Graduation Thesis
School of Engineering, Tohoku University

An Artificial Neural Network-Assisted Genetic Algorithm with Application to Multi-Objective Transonic Airfoil Shape Optimization

Department of Mechanical and Aerospace Engineering
Fluids Engineering with Data Science Lab.
Supervisor: Assoc. Prof. Koji Shimoyama

B7TB1705, Muhammad Alfiyandy Hariansyah
（September, 2021）

TOHOKU UNIVERSITY

School of Engineering

An Artificial Neural Network-Assisted Genetic Algorithm with Application to Multi-Objective Transonic Airfoil Shape Optimization

（人工ニューラルネットワーク支援型遺伝的アルゴリズムの多目的遷音速翼型形状最適化への応用）

A thesis submitted for the bachelor's degree (Engineering)

Department of Mechanical and Aerospace Engineering

by

Muhammad Alfiyandy HARIANSYAH

January 26, 2021

Contents

Chapter 1 Introduction

1.1 Research background

1.1.1 Engineering design optimization

Engineering design optimization is the process of finding design candidates with the best performance subject to some objective function(s) and constraint(s). The search process is done on the design space that has been pre-defined beforehand. Unlike conventional design process that relies on the designer experience (draw and build), engineering design now shifts to approaches which are less dependent on the designer knowledge, experience, and intuition. The approaches include advanced numerical optimization techniques such as evolutionary algorithms. The advancement of computing power has also driven the design process to use numerical simulations to evaluate the design (e.g., computational fluid dynamics: CFD). The marriage between optimization algorithms and numerical simulations gives rise to the field of engineering design optimization.

In the field of fluid machinery, examples include the design optimization of tailless aircraft, micromixer, and car-air conditioner, as depicted in Figure 1.1 The objective is to minimize the fuel burn (or drag coefficient), for example. The design evaluation is done using CFD without having to build and test the real design. The optimization algorithm is thus responsible for conducting the process of finding the best performing design. At the end, the final design surely needs to be built and tested before entering the market.

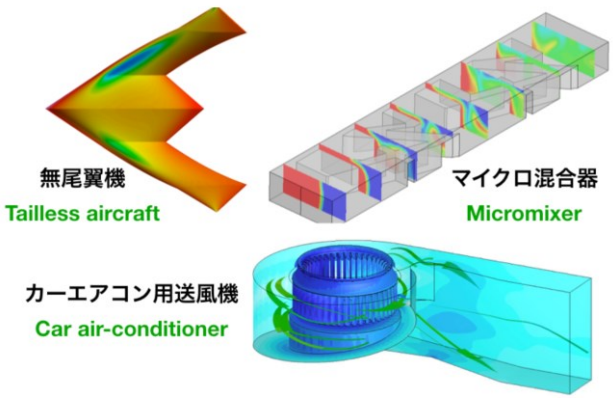


Figure 1.1 Examples of design optimization of fluid machinery [1].

1.1.2 Evolutionary optimization

Due to the rise of optimization algorithms, the designer's task is to formulate the optimization problem. It includes the definition of design variable(s) $\boldsymbol{x}$, objective function(s) $\boldsymbol{f}(\boldsymbol{x})$, inequality constraint(s) $\boldsymbol{g}(\boldsymbol{x})$, equality constraint(s) $\boldsymbol{h}(\boldsymbol{x})$. This is a crucial task because the obtained optimal design highly depends on the problem formulation as follows.

$$\text{Minimize} \quad f_m(\boldsymbol{x}), \qquad \mathrm{m} = 1,2,\ldots,\mathrm{M}; \tag{1.1}$$

$$\text{subject to} \quad g_j(\boldsymbol{x}) \leq 0, \qquad \mathrm{j} = 1,2,\ldots,\mathrm{J}; \tag{1.2}$$

$$h_k(\boldsymbol{x}) = 0, \qquad \mathrm{k} = 1,2,\ldots,\mathrm{K}; \tag{1.3}$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)}, \qquad \mathrm{i} = 1,2,\ldots..\mathrm{n}; \tag{1.4}$$

It basically translates to: "find design variables $\boldsymbol{x}$ with the most minimum objective functions $\boldsymbol{f}$ that meet the criteria given by constraints $\boldsymbol{g}$ and $\boldsymbol{h}$. When there is only one objective function to be minimized, the problem is called single-objective problem. Problems with more than one objective function are called multi-objective problems. The real-world application in this study focuses on the multi-objective problems.

For illustrations, consider a car-buying decision problem. In this problem, we want to buy a car that is cheap but comfortable. So, the objectives are cost and comfort. However, the cars must meet certain criteria such as noise level, CO_2 emission, and safety factors. Those are standards set by the regulators and treated as constraints. The design variables include the shapes, dimensions, and materials of the car. Multiple tradeoff solutions (solution 1, A, B, C, and 2) will be obtained in this problem, depicted in Figure 1.2. Solution 1 and 2 correspond to the extreme solutions while solution A, B, and C indicate the solutions that balance both objectives. They are not superior to each other. The final decision is given to the designer.

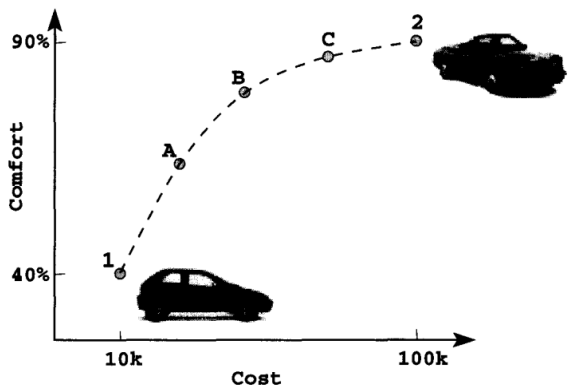


Figure 1.2 Car-buying decision problem [2].

To solve this kind optimization problem (especially multi-objective), researchers have been using population-based approaches such as evolutionary algorithm (EA). Unlike gradient methods, EAs do not need gradient information which is often difficult to calculate. These algorithms are bio-inspired and mimic the process of natural evolution. One of the variants of EAs is NSGA-II, a genetic algorithm proposed by Deb et al. [3]. EAs are attractive since they can locate the optimal solutions close to global solutions.

1.1.3 Surrogate modeling

However, population-based approaches like EA, often need numerous evaluations. In other words, numerous designs must be evaluated numerically or by experiments. In the numerical optimization where numerical simulation is used (e.g., CFD), the evaluation refers to the CFD evaluation of a design. CFD is, most of the time, expensive. One CFD evaluation might take minutes, hours, days or even weeks, depending on the fidelity. In the industry, high-fidelity CFD is preferred to ensure the accuracy, making the use of EAs in engineering design intractable. Alternatively, data-driven surrogate models that can cheaply and analytically map inputs to outputs are often deployed to replace the expensive evaluations. The idea is to perform several design point evaluations using true evaluations (e.g., CFD) and then build the surrogate models based on these data. The surrogate models then replace the expensive evaluations in the EA optimization process to obtain new (hopefully better performing) designs. These new designs can be further used to rebuild the surrogate models and the optimization process repeats until one run out of the computational budget. The examples of surrogate models are Kriging [4], radial basis function (RBF) [5], support vector machine (SVM) [6], and artificial neural network (ANN) [7]. Of them, Kriging is the most popular one, while ANN has not drawn much attention. This fact drives us to study its feasibility when combined with an EA.

1.2 Research objectives

We propose a surrogate-based optimization method that combines NSGA-II and NN (we call it NN+GA). The objectives are to study its efficacy and compare the performance with the NSGA-II without surrogates in the test problems and multi-objective transonic airfoil shape optimization problems. The use of NN is hoped to cut the number of true evaluations needed and reduce the computational time when applied to expensive design optimization problems. Furthermore, the comparison with Kriging is performed as additional study at the end.

1.3 Thesis structure

The present thesis is organized into 6 chapters. In chapter 1, the background and the objectives of the current research are presented. Chapter 2 explains the basics of the artificial neural network (ANN) based surrogate model along with its training procedure and techniques. Then, the surrogate-based optimization that combines the ANN-based surrogate model with a genetic algorithm (GA) is presented in chapter 3. The method is called NN+GA for combining ANN with GA. Chapter 4 presents the application of the proposed method (NN+GA) to the known test problems with analytical functions. Then, the application of NN+GA extends to real-world optimization problem, presented in chapter 5. The problem to be solved is the multi-objective aerodynamic shape optimization of transonic airfoils with CFD as its true evaluation. Chapter 6 tries to compare the proposed surrogate model with a popular model called Kriging for aerodynamic design (the similar problems as in chapter 5). Lastly, the thesis is closed with conclusion, reference, and acknowledgment.

Chapter 2 Artificial neural network

2.1 Introduction

This chapter covers the mathematical background of the artificial neural network (ANN) based surrogate model. The training procedure and techniques to construct the surrogate model is also covered. The techniques include supervised and unsupervised learning. The supervised learning learns from the given sample dataset that has labeled input and output. The unsupervised learning, like the clustering technique, works on their own to discover the inherent structure of unlabeled data. Both techniques constitute the whole process of building the model.

2.2 Definition

Artificial neural network (ANN) is an abstract computational model of the human brain, which is a highly complex, nonlinear, and parallel information processing system. ANN analytically models the relationship between the input variables $\boldsymbol{x} = \{x_1, x_2, \dots, x_n\}^T$, where n is the dimensionality of the design variables, and the output $\boldsymbol{f} = \{f_1, f_2, \dots, f_m\}^T$, where m is the number of expensive-to-evaluate functions to approximate.

Artificial neuron is a building block of ANN that acts as an information-processing unit. It consists of an adder and an activation function. The former acts as a linear combiner that sums the input signals while the latter allows ANN to map nonlinear functions. An externally applied bias b_k can be applied to the adder. A set of neuron makes up a layer. Each neuron is connected to other neurons via links characterized by weight values. Figure 2.1 depicts the whole mechanism of the artificial neuron.

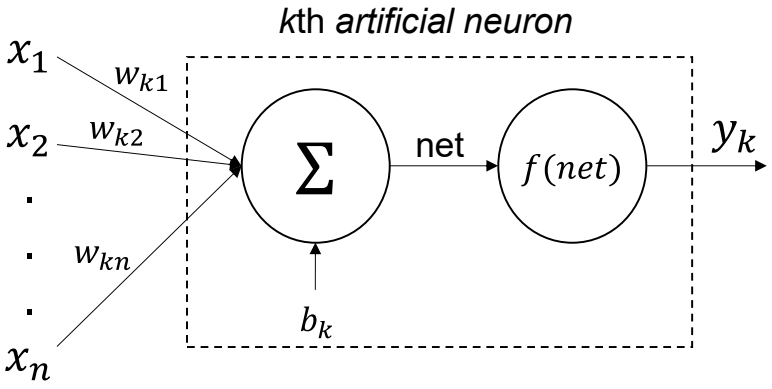


Figure 2.1 Mechanism of an artificial neuron.

The net of a neuron k can be represented as follows.

$$net_k = x_1 w_{k1} + x_2 w_{k2} + \cdots + x_n w_{kn} + b_k \tag{2-1}$$

$$net_k = \sum_{i=1}^{n} x_n w_{kn} + b_k \tag{2-2}$$

Then the neuron computes the output y_k as a certain function f of net_k value as follows.

$$y_k = f(net_k) \tag{2-3}$$

The function f is called the activation function, e.g., sigmoid, Tanh, ReLU, etc.

2.3 Multilayer perceptron

2.3.1 Architectures

The type of neural network being used throughout this study is a multilayer perceptron (MLP). MLP is one of the classes of feedforward neural network (FNN), wherein the connections between the nodes do not form a cycle. An MLP consists of multiple layers of perceptron (at least three): an input layer, an output layer, and hidden layer(s). This study uses an MLP with five layers consisting of an input layer, three hidden layers, and an output layer, shown in Figure 2.2.

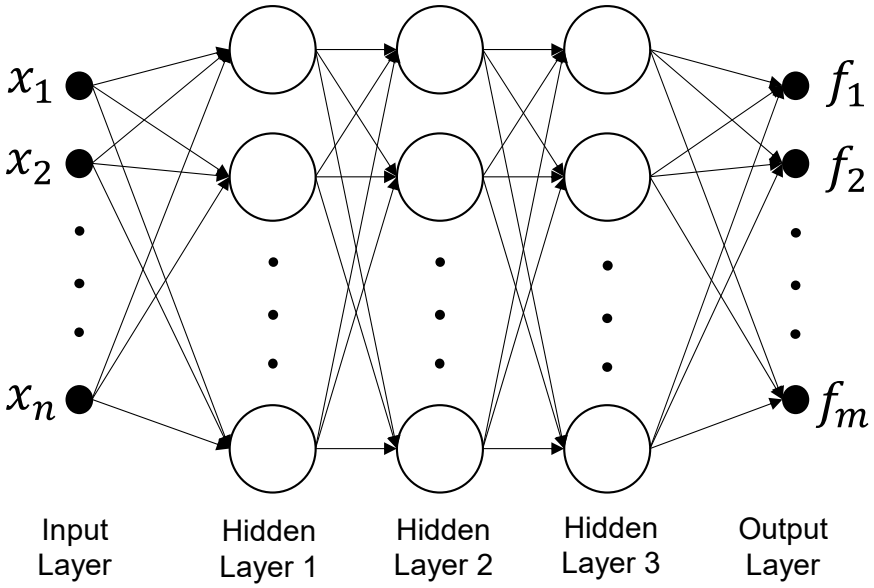


Figure 2.2 Architecture of the multilayer perceptron.

The design variables are fed into the input layer, while the expensive-to-evaluate functions are fed into the output layer. The three hidden layers play a vital role to map the input to output. It has been reported that three hidden layers of neural network are enough to map any highly nonlinear function (e.g., Shen et al. [8]). The activation function being used is LeakyReLU activation function, which is the extended version of ReLU activation function. The activation functions are embedded to every layer in the hidden layers. ReLU always outputs zero in the negative range leading to saturation (also called dying ReLU problem). LeakyReLU is preferable since it introduces a new term in the negative region, as expressed below.

$$ReLU(x) = \max(0, x) \tag{2-4}$$

$$LeakyReLU(x) = \max(0, x) + 0.01 \cdot \min(0, x) \tag{2-5}$$

2.3.2 Training

The surrogate model is constructed by learning a set of data samples consisting of input $\boldsymbol{x}$ (design variables) and target output $\boldsymbol{t}$ (the true evaluation of objective and constraint functions to approximate). A loss function is a measure of how good the model in terms of predicting the desired response. One type of loss function that works in regression tasks is mean squared error (MSE). MSE loss function E bis defined as follows,

$$E(\boldsymbol{w}) = \sum_{j=1}^{J} \left\| \boldsymbol{t}_j - \boldsymbol{y}_j \right\|^2, \tag{2-6}$$

where $\boldsymbol{y}_j$ is the predicted value vector, $\boldsymbol{t}_j$ is the desired response vector, J is the number of samples trained in one batch (batch size), and $\boldsymbol{w}$ represents the weight values of the network. Equation (2-6) can also be divided by J to obtain the mean value. The learning process is conducted by minimizing the loss function by updating the weight values sequentially. The weight adjustment is conducted by error back-propagation technique [9] with gradient descent method [10].

The design database is first divided into two sets: training set and validation set. The former is used to train the model, while the latter is used to validate the model. This method is called cross-validation. In this study, the training and the validation set are randomly chosen from the design database with ratio 4:1. Again, the term ‘training’ refers to the process of sequentially updating the weight values of the network so that the model can match the training data. The training is done in two phases: feedforward and back-propagation. In the feedforward phase, the training set is passed into the network. The network with initial random weight values

predicts the output. In the backpropagation, gradient descent is used to calculate the slope of the function and uses this value to update weight values using the Widrow-Hoff rule as follows,

$$w_{k,l} := w_{k,l} - \eta \frac{\partial E(\boldsymbol{w})}{\partial w_{k,l}}, \tag{2-7}$$

where $w_{k,l}$ represents the weight value that connects neuron k in a layer l in the next layer, η is the learning rate, and $E(\boldsymbol{w})$ is the cost function. Adam [11], a gradient based optimizer, is used to do the gradient descent task, which is to find a set of weight values $\boldsymbol{w}$ that minimizes the cost function $E(\boldsymbol{w})$. Specifically, a variant of gradient descent called mini batch gradient descent is adopted, in which only a portion of the training set is used at a time to calculate the cost function. This portion is called a batch size, which is set to 5% of the training set. The cost function calculation is done until all the training set has been used. The average cost function is the obtained, and the weight values are updated.

The validation process only includes the feedforward phase, where the validation set is passed into the network to calculate the cost function. As the training progresses, the cost function with respect to the training set decreases, while the cost function with respect to the validation set eventually increases. This is the sign of overfitting, as depicted in Figure 2.3. The training is stopped whenever this sign is observed (early stopping). If not observed, the training progresses, indicating an epoch. The maximum epoch is set to 2000.

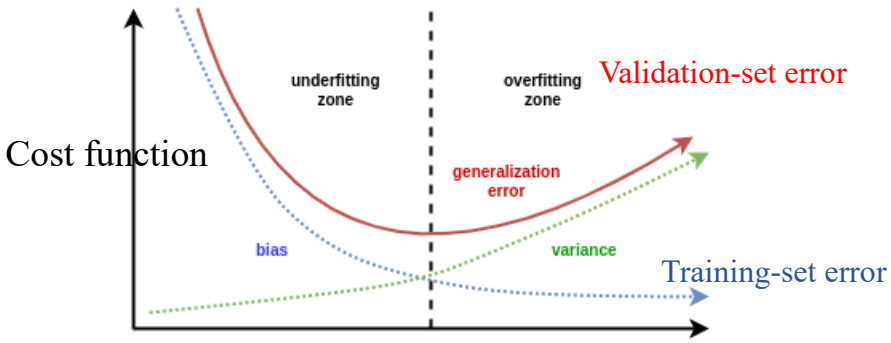


Figure 2.3 Early stopping is done whenever the overfitting case is observed.

2.4 Data treatments

2.4.1 Standardization

Prior to every training, data treatments should be done. The first step is called standardization. The design variables $\boldsymbol{x}$ and the functions $\boldsymbol{f}$ (objective and constraint functions) to approximate are standardized between $0-1$. To do this, $\boldsymbol{x}^{(L)}$ and $\boldsymbol{x}^{(U)}$ are used as the lower and upper boundaries for $\boldsymbol{x}$, while min($\boldsymbol{f}$) and max($\boldsymbol{f}$) are the lower and upper boundaries for $\boldsymbol{f}$. This process is reported to speed up the training [12].

2.4.2 Duplicates removal

The next is step is duplicates removal. Euclidean distances between samples in the design variable domain are calculated. As the name suggests, if there are two samples that have a distance < 0.001, one of them is removed. In other words, samples that are closely similar with each other in terms of design variable (similar shapes/dimensions). The distance limitation is user-defined and might depend on the optimization problem being solved. This step prevents the training put more weights and bias towards the crowded samples. This can also slow down the early stopping process by delaying the overfitting.

2.4.3 Oversampling

This step is initiated by using *K*-means algorithm to cluster the samples into *K* clusters, in the $\boldsymbol{x}$ domain. The value of *K* is chosen using a method called gap statistics [13]. The samples are duplicated so that the number of samples allocated to each cluster is evenly distributed. In other words, sample points in a less dense cluster region are oversampled. Suppose *N* is the largest number of training data that belong to a cluster and n_l is the number of training data that belong to *l*-cluster. The sample points that belong to the *l*-cluster are duplicated $round(N/n_l)$ times. This step is thus called oversampling, which is done to delay overfitting as well.

Chapter 3 Surrogate-based optimization

3.1 Introduction

This chapter covers the basics of the core optimizer: genetic algorithm (GA). GA is one type of evolutionary algorithm that relies in the evolutionary principles. Their basic operators drive the evolutionary search. In the search process, numerous sample evaluations must typically be done. In the optimization where the outcome of interest cannot be easily measured (expensive evaluations), it is then intractable, if not impossible to use GA. One way to alleviate this is to use surrogate models to assist the evaluation in the GA search process. Hence, this method is called surrogate-based optimization. The procedures of how to do the surrogate-based optimization and how to apply it to a real-world design problem is explained in this chapter.

3.2 Genetic algorithm: NSGA-II

3.2.1 Definition

Evolutionary algorithms (EAs) mimic natural evolutionary principles to perform the search and optimization procedures. The biological evolution and natural selection emphasize good solutions to survive through evolutionary process, as illustrated in Figure 3.1. Human is the living example of a species that has undergone evolution. One unique feature of evolutionary optimization using EAs is to find and maintain multiple optimal solutions in one single optimization run, which is very promising to be used in multi-objective optimization problems. This is because the search is performed in a population-based approach, where a population of solutions are evaluated per iteration. One of the EAs is called genetic algorithm (GA), which was first conceived by John Holland of the University of Michigan, Ann Arbor. The development of GA has grown over decades that leads to the birth of NSGA-II (non-dominated sorting GA – II) by Deb et al. [3]. NSGA-II specializes in solving multi-objective optimization problems, although it can also be used to solve single-objective optimization problems. This algorithm has drawn much attention due to its simplicity to be applied in wide variety of fields and the codes accessibility. This study uses NSGA-II as the core optimizer.

