Tuning of Optimization Strategies

R.A. Van den Braembussche
von Karman Institute for Fluid Dynamics
Waterloose steenweg, 72
1640, Sint-Genesius-Rode

vdb@vki.ac.be

ABSTRACT

Present lecture describes some of the techniques that are available, to speed up the convergence to the optimum geometry when using Evolutionary Strategies. They are: optimum parameter setting of the evolutionary search mechanism, the use of a surrogate mode and improvement of the database. The different ways to accelerate the convergence are described and illustrated by some practical examples. A last section shortly discusses the influence of the coding technique

1.0 INTRODUCTION

One of the major problems in optimization is the large computer effort that is often required to reach an optimum geometry [1, 2]. One way to reduce it, is by improving the convergence of the search mechanism to the optimum geometry. An optimization of the parameters of the Evolutionary Algorithm (EA) may not only reduce the computational effort, i.e. the number of performance evaluations that are needed to find that optimum (efficiency), but also improve the value of the optimum (effectiveness).

Another way to reduce the problem is by using a metafunction or surrogate model to make a two level optimization (Fig.1). The EA is driven by surrogate models of the accurate solvers and only the optimized geometry is verified by the accurate analysis tools. Different metafunctions are available. Only a few of them will be presented and their characteristics discussed.

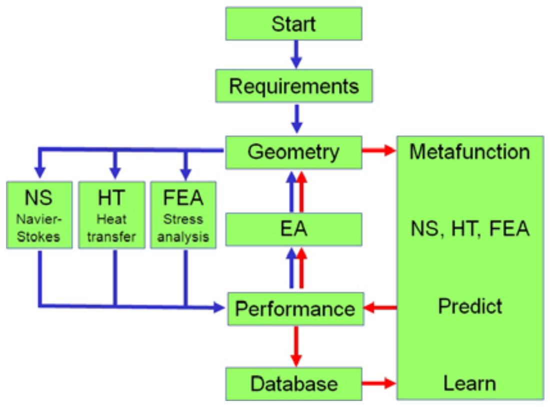


Fig. 1 Optimization scheme with metafunction

The accuracy of the metafunction strongly depends on the quality of the database used for the learning of it. The more general and complete this information, the more accurate the ANN can be and the closer the first optimized geometry will be to the real optimum. Hence a good database may considerably speed up the convergence to the optimum.

The way of coding the geometrical parameters may also have an impact on convergence

2.0 EA OPTIMAL PARAMETER SETTING

The advantage of using optimized parameters of the EA algorithm is illustrated here for the GA software developed by David L. Carroll [3]. It is known that the "optimum parameter setting" is problem dependent and that a correct setting has an impact on the convergence. The optimal parameters have been defined here by means of a systematic study on two typical design cases: one geometry defined by 7 parameters and one defined by 27 parameters [4]. These values result from the experience with typical turbomachinery optimizations. Conclusions are based on the solution quality Q, i.e. the degree to which the GA optimum approaches the real one within a given effort (5000 function evaluations). It is defined by:

$$Q = \frac{OF_{AV} - OF_{GA}}{OF_{AV} - OF_{\min}} 100\%$$

where: OF_{AV} is the average of the objective function over the complete design space, OF_{min} is the global minimum value of the objective function obtained from a systematic (numerically very expensive) scanning of the whole design space, OF_{EA} is the minimum value of the objective function obtained from the EA optimization. A Q value of 100% indicates that the GA has found the global minimum value.

The function evaluations for the numerical experiments are made by means of an ANN approximation of the NS solver based. Other possibilities to verify the optimum GA parameter setting is by means of an analytical test function as shown on Fig. 2.

