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

ハリヤンシャ ムハマド アルフィヤンディ，下山 幸治（東北大学）

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

HARIANSYAH Muhammad Alfiyandy, SHIMOYAMA Koji (Tohoku University)

ABSTRACT

Evolutionary algorithms (EAs) have been widely used in design optimization that often requires expensive evaluations (e.g., computational fluid dynamics: CFD). An artificial neural network (ANN) based surrogate model is used in the EA optimization routines to reduce the time for these expensive evaluations. The ANN can model the relationship between many design variables and objective functions in a single surrogate model, unlike other surrogate models (e.g., Kriging). In this study, a genetic algorithm (GA) coupled with a dynamically retrained ANN is proposed and applied to multi-objective transonic airfoil shape optimization where aerodynamic performances are evaluated with CFD. The proposed method is shown to converge more quickly towards the Pareto-optimal front with fewer CFD evaluations compared to a stand-alone GA, proving the efficacy of ANN as the surrogate model in the GA.

I. 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 its airfoil section.[1] Aerodynamic shape optimization of transonic airfoil (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 objective of ASO-TA is drag minimization to reduce fuel consumption at cruise. However, it comes with a tradeoff with lift, for example. The induced drag increases in proportion to the square of lift. Zero induced drag means zero lift. This study includes several ASO-TA problems with two objectives: drag minimization and lift maximization.

Population based metaheuristic approaches, such as evolutionary algorithms (EAs)[2], have been used to solve multi-objective optimization problems (MOPs) due to their ability to find multiple tradeoff solutions. The obtained tradeoff solutions are called non-dominated solutions that approximate unknown Pareto-optimal solutions (the ideal solutions in which no other solutions in the design space are better than them in terms of all the objectives).

The discipline of aerodynamic shape optimization (ASO) has benefitted from the increasing use of multi-objective EAs (MOEAs). Among MOEAs, multi-objective genetic algorithms (MOGAs) are gaining popularity in recent years (e.g., NSGA-II algorithm[3]). However, MOEAs require a considerable number of objective and constraint function evaluation calls. This is a drawback if applied to ASO, where computational fluid dynamics (CFD) is used as function evaluation (often computationally expensive). Alternatively, surrogate (or meta) models are used as approximate models that analytically map inputs to outputs based on a sample dataset given. All direct calls to the expensive evaluations are replaced with the surrogate models, hence called surrogate-based optimization (SBO).[4] To improve the accuracy of the surrogate models, they often need to be updated with additional sample datasets given by infilling criteria in a sequential process.

The use of surrogate models is nothing new in the field of engineering design optimization and exploration, especially ASO. Palar et al. pointed out several key issues concerning the applications of SBO techniques in real-world problems.[5] Of several surrogate models, Kriging[6] is arguably the most popular surrogate model relied in

Bayesian optimization approach.[7] Kriging provides the function prediction along with its estimation error. Zuhal et al., for example, did the benchmarking multi-objective Bayesian global optimization for aerodynamic designs with ordinary Kriging as the surrogate model.[8] One Kriging model, however, can only do mapping for one function. This is a pitfall in MOPs since *K* Kriging models must be constructed for *K* expensive objective functions. It gets worse when dealing with high number of design variables which translates to high number of hyperparameters to be optimized leading to longer model generation time.

Artificial neural network (ANN), on the other hand, can do mapping between many design variables (input) and multiple functions (output) in a single model. ANN 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.

In this paper, we develop an ANN-assisted NSGA-II (NN+GA) with application to several multi-objective ASO-TA problems. The objective is to investigate the efficacy of the proposed method in the field of aerospace systems that have low to moderate number of design variables. This paper describes a preliminary study before the proposed method is used in high dimensional problems. The performance metric used is hypervolume (HV) that measures both the convergence and the spread of the obtained non-dominated solutions. We also compare the results obtained using NN+GA and standard NSGA-II by tracing the history of HV values per iteration.

The remainder of this paper is structured as follows: Section II explains the basic of ANN surrogate model. Section III elaborates the flow and the techniques used in the SBO procedure. Section IV details the multi-objective ASO-TA problems. Section V presents the results along with the discussions. Finally, conclusions and future works are summarized in Section VI.

II. Artificial Neural Network

ANN is an abstract computational model of the human brain (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 in a single model. Unlike Kriging that interpolates data, ANN is a regression type model and thus has the potential to be inherently fault tolerant or capable of robust computation.[9]

Artificial neuron is a building block of ANN that 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, as in Figure 1.