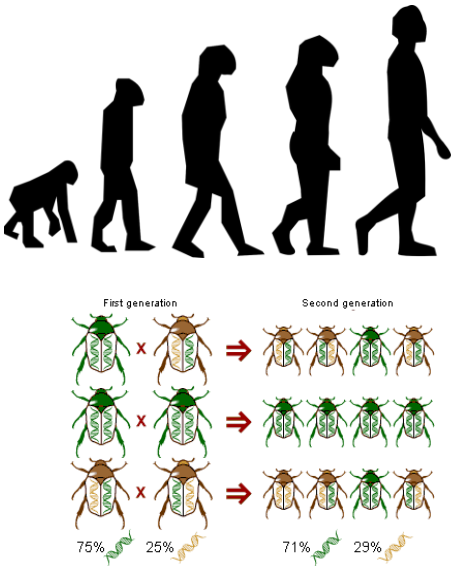


Figure 3.1 Illustrations for biological evolution and natural selection process [14].

3.2.2 Basic operators

Any GA, including NSGA-II, basically relies on some basic operators to drive the evolutionary search. The bio-inspired operators include ***selection***, ***crossover***, and ***mutation***. The selection operator (also called reproduction operator) is subject to following tasks: identify good solutions in a population, make multiple copies of good solutions and eliminate bad solutions. This marks as the feature of exploitation of the existing solutions. The crossover operator and the mutation operator are responsible for creating new (and hopefully better) solutions. The former relies on the principle of marriage between fit parents, in hoping to reproduce good offspring. The latter alters the genetic features of the solutions to find new solutions. The last two operators mark the feature of exploration. The most important thing to be kept in mind when solving an optimization problem is to keep the balance between exploitation and exploration, since we want the optimal solutions to have good proximity and diversity towards the global optimums. Further explanation and mathematical backgrounds of these operators are out of scope of this study and can be found in a good book by Kalyanmoy Deb [15].

The abstract flow is depicted in Figure 3.2. The population is initialized by doing initial sampling. The first population is then evaluated. The next step is survival where the fitness is assigned to each solution to rank them up. Based on the ranks, the selection process takes place, followed by the crossover and mutation process. This genetic procedure produces new solutions to be evaluated again, marking one optimization iteration. This iteration is repeated iteratively.

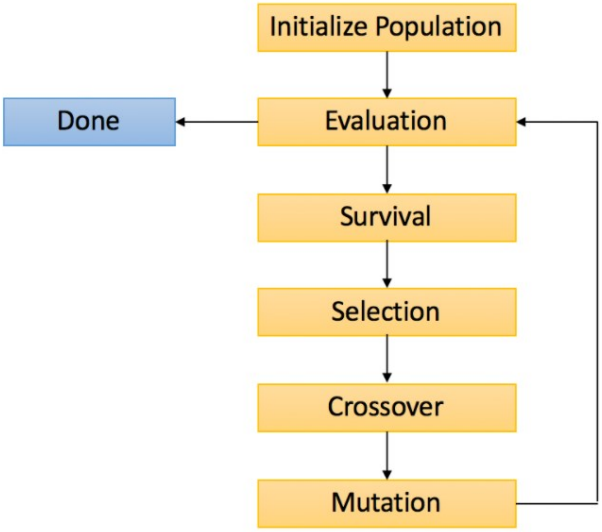


Figure 3.2 General flow of a genetic algorithm [16].

3.3 Combining neural network with NSGA-II (NN+GA)

3.3.1 The driving idea

To perform the search in an optimization problem, one must evaluate the solutions. The term 'evaluation' refers to analytical function, numerical simulation (e.g., CFD) or experiment to assign fitness or score to the given solution. In the test problems, analytical functions are known and can be evaluated cheaply. However, in expensive problems, rigorous numerical simulations must be carried out to evaluate the solution. Since NSGA-II is a population-based approach, it is expected to have many solutions to be evaluated. This is not an obstacle when the problems only need analytical functions for evaluation. But this is clearly a disadvantage when the problems include expensive evaluations, leading to longer computational time. To reduce the time for these expensive evaluations, one might consider a surrogate model. In this study, a neural network (NN), specifically a multilayer perceptron offers a fast evaluation of new unknown solutions based on the model trained using the existing solutions. This leads to the idea of combining the NN with NSGA-II (we call it NN+GA). This idea is natural and common-sense that has been around for decades. But again, as to my knowledge, few studies have considered NN to be integrated with a GA like NSGA-II.

3.3.2 General procedure of NN+GA

The general procedure of NN+GA can be applied to any optimization problem, including single or multiple objectives. Since the NN model is a data-driven model, initial sampling must be done to get the idea of the objective space. A Latin hypercube sampling (LHS) [17] is used to perform the task in the decision space. LHS points are then evaluated using true evaluations. An initial NN model is constructed using the initial dataset. After that, optimization is carried out using the NSGA-II algorithm with every evaluation being substituted by the NN model. Some optimized solutions are found at the end of the optimization by NSGA-II. The optimized solution(s) on the NN model is(are) then evaluated using true evaluation. We call this(these) solution(s) as the infilling point(s). It is noteworthy that only one infilling point per iteration is evaluated in single-objective problems, while in multi-objective problems, infilling criteria are used to decide the next infilling points to be evaluated. Lastly, the infilling point(s) is(are) compiled in a new dataset that will be used to retrain the NN model. This process is repeated until several iterations, defined by the user considering the budget limitation.

3.3.3 NN+GA procedure applied to design optimization

Things are getting more technical since our focus now is to apply the NN+GA procedure to design optimization of an aerospace system. In this study, a shape optimization of transonic airfoil is introduced as the example of real-world application and illustrated in Figure 3.3. The procedure is summarized in the following steps.

1. Firstly, the geometry parameterization is done. The geometry is characterized by a set of design variables that determine the shape of a design.
2. Latin hypercube sampling (LHS) is done as the initial sampling in the design space. N is chosen as the number of initial LHS points, defined by the user. If the geometry parameterization has any expensive constraints to be considered, the LHS is combined with a constraint handling technique, so that the obtained initial design variables $\boldsymbol{x}_{initial}$ are geometrically feasible. Note that the constraint handling technique affects the obtained initial samples, make sure that only important constraints are considered.
3. The objective $\boldsymbol{f}$ and constraints $\boldsymbol{g}$ are evaluated using true evaluations, in this case, CFD is done. Other analytical functions included in the evaluation are also conducted.
4. The so-far obtained solutions are compiled in a database containing $\boldsymbol{x}$ and expensive-to-evaluate functions from $\boldsymbol{f}$ and $\boldsymbol{g}$ to approximate.

5. The best N solutions are reported as the current generation by using survival selection of NSGA-II (considering the non-dominated sorting and the crowding distance sorting).
6. The NN based surrogate model is constructed by training using design database. An approximate model $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{g}}$ that can do analytical evaluation is obtained.
7. Multi-objective optimization is carried out using NSGA-II algorithm, coded in Python [18]. Any call to expensive evaluations of $\boldsymbol{f}$ and $\boldsymbol{g}$ is replaced with the model $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{g}}$. At the end, a set of non-dominated solution $\boldsymbol{x_{best}}$ is obtained which corresponds to $\hat{\boldsymbol{f}}_{\boldsymbol{best}}$.
8. K-means algorithm [19] for clustering data is used as the infilling criterion to down-select K points from $\boldsymbol{x_{best}}$ to be evaluated as the infilling points. They are selected from the closest points from the centroids. The clustering is done in the objective space.
9. Steps 3-8 are repeated until the computational budget is exhausted, or the stopping criteria are satisfied. The criteria are defined by the users.
10. A set of solutions $\boldsymbol{x_{last\_gen}}$ which correspond to the best N solutions from the design database are obtained by survival selection of NSGA-II. They are then sorted based on their objective and constraint functions (non-dominated sorting) to find solutions that lie on the first non-dominated front to approximate the Pareto-optimal front (POF) (the front where the global optimums are located).

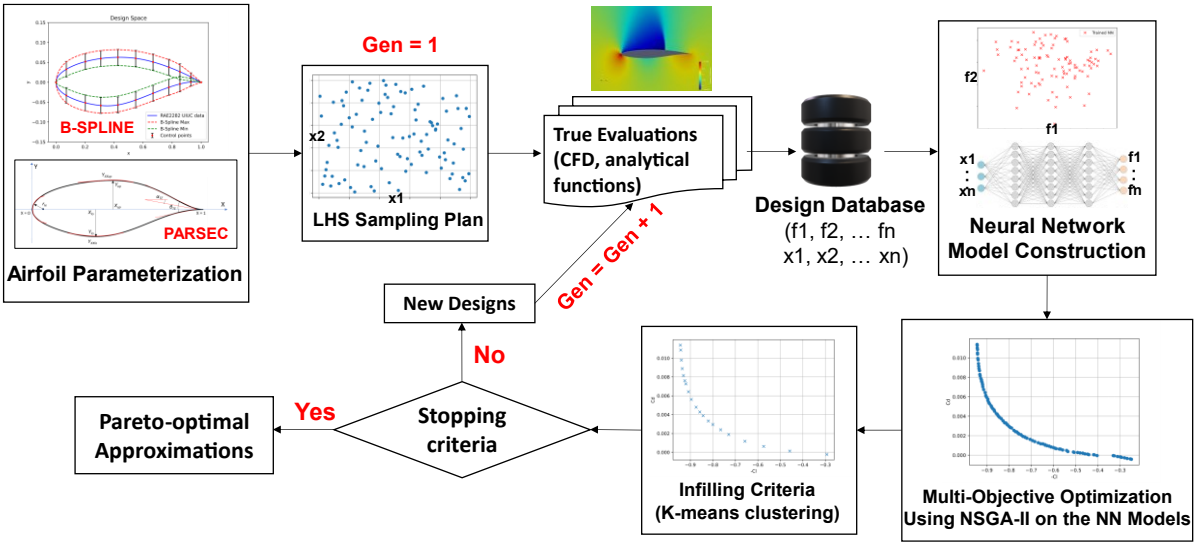


Figure 3.3 NN+GA procedure applied to airfoil shape optimization.

This procedure is used throughout this study when solving the airfoil optimization problems. The NN surrogate model can also be replaced with other techniques (e.g., Kriging). When comparing NN and Kriging (Chapter 6), the same framework/flow is used, but with different models (done in the 5th step) to be optimized by the NSGA-II, as shown below.

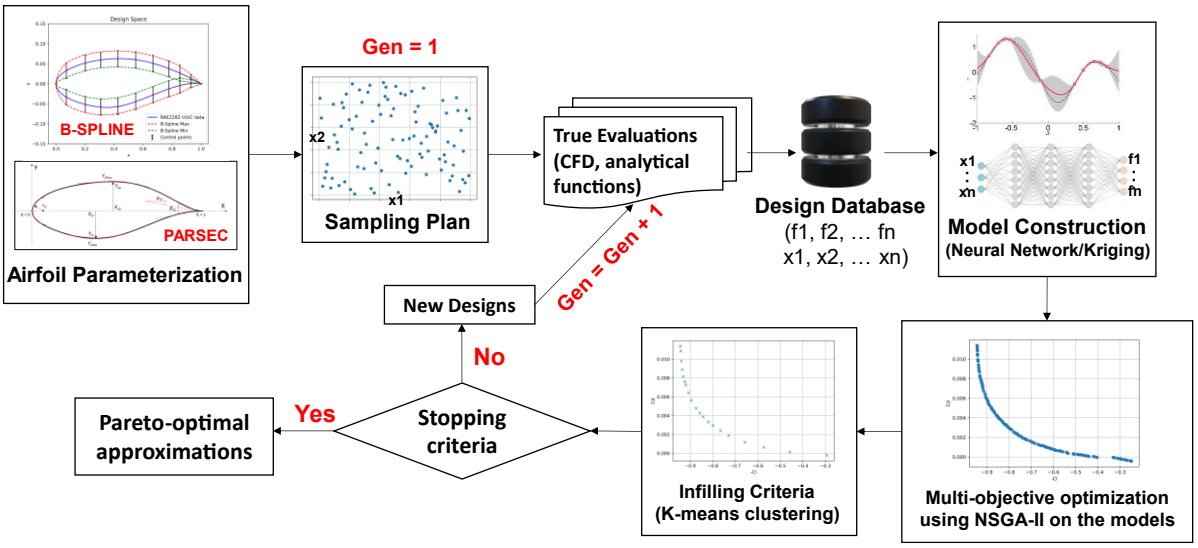


Figure 3.4 Flow applied to airfoil shape optimization with different models.

Chapter 4 Application to test problems

4.1 Introduction

After setting the optimization procedure, the next task is to apply the method to some optimization problems. This chapter covers the application of the proposed method (NN+GA) to some popular test problems where the evaluations are cheap and analytical. The main objective is to investigate whether integrating ANN-based surrogate model to a GA can cut the required number of evaluations. If the result is positive, meaning that the required number of evaluations is indeed reduced, the method is ready to be used in a real-world problem.

4.2 Description of the test problems

Prior to applying the proposed method (NN+GA) to real-world optimization problems, the investigation should be first done on the test problems. The test problems are well-known optimization problems that only include analytical functions. Their global optimums are known, so that it is easier to judge the performance of the algorithm. The test problems are divided into two types: single- and multi- objective problems. The main difference is that in single-objective problems, only one infilling point is found per iteration, while in multi-objective problems, a set of non-dominated solutions is expected in every iteration. Table 4.1 summarizes all the test problems used to verify the method. The efficacy of incorporating NN into NSGA-II is the main objective here, to see whether it is more efficient than a standard NSGA-II without surrogates. The detailed explanation is not covered here since it can be found anywhere else (e.g., [20]).

Table 4.1 Test problems with various complexities

Name	Type	Feature(s)	Objective(s)	Constraint(s)	Variables
Ackley	Single-objective	Flat w/ deep hole	1	0	2
Griewank	Single-objective	Widespread locals	1	0	2
Pressure Vessel	Single-objective	Mixed-integer	1	4	4
ZDT1	Multi-objective	Convex Pareto	2	0	30
ZDT2	Multi-objective	Non-convex Pareto	2	0	30
ZDT3	Multi-objective	Disconnected Pareto	2	0	30
OSY	Multi-objective	Many constraints	2	6	6

4.3 Methods

4.3.1 Experimental setup

Due to the GA stochastic nature, 10 optimization runs with 10 different initial samples have been conducted for the test problems. For every optimization, 100 initial samples are used to construct the NN model. The infilling point for single-objective test problems is one every iteration (the best in terms of objective function predicted by the NN model), while for multi-objective test problems, 100 optimized solutions found by NSGA-II (with 100 population size) on the NN model are directly treated as the new infilling points. The budget for NN+GA in one optimization run is 400 and 2000 true evaluations to solve single- and multi-objective test problems, respectively. The standard NSGA-II with no surrogates is also used with the budget of 5000 and 150000 true evaluations per optimization run to solve single- and multi-objective test problems, respectively, except for Griewank that only needs 2500. The NSGA-II settings for the standard NSGA-II without surrogates are given in Table 4.2. The NSGA-II in NN+GA is the same as given in Table 4.2, but with 250 generations.

Table 4.2 The standard NSGA-II settings

	Test functions		**Real-world problem**
	Single-objective	**Multi-objective**	
Pop size	100	100	100
Crossover	$\eta_c = 15, rate = 0.9$	$\eta_c = 15, rate = 0.9$	$\eta_c = 15, rate = 0.9$
Mutation	$\eta_m = 15, rate = 0.01$	$\eta_m = 20, rate = 0.01$	$\eta_m = 20, rate = 0.01$
No. of generations	50, for Griewank: 25	150	10

4.3.2 Performance indicator

To judge the performance of NN+GA and NSGA-II, the objective function and a hypervolume (HV) value is used as the indicator for the single- and multi-objective test problems, respectively. In single-objective case, it is straightforward to use the objective value since there is only one objective. The obtained solutions with lower value are deemed better in the minimization problem. In multi-objective case, given the reference points, HV calculates the hatched area, shown in Figure 4.1. In the case where both objectives are minimized, the higher the value of HV, the better the obtained solutions are. This is because the direction that we want for our solutions to evolve is to the lower left direction.

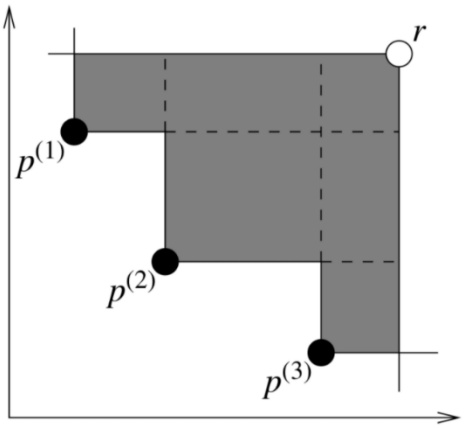


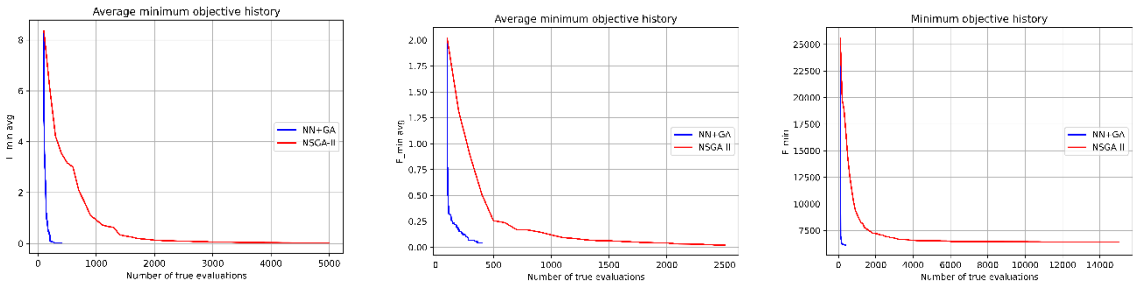
Figure 4.1 HV calculates the hatched area.

When all objectives are minimized, we surely want the larger hatched area. In other words, the higher the value of HV, the better. The performance of the algorithm can be investigated by plotting the indicator value per iteration, and we can see how it evolves over the iterations. An algorithm that can achieve better indicator value with lower number of true evaluations is preferable, showing its efficiency in solving the problem.

4.4 Optimization results

4.4.1 Results for the single-objective test problems

The average objective values (over 10 runs) of new solutions per iteration is plotted against the number of true evaluations. In the case of NN+GA, one new solution is evaluated per iteration. As for NSGA-II without surrogates, 100 new solutions must be evaluated. Figure 4.2 shows the performance comparison for the single-objective test problems. The indicator is the minimum objective found every iteration. From Figure 4.2, NN+GA can find optimal solutions with fewer true evaluation calls, compared to NSGA-II without surrogates.

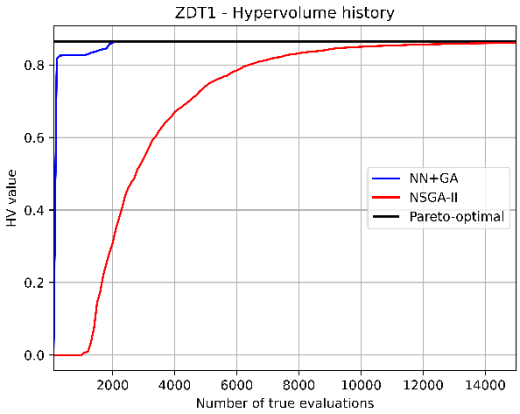


(a) Ackley (b) Griewank (c) Pressure vessel

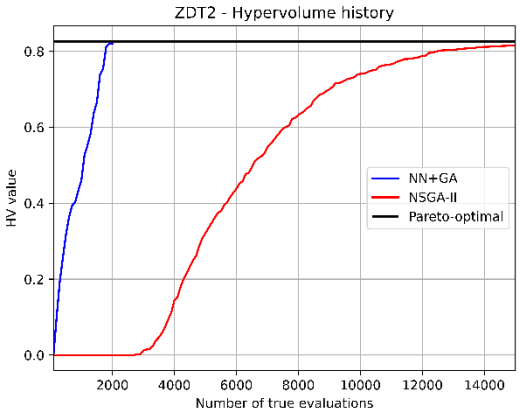
Figure 4.2 Performance measure histories for single-objective test problems

4.4.2 Results for the multi-objective test problems

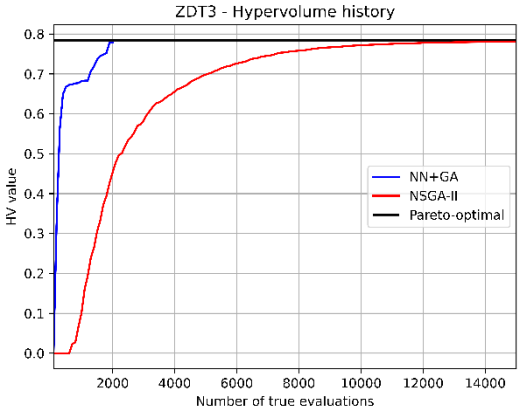
The HV values are averaged over 10 optimization runs and plotted against the number of true evaluations, shown in Figure 4.3. The HV value basically measures the proximity and diversity of the non-dominated solutions. So, the higher the value, the better the solutions are.