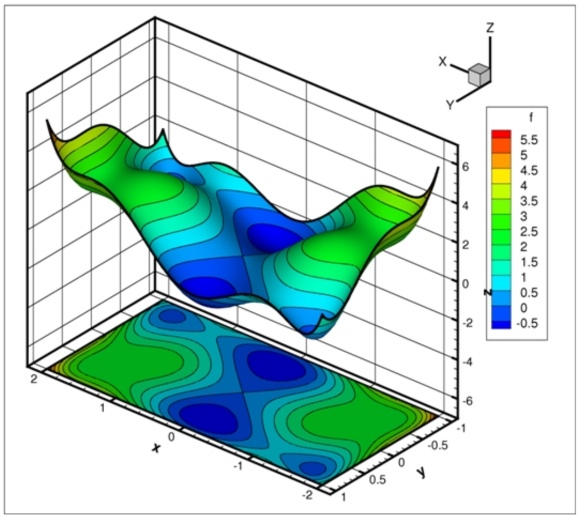


Fig. 2 Six hump camel back test function

2.1 Optimum Substring Length

In a standard binary-coded GA, the *n* real-valued design parameters x_i, defining a geometry, are jointly represented by one binary string:

$$\underbrace{1101\ldots0}_{x_1}\ \underbrace{1001\ldots1}_{x_2}\ \underbrace{0011\ldots0}_{x_3}\ \ldots\ldots\ldots\ \underbrace{0101\ldots1}_{x_n}$$

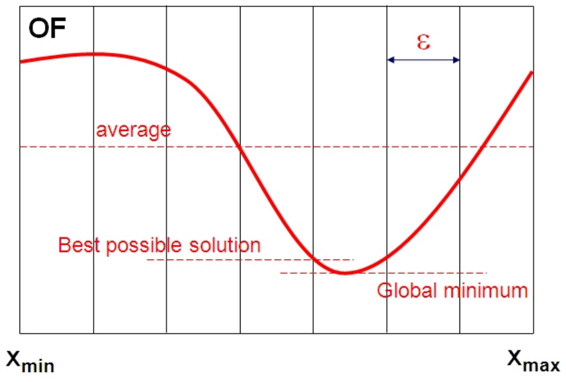


Fig. 3 Impact of substring length

The substring length, denoted by *l* (number of bits per variable), determines the total number of values (2^l) that each design parameter can take. Fig. 3 shows how the minimum substring length l_i for the i^{th} design variable depends on the upper and lower bound respectively x_{min} and x_{max}, as well as on the desired resolution (ε_i) for this variable:

$$l_i = \log_2 \frac{(x_{i,\max} - x_{i,\min})}{\varepsilon_i}$$

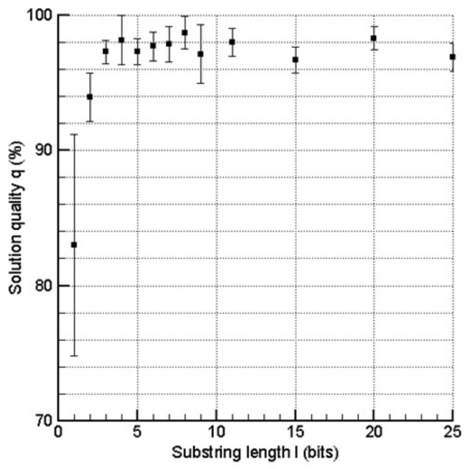


Fig. 4 Impact of substring length on solution quality

Very short substrings (l< 3) result in a too low resolution and the GA may not be able to accurately approach the global minimum. Longer substrings ($3 < l < 10$) enable a sufficiently high resolution but cause a larger search space and make it difficult to find the complete optimal binary string. Systematic testing has shown that $l = 5$ is the optimum substring length independent of the number of unknown parameters (Fig. 4).

2.2 Selection scheme

Different selection schemes can be used. The roulette and tournament selection have been presented in another lecture [1]. The criteria for the selection of the main parameters relate to the convergence rate and the risk to get stuck in a local minimum.