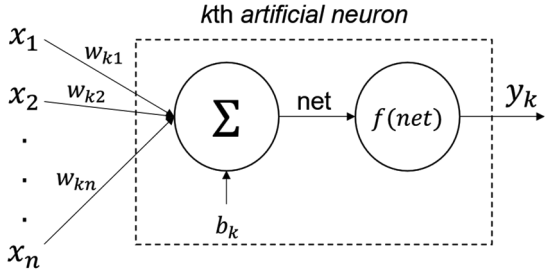


Fig. 1: A mechanism of an artificial neuron

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

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

$$net_k = \sum_{i=1}^{n} x_i w_{ki} + b_k \quad (1)$$

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

$$y_k = f(net_k) \quad (2)$$

The function f is called the activation function. Some popular activation functions include logistic sigmoid, hyperbolic tangent, and rectified linear unit (ReLU).

The regression model is constructed by learning a set of data samples consisting of input $\boldsymbol{x}$ (design variables) and target output $\boldsymbol{t}$ (in this case, CFD). 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 is defined as follows:

$$E(\boldsymbol{w}) = \sum_{j=1}^{J} ||\boldsymbol{t_j} - \boldsymbol{y_j}||^2 \quad (3)$$

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 (3) 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[10] with gradient descent method[11].

III. Surrogate Based Optimization using NN+GA

An MOP includes more than one objective $\boldsymbol{f}$ to be simultaneously optimized with respect to some constraints $\boldsymbol{g}$ and $\boldsymbol{h}$, if any. The search space is done on the design space $\boldsymbol{x}$. The formulation is generally written as follows.

$$\begin{aligned} &\text{Minimize } f_m(\boldsymbol{x}), && \text{m} = 1, 2, \dots, \text{M};\\ &\text{subject to } g_j(\boldsymbol{x}) \le 0, && \text{j} = 1, 2, \dots, \text{J};\\ &h_k(\boldsymbol{x}) = 0, && \text{k} = 1, 2, \dots, \text{K};\\ &{x_i}^{(L)} \le x_i \le {x_i}^{(U)}, && \text{i} = 1, 2, \dots, \text{n}; \end{aligned}$$

Maximization problem can be treated the same way as minimization problem by multiplying the objective function with -1. Many kinds of EAs have recently been introduced to solve MOPs. Deb wrote a good summary of most of the methods in his book.[12]

Many EAs guarantee the findings of global optimum, but then again, they typically require numerous calls to function evaluation, which is not favorable when the function evaluation is expensive to evaluate. It thus drives the development of SBO method to reduce the number of evaluations which translates to low computational cost.

III.A. General procedure of SBO

1. Geometry parameterization is done as the first step. The geometry is characterized by a set of design variables that determine the shape of a design.
2. Latin hypercube sampling (LHS)[13] is done as the initial sampling in the design space. N is chosen as the number of initial LHS points. If the geometry parameterization has any inexpensive constraints, the LHS is combined with a constraint handling technique to obtain geometrically feasible shapes.
3. The objective $\boldsymbol{f}$, constraint $\boldsymbol{g}$ and $\boldsymbol{h}$ are evaluated using true function evaluations (e.g., CFD, etc).
4. The so-far obtained solutions are compiled in a database containing $\boldsymbol{x}$ and expensive-to-evaluate functions from $\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}$ to approximate.
5. ANN based surrogate model is constructed by training using design database. An approximate model $\hat{\boldsymbol{f}}(\boldsymbol{x}), \hat{\boldsymbol{g}}(\boldsymbol{x}), \hat{\boldsymbol{h}}(\boldsymbol{x})$ that can do analytical evaluation is obtained.
6. Multi-objective optimization is done using NSGA-II algorithm, coded in Python.[14] Any call to expensive evaluation from $\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}$ is replaced using the model $\hat{\boldsymbol{f}}(\boldsymbol{x}), \hat{\boldsymbol{g}}(\boldsymbol{x}), \hat{\boldsymbol{h}}(\boldsymbol{x})$. At the end of NSGA-II procedure, a set of non-dominated solution $\boldsymbol{x_{best}}$ is obtained which correspond to $\hat{\boldsymbol{f}}_{\boldsymbol{best}}$.
7. K-means algorithm[15] for clustering data is used as an infilling criterion to down-select K points from $\boldsymbol{x_{best}}$ to be used as the new design points. The clustering is done on the objective space.
8. If the computational budget is still available, proceed to Step 9. If not available, proceed to Step 10.
9. The new design points are prepared to be evaluated by the true evaluations. Step 3-8 are repeated.
10. A set of solutions $\boldsymbol{x_{last\_gen}}$ which correspond to the best N non-dominated solutions from the design database are obtained. They are then sorted to find solutions that lie on the first non-dominated front to approximate the Pareto-optimal front (POF)

Figure 2 illustrates the SBO procedures applied to the multi-objective ASO-TA problems.