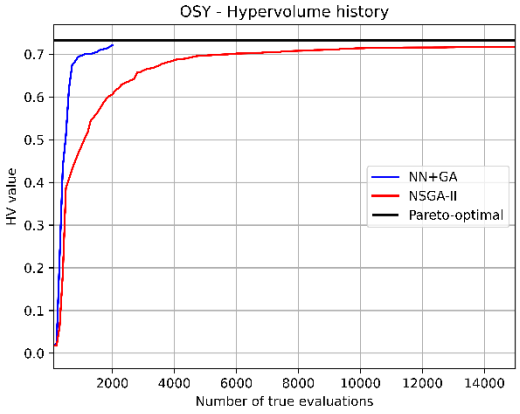
(a) ZDT1



(b) ZDT2



(c) ZDT3



(c) OSY

Figure 4.3 Performance measure histories for multi-objective test problems

Again, NN+GA is superior to NSGA-II without surrogates, in a way that the former can find optimal solutions with HV values close to the HV values of the Pareto-optimal (the true optimal) solutions but spend much fewer true evaluations in every test multi-objective test problem, as observed from Figure 4.3.

4.5 Summary

In this chapter, NN+GA is applied to several test problems with various complexities. Undoubtedly, the use NN surrogate model in NSGA-II can reduce the number of true evaluations in every test problem. This might not seem advantageous in the case of test problems, since the optimization time takes longer when using NN+GA due to the time needed for training the NN models. However, this can be a great advantage when NN+GA is applied to problems with expensive evaluations, especially when the model training time is much shorter than the true evaluation time.

The reason behind the better performance of NN+GA compared to NSGA-II includes the fact that we use cheap approximation models when performing the evolutionary search. It thus allows us to set a high number of generations (250, in this case) without concerning the evaluation time. However, this finding should be further investigated by applying it to real-world problems with expensive evaluations.

Chapter 5 Real-world application

5.1 Introduction

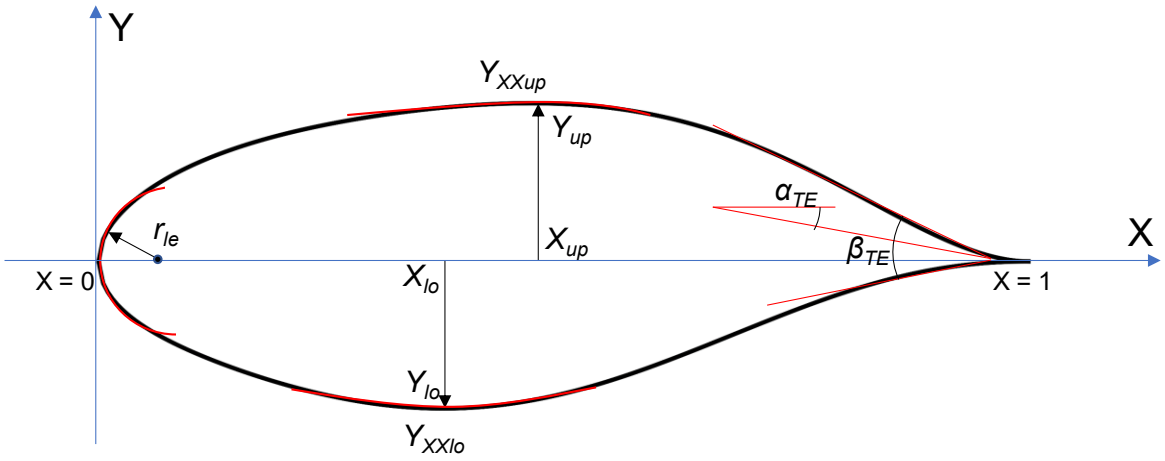
Aerodynamic design of transonic wing is important since most of commercial aircrafts today cruise at transonic speeds, near the speed of sound. The aerodynamic characteristics of the wing are strongly affected by the shape of airfoil section. Aerodynamic shape optimization of transonic airfoils (ASO-TA) thus becomes a crucial task to find candidates of shapes with optimum aerodynamic performance, given the set of design variables, objective, and constraint functions. One of the objectives is drag minimization to reduce fuel consumption at cruise. However, it comes with a tradeoff with lift, for example. The drag component called induced drag increases in proportion to the square of lift. Zero induced drag means zero lift.

In line with our objective of applying NN+GA to a real-world problem, ASO-TA seems to be a good start. Therefore, in this chapter, we apply NN+GA and NSGA-II without surrogates to solve ASO-TA problems. The evaluation includes computational fluid dynamics as the true evaluation. Unlike in the test problems in the previous chapter in which one evaluation virtually takes no time (<0.001 s), the evaluation now takes 1-3 minutes. So, if 1000 designs are to be evaluated, 1000-3000 minutes (or 16-50 hours) must be allocated.

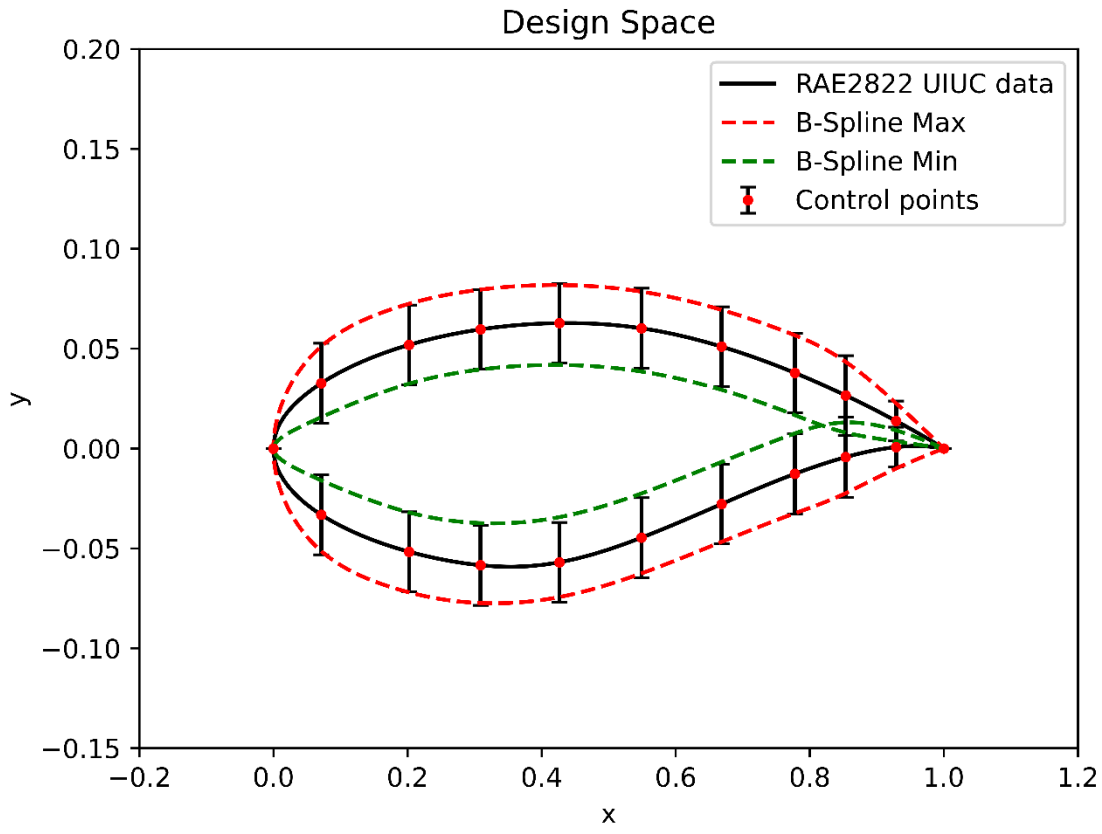
5.2 Description of the problems

5.2.1 Airfoil parameterization

Two techniques are used to parameterize the airfoil: PARSEC [21] and B-Spline, illustrated in Figure 5.1. All airfoils have a sharp trailing edge, for simplicity. The 9 design variables for the PARSEC airfoil are listed in Table 5.1. As for the B-Spline airfoil, we start by selecting 18 coordinates of the baseline airfoil (RAE2822). The selection is done by roughly distributing them evenly on the airfoil surface (purely up to the designers). Note that larger number of control points give more flexibility, but bumpy airfoils that can lead to bad-performing designs. The 18 coordinates are treated as the control points of the B-Spline of order 3. We then introduce some perturbations to the 18 control points in the y-axis (with fixed x-axis) to create new airfoils. The upper and lower limit of y-axis are listed in Table 5.2. The B-Spline curves are generated using NURBS-Python [22].



(a) PARSEC parameterization



(b) B-Spline parameterization

Figure 5.1 Two techniques for parameterizing the airfoils: PARSEC and B-Spline.

Table 5.1 PARSEC variables and their boundaries.

No	Variables	Lower Bound	Upper Bound
1	r_{LE}	0.0065	0.0092
2	X_{up}	0.3466	0.5198
3	Y_{up}	0.0503	0.0755
4	Y_{XXup}	–0.5094	–0.3396
5	X_{lo}	0.2894	0.4342
6	Y_{lo}	–0.0707	–0.0471
7	Y_{XXlo}	0.5655	0.8483
8	α_{TE}	–0.1351	–0.0901
9	β_{TE}	0.1317	0.1975

Table 5.2 B-Spline control points and their boundaries.

No	X	Vars	Lower Bound	Upper Bound
1.	0.928864	Y_1	-0.009306	0.010694
2.	0.853553	Y_2	-0.024314	0.015686
3.	0.777785	Y_3	-0.032689	0.007310
4.	0.668445	Y_4	-0.048139	-0.007814
5.	0.549009	Y_5	-0.064642	-0.024642
6.	0.426635	Y_6	-0.076979	-0.036979
7.	0.308658	Y_7	-0.078459	-0.038459
8.	0.202150	Y_8	-0.071694	-0.031694
9.	0.071136	Y_9	-0.053169	-0.013169
10.	0.071136	Y_{10}	0.012644	0.052644
11.	0.202150	Y_{11}	0.031885	0.071885
12.	0.308658	Y_{12}	0.039629	0.079629
13.	0.426635	Y_{13}	0.042779	0.082779
14.	0.549009	Y_{14}	0.040194	0.080194
15.	0.668445	Y_{15}	0.030993	0.070993
16.	0.777785	Y_{16}	0.017847	0.057847
17.	0.853553	Y_{17}	0.065540	0.046554
18.	0.928864	Y_{18}	0.037689	0.023769

5.2.2 Computational fluid dynamics

CFD is used to evaluate the true aerodynamic performance (i.e., drag coefficient C_d and lift coefficient C_l). The condition includes a 2D inviscid flow, solved by SU2 open-source code [23]. The inviscid Euler solver is not realistic to simulate real-world aerodynamics with viscosity and thermal conductivity. Nevertheless, it is cheap enough for the current machine being used: Intel(R) Xeon(R) CPU E5-1630 v4 3.70 GHz with 4 cores. It thus allows us to

perform numerous CFD evaluations in the present numerical experiments, since our interests are more on the comparison between optimization algorithms, rather than the results of the optimization. Moreover, using the Euler solver is complex enough for the surrogate models and is still a representative of a real-world optimization problem.

The inviscid and compressible Euler equations can be obtained as a simplification of the compressible Navier-Stokes equations in the absence of viscosity and thermal conductivity, expressed in differential form as:

$$\frac{\partial U}{\partial t} + \nabla . \bar{F}^c(U) - S = 0, \tag{5-1}$$

where the conservative variables U are given by:

$$U = \{\rho, \rho\bar{v}, \rho E\}^{\mathrm{T}}. \tag{5-2}$$

S is a generic source term, while the convective flux $\bar{F}^c$ is given by:

$$\bar{F}^c = \begin{Bmatrix} \rho\bar{v} \\ \rho\bar{v} \otimes \bar{v} + \bar{\bar{I}}p \\ \rho E\bar{v} + p\bar{v} \end{Bmatrix}, \tag{5-3}$$

where ρ is the fluid density, $\bar{v} = \{u, v, w\}^T \in \mathbb{R}^3$ is the flow speed in Cartesian system of reference, E is the total energy per unit mass, p is the static pressure, and T is the temperature. Assuming a perfect gas with a ratio of specific heats γ and gas constant R, one can close the system by determining pressure p from:

$$p = (\gamma - 1)\rho[E - 0.5(\bar{v} \cdot \bar{v})], \tag{5-4}$$

and the temperature T from the ideal gas equation:

$$T = \frac{p}{\rho R}. \tag{5-5}$$

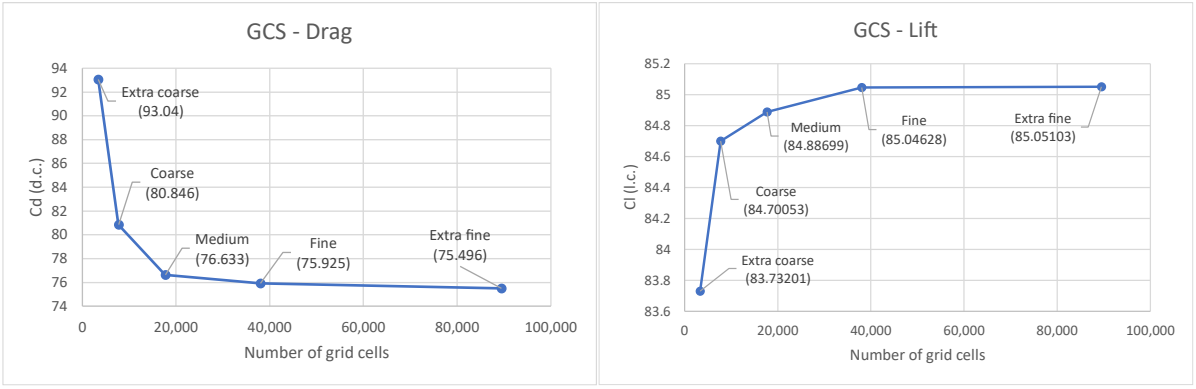
The CFD solver uses Jameson-Schmidt-Turkel (JST) [24] for the convective flux scheme and Euler implicit scheme for the time discretization method.

5.2.3 Grid configuration

The CFD results are dependent on the grid resolution. Thus, we should conduct a grid convergence study (GCS) to decide our grid configuration so that our aerodynamic values of interest are grid independent. In this, the grid independence is marked by the convergence of C_d and C_l. The GCS is done on the RAE2822 in a 2D inviscid flow condition with angle of attack $\alpha = 2^o$ and Mach number $M = 0.73$. We use a standard C-grid topology and survey five types of grid resolution, with various grid densities as illustrated in the Figure 5.3. The GCS results are shown in Table 5.3 and Figure 5.2.

Table 5.3 The results of grid convergence study

Types	Mesh size	C_d	C_l
Extra coarse	3,344	0.0093040	0.8373201
Coarse	7,714	0.0080846	0.8470053
Medium	17,688	0.0076633	0.8488699
Fine	38,016	0.0075925	0.8504628
Extra fine	89,496	0.0075496	0.8505103



(a) Drag convergence

(b) Lift convergence

Figure 5.2 Grid convergence for drag and lift coefficients

We can observe that C_d and C_l have converged between fine and extra fine grids. The CFD on the fine grid took about 1 minute and 10 seconds while it took about 2 minutes and 47 seconds for the extra fine grid. Based on this result, we decide to use the fine grid, as in Figure 5.4. The airfoil is resolved by 338 computational points. The C-grid has the far-field boundary located 50 chord lengths away from the airfoil surface (65 steps for the half-circle and 130 steps for the downstream far-field), the first layer grid thickness as 0.001 chord length. Hence, the grid has 337 x 65 + 2 (130 x 65) = 38,805 computational cells. All the grids are generated by Pointwise [25], a commercial meshing software.

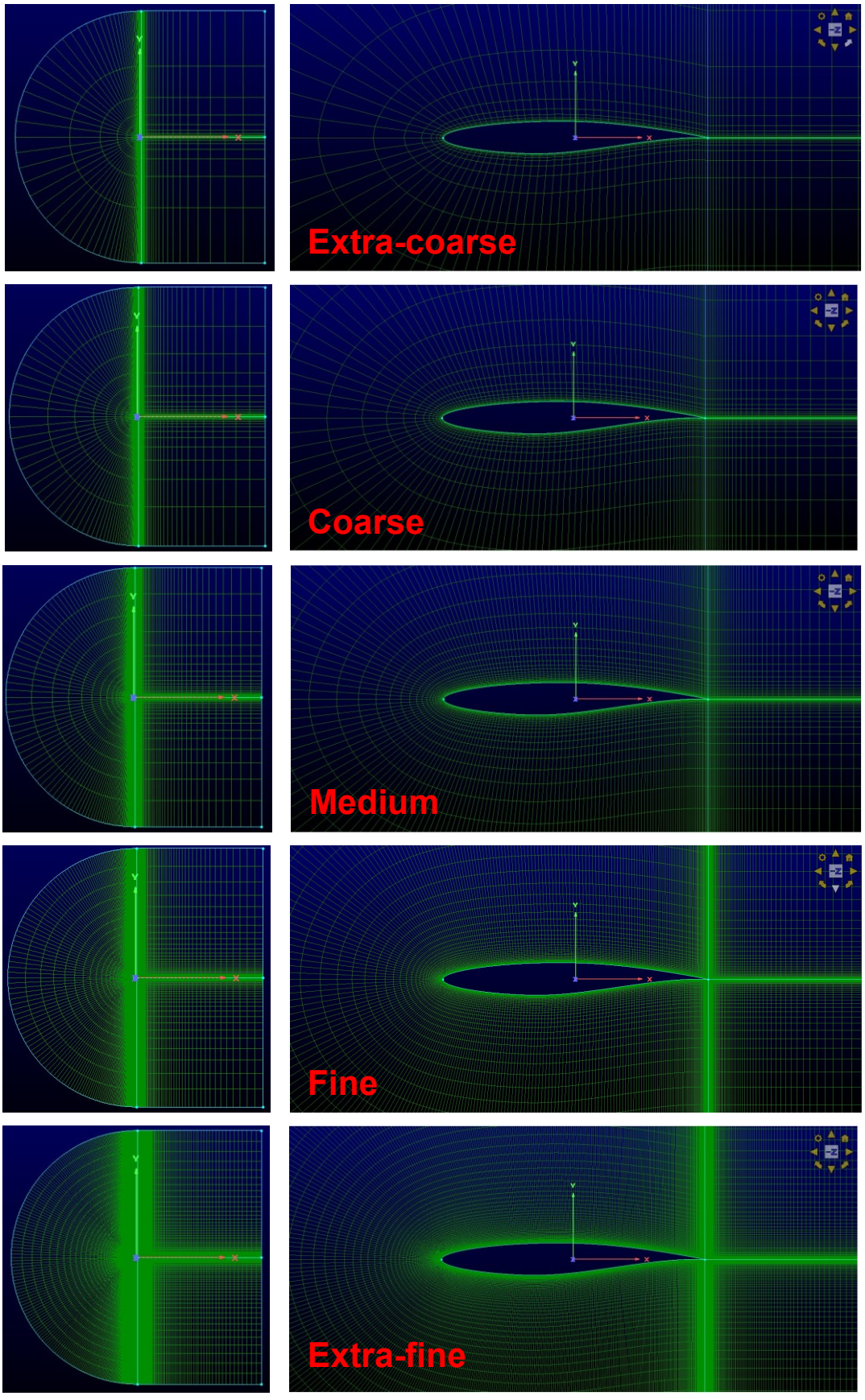
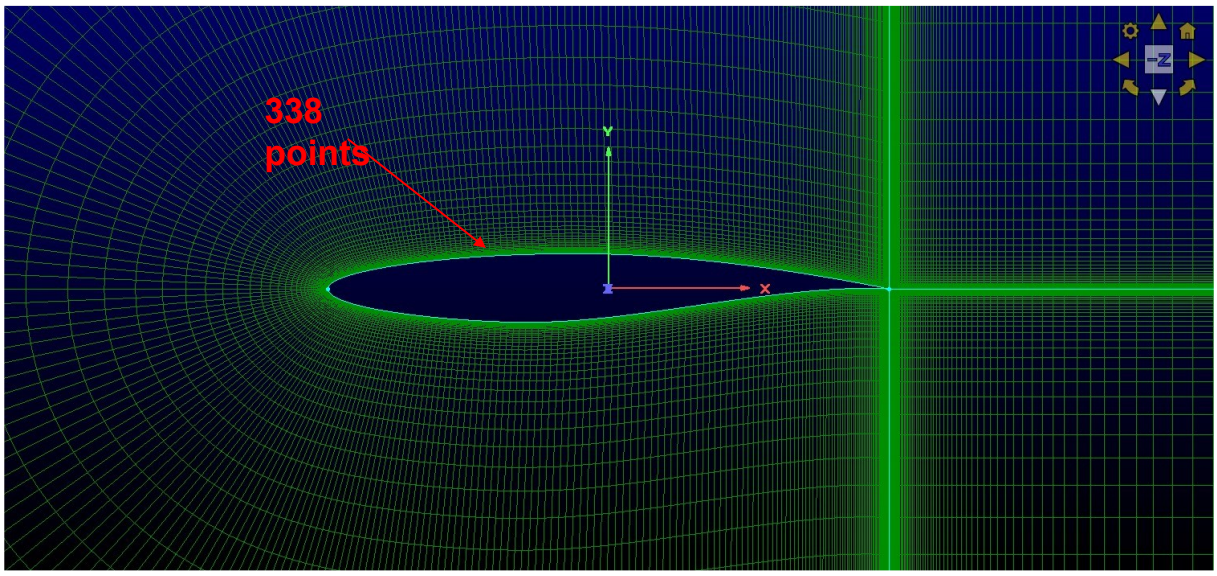
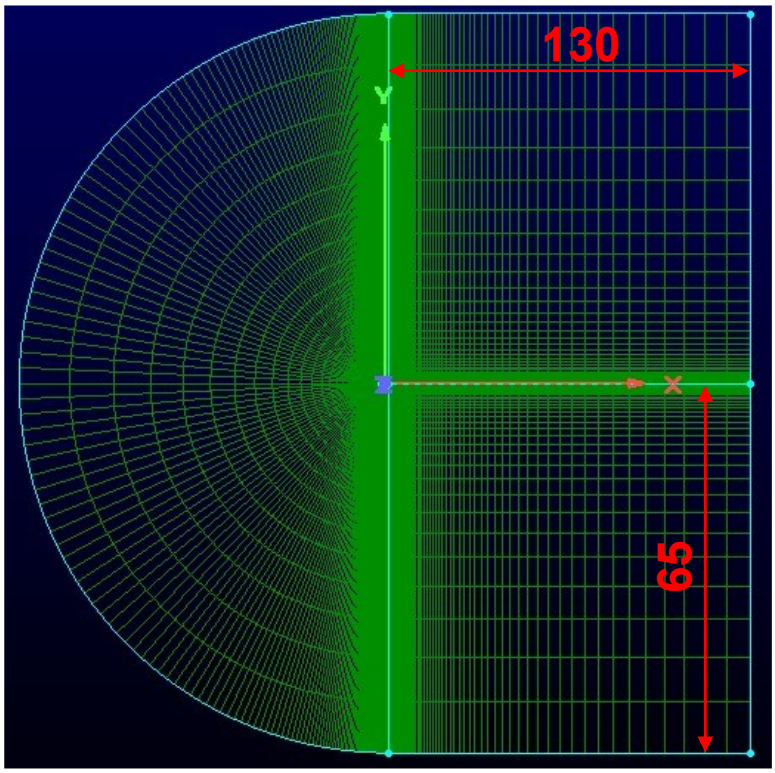