2.3 Population size

Fixing the total number of function evaluations at 5000, the number of generations t is a consequence of the population size N ($N*t = 5\ 000$). Fig. 5 shows the evaluation of the solution quality at the end of the GA run for different values of the population size. The solution quality is maximum for $N = 11$ to 20. Small populations (N<10) converge prematurely to suboptimal solutions, due to a lack of diversity and high performing samples in the initial population. Larger populations (N>25) have a sluggish convergence to the optimal geometry because less generations are allowed

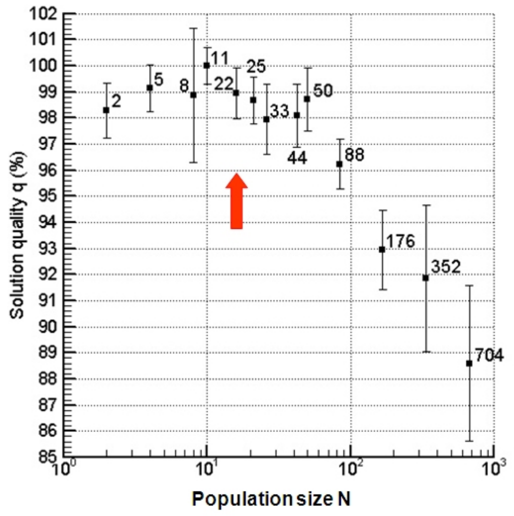


Fig. 5 Dependence of GA solution quality on population size for the 27 parameter test case

2.4 Crossover probability

In a single-point crossover operator, both parent strings are cut at a random place and the right-side portions of both strings are swapped. In case of a uniform crossover, the value of p_c defines the probability that crossover is applied per bit of the complete parent string. High values of p_c increase mixing of string parts and at the same time, increase disruption of good string parts. Low values limit the search to combinations of samples in the existing design space. Experiments confirm that a single point crossover is optimal (Fig. 6).

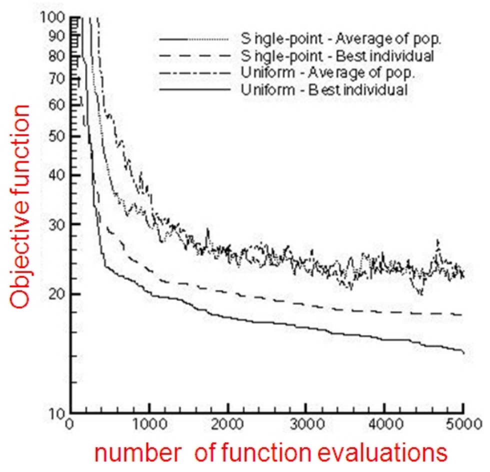


Fig. 6 Impact of cross over on convergence

2.5 Mutation probability

The mutation operator creates new individuals by changing a “1” to a “0” or vice versa in the off-spring string. The mutation probability p_m is defined as the probability a bit of a string is flipped. Systematic numerical experiments confirm that the optimum setting for the mutation probability is $p_m=1/(l.N)$ for all optimizations (Fig. 7). This corresponds to changing on average only one bit at every generation.

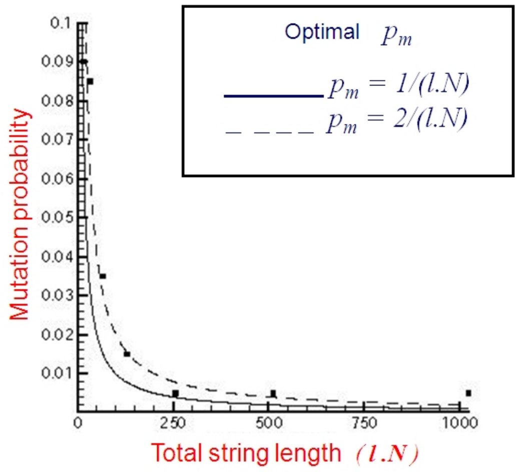


Fig. 7 Optimal mutation probability

2.6 New generation

The convergence is also influenced by the way the new generation is defined. The *(N,ch)* definition means that the *N* best of *ch* offspring's replace the old population. In this procedure the best individuals of the previous generation are lost, even if they are better that the best of present generation.

The *(N+ch)* generation means that the *N* best of the *ch* offspring's and the *N* members of the old population replace the old population (elitism). Members of the old and new generation are in full competition. This system has the characteristic of elitism and favors the convergence. Ther best samples are never lost.

The *(N/i+ch)* generation definition means that only the *N/i* best of the previous generation contribute to new generation. This limits the elitism.

Fig. 8 shows how an optimization of the GA parameter settings can lead to a more efficient and more effective GA convergence.