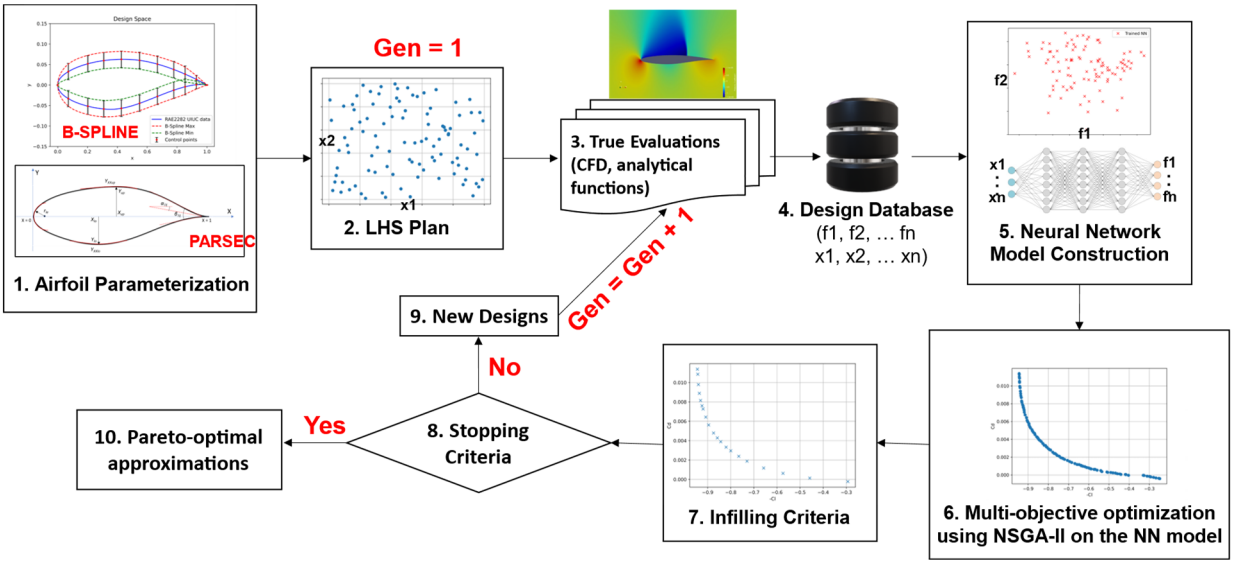


Fig. 2: Surrogate-based optimization procedure applied to multi-objective ASO-TA problems

III.B. Neural network configuration

This study uses a fully connected feedforward neural network with five layers consisting of an input layer, three hidden layers, and an output layer, shown in Fig. 3.

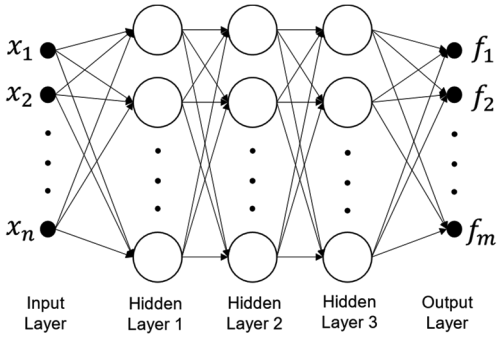


Fig. 3: Fully connected feedforward neural network

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 non-linear function (e.g., Shen et. al.[16]).

This study uses 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 avoids this saturation which often happens in a dense network. Equations 4 and 5 explain how they work.

$$ReLU(x) = \max(0, x) \quad (4)$$

$$LeakyReLU(x) = \max(0, x) + 0.01 \min(0, x) \quad (5)$$

III.C. Data treatment prior to training

Firstly, the design variable $\boldsymbol{x}$ and the objective functions $\boldsymbol{f}$ are normalized 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 $\max(\boldsymbol{f})$ of the current sample dataset are the lower and upper boundary for $\boldsymbol{f}$. This process can speed up the training.[17]

Secondly, Euclidean distances between samples in the design variable domain are calculated. If there are two samples that have a distance < 0.001, one of them will be deleted. Lastly, the *K*-means algorithm is used on the $\boldsymbol{x}$ domain to cluster the samples into *K* clusters. The value of *K* is chosen using a method called gap statistics[18]. 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 belongs to a cluster and n_l is the number of training data that belongs to l-cluster. Then, the sample points that belong to l-cluster are duplicated $round(N/n_l)$ times. The last two steps prevent the network from adding more weights to the crowded samples, leading to overfitting them.