Figure 5.3 The C-grid with various densities from lower (top) to higher (bottom) density



(a) Near-field boundary



(b) Far-field boundary

Figure 5.4 Example of the C-grid configuration on RAE2822

5.2.4 Problem definitions

The complexity of an optimization problem is influenced by the dimensionality, the governing physics, the presence of constraints, etc. The higher the dimensionality, the more complex the problem is. The presence of constraints also adds more complexity to the problem. Therefore, we define the following three multi-objective ASO-TA problems, aiming at presenting different complexities. Note that we transform the maximization of C_l to the minimization of $-C_l$.

ASO-TA1: (2 objectives, 0 constraint, 9 variables)
minimize : C_d and $-C_l$
with respect to : PARSEC variables in Table 5.1
subject to : -
@ $\alpha = 2^o, M = 0.73$

ASO-TA2: (2 objectives, 0 constraint, 9 variables)
minimize : C_d and $-C_l$
with respect to : PARSEC variables in Table 5.1
subject to : -
@ $\alpha = 2^o, M = 0.80$

ASO-TA3: (2 objectives, 3 constraints, 18 variables)
minimize : C_d and $-C_l$
with respect to : B − Spline control points in Table 5.2
subject to : $0.8 * A_{baseline} - A \leq 0$
$Y_1 - Y_{18} \leq 0$
$Y_2 - Y_{17} \leq 0$
@ $\alpha = 2^o, M = 0.73$

The above problems are in an order of increasing complexity, with two expensive objective functions: C_d and C_l. ASO-TA1 and ASO-TA2 have no constraint functions, but the latter has a slightly larger Mach number. This is done to present a more complex problem since the shock wave is expected to be more intense. ASO-TA3 has higher dimensionality with addition of three cheap constraints. The area constraint ($0.8 * A_{baseline} - A \leq 0$) prevents the airfoil from going too slender compared to baseline, and the trailing edge constraints ($Y_1 - Y_{18} \leq 0, Y_2 - Y_{17} \leq 0$) ensure geometrically feasible shapes. Look at the green dashed line in Figure 5.1(b). The area and trailing edge constraints prevent such airfoil from being created.

5.3 Methods

5.3.1 Experimental setup

We use NN+GA and three different configurations of NSGA-II algorithm without surrogates to approach the ASO-TA problems. The same exact initial samples found by LHS are used as the initial population for all algorithms so that we can do a fair comparison. The number of initial samples is 100. In the NN+GA, an initial surrogate model is obtained by training the model based on these 100 initial samples. The training parameters are listed in Table 5.4.

NSGA-II with the configuration listed in Table 5.5 is used with the NN model replacing all the expensive function evaluations (C_d and C_l) to obtain 100 optimized solutions on the MLP model. *K*-means algorithm is used to cluster these samples into $N_{infilling}$ clusters. The samples closest to the centroids are chosen as the next sample points (infilling points). For ASO-TA1 and ASO-TA2, 20 infilling points are added 5 times, while for ASO-TA3, 10 infilling points are added 30 times. Thus, if the optimization is done until n^{th} generation, the number of CFD evaluations using MLP+GA can be written as follows:

$$N_{CFD} = 100 + N_{infilling}(n_{gen} - 1). \tag{5-6}$$

Table 5.4 The NN training parameters.

	ASO-TA1	**ASO-TA2**	**ASO-TA3**
$\boldsymbol{N_{neuron}}$ **(input)**	9	9	18
$\boldsymbol{N_{neuron}}$ **(hidden)**	128	128	2048
$\boldsymbol{N_{neuron}}$ **(output)**	2	2	2
Learning rate	0.001		
$\boldsymbol{N_{epoch}}$	2000		
Train ratio	80% of the current database		
Batch size	5% of the training set		

Table 5.5 Parameter values for NSGA-II inside NN+GA.

Population size	100
Max number of generations	250
Crossover	$\eta_c = 15, rate = 0.9$
Mutation	$\eta_m = 20, rate = 1/100$

In the second algorithm, NSGA-II with no surrogates is used with a population size of 100, producing the next 100 sample points to be evaluated. Due to budget limitation, the NSGA-II$_{100pop}$ is run until 10th generation (1000 CFD evaluations). In the third and fourth algorithm, NSGA-II with fewer population size is used. Now, only a population size of 20 participates in the genetic process, producing the next 20 samples to be evaluated. Since we must start with the same 100 initial samples, the *K*-means algorithm is used to down-select 20 out of 100 initial LHS samples. The *K*-means algorithm is run on the design space $\boldsymbol{x}$ for the third algorithm, and on the objective space $\boldsymbol{f}$ for the fourth algorithm. The fourth algorithm is the least efficient, because the 100 LHS samples must be evaluated first. Since 20 samples are chosen, the remaining 80 samples are redundant. For ASO-TA3 only, the population size is set to 10. Thus, if the optimization is done until n^{th} generation, the number of CFD evaluations using NSGA-II without surrogates can be written as follows:

$$N_{CFD} = 100 + N_{pop}(n_{gen} - 1). \tag{5-7}$$

The third and fourth algorithm are done to lower the budget to make a fair comparison with our proposed method (NN+GA). Since the N_{pop} for the second algorithm is higher than $N_{infilling}$ of the NN+GA method, it might be influenced by this setting. The parameters for the second, third, and fourth algorithm are listed in Table 5.6. Note that P1, P2, and P3 are problem 1, problem 2, and problem 3, respectively.

Table 5.6 Parameter values for the three NSGA-II configurations.

	2nd algo	**3rd algo**	**4th algo**
Pop size (P1, P2)	100	20	20
Pop size (P3)	100	10	10
$\boldsymbol{n_{gen\_max}}$ **(P1, P2)**	10	11	11
$\boldsymbol{n_{gen\_max}}$ **(P3)**	10	31	31
Crossover	$\eta_c = 15$ $rate = 0.9$	$\eta_c = 15$ $rate = 0.9$	$\eta_c = 15$ $rate = 0.9$
Mutation	$\eta_c = 20$ $r = 1/100$	$\eta_c = 20$ $r_{1,2} = 1/20$ $r_3 = 1/10$	$\eta_c = 20$ $r_{1,2} = 1/20$ $r_3 = 1/10$
Initial pop	LHS samples	K-Means on $\boldsymbol{x}$	K-Means on $\boldsymbol{f}$

5.3.2 Performance indicators

The HV indicator is used as the performance metric for each optimization. HV measures the proximity and diversity of the obtained non-dominated solutions (optimized solutions). For NN+GA, the HV of current population is calculated every time the infilling points are evaluated, while for NSGA-II, it is calculated every time the new generation is evaluated. To calculate HV, two reference points, $[0.0, -1.5]$ and $[0.1, 0.0]$, in the objective space are used to normalize both C_d and C_l. Due to the stochastic nature of the algorithms, each optimization problem is solved three times with different initial populations (LHS is done three times). Thus, the HV value is averaged among three optimization runs.

5.4 Results and discussions

5.4.1 Results for ASO-TA1

Figure 5.5 shows the average HV history for all algorithms performance in ASO-TA1. It basically shows how HV value progresses as the number of true evaluations increases. Since HV values represent the proximity towards the POF and its spread, the higher the HV value is, the better. It can be observed from Figure 5.5, the proposed method (NN+GA) can achieve higher HV value with significantly fewer number of CFD evaluations compared to the standard NSGA-II without surrogate model. We defined our budget for ASO-TA1 to be: 3 x 200 = 600

evaluations for NN+GA; 3 x 1000 = 3000 evaluations for the second algorithm; and 3 x 300 = 900 evaluations for the third and fourth algorithm. With only 600 CFD evaluations, the NN+GA achieves an HV value of around 0.675 while NSGA-II (2nd algo) can only achieve 0.650 with 3000 CFD evaluations. We can say that the proposed method is superior to the rest of algorithms in solving ASO-TA1. If one CFD takes around 1 minute and 10 seconds, it means NN+GA cuts the computational time for around 48 hours.

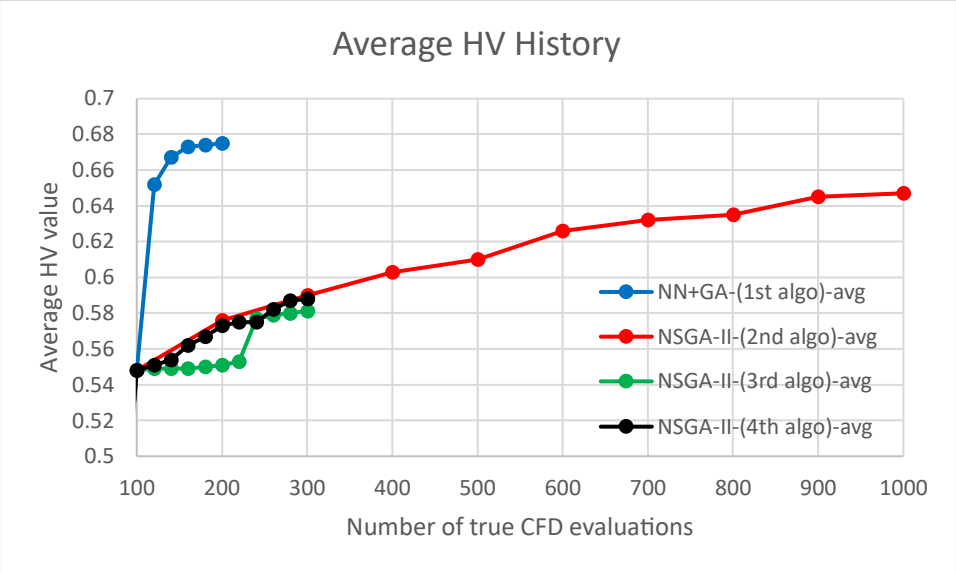
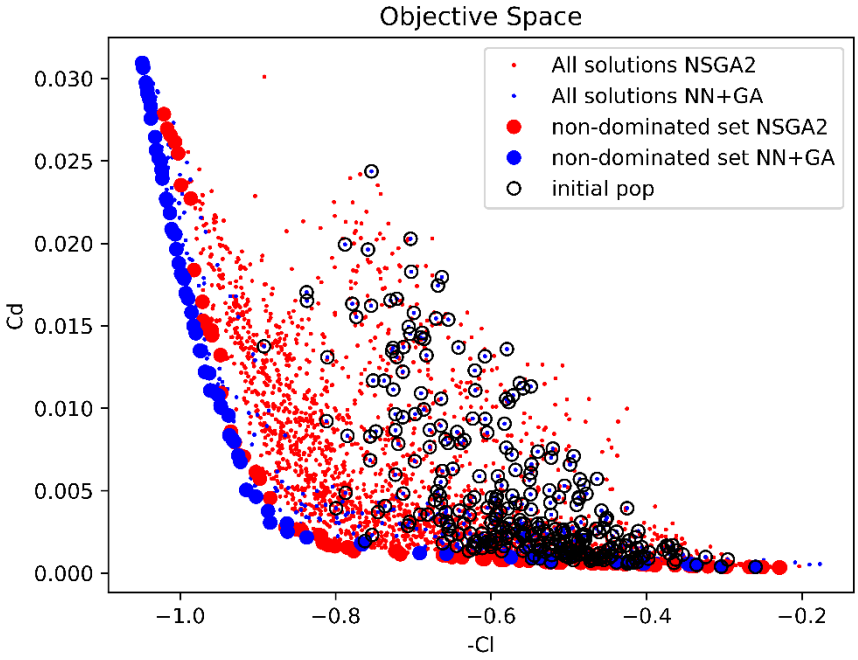


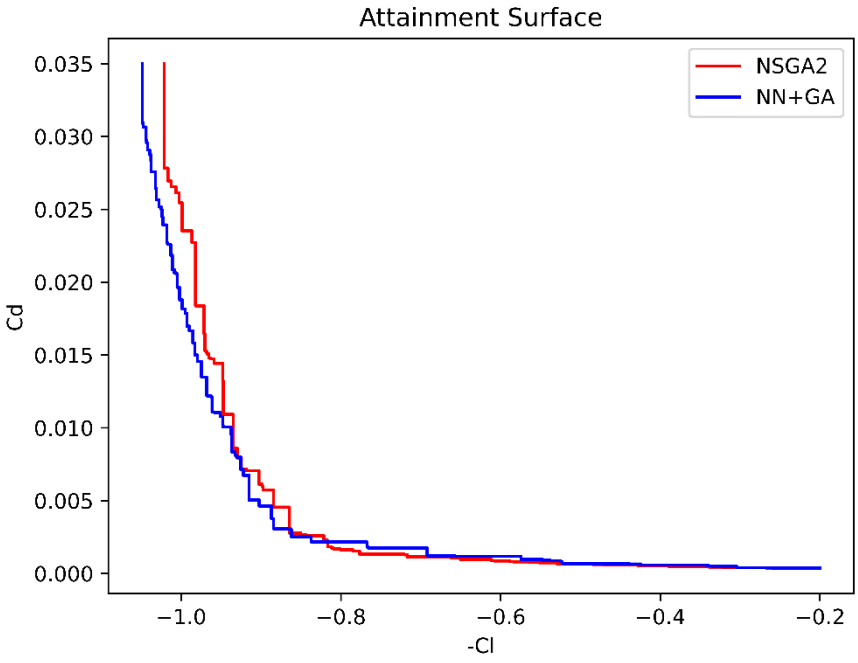
Figure 5.5 Average HV values for ASO-TA1.

The NSGA-II 3rd and 4th algorithms, again, are used to eliminate the doubt that claims the superiority of NN+GA, is because of lower $N_{infilling} = 20$ compared to $N_{pop} = 100$. Even with $N_{pop} = 20$, there is no significant improvement in the 3rd and 4th algorithm, that can make them compete with NN+GA.

These results are visualized in Figure 5.6 that shows the plot of all solutions found by NN+GA and NSGA-II (2nd algo) and the attainment surface of their non-dominated sets. The plot of the initial population and the non-dominated set indicates the complexity of the problem. From Figure 5.6, we can observe that most of the initial population lie in the low C_d region ($C_l = 0.4 - 0.6$). The only task of the optimizer is then to expand the search to cover the high C_l region. The CFD results of the optimized solutions are presented in Figure 5.11-5.13. Note that the low-drag airfoils have less intense shock, shown in the pressure contours, which in turn reduces the wave drag.



(a) All solutions



(b) Attainment surfaces

Figure 5.6 Plots in the true objective space for ASO-TA1.

5.4.2 Results for ASO-TA2

Both ASO-TA1 and ASO-TA2 have all identical conditions, except for the slightly higher Mach number in the latter. This slight difference results in a higher complexity. This is indicated in the plot of the initial population and the non-dominated set (Figure 5.8(a)). The optimizer's task now is to find both extreme regions. A slightly higher Mach number induces a more intense shock wave, as found from a comparison between Figure 5.11 and 5.12. The shock wave induces higher wave drag that in turn increases the drag coefficient C_d. This is the reason why the initial population in ASO-TA2 does not lie in the low C_d region as in ASO-TA1.

Figure 5.7, again, shows the average HV history for all algorithms' performance in solving ASO-TA2. In the same way, it shows the superiority of NN+GA over the standard NSGA-II in solving ASO-TA2. With only 3 x 200 = 600 CFD evaluations, the NN+GA achieves an HV value of around 0.535, while the NSGA-II (2nd algo) can only achieve 0.505 with 3 x 1000 = 3000 CFD evaluations. If one CFD takes around 1 minute and 10 seconds, it means NN+GA cuts the computational time for around 48 hours. The training time is only around 20 seconds. The third and fourth algorithms give a slight improvement for NSGA-II, with the same 3 x 300 = 900 evaluations, but are still inferior to NN+GA.

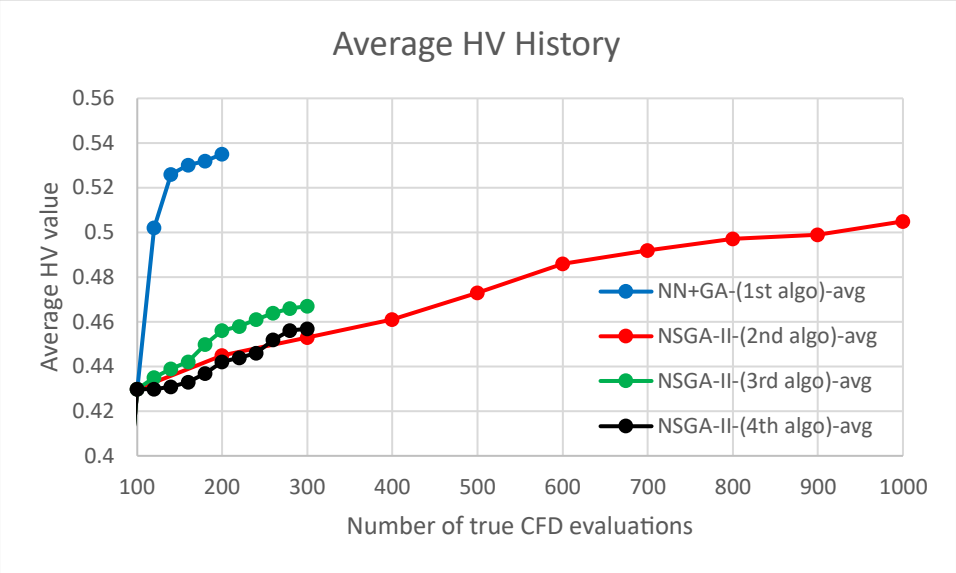
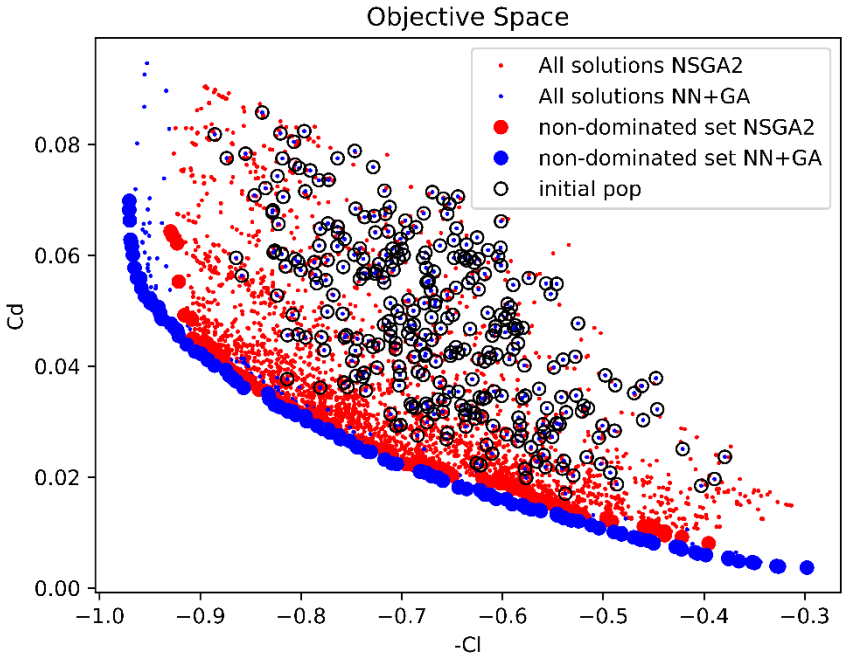
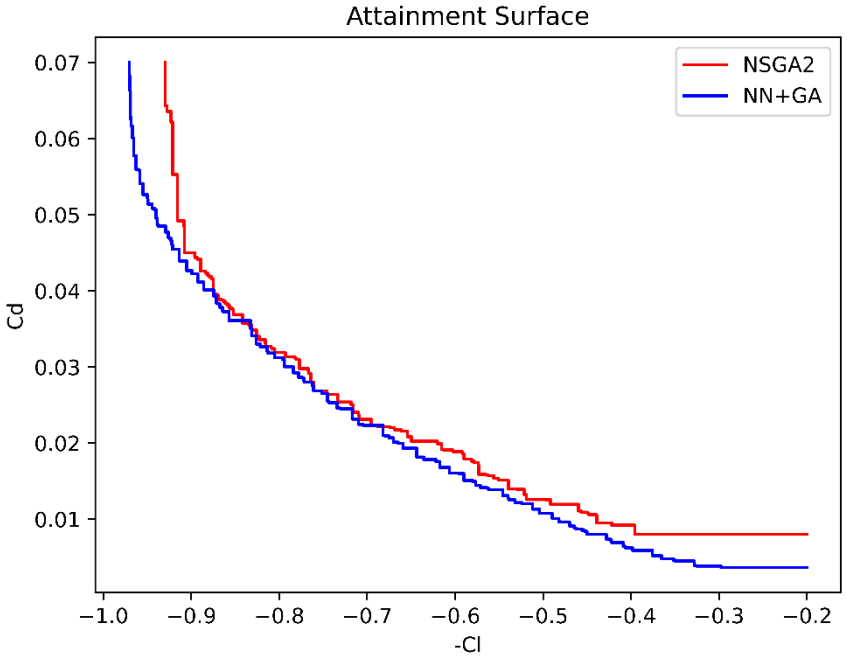


Figure 5.7 Average HV values for ASO-TA2.

