτομής, παρουσιάζονται στον Πίνακα 6.5, ενώ οι συγκλίσεις στο Σχήμα 6.6. Από τον Πίνακα, βλέπουμε ότι όλες οι δοκιμές πέτυχαν ικανοποιητικά αποτελέσματα, αλλά μόνο η δοκιμή με MAEA (RBF on-line) στην υποπεριοχή της Try2 βρήκε καλύτερη λύση από το MAEA (RBF on-line).

Για την περίπτωση του αγωγού, τα αποτελέσματα φαίνονται στον Πίνακα 6.6 και οι συγκλίσεις στο Σχήμα 6.7. Παρατηρείται ότι μόνο η δοκιμή στην υποπεριοχή Try3 με MAEA (RBF on-line), κατάφερε να ξεπεράσει την απόδοση του ΕΑ, με βελτίωση της συνάρτησης στόχου κατά ∼21%. Σημειώνεται ότι για την επιλογή αυτής της



Σχήμα 6.6: *Περίπτωση Αεροτομής: Σύγκριση των συγκλίσεων της συνάρτησης στόχου για διαφορετικές δοκιμές στην τρίτη προσέγγιση.*

υποπεριοχής χρησιμοποιήθηκε γνώση από προηγούμενες δοκιμές, εγείροντας ανησυχίες για την κατάλληλη μέθοδο επιλογής της βέλτιστης υποπεριοχής.



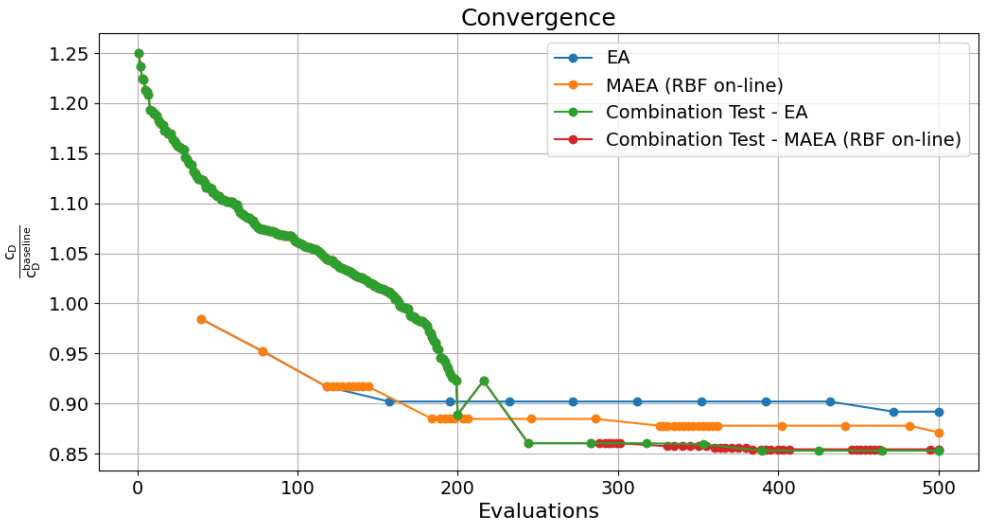
Σχήμα 6.7: *Περίπτωση Αγωγού: Σύγκριση των συγκλίσεων της συνάρτησης στόχου για διαφορετικές δοκιμές στην τρίτη προσέγγιση.*

Συνδυασμός Μεθόδων

Τέλος, η δεύτερη και η τρίτη προσέγγιση συνδυάζονται προκειμένου να αξιολογηθεί ο συνδυασμός αυτών. Καθώς, η ΒΔ είναι απαραίτητη και για τις δύο αυτές προσεγγίσεις, δεν υπάρχει επιπλέον υπολογιστικό κόστος για να συνδυαστούν σε μία δοκιμή. Σε κάθε

περίπτωση συνδυάζονται οι καλύτερες δοκιμές από κάθε προσέγγιση και δοκιμάζονται και με ΕΑ και MAEA (RBF on-line). Ακόμη, το καλύτερο άτομο της ΒΔ που ανήκει στην εκάστοτε περιοχή και έχει την κατάλληλη τιμή κλάσης τροφοδοτείται κάθε φορά στον αρχικό πληθυσμό.

Συγκεκριμένα, στην περίπτωση της αεροτομής, το μοντέλο ταξινόμησης Random Forest με το σχήμα "Test 2" εφαρμόζεται στην υποπεριοχή της δοκιμής "Try2", με ΕΑ και MAEA (RBF on-line). Τα αποτελέσματα παρουσιάζονται στον Πίνακα 6.5 και οι συγκλίσεις στο Σχήμα 6.8. Και οι δύο δοκιμές πέτυχαν καλύτερες λύσεις από όλες τις υπόλοιπες δοκιμές (βελτίωση της συνάρτησης στόχου κατά ∼14.5%) σε πολύ μικρό αριθμό αξιολογήσεων.