III.D. Training the neural network

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 chosen randomly from the design database with ratio 4:1.

The neural network model is constructed by training it using the training set. 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 initially random weight values predicts the output. In the back propagation phase, 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,

$$w_{k,l} := w_{k,l} - \eta \frac{\partial E(\boldsymbol{w})}{\partial w_{k,l}} \quad (6)$$

where $w_{k,l}$ represents the weight value that connects neuron k in a layer and neuron l in the next layer, η is the learning rate, and $E(\boldsymbol{w})$ is the cost function. Adam[19], 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 $E(\boldsymbol{w})$.

This study adopts a mini batch gradient descent method 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 then 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. The training is stopped whenever this sign is observed. If not observed, the training progresses, indicating an epoch. The maximum epoch is set to 2000.

IV. Transonic Airfoil Shape Optimization

We aim to apply our proposed algorithm (NN+GA) introduced in Section III to several ASO-TA problems. The same problems are also solved using standard NSGA-II algorithm with different configurations. The obtained results are compared to investigate the efficacy of the NN+GA as well as to show its superiority over the standard NSGA-II in this real-world problem.

IV.A. Airfoil parameterization

We use two techniques to parameterize the airfoil: PARSEC[20] and B-Spline, illustrated in Figure 4. All airfoils in this study have a sharp trailing edge. The nine design variables for the PARSEC airfoil are listed in Table 1. For the B-Spline airfoil, the 18 control points are selected from the coordinates of RAE2822 (the baseline) that are listed in Table 2.

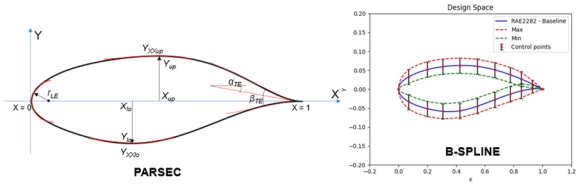


Figure 4. Airfoil parameterization

Table 1: The boundaries of PARSEC variables

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 2: The boundaries of B-Spline control points

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

IV.B. Computational fluid dynamics

We use CFD to evaluate the aerodynamic performance (i.e., C_d and C_l) of the transonic airfoil shape in a 2D inviscid flow which was solved by SU2 open-source code.[21] Using the inviscid Euler solver is not realistic to simulate real-world aerodynamics with viscosity and thermal conductivity. Nevertheless, it is cheap and allows us to perform numerous function evaluations in the present numerical experiments to compare the efficacy between optimization algorithms.

IV.C. Problem definitions

The complexity of a problem is influenced by the dimensionality, the governing physics, the presence of constraints, etc. 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 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 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 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. All problems have two expensive objective functions. 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 prevents the airfoil from going too slender compared to baseline, and the trailing edge constraints ensure geometrically feasible shapes.

V. Results and Discussions

In this section, we present the computational results. To ensure the computational accuracy, we first conduct grid convergence study (GCS). We then apply the NN+GA and NSGA-II to the three ASO-TA problems. Finally, we compare the results using the HV metric.

V.A. Grid convergence study

The CFD results are dependent on the grid resolution. Thus, we should conduct GCS to decide our mesh configuration so that our aerodynamic values of interest are grid independent. In this study, 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 $\alpha = 2^o$, $M = 0.73$. We used a standard C-grid topology and surveyed five types of grid resolution. The GCS results are shown in Table 3 and in Figure 5.

Table 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

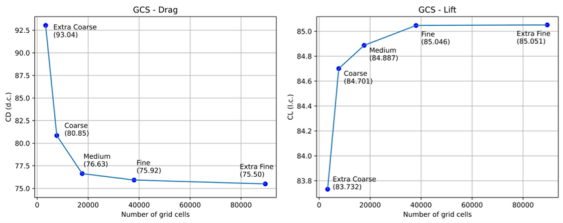


Fig. 5: Grid convergence for C_d and C_l

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 47 seconds for the extra fine grid. The CFD was done on an Intel(R) Xeon(R) CPU E5-1630 v4 3.70 GHz with 4 cores. Based on this result, we decide to use the fine grid. Figure 6 shows an example of the structured grid used in this study. All the meshes are created using Pointwise, a commercial meshing software.