Question arises as to why smaller population size of NSGA-II gives an improvement in ASO-TA2, but not in ASO-TA1. There is no obvious reason for this, but the fact that any GA algorithm is stochastic might be one of the factors. Moreover, we did not give more budget to the 3rd and 4th algorithm, so future generations are unknown.



(a) All solutions



(b) Attainment surfaces

Figure 5.8 Plots in the true objective space for ASO-TA2.

5.4.3 Results for ASO-TA3

The B-Spline parameterization has 18 control points as the design variables. Unlike the 9 design variables in PARSEC that ensures the smoothness of the airfoil, the B-Spline control points offer much more flexibility that allows the creation of rough surfaces. Thus, ASO-TA3 is the most complex problem among the three problems. It can be observed in Figure 5.10(a) that the initial population is located far away from the POF. Some of the solutions found by NSGA-II are even infeasible, violating the area constraints. Due to this difficulty, we define the budget to be: 3 x 400 = 1200 CFD evaluations for NN+GA, the third and fourth algorithm, and 3 x 1000 = 3000 CFD evaluations for NSGA-II (2^{nd} algo). From Figure 5.10(b), it is observed that both NN+GA and NSGA-II (2^{nd} algo) can find solutions that dominate the baseline (in terms of both objectives), located in the third quadrant position relative to the baseline.

Figure 5.9 shows the average HV history in solving ASO-TA3 which corroborates the superiority of NN+GA over NSGA-II even in the more complex problem. However more CFD evaluations are needed compared to the previous two problems. If one CFD takes around 3 minutes, it means NN+GA cuts the computational time for around 90 hours. The CFD result of the baseline airfoil and its geometry comparison between the best two extreme solutions with the baseline in ASO-TA3 are also plotted in Figure 5.14. It can be observed that the camber of the highest C_l airfoil aft portion is increased to gain more lift. In contrast, the camber of the lowest C_d airfoil aft portion is decreased to reduce the wave drag.

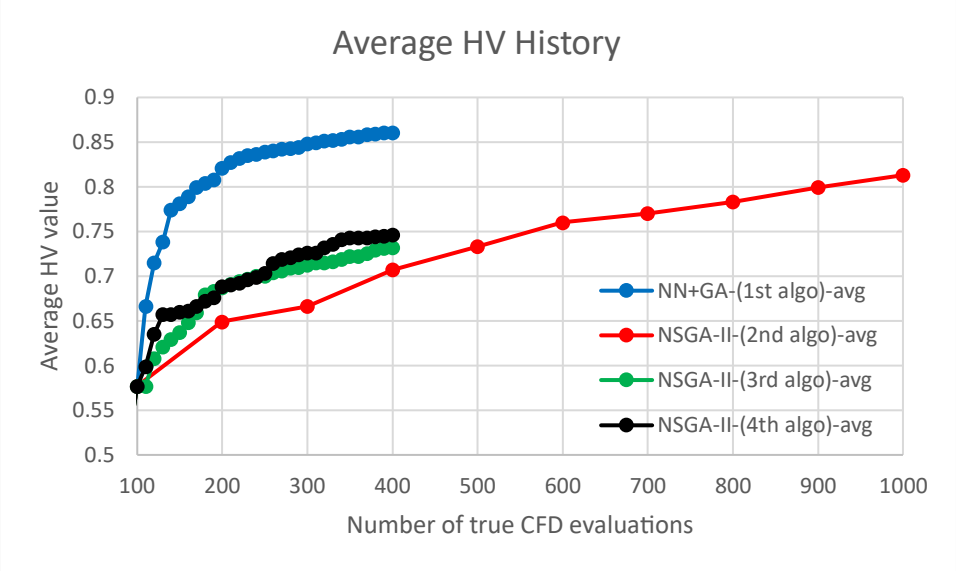
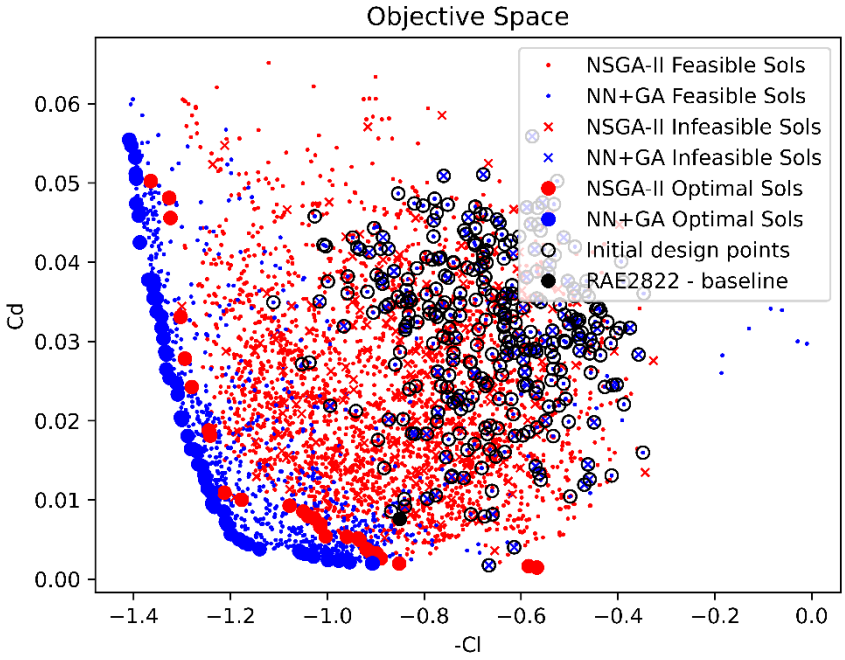
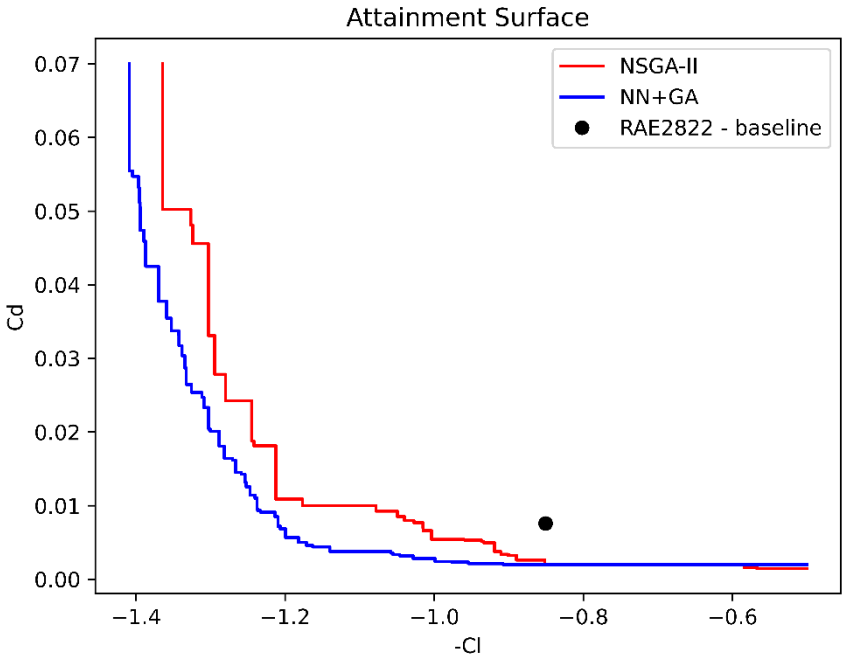


Figure 5.9 Average HV values for ASO-TA3.



(a) All solutions



(b) Attainment surfaces

Figure 5.10 Plots in the true objective space for ASO-TA3.

5.4.4 CFD results

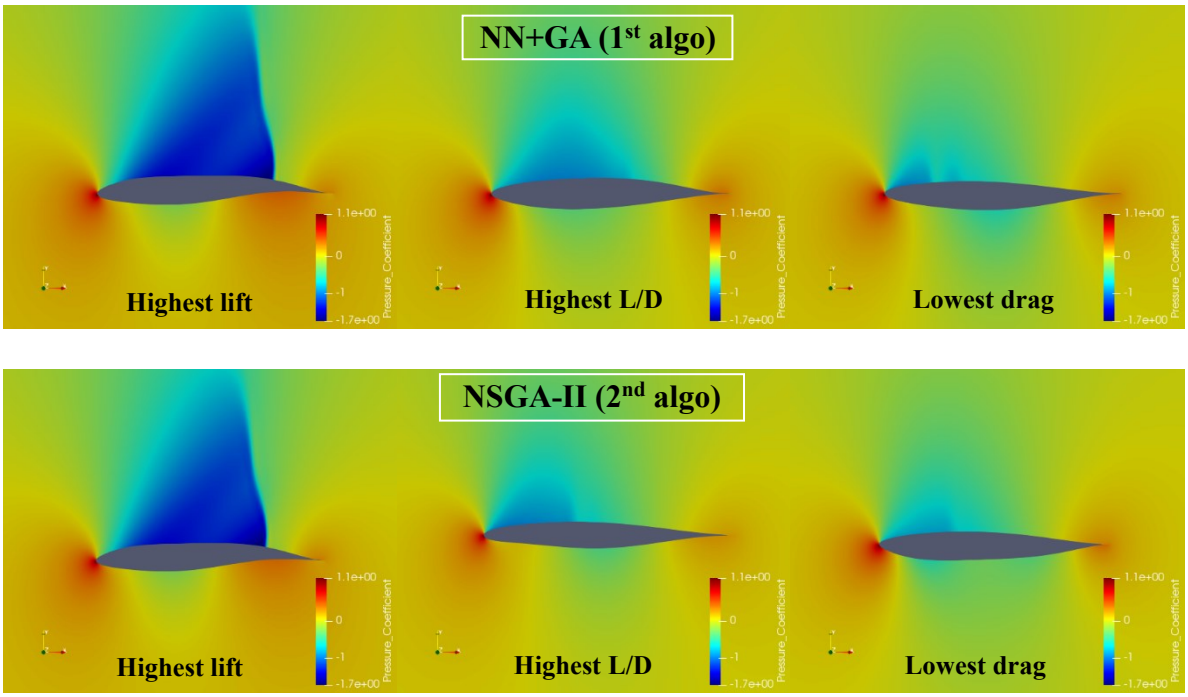


Figure 5.11 Pressure contours of extreme solutions for ASO-TA1.

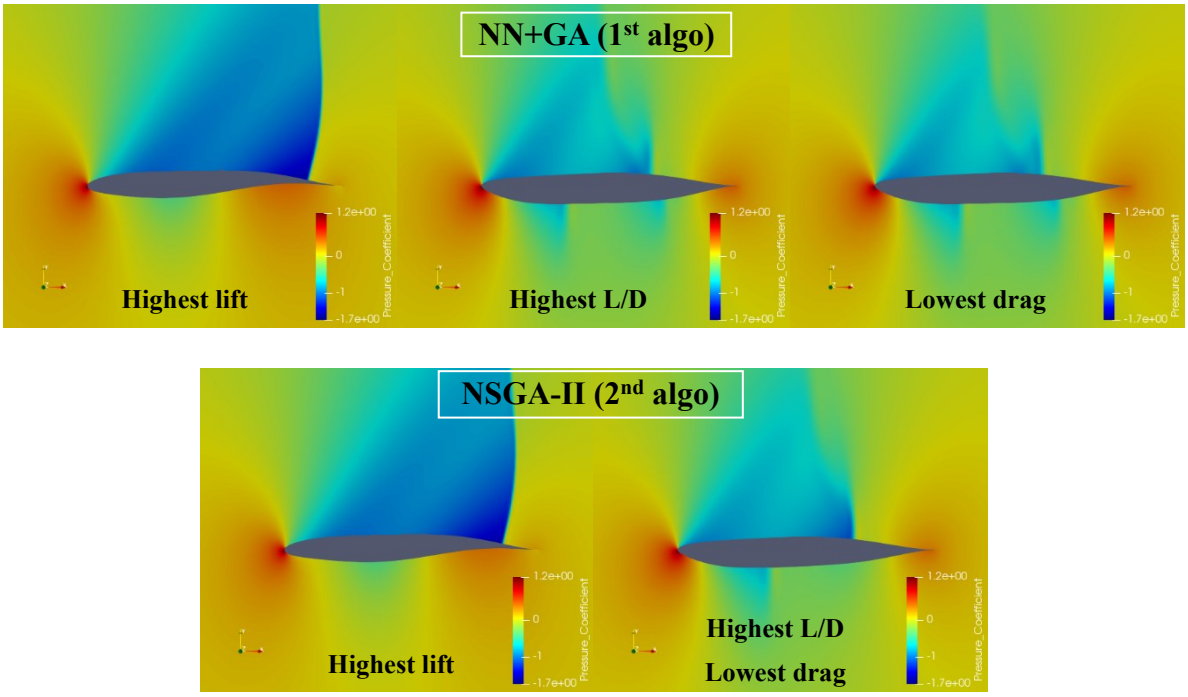


Figure 5.12 Pressure contours of extreme solutions for ASO-TA2.

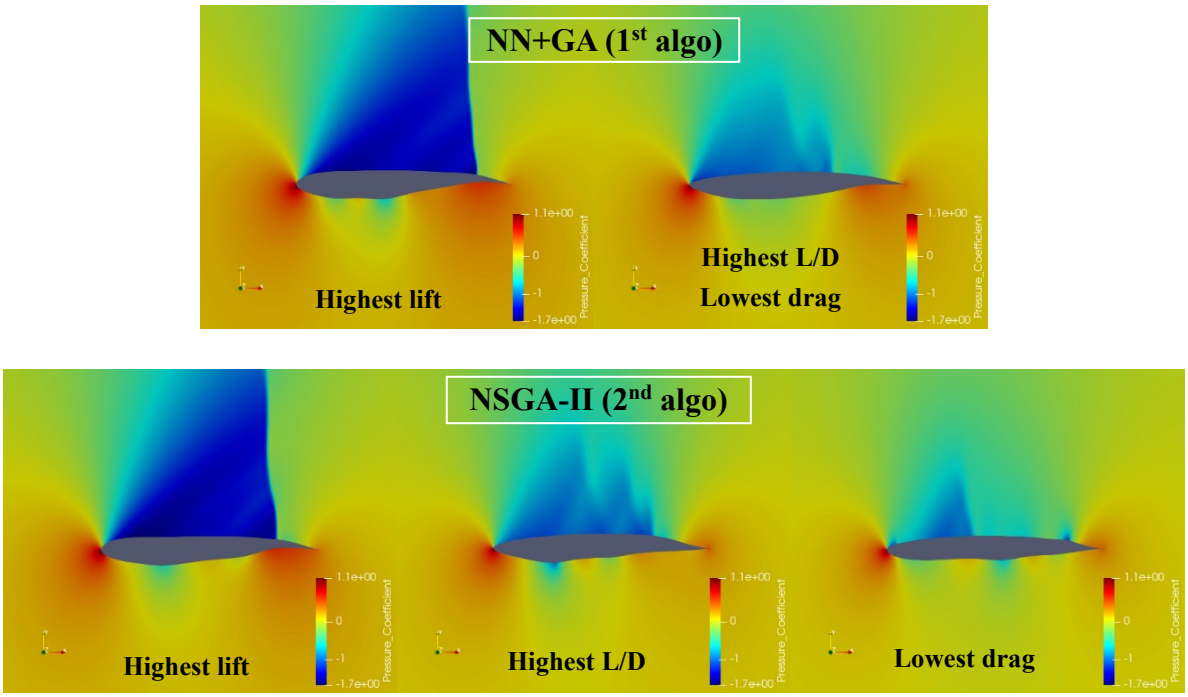


Figure 5.13 Pressure contours of extreme solutions for ASO-TA3.

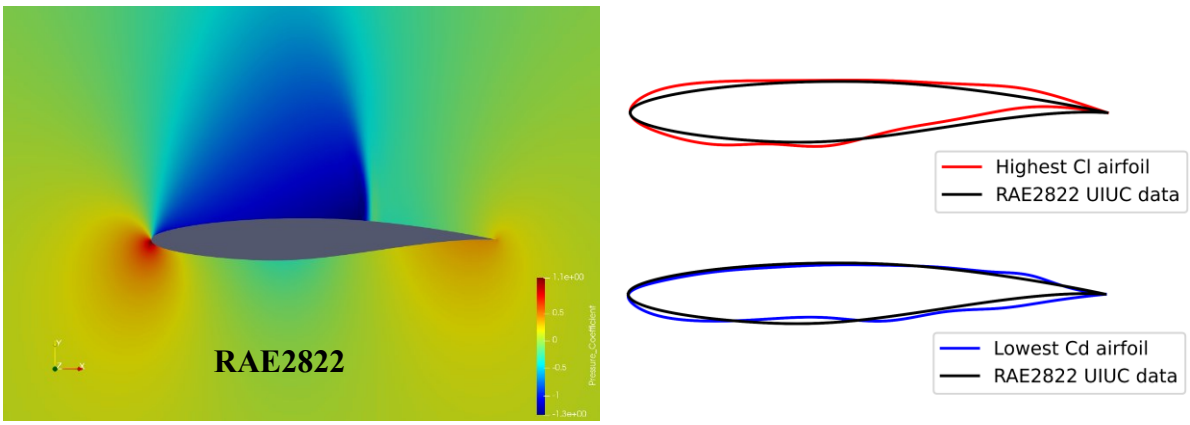


Figure 5.14 Pressure contour of the baseline and its geometry comparison with the best two extreme solutions found by NN+GA and NSGA-II

The pressure coefficient contours are plotted above for every extreme solution: highest lift, lowest drag, and highest L/D. They are also shown in the true objective space as in the Figures 5.15-5.17. The results show unnatural shock waves which are too intense. It is because of the use of Euler solver, which in its nature, is not realistic and this nature seems to be exploited by the optimizers that have no information about the physics (physics-uninformed models).