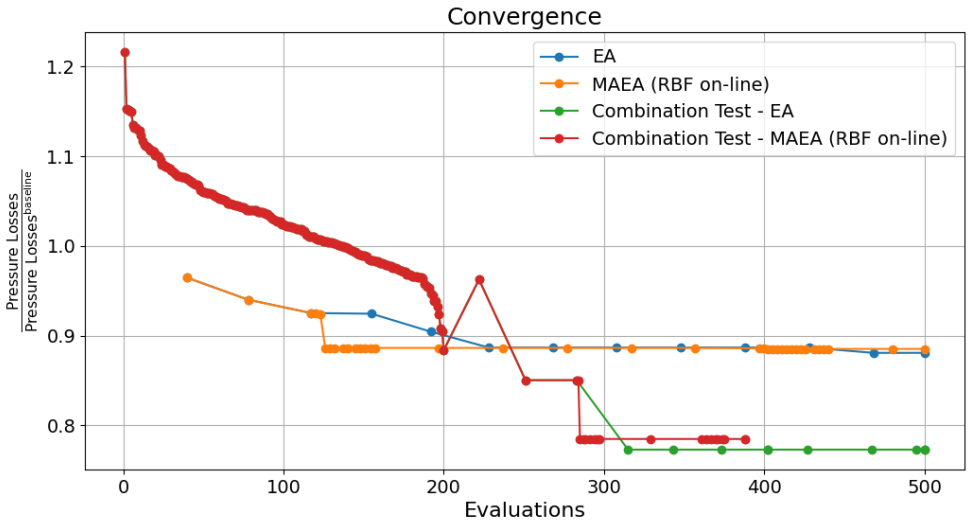
Σχήμα 6.8: *Περίπτωση Αεροτομής: Σύγκριση των συγκλίσεων της συνάρτησης στόχου για τις συνδυαστικές δοκιμές.*

Στην περίπτωση του αγωγού, το μοντέλο ταξινόμησης Random Forest με το σχήμα "Test 1" συνδυάζεται με την υποπεριοχή της δοκιμής "Try3", κάνοντας δύο δοκιμές: μία με ΕΑ (βελτίωση της συνάρτησης στόχου κατά ∼23%) και μία με MAEA (RBF on-line) (βελτίωση της συνάρτησης στόχου κατά ∼22%). Τα αποτελέσματα παρουσιάζονται στον Πίνακα 6.6 και στο Σχήμα 6.9. Και οι δύο δοκιμές βρήκαν καλύτερες λύσεις από αυτήν του συμβατικού ΕΑ και MAEA (RBF on-line) με γρήγορη σύγκλιση, όμως δεν κατάφεραν να βρουν καλύτερες λύσεις από την δοκιμή "Test 1 (Random Forest) - ΕΑ".

Συμπεράσματα

Μετά την ολοκλήρωση των προηγούμενων μελετών, εξάγονται τα ακόλουθα συμπεράσματα:

- Εκφράζονται επιφυλάξεις σχετικά με τη χρήση των μοντέλων παλινδρόμησης με βάση τα ΔΑ ως off-line εκπαιδευμένα μεταμοντέλα στη βελτιστοποίηση.



Σχήμα 6.9: *Περίπτωση Αγωγού: Σύγκριση των συγκλίσεων της συνάρτησης στόχου για τις συνδυαστικές δοκιμές.*

- Η προσέγγιση ταξινόμησης με βάση τους περιορισμούς μπορεί να επιτύχει καλύτερες λύσεις από τον EA και MAEA (RBF on-line). Ωστόσο, για το βέλτιστο σχήμα απαιτείται περαιτέρω διερεύνηση, ιδίως μεταξύ των "Test 1" και "Test 2". Το μοντέλο Random Forest παρουσιάζει την καλύτερη απόδοση μεταξύ των μοντέλων.
- Η προσέγγιση εξερεύνησης του χώρου σχεδιασμού βοηθά αποτελεσματικά τον EA ή τον MAEA να συγκλίνει στη βέλτιστη λύση, έχοντας επιλέξει την κατάλληλη υποπεριοχή. Μέσα από τις προηγούμενες μελέτες δεν βρέθηκε συστηματικός τρόπος προσδιορισμού της βέλτιστης υποπεριοχής. Εναπόκειται στον χρήστη να αξιοποιήσει τις γνώσεις, σε κάθε πρόβλημα, για να επιλέξει την καλύτερη υποπεριοχή, με τον κίνδυνο ότι το λογισμικό βελτιστοποίησης θα μπορούσε να περιοριστεί σε μη-βέλτιστη υποπεριοχή.
- Ο συνδυασμός της δεύτερης και της τρίτης προσέγγισης μπορεί να βρει καλύτερο αποτέλεσμα από άλλες μεθόδους με γρήγορη σύγκλιση, αρκεί να επιλεχθεί το κατάλληλο σχήμα ταξινόμησης και η βέλτιστη υποπεριοχή.