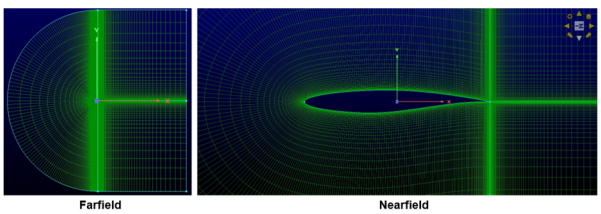


Fig. 6: An example of the fine grid used in this study.

V.B. Optimization algorithms

We use NN+GA algorithm and three different configurations of NSGA-II algorithm 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 network based on these 100 initial samples. The training parameters are listed in Table 4. NSGA-II with the configuration listed in Table 5 is used with the NN model replacing all the expensive function evaluations (C_d and C_l). After obtaining 100 optimized samples predicted by the NN+GA, *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 NN+GA can be written as follows:

$$N_{CFD} = 100 + N_{infilling} * (n_{gen} - 1) \qquad (7)$$

Table 4: The training parameters

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

Table 5: Parameter values for NSGA-II inside the NN+GA

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

In the second algorithm, NSGA-II is used with a population size of 100 that participates in the genetic process, producing the next 100 sample points to be evaluated. Due to the 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

sample points to be evaluated. Since we have to start with the same 100 initial LHS 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 only 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 can be written as follows:

$$N_{CFD} = 100 + N_{pop} * (n_{gen} - 1) \qquad (8)$$

The parameters for the second, third, and fourth algorithm are listed in Table 6.

Table 6: Parameter values for the three NSGA-II algorithms

	The second algorithm	The third algorithm	The fourth algorithm
Pop size for prob 1 & 2	100	20	20
Pop size for prob 3	100	10	10
Max n_{gen} for prob 1 & 2	10	11	11
Max n_{gen} for prob 3	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 $\boldsymbol{f}$

The HV indicator is used as the performance metric for each optimization. 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 was done three times). Thus, the HV value is averaged among three optimization runs.

V.C. Results of ASO-TA1

Figure 7 shows the average HV history for all algorithms performance in ASO-TA1. It basically shows how the HV value progresses as the number of true evaluations increases. Since the 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 7, that 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.

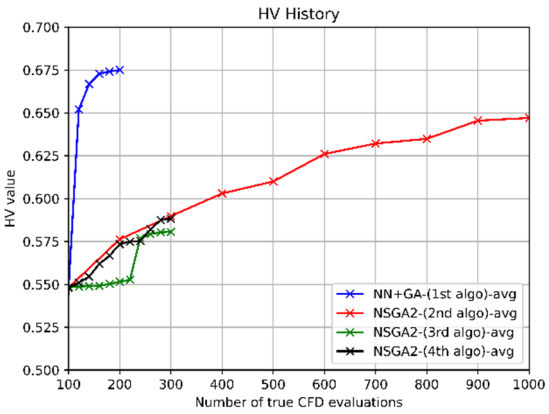
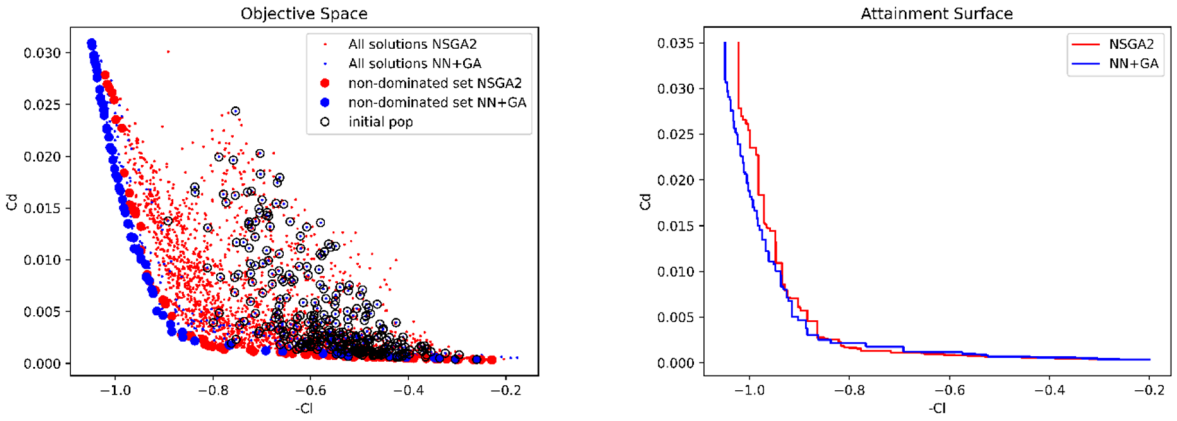


Fig. 7: Average HV history for ASO-TA1

The NSGA-II 3rd and 4th algorithms 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 $N_{pop} = 20$, as in the third and fourth algorithm, there is no significant improvement that can make them compete with NN+GA.