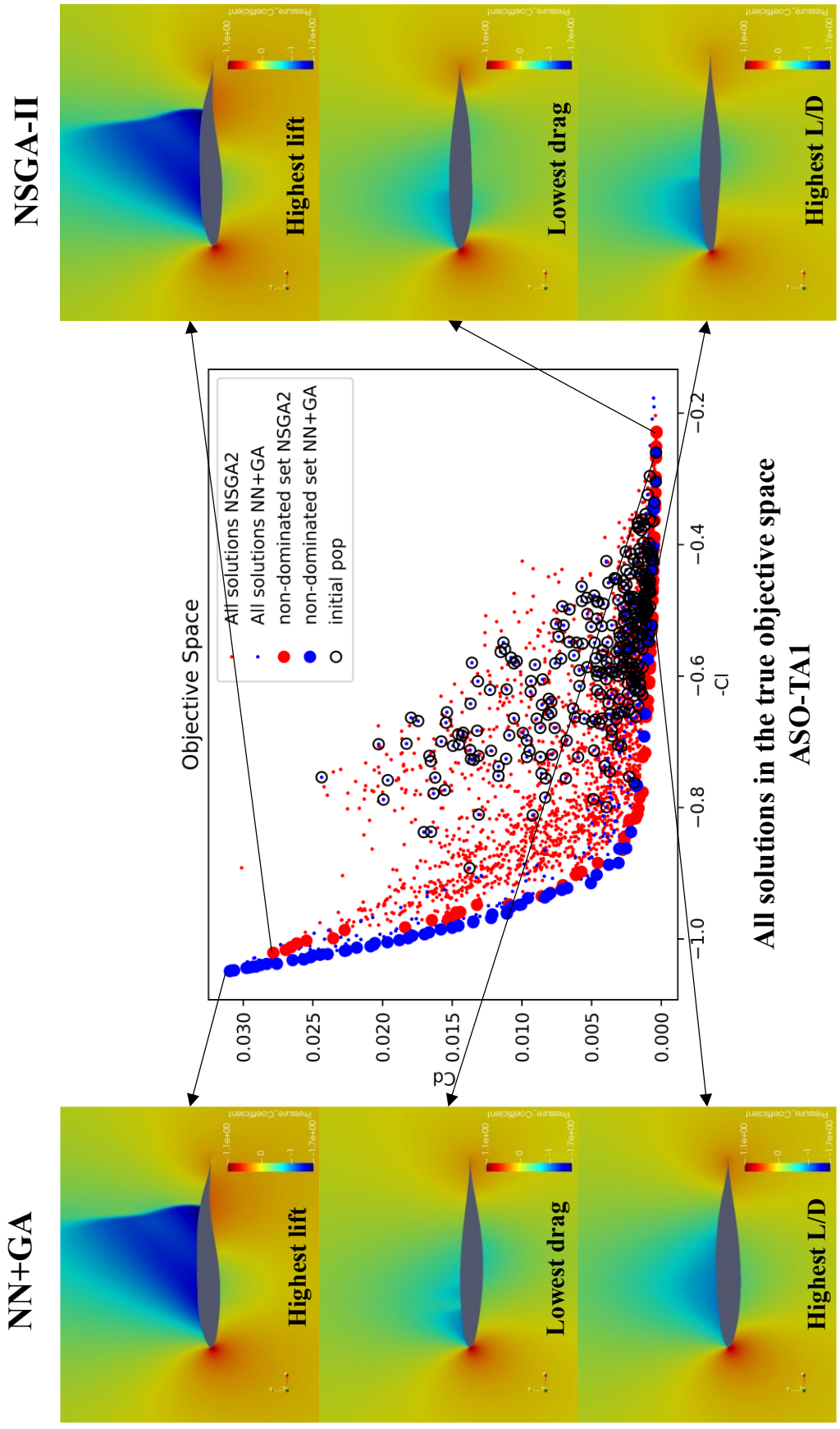


Figure 5.15 Extreme solutions shown in the true objective space for ASO-TA1

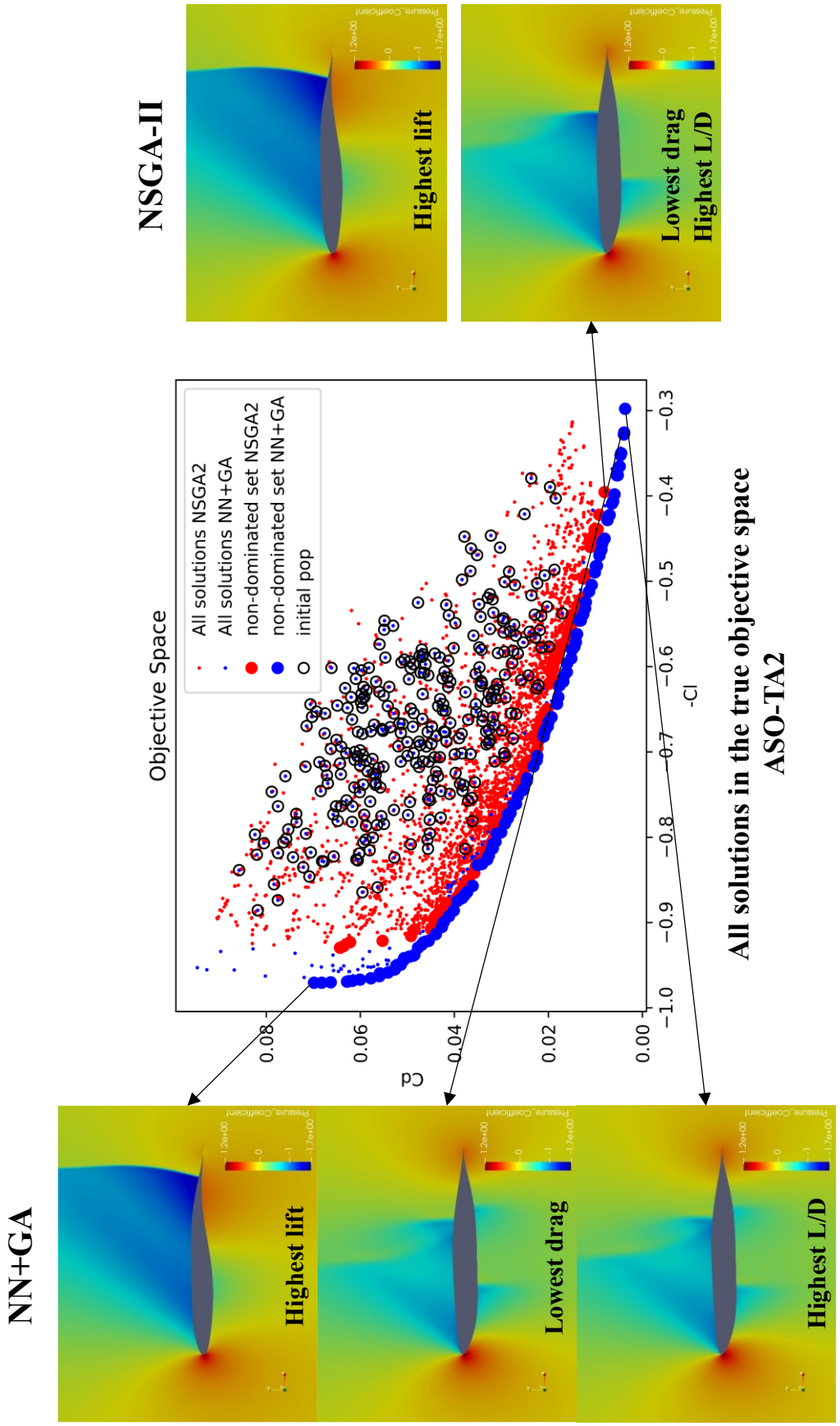


Figure 5.16 Extreme solutions shown in the true objective space for ASO-TA2

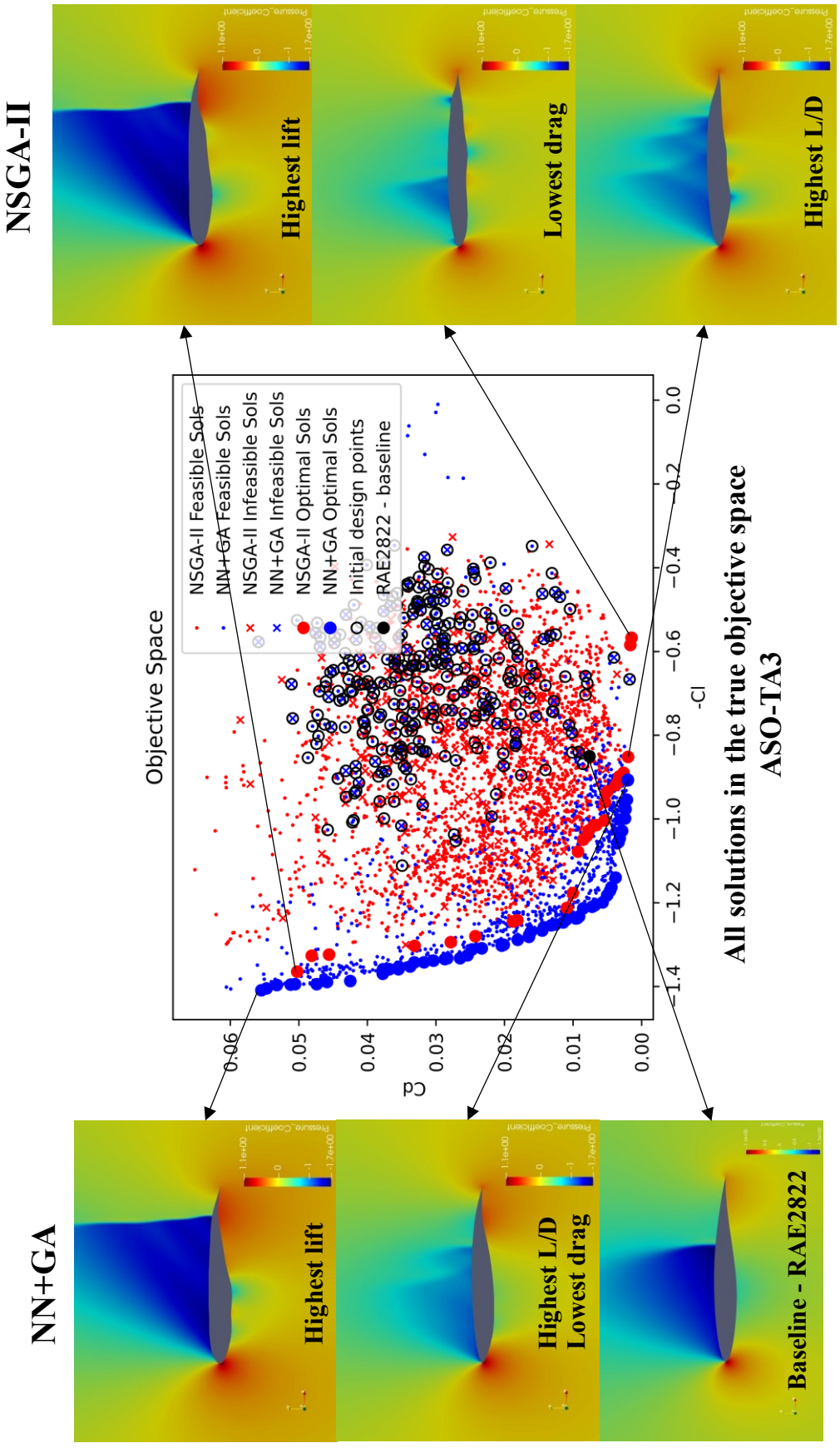


Figure 5.17 Extreme solutions shown in the true objective space for ASO-TA3

5.5 Summary

In this chapter, we have applied NN+GA and compared it with NSGA-II without surrogates and with different configurations. From all the results, NN+GA is shown to be more efficient than NSGA-II without surrogates in solving real-world problems, especially the transonic airfoil optimization. NN+GA cuts the computational time for around 48 hours in ASO-TA1 and ASO-TA2, while it reduces the CFD computational time for around 90 hours. Moreover, the obtained optimal solutions are better than the ones found by NSGA-II without surrogates.

Again, same as in the summary of Chapter 4, the reasons behind the superiority of NN+GA is since NN+GA uses surrogate models by NN that can do analytical evaluations that in turn becomes useful when doing population-based optimization routines. The number of generations can be set high since the aerodynamic performance is now approximated by the NN model, not evaluated by the CFD, as in the NSGA-II without surrogates.

Although the NN model is only an approximate model, it turns out to be sufficient to be used as the response surface that can assist the optimization routines. Now, it is undoubtedly true that NN sufficiently has the approximation power to map non-linear relationship between input and output. The next task is to compare the NN-based surrogate models with other surrogate models, e.g., Kriging. This task is subject to the next chapter.

Chapter 6 Comparing NN and Kriging surrogates for aerodynamic design

6.1 Introduction

The use of surrogate (or meta) modeling techniques in engineering design and exploration has been around for decades. The driving idea is to substitute the analysis tools that are often expensive with cheap and data-driven models. In aerospace systems, for example, the analysis involves aerodynamics, structures, propulsion, etc. Since exhaustive real experiments are impractical to analyze such systems, it has been a norm to use computer simulations to predict the design performance (e.g., computational fluid dynamics: CFD). However, to conduct an accurate design optimization, high-fidelity computer simulations that might take hours or days for a single design evaluation often must be employed. This makes population-based methods such as evolutionary algorithms (EAs) not suitable to be used in engineering design. Thus, combining surrogate models and EAs becomes a natural and common-sense method to effectively approach design optimization. This method is referred to as surrogate-based optimization (SBO) [26].

In real-world design optimization problems where the computational budget is limited, it is important to carefully decide which design points to sample next. EAs are usually used to perform the search, mainly due to their ability to locate optimal solutions close to the global optimum. The search is done by optimizing the acquisition functions, provided by data-driven surrogate models that can map input(s) to output(s) analytically. Therefore, the choice of surrogate models indirectly affects search performance.

Of several surrogate models, Kriging [4] is arguably the most popular surrogate model in the Bayesian optimization approach [27]. Kriging provides a function approximation along with its uncertainties. One Kriging model, however, can only do mapping for one function. This is a pitfall in multi-objective optimization problems (MOPs) since K Kriging models must be constructed for K expensive objective (and constraints, if any) functions. Neural network (NN), on the other hand, can do mapping between many design variables (inputs) to multiple functions (outputs) in a single model. NN thus has a great potential to be a surrogate model for optimization problems that have multiple expensive functions (e.g., MOPs) and many design variables.

Both Kriging and NN can provide a fitness approximation used in the EA optimization to drive the search. However, unlike NN, Kriging can also provide another type of acquisition function (e.g., expected improvement: EI) as a new search space, utilizing the readily available uncertainties information. The EI acquisition function promises a balance between exploitation and exploration when searching for new designs [28]. This chapter compares NN and Kriging with their respective acquisition functions combined with NSGA-II applied to multi-objective aerodynamic design optimization of transonic airfoil (ADO-TA) problems. Two aspects to be considered are the search performance and the model generation time. NSGA-II without surrogates is also performed in some of the problems to demonstrate that SBO techniques are efficient to approach design optimization.

6.2 Surrogate model: ordinary Kriging

The idea of any data-driven surrogate model is to construct an analytical model based on a sampled dataset, allowing fast prediction at unsampled locations. The model analytically maps the relationship between the input $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}^T$, where n is the dimensionality of the decision variables, and the output of interest $y = f(\boldsymbol{x})$ (black-box function).

There are many variants of Kriging. In this chapter, an ordinary Kriging (OK) is used that expresses the black-box function $y(\boldsymbol{x})$ as:

$$y(\boldsymbol{x}) = \mu + Z(\boldsymbol{x}), \tag{6-1}$$

where μ is the mean of Kriging approximation and $Z(\boldsymbol{x})$ represents its deviation. Sampled points are interpolated with the Gaussian process to estimate the distribution of the function value at unsampled locations. The correlation $Corr\left(Z(\boldsymbol{x}^{(i)}), Z(\boldsymbol{x}^{(j)})\right) \equiv \boldsymbol{R}_{ij}$ between $Z(\boldsymbol{x}^{(i)})$ and $Z(\boldsymbol{x}^{(j)})$ is modeled by the Gaussian autocorrelation function:

$$\boldsymbol{R}_{ij} = \exp\left(-\sum_{k=1}^{n} \theta_k \left| x_k^{(i)} - x_k^{(j)} \right|^2\right), \tag{6-2}$$

where $\boldsymbol{\theta} = \{\theta_1, \theta_2, \dots, \theta_n\}^T$ is the hyperparameter vector that needs to be optimized.

For a new unsampled point, OK predicts the output $y(\boldsymbol{x})$ and its uncertainty as follows:

$$\hat{y}(\boldsymbol{x}) = \mu + \boldsymbol{r}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\mu), \tag{6-3}$$

$$\hat{s}^2(\boldsymbol{x}) = \left[1 - \boldsymbol{r}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}) + \frac{1 - (\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}))}{(\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\mathbf{1})}\right], \tag{6-4}$$

where $\boldsymbol{R}$ denotes the n x n matrix whose (i, j) entry is $\boldsymbol{R}_{ij}$, $\boldsymbol{r}(\boldsymbol{x})$ is the correlation between any $\boldsymbol{x}$ and sampled points $\boldsymbol{x}_{sampled}$ whose $(i, 1)$ entry is $Corr\big(Z(\boldsymbol{x}), Z(\boldsymbol{x}^{(i)})\big)$, $\boldsymbol{y}$ is the observation at sampled locations, and $\mathbf{1}$ is the vector of ones.

6.3 Problem definitions

In this chapter, we present three multi-objective aerodynamic design optimization of transonic airfoils (ADO-TA, not to confuse with ASO-TA in the previous chapter). The airfoil parameterization, the grid configuration and the CFD solver are identical as before.

***ADO-TA1*:** (2 objectives, 0 constraint, 9 variables)
minimize : C_d and $-\,C_l$
with respect to : PARSEC variables in Table 5.1
subject to : -
@ single point, $\alpha = 2^o$, $M = 0.73$

***ADO-TA2*:** (2 objectives, 3 constraints, 18 variables)
minimize : C_d and $-\,C_l$
with respect to : B-Spline control points in Table 5.2
subject to : $0.8 \cdot A_{\text{baseline}} - A \le 0$
$Y_1 - Y_{18} \le 0$
$Y_2 - Y_{17} \le 0$
@ single point, $\alpha = 2^{\text{o}}$, M = 0.73

***ADO-TA3*:** (2 objectives, 6 constraints, 18 variables)
minimize : $0.25\ C_{d\,\text{M}_1} + 0.50\ C_{d\,\text{M}_2} + 0.25\ C_{d\,\text{M}_3}$
minimize : $-0.25\ C_{l\,\text{M}_1} - 0.50\ C_{l\,\text{M}_2} - 0.25\ C_{l\,\text{M}_3}$
with respect to : B-Spline control points in Table 5.2
subject to : $0.75 - C_{l\,\text{M}_1} \le 0$
$0.75 - C_{l\,\text{M}_2} \le 0$
$0.75 - C_{l\,\text{M}_3} \le 0$
$0.8 \cdot A_{\text{baseline}} - A \le 0$
$Y_1 - Y_{18} \le 0$
$Y_2 - Y_{17} \le 0$
@ multi points, $\alpha = 2^{\text{o}}$, $\text{M}_1 = 0.73$, $\text{M}_2 = 0.77$, $\text{M}_3 = 0.80$

While ADO-TA1 and ADO-TA2 include a single-point optimization, ADO-TA3 includes a multi-point optimization where the objective functions now become the weighted sums of C_d and $-C_l$ at the three flight conditions. The weights entirely depend on the flight requirements. We put more weight on the flight $\mathrm{M} = 0.77$. In the problems where B-Spline parameterization is used, 3 additional cheap constraints are introduced. The area constraint ($0.8 \cdot A_{\text{baseline}} - A \leq 0$) prevents the airfoil from going too slender compared to the baseline, and the trailing edge (TE) constraints ($Y_1 - Y_{18} \leq 0$, $Y_2 - Y_{17} \leq 0$) ensure geometrically feasible shapes. A is the 2D area of the airfoils calculated by approximating it with many triangles. Y_1 and Y_2 are y-coordinates of the two points in the lower part of the airfoil near the TE, while Y_{17} and Y_{18} are y-coordinates of the two points in the upper part near the TE. Look at the green dashed line in Fig 5.1(b). The area and trailing edge constraints prevent such airfoils from being created. In ADO-TA3, three new black-box constraints (C_l constraints) are introduced.

6.4 Methods

6.4.1 Acquisition functions

The general task in any design optimization problem is to find design candidates with optimum design performance, given the set of design variables $\boldsymbol{x}$, objective functions $\boldsymbol{f}$, and constraints $\boldsymbol{g}$. Since the true fitness evaluation of $\boldsymbol{f}$ and $\boldsymbol{g}$ is often expensive, the problem is reformulated: optimizing approximate objective functions $\hat{\boldsymbol{f}}$, subject to approximate constraints $\hat{\boldsymbol{g}}$. In the case where analytical constraints exist, they are evaluated using the true analytical functions. In this chapter, the value of $\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{g}}$ are calculated by the NN and OK models. The optimizer task is then to perform the search in the approximate objective space. We call these methods as **NN+GA** and **EST-OK+GA**, for combining the neural network and the ordinary Kriging, respectively, with a genetic algorithm (NSGA-II). The term *EST* comes from the fact that only the *estimated* value is used, obtained from the OK model.

Since the estimated value provided by the OK model includes uncertainty in it, there is a possibility of missing the global optimum. Another technique is to utilize the uncertainty information provided by the OK model and then construct a new acquisition function to be optimized: expected improvement (EI).

The probability of being the global optimum is given by the following expressions [29]:

$$I(x) = \max(f_{\min} - y, 0), \tag{6-5}$$

$$\mathrm{EI} = E(I) = \int_{-\infty}^{f_{\min}} (f_{\min} - y)\phi(y)dy, \tag{6-6}$$

where ϕ is the probability density function representing uncertainty about y. The problem being solved now by the optimizer is to find the next sample points to maximize EI.