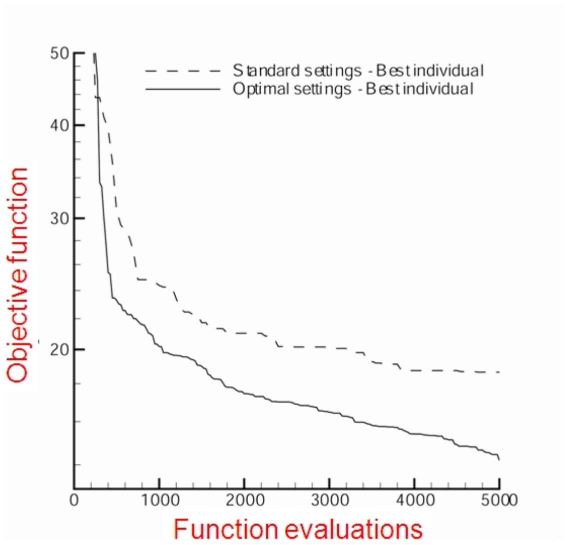


Fig. 8 GA convergence for a 27 parameter test case (standard versus optimized parameter setting).

3.0 METAFUNCTIONS

Meta-functions need to be fast (to limit the time required to analyze a large number of samples) but also accurate to drive the optimizer to the real optimum. Different types of meta-functions have been proposed. Only a few of them (ANN, RBF, Kriging) are presented here.

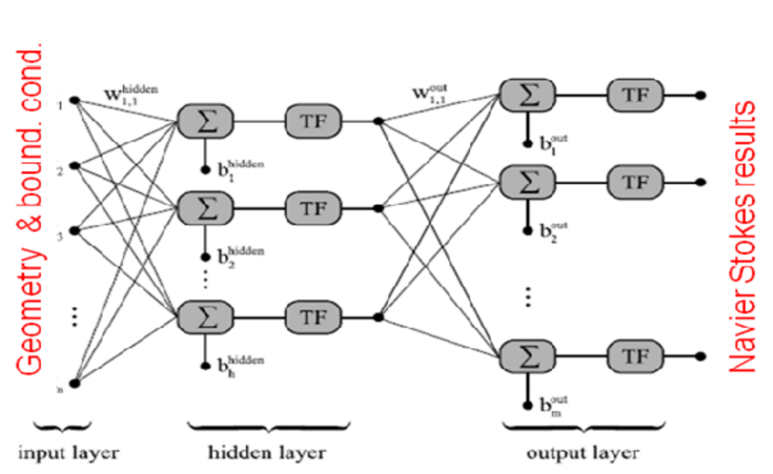


Fig. 9 Artificial Neural Network architecture

An ANN (Fig. 9) is composed of several elementary processing units called neurons or nodes. These nodes are organized in layers and joined with connections (synapses) of different intensity, called the connection weight (W) to form a parallel architecture. Each node performs two operations: the first one is the summation of all the incoming signals and the second one is the transformation of the signal by using a transfer function (*TF*) after a bias b_i has been added.

$$a_1(i) = TF_1\left(\sum_{j=1}^{n} W_1(i,j)\ p(j) + b_1(i)\right)$$

Sigmoidal functions $TF(x) = \dfrac{1}{1+e^{(-x)}}$ are mostly used as transfer function (Fig. 10). These functions introduce power series (given implicitly in the form of an exponential term) and do not require any hypotheses concerning the type of relationship between the input and the output variables. In order to avoid saturation of the function, it is important to verify that the variation takes place in the central non zero slope part of the curve.