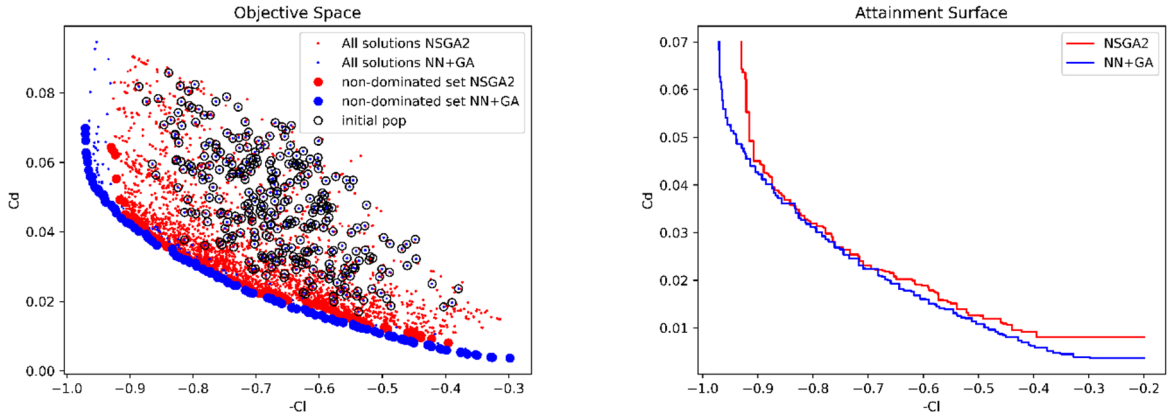
These results are visualized in Figure 8 that shows the plot of all solutions found by NN+GA and NSGA-II (2nd algo) and the attainment surface of their non-dominated sets. The plot of the initial population and the non-dominated set indicates the complexity of the problem. From Figure 8(a), we can observe that most of the initial population lie on 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 13-15. Note that the low-drag airfoils have less intense shock, shown in the pressure contours, which in turn reduces the wave drag.



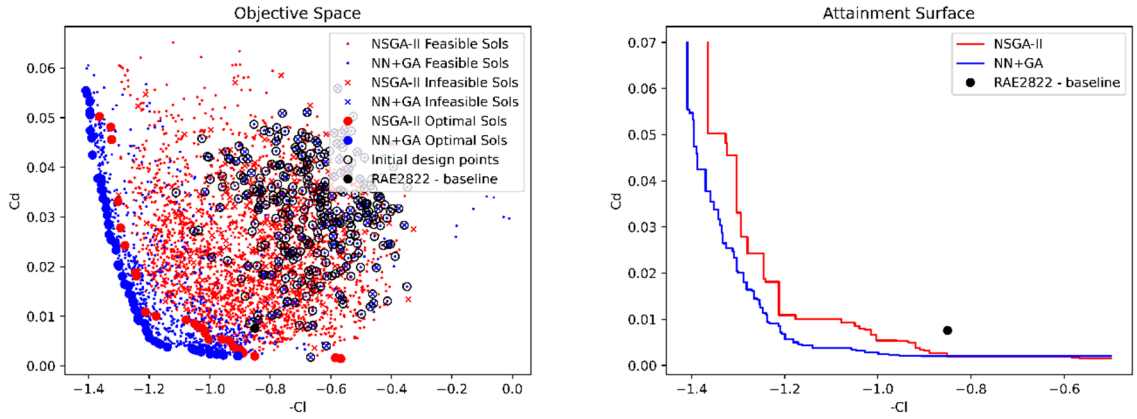
(a) All solutions (b) Attainment surface

Fig. 8: Plot in the objective space for ASO-TA1 solved by NN+GA and NSGA-II (2nd algo)



(a) All solutions (b) Attainment surface

Fig. 9: Plot in the objective space for ASO-TA2 solved by NN+GA and NSGA-II (2nd algo)



(a) All solutions (b) Attainment surface

Fig 10: Plot in the objective space for ASO-TA3 solved by NN+GA and NSGA-II (2nd algo)

V.D. Results of 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 of the latter compared to the former. This is indicated in the plot of the initial population and the non-dominated set (Figure 9(a)). In ASO-TA2, 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 13 and 14. This

shock wave induces higher wave drag, increasing the drag coefficient C_d. This is why the initial population in ASO-TA2 does not lie in the low C_d region as in ASO-TA1.