In the Bayesian approach, it is common to convert a multi-objective optimization problem (MOP) into a single-objective optimization problem (SOP) with EI as the acquisition function. However, only one new sample point per iteration will be found in such a framework. Since we include multi-objective problems in this paper and to ensure fairness between the methods, EI is constructed for every objective. Therefore, the MOPs do not need to be converted into SOPs. In this way, Pareto-optimal solutions of EIs can be found, and several new points can be sampled per iteration. This technique is used in [30].

If there are constraints $\boldsymbol{g}$:

$$g_i \leq 0, \quad i = 1, \ldots, k, \tag{6-7}$$

the EI subject to constraints are modified as follows:

$$\mathrm{EI} = E(I) = E_y(I) \cdot \prod\nolimits_{i=1,\ldots,k} \phi_{g_i}(0). \tag{6-1}$$

To calculate Equation (6-1), the OK is constructed for all black-box objective functions and constraints. The term $E_y(I)$ is calculated in the OK for objective functions, while the term $\prod_{i=1,\ldots,k} \phi_{g_i}(0)$ is calculated in the OK for constraints. The last term represents the probability of satisfying the constraints. Noted only black-box constraints need to be included. This technique is used in [28]. We call this method **EI-OK+GA**.

6.4.2 Optimization flow and budget allocation

The optimization flow is the same as explained in the Chapter 4 – 3.2.3. Three SBO methods (NN+GA, EST-OK+GA, and EI-OK+GA) and a stand-alone NSGA-II without surrogates (hereafter called NSGA-II-WS) are done to solve ADO-TA1 and ADO-TA2. The setting for NSGA-II-WS and the NSGA-II optimizer inside the SBO is summarized below.

Table 6.1 Parameter values for NSGA-II

Population size	100
Max number of generations	250
Crossover	$\eta_c = 15$, *rate* = 0.9
Mutation	$\eta_m = 20$, *rate* = 0.01

Due to the budget limitation, only the SBO methods are done to solve ADO-TA3. For every method, we start with 100 evaluated initial design points. In ADO-TA1, 20 infilling points are then added per iteration for the SBO methods and 100 new points for NSGA-II-WS (since it produces the same size of the child population). In ADO-TA2 and ADO-TA3, 10 infilling points are added per iteration for the SBO methods. Since each problem is solved three times independently, so the total budget is 3 x 200 = 600 true evaluations for the SBO methods and 3 x 1000 = 3000 true evaluations for NSGA-II-WS. Only for NN+GA, the budget for solving ADO-TA2 is 3 x 400 = 1200 true evaluations.

6.4.3 Experimental settings

For EI-OK+GA and EST-OK+GA, the OK model is built for every black-box function evaluated by CFD. In ADO-TA1 and ADO-TA2, two OK models are built, while in ADO-TA3, three OK models are required. As for NN+GA, only one model is needed for every problem. The hyperparameters in OK are optimized using L-BFGS-B [31], while the connection weights in NN are optimized using Adam [11]. The number of OK hyperparameters is the same as the dimensionality of the problem, as stated in (2). The OK models are trained three times, the best one is picked based on the smallest leave-one-out-cross-validation (LOOCV) error. For training NN, gradient descent [10] with the mini-batch technique is used. The cross-validation technique is also used to stop the NN training when the validation error increases (overfitting sign). Unlike training OK, training NN requires more parameters to be defined. The training parameters for NN are listed in Table IV. No optimization is done to decide the values of these parameters, they are manually fine-tuned.

6.5 Comparison metrics

6.5.1 Search performance

The design optimization problems are treated as MOPs. In MOPs where the objectives are in a trade-off relationship, multiple optimal (or non-dominated) solutions will be found instead of a single solution. In SOPs, the search performance can be directly quantified by the objective function value. In MOPs, however, new metrics must be used. One of the ways to measure the search performance is, again, to use hypervolume (HV) metrics. HV can quantitatively measure the convergence and the spread of the non-dominated solutions. Given a reference point, the HV is obtained by calculating the area between the reference point and the current non-dominated solutions, in the true objective space. In the cases where all objectives are minimized, the higher the HV value, the better.

Prior to calculating HV, both objectives are normalized using two reference points: [0.1, 0.0] and [0.0, -1.5]. Hence, the reference point for calculating HV is [1.0, 1.0]. In this paper, the HV value is calculated every time new design points are sampled. Due to the stochastic nature of the optimizer, each problem is solved three times independently with different initial samples. The HV value is thus averaged and plotted against the number of true evaluations.

6.5.2 Model generation time

Using surrogates can cut the time for expensive evaluations. However, there should be additional time allocated for constructing the surrogates. In the iterative design process, the new surrogate models are reconstructed every time new design (or infilling) points are sampled. Hence, to realize a fast design process, the model generation time should be considered. The time to construct (or reconstruct) the model is plotted against the size of the dataset.

6.6 Results and discussions

6.6.1 Results for ADO-TA1

Figure 6.1 shows the average HV history for solving the first problem. All the SBO methods achieve higher average HV value with significantly fewer CFD evaluations, compared to the NSGA-II-WS. Although there is not much difference in the average HV value obtained between the SBO methods, it is quite clear that EST-OK+GA performs the best. Using the EI acquisition function from the OK models gives a little improvement over the EST counterpart in the first iteration. However, as the iteration goes on, the EI-OK+GA performs the worst. Note that the future iterations are still unknown.

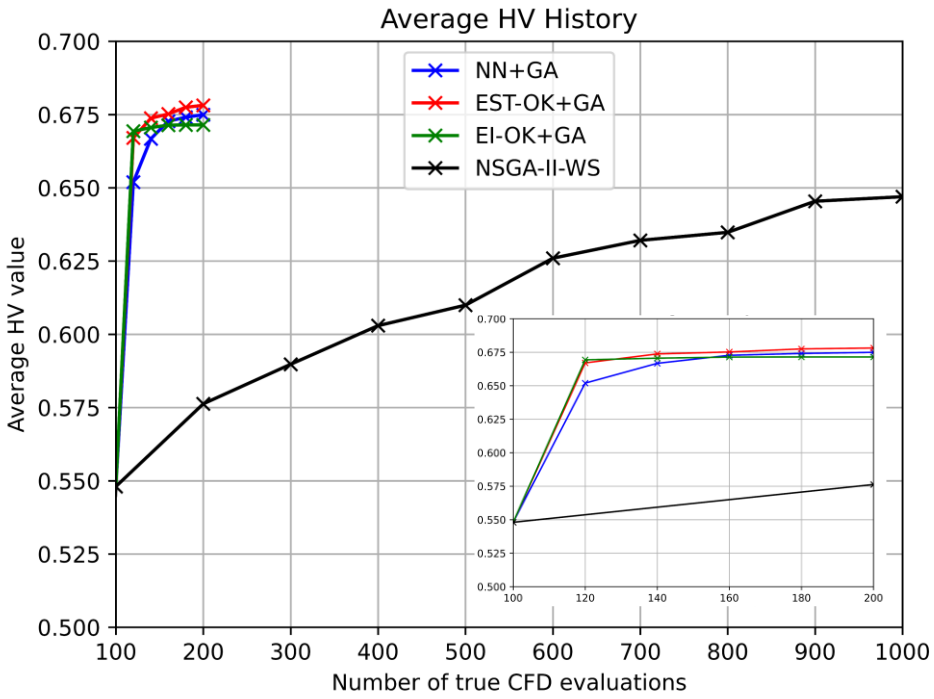
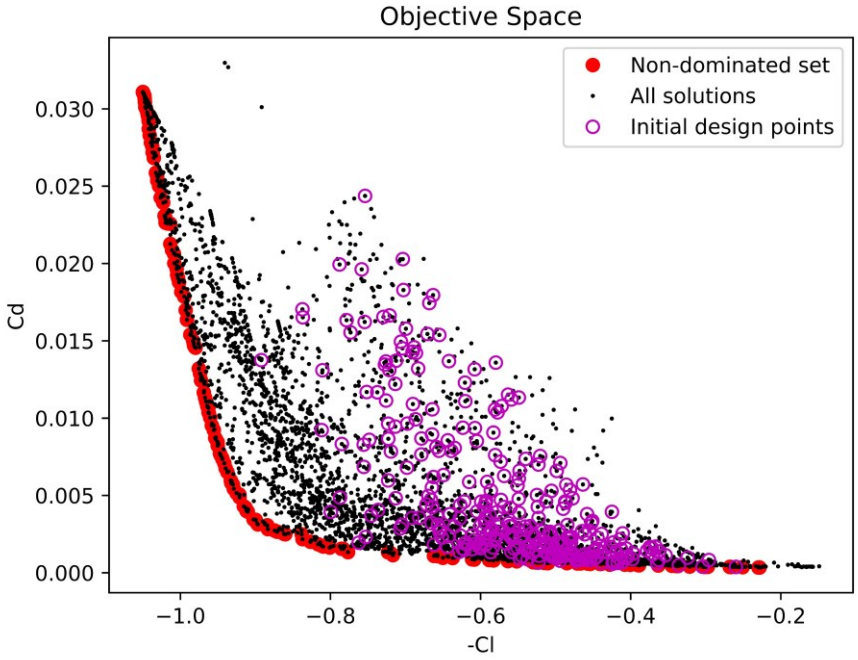
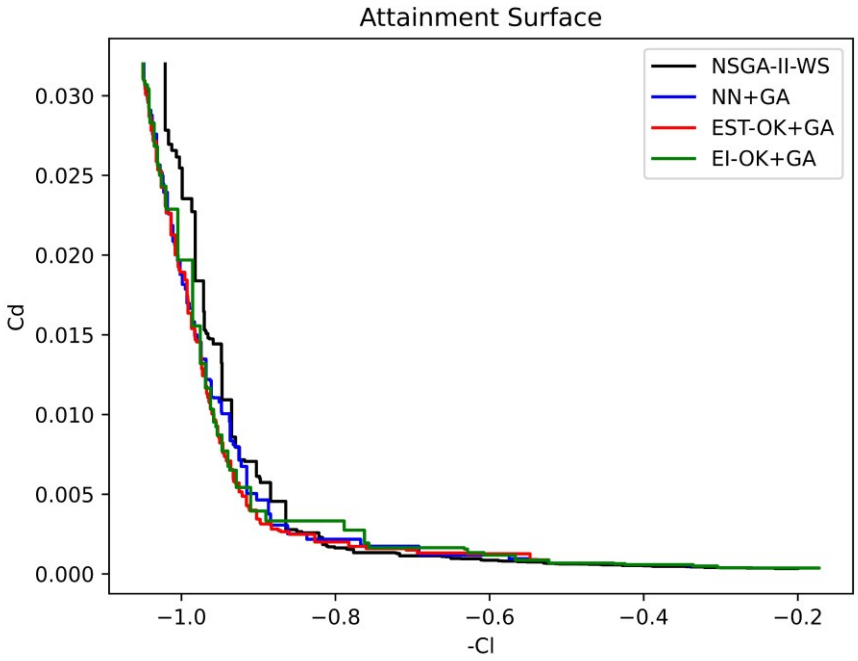


Figure 6.1 Average HV history for ADO-TA1.

From Figure 6.2(a), it is shown that most of the initial design points are in the low C_d region, hence the optimizers' task is to expand the solutions to cover the high C_l region. Figure 6.2(b). illustrates the attainment surface that is related to the average HV value.



(a) All solutions



(b) Attainment surfaces

Figure 6.2 Plots in the true objective space for ADO-TA1.

6.6.2 Results for ADO-TA2

Figure 6.3 shows the average HV history for solving the second problem. Again, the SBO methods show that they are more efficient than the NSGA-II-WS. In this problem, NN+GA performs the worst, while EST-OK+GA once again performs the best, by achieving the average HV value of 0.90 with only 3 x 200 = 600 true evaluations.

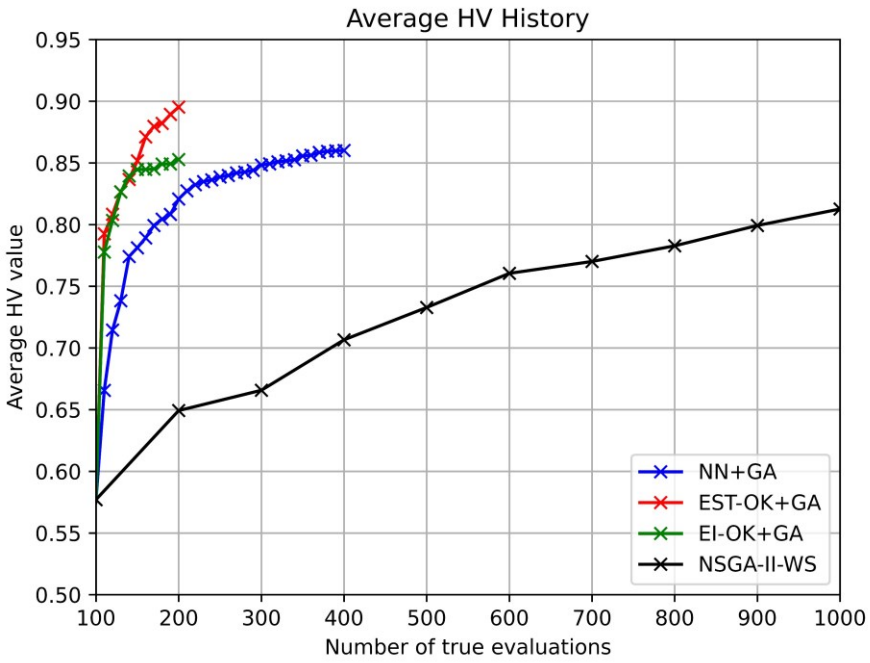
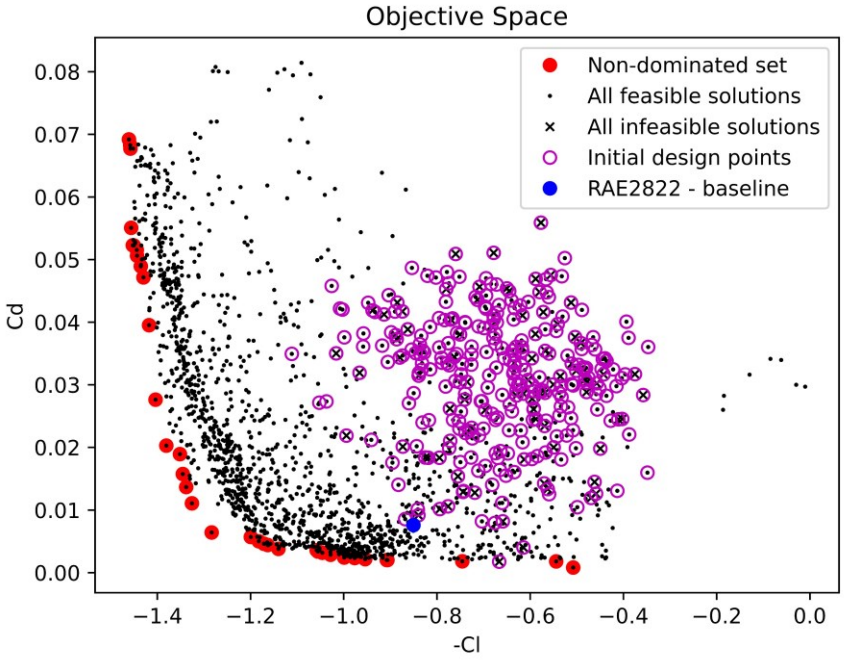
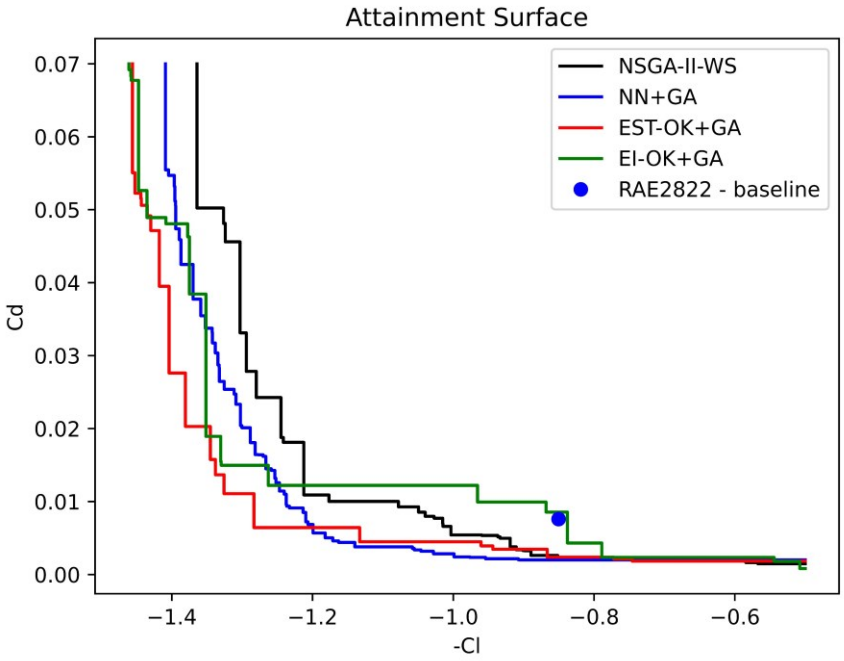


Figure 6.3 Average HV history for ADO-TA2

From Figure 6.4(a), most of the initial design points are now located far from both extreme solutions, hence the optimizers' task is to find and cover both extreme solutions. From Figure 6.4(b), it is observed that EI-OK+GA cannot find solutions that dominate the baseline (better in terms of both objectives). Note that NN+GA is given more budget to see how it can catch up with OK-based methods since it shows a poor performance with the same budget.



(a) All solutions



(b) Attainment surfaces

Figure 6.4 Plots in the true objective space for ADO-TA2

6.6.3 Results for ADO-TA3

Figure 6.5 Shows the average HV history for solving the third problem. Now, only the SBO methods are compared, due to the budget limitation, since 1 true evaluation in this multi-point problem includes the evaluation of 3 flight conditions. Figure 6.5 also agrees that EST-OK+GA performs the best. In the first two iterations, EI-OK+GA performs better than NN+GA, the latter then surpasses the former's performance in the next iterations. Again, EI does not give improvement over EST.

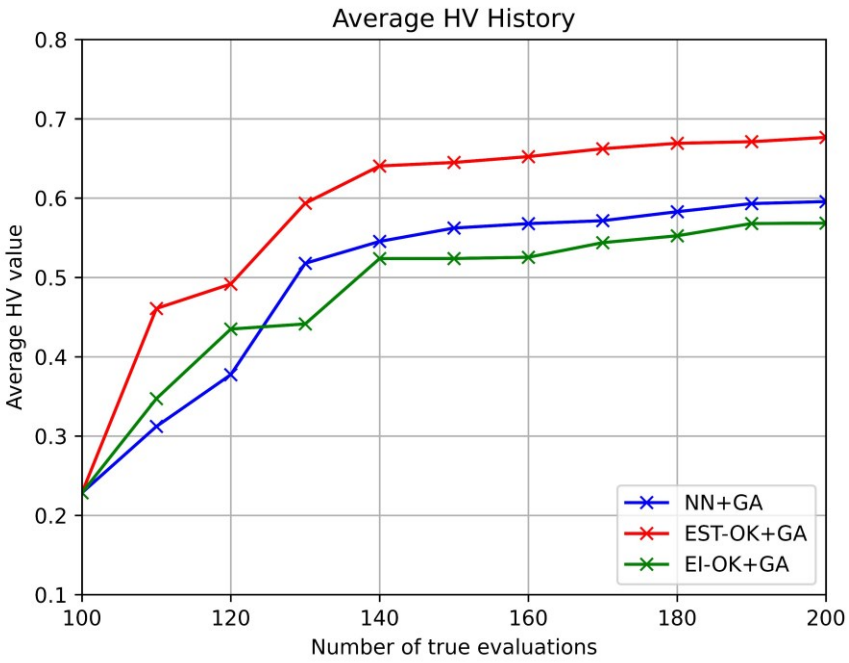
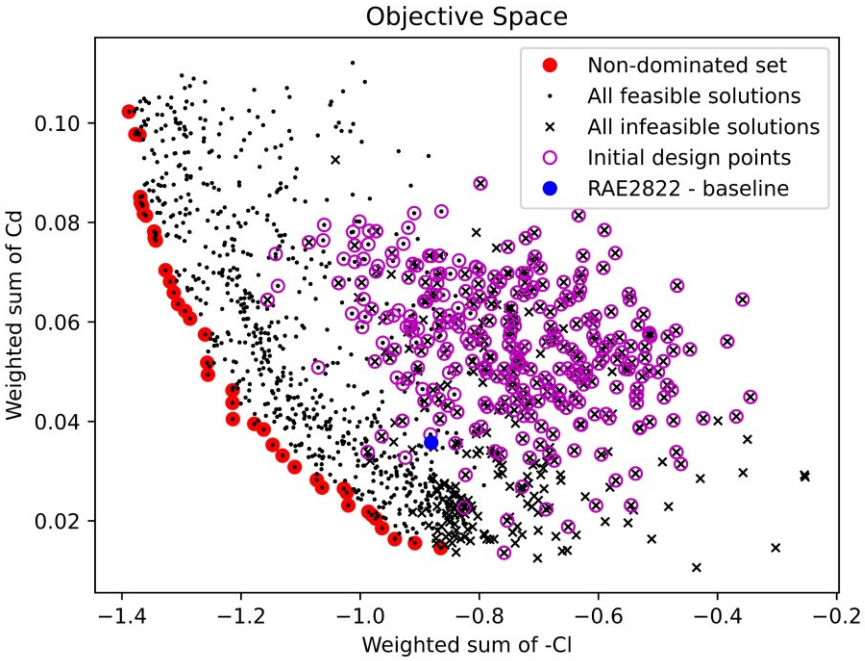
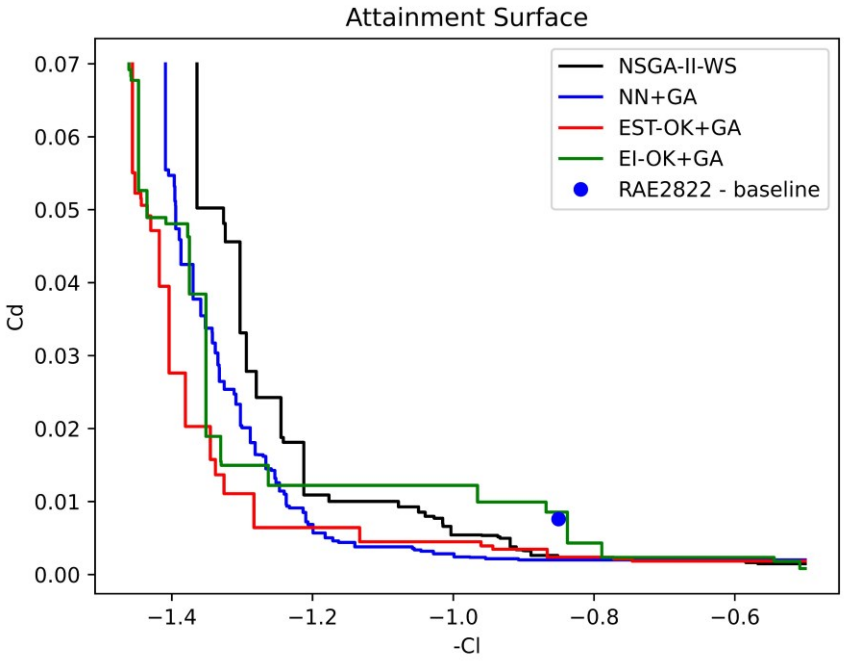


Figure 6.5 Average HV history for ADO-TA3

The initial design points are located far from both extreme solutions, as shown in Figure 6.6(a). The attainment surface plot shows that all the SBO methods can find solutions that dominate the baseline. In this problem, many infeasible solutions are found since C_l constraints are introduced. No feasible solutions are found higher than -0.80 for the weighted sum of $-C_l$, due to these constraints.