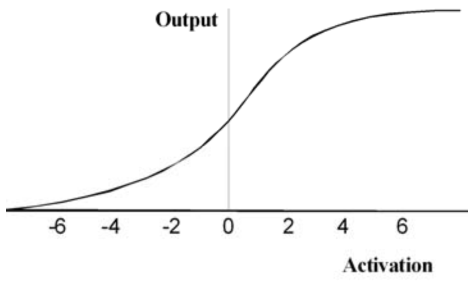


Fig.10 Sigmoid activation function

The coefficients (weights and bias) are defined by a LEARNING procedure relating for all the samples of the database the output (performance i.e.: η, β_2 and the Mach number distribution M_i $,i=1,n_M$ to the input (boundary conditions and geometry parameters x_i $,i=1,n$). The purpose is not to reproduce the existing database with maximum accuracy but to predict the performance of new geometries i.e. to generalize. This is illustrated on Fig. 11 showing how overfitting can result in a higher degree curve with large oscillations between the input data (x) and predict incorrect values (o) for the intermediate and extrapolated points (generalization).

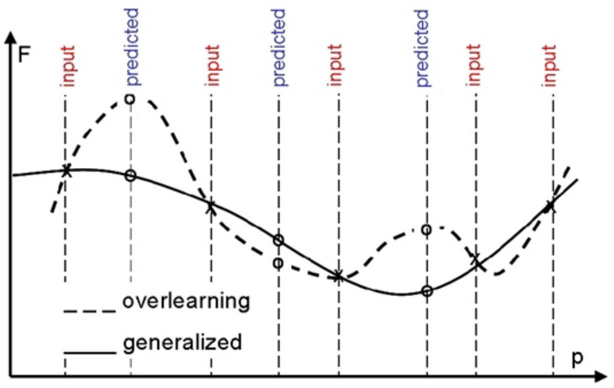


Fig. 11 Properly generalized ___ and overfitted _ _ _ data

Three conditions are necessary, although not sufficient, for good generalization [5]:

- The first one is that the inputs to the ANN contain sufficient information pertaining to the target, so that one can define a mathematical function relating correctly outputs to inputs with the desired degree of accuracy. The designer should select design parameters that are relevant, i.e. that have an influence on performance
- The second one is that the function to be learned (relating inputs to the correct outputs) is smooth and well defined. In other words, a small change in the inputs should, produce a small change in the outputs. Most physical problems are well defined in this respect.
- The third one requires that the training set is a sufficiently large and a representative sample of all cases that one wants to generalize (the "population" in statistical terminology). One should avoid that the ANN predictions are extrapolations instead of the more accurate interpolations. As will be shown later this condition is more or less taken into account when defining the database.

Generalization of the ANN learning is favored by dividing the available samples into "training", "test" and "validation" sets. Each of them has its own purpose.

The **Training set** contains the samples used for LEARNING; that is to fit the parameters (i.e., weights and bias) of the classifier.

The **Test set** contains the samples used only to assess the performance (generalization) of a fully-specified ANN (given weights and architecture).

The **Validation set** contains the samples used to tune the parameters (i.e., architecture, not the weights) of a classifier, for example to choose the number of hidden units in a neural network.

The learning process results in a rapid initial decrease of the training set error and then continues to decrease slowly as the network makes its way to a minimum on the error surface (Fig. 12). Good generalization can be achieved by the use of cross-validation by the test set. In this procedure, the training is periodically stopped (i.e., every so many training epochs), and the network is tested on the Test Set. The learning is stopped when the minimum error of the Test Set is reached.

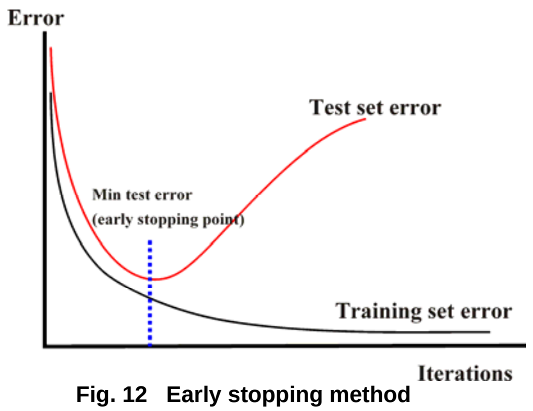


Fig. 12 Early stopping method

A complete design cycle including ANN learning, GA optimization and verification by a Navier Stokes solver requires typically 35% more time than a Navier Stokes analysis.

The Radial Basis Function (RBF) network is a three layer network with a non-linear mapping from the input layer to the hidden layer and a linear mapping from the hidden layer