Figure 11, 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. 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 the NN+GA.

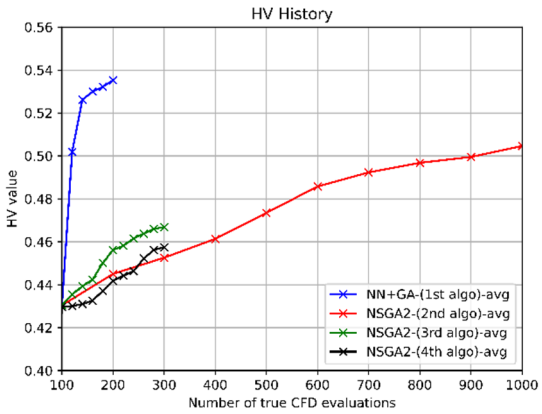


Fig. 11: Average HV history for ASO-TA2

V.E. Results of ASO-TA3

Unlike the PARSEC variables that ensure the smoothness of the airfoil, the B-Spline control points offer much more flexibility that allows the creation of rough surfaces. The existence of constraints also makes some obtained solutions infeasible. Thus, ASO-TA3 is the most complex problem among the three problems. It can be observed in Figure 10(a) that the initial population is located far away from the PO and some solutions are infeasible. It is expected to be difficult for the optimizer to find the POF. That is why we defined 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 (2nd algo). Both algorithms can find better solutions than the baseline.

Figure 12 corroborates the superiority of NN+GA over standard NSGA-II even in a more complex problem. We can observe from Figure 13-15, that the shock waves seem to be not realistic and too intense. It is because of the use of Euler solver, which in its nature, is not realistic and this nature is exploited by the optimizers that have no information about the physics.

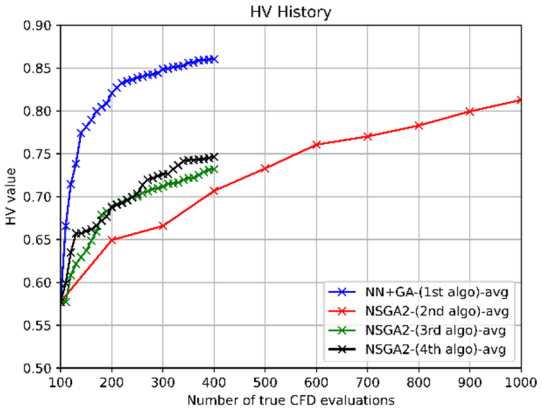


Fig. 12: Average HV history for ASO-TA3

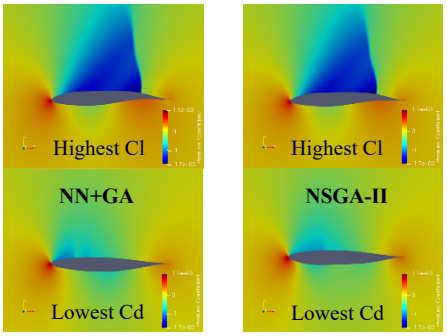


Fig. 13: Pressure contours of two extreme solutions found by NN+GA and NSGA-II on ASO-TA1

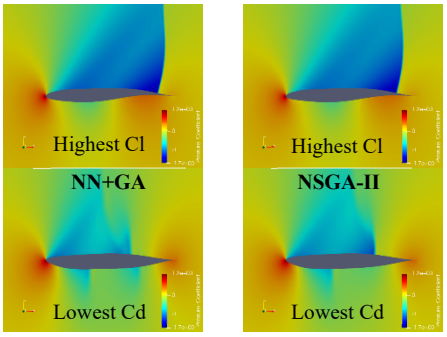


Fig. 14: Pressure contours of two extreme solutions found by NN+GA and NSGA-II on ASO-TA2

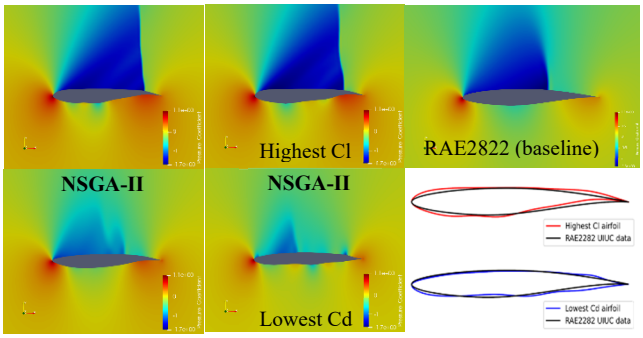


Fig. 15: Pressure contours of two extreme solutions on ASO-TA3 and the baseline with their geometries

Although this study focuses on the comparison between optimization algorithms (not on the CFD results) and only uses Euler solver, it is sufficient to say that the use of ANN makes the GA more efficient. This finding, however, must be further justified in high-fidelity CFD.