(a) All solutions



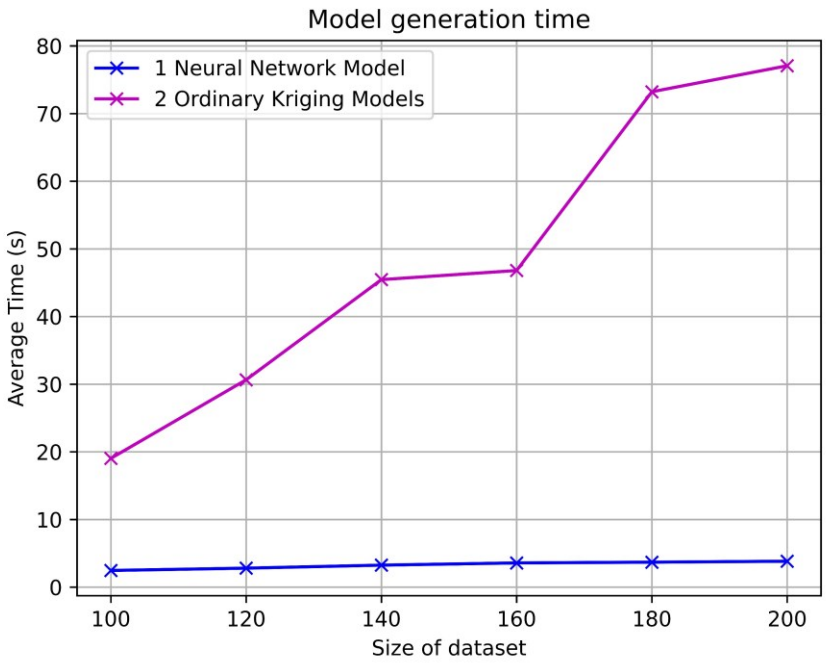
(b) Attainment surfaces

Figure 6.6 Plots in the true objective space for ADO-TA3

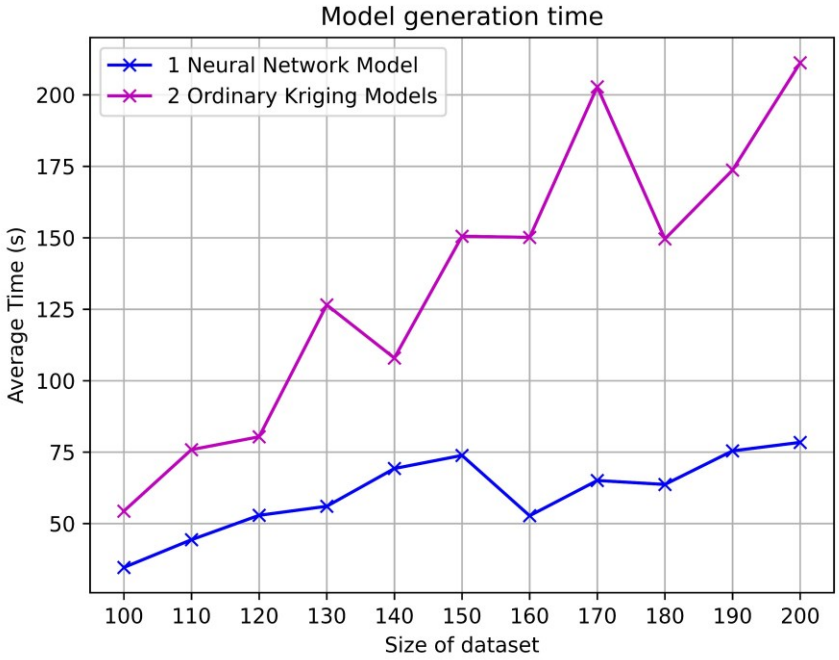
6.6.4 Model generation time comparison

The times associated with constructing the NN and OK models are plotted in Figure 6.7, against the size of the dataset. Note that the models are reconstructed when new sampled points are added to the dataset. The times are averaged among the three optimization runs. One cannot deny that the time for constructing OK models aggressively increases as the dataset grows. As for NN, it slowly increases. This is the reason why Kriging is often used in problems with small datasets. When it comes to big data (large datasets), NN is often deployed as the surrogate model.

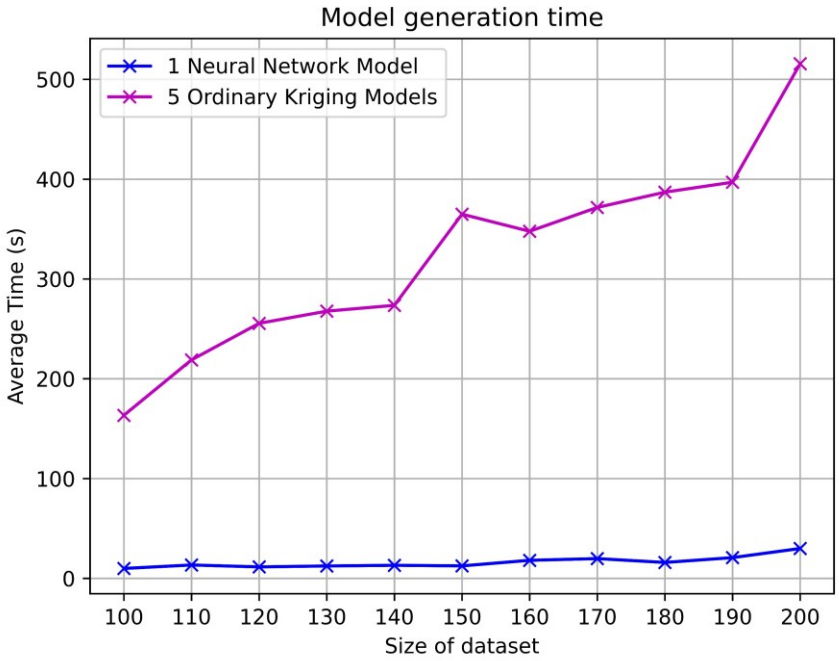
The OK models' generation time is also influenced by the dimensionality of the problem. From Figures 6.7(a) and 6.7(b), it is observed that it needs a longer time to construct OK models when the dimensionality is higher (dimensionality: 18 in ADO-TA2 compared to 9 in ADO-TA1). Note that both problems include the same number of OK models. As for the NN models, it is mainly influenced by the architecture of the network, e.g., the number of hidden neurons, type of network, etc. These different behaviors come from the way they are constructed. Constructing OK models includes the calculation of the inverse of matrix $\boldsymbol{R}$, as in the equations (6-3) and (6-4). A matrix $\boldsymbol{R}$ is an n x n matrix, where n is the dimensionality. Constructing NN models includes simple algebraic calculations. The activation function is often simple as well, as in (2-4) and (2-5). However, these calculations must be done for every neuron in the network. Hence, having many neurons or deeper networks makes the NN model generation time longer.



(a) ADO-TA1



(b) ADO-TA2



(c) ADO-TA3

Figure 6.7 Models' generation time

6.6.5 Comparing NN and Kriging

Using Kriging as the surrogate model in optimization tasks allows us to select various acquisition function types, thanks to the readily available uncertainty information in the Kriging. In NN, implementing uncertainty is not quite straightforward. Moreover, there are fewer parameters to be tuned when constructing Kriging (especially OK). On the other hand, parameterization in NN (especially MLP) is rather difficult when no optimization is done.

EST-OK+GA performs the best in all the problems, in terms of search performance. It turns out that using EST rather than EI is enough to solve the current problems. One possible explanation is that the current problems are still categorized as simple problems with low dimensionality, in a way that applying EI as the acquisition functions might introduce a more complex and rugged search space for the optimizer. Another explanation is that the techniques of creating EI for every objective function as in [28], rather than creating a single EI for all objectives (converting MOPs into SOPs), should be further justified and studied. Although EST-OK+GA performs better than NN+GA in the current problems, it is still hard to say that Kriging is better than NN in all types of aerodynamic design optimization problems, especially when dealing with high dimensionality (>100) problems. Díaz-Manríquez et al. in [32] suggest that Kriging is the best approach to be used only in low dimensionality problems since the accuracy deteriorates as the dimensionality increases.

In the present chapter, only MLP and OK are used to represent NN and Kriging, respectively. However, there are many variants of them, e.g., RBNN, Universal Kriging, etc. Another weakness of the present NN training is that no optimization is done on selecting the training parameters. Thus, the optimized and the best architecture is unknown. Despite all that, NN offers simplicity when it comes to constructing the surrogate model. In contrast to Kriging, many variables input and multiple functions output can be modeled in a single NN model.

6.7 Summary

In this chapter, Kriging and neural network (NN) were combined with NSGA-II to solve three different multi-objective aerodynamic design optimization of transonic airfoil problems. Three SBO methods with different acquisition functions provided by the models were used: EST-OK+GA, EI-OK+GA, and NN+GA. NSGA-II without surrogates (NSGA-II-WS) was also done to solve the first two problems. It was shown that EST-OK+GA performed the best in all the problems, and all the SBO methods are superior to NSGA-II-WS, highlighting their efficiency.

Considering the model generation time, unlike NN, Kriging is not suited to deal with large datasets. In high dimensionality problems (>100) with multiple black-box functions, it is also not recommended to use Kriging. Unless parallel computation is feasible, it can be time-consuming to construct multiple Kriging models for multiple black-box functions to approximate. However, in simple problems with low dimensionality (<100) as in this paper, Kriging model generation time is still reasonably short, while giving the best search performance (using EST). In this paper, only aerodynamic design optimization problems with low dimensionality (<100) were introduced to compare between Kriging and NN. It is therefore impossible to discuss the search performance scalability, that is the performance when applied to high dimensionality problems. To compare the methods in high dimensionality aerodynamic design problems is thus subject to future works.

Conclusion

To aid the process of design optimization that often includes expensive evaluations, surrogate models have been used over decades. One of the types of surrogate models that has grown recently is the artificial neural network (ANN). The surrogate models are often coupled with various types of population-based optimization algorithms, such as evolutionary algorithms (EAs). One variant of them is a genetic algorithm called NSGA-II.

In this thesis, I studied the feasibility and the efficacy of integrating ANN into NSGA-II (we call it NN+GA). The whole framework of optimization of NN+GA has also been proposed, utilizing supervised and unsupervised machine learning techniques. The NN+GA was first applied to test problems with various complexities, including single- and multi-objective optimization test problems. It was found that NN+GA can cut the number of true function evaluations, that is expected to be useful for expensive optimization problems.

The investigation continues by applying the NN+GA to a real-world problem, i.e., multi-objective aerodynamic shape optimization of transonic airfoils. The optimization includes three different problem definitions. It was found that integrating ANN into NSGA-II is more efficient than directly using NSGA-II with true evaluations, in the given aerodynamic design problems. NN+GA can cut the computational time for around 48 and 90 hours, by replacing the CFD evaluations in the optimization procedure. In other words, ANN can map non-linear functions and is sufficient to be a surrogate model that can replace CFD in the optimization procedure.

Lastly, ANN is compared with the most popular surrogate model in the field of aerospace engineering: Kriging. Some advantages and disadvantages of using ANN and Kriging surrogate models for aerodynamic design optimization were discussed. The comparison also includes three different acquisition functions when solving multi-objective aerodynamic design optimization of transonic airfoils. Kriging was shown to be more efficient than ANN in the current low-dimensionality and low-fidelity design problems. However, the performance scalability in the high-dimensionality problems is subject to future research.

References

[1] Multi-Objective Design Optimization of Fluid Machinery | Shimoyama Lab Research. http://www.ifs.tohoku.ac.jp/shimoyama/research/

[2] K. Deb, “Multi-objective optimization using evolutionary algorithms,” Wiley, Chichester, 2001, p. 2.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE TEC, vol. 6, 2002, pp. 182-197.

[4] D. G. Krige, “A statistical approach to some mine valuation and allied problems on the Witwatersrand,” 1951.

[5] M. J. D. Powell, “Restart procedures for the conjugate gradient method,” Mathematical Programming, vol. 12, 1977, pp. 241-254.

[6] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, 1995, pp. 273-297.

[7] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” The Bulletin of Mathematical Biophysics, vol. 5, 1943, pp. 115-133.

[8] Z. Shen, H. Yang, and S. Zhang, “Neural network approximation: Three hidden layers are enough,” Neural Networks, Elsevier, vol. 141, 2021, pp. 160-173.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. William, “Learning representations by back-propagating errors,” Nature, vol. 323, no. 9, 1986, pp. 533-536.

[10] S. Amari, “A theory of adaptive pattern classifiers,” IEEE Transactions on Electronic Computers, EC-16, 1967, pp. 299-307.

[11] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” 3rd International Conference for Learning Representations, San Diego, 2015.

[12] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” Proceedings of the 3 International Conference on Machine Learning, vol. 37, 2015, pp. 448-456. 2nd

[13] R. Tibishirani, W. Guenther, and H. Trevor, “Estimating the number of clusters in a data set via the gap statistic,” Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 63, issue 2, 2001, pp. 411-423.

[14] Natural selection images | https://edu.glogster.com/glog/natural-selection/1kstclxlg4d

[15] K. Deb, “Multi-objective optimization using evolutionary algorithms,” Wiley, Chichester, 2001

[16] Genetic algorithm in pymoo | https://pymoo.org/algorithms/genetic_algorithm.html

[17] M. D. McKay, R. J. Beckman, and W. J. Conover, “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” Technometrics, vol. 21, 1979, pp. 239-245.

[18] J. Blank and K. Deb, “Pymoo: Multi-objective optimization in python,” IEEE Access, vol. 8, 2020, pp. 89497-89509.

[19] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, 1967, pp. 281-297.

[20] Optimization test functions and datasets | http://www.sfu.ca/~ssurjano/optimization.html

[21] H. Sobieczky, “Parametric airfoils and wings,” Recent Development of Aerodynamic Design Methodologies, 1999, pp. 71-87.

[22] O. R. Bingol and A. Krishnamurthy, “NURBS-Python: An open-source object-oriented NURBS modeling framework in Python,” SoftwareX, vol. 9, 2019, pp. 85-94.

[23] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso, “SU2: An open-source suite for multiphysics simulation and design,” AIAA Journal, vol. 54, 2016, pp. 828-846.

[24] A. Jameson, “Origins and further development of the Jameson-Schmidt-Turkel scheme,” AIAA Journal, vol. 55, no. 5, 2017, pp.1-23.

[25] Mesh Generation Software for CFD: Pointwise, Inc. | http://www.pointwise.com/

[26] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, “Surrogate-based analysis and optimization,” Progress in Aerospace Sciences, vol. 41, 2005, pp. 1-28.

[27] R. Lam, M. Poloczek, P. Frazier, and P. Willcox, “Advances in Bayesian optimization with applications in aerospace engineering,” AIAA NDAC, 2008, p. 1696.

[28] S. Jeong, S. Obayashi, and K. Yamamoto, “Aerodynamic optimization design with Kriging model,” Trans. Japan Soc. Aero. Space Sci., vol. 48, 2005, pp. 161-168.

[29] S. Matthias, J. W. William, and R. J. Donald, “Global versus local search in constrained optimization of computer models,” NDAED, vol. 21, 1979, pp. 239-245.

[30] S. Jeong, Y. Minemura, and S. Obayashi, “Optimization of combustion chamber for diesel engine using Kriging model,” JFST, vol. 1, 2006, pp. 138-146.

[31] R. H. Byrd, P. Lu, and J. Nocedal, “A limited memory algorithm for bound constrained optimization,” SIAM JSSC, vol. 16, 1995, pp. 1190-1208.

[32] A. Díaz-Manríquez, G. Toscano-Pulido, and W. Gómez-Flores, “On the selection of surrogate models in evolutionary optimization algorithms,” 2011 IEEE Congress of Evolutionary Computation (CEC), 2011, pp. 2155-2162.

Acknowledgment

Firstly, all praise to Allah, I have finished this thesis as a completion of my bachelor study.

I would like to dedicate this study to my parents who have supported me in all ways. They have raised me and guaranteed my education since I was in the kindergarten. To my little sister, you are my frenemy at home, although I have not come home for almost two years, and surely, I will soon. I must finish what I have begun first: my study.

I am expressing my gratitude to the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan for becoming my sponsor throughout my stay in Japan. I hope I can fulfill my responsibility well. This study is also partially supported by MEXT as "Program for Promoting Researches on the Supercomputer Fugaku" (Leading research on innovative aircraft design technologies to replace test flight).

I would like to express my heartiest thanks with a deep sense of gratitude to my academic supervisor, Shimoyama-sensei, for his guidance throughout my bachelor study. He offered me to work with him in a Fugaku supercomputer project, which caught my interest. He came in the right time and became my savior when I was so confused of what I would do for my research. Looking forward to working with him again for the next two years in my master study. This study is also an extended study from the previous student of his, Kato-senpai. Although I don't have a chance to meet him in person, I can somehow interact with him through his codes that was handed in to me.

I would also like to appreciate all the supports and togetherness from my lab mates: Tim, Hilmi, Foong, Kamada, Tanguy, and others. We often had dinner together and talked about things beside our research to nurture my mental health. I had a great discussion and guidance, especially from Tim. He is like 24/7 answering my questions. I messaged him 3 am in the morning, but he replied in a blink of an eye.

I must express my thanks to my IMAC-U friends and my Indo FGL 17 friends: Hanif, Zein, Amanda, Atika, Nadya, Maul, Joshua, Clifford, Walance. and others. To Naoki and Lisa, my only Japanese friends in the class, who have given me language support. They are so patient answering my silly questions in Japanese sometimes. We started this course together in October 2017 and I am glad we can end it together with great presentations, and soon graduate in September 2021. We often had great parties before the pandemic, quite few during the pandemic. Hopefully we can still have farewell parties after the pandemic since most of us continue to master program here.

Lastly, I would like to thank my support system in Sendai: *grup Tahsin* (Tahlil dan Yasin). Hilmi and Adam, my senpais but I treat them like a friend, lol. They introduced me to this great university. Maul and Aghi, who are very *nano-nano*. Good luck for your career, conquer Europe, Maul! Mas Pras, Mbak Yanti, Mbak Farah, Mbak Halida, Mbak Pinka, Bang Ijat, and Kang Shiddiq who often invited me for free foods hahaa XD. Stay connected guys! I am grateful I could get to know some of you who will soon be great lecturers and professors in Indonesia.

The end.