VI. Conclusion and Future Works

In this study, we proposed a state-of-the-art surrogate-based optimization methodology called NN+GA that uses an artificial neural network-assisted genetic algorithm. The NN+GA and the standard NSGA-II were applied to three multi-objective aerodynamic shape optimization of transonic airfoil problems with different complexities. The NN+GA was shown to converge more quickly towards the POF compared to the standard NSGA-II in all problems.

Although we only used problems with low (<10) and moderate (10-50) dimensionality, the NN+GA algorithm has the potential to be used in multi-objective optimization problems that have many design variables. We are planning to use the proposed method in a problem with much higher dimensionality (>100) and high-fidelity CFD.

References

[1] Abbot, I. H. and Doenhoff, A. E. V., "Theory of Wing Sections Including a Summary of Airfoil Data," *Dover Books on Aeronautical Engineering*, 1959, p. 2.

[2] Holland John, H., "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence," *USA: University of Michigan*, 1975.

[3] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182-197.

[4] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., Tucker, P. K., "Surrogate-based Analysis and Optimization," *Progress in Aerospace Sciences*, 2005.

[5] Palar, S. P., Zuhal, L. R., Liem, R. P., Shimoyama, K., "On The Use of Surrogate Models in Engineering Design Optimization and Exploration: The Key Issues," *GECCO '19 Companion*, 2019.

[6] Krige, D. G., "A Statistical Approach to Some Mine Valuation and Allied Problems on the Witwatersrand: By DG Krige," 1951.

[7] Lam, R., Poloczek, M., Frazier, P., Willcox, P., "Advances in Bayesian Optimization with Applications in Aerospace Engineering," *AIAA Non-Deterministic Approaches Conference*, 2018.

[8] Zuhal, L. R., Amalinadhi, C., Dwianto, Y. B., Palar, P. S., Shimoyama, K., "Benchmarking Multi-Objective Bayesian Global Optimization Strategies for Aerodynamic Design," *AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018.

[9] Kantardzic, M., "Data Mining: Concepts, Models, Methods, and Algorithms, 3rd Edition," *Wiley-IEEE Press*, 2019, p. 233.

[10] Rumelhart, D. E., Hinton, G., E., and William, R. J., "Learning Representations by Back-propagating Errors," *Nature*, Vol. 323, No. 9, pp. 533-536, 1986.

[11] Amari, S., "A Theory of Adaptive Pattern Classifiers," *IEEE Transactions on Electronic Computers*, EC-16(3), pp. 299-307,1967.

[12] Deb, K., "Multi-objective optimization using evolutionary algorithms," Wiley, Chichester, 2001.

[13] McKay, M. D., Beckman, R. J., Conover, W. J., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, pp. 239-245, 1979.

[14] Blank, J., and Deb, K., "Pymoo: Multi-Objective Optimization in Python," *IEEE Access*, 2020.

[15] MacQueen, J., "Some methods for classification and analysis of multivariate observations,", *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, No. 14, pp. 281-297, 1967.

[16] Shen, Z., Yang, H., and Zhang, S., "Neural Network Approximation: Three Hidden Layers Are Enough," *Neural Networks*, Elsevier, Vol. 141, pp. 160-173, 2021.

[17] Ioffe, S., and Szegedy, C., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37, pp. 448-456, 2015.

[18] Tibishirani, R., Guenther, W., and Trevor, H., "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B*, Vol. 63, pp. 411-423, 2001.

[19] Kingma, D. P., and Ba, J. L., "Adam: A Method for Stochastic Optimization," *3rd International Conference for Learning Representations*, San Diego, 2015.

[20] Sobieczky, H., "Parametric airfoils and wings," *Recent Development of Aerodynamic Design Methodologies*, Springer, 1999, pp. 71-87.

[21] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., "SU2: An Open-Source Suite for Multiphysics Simulation and Design," *AIAA Journal*, Vol. 54, No. 3, 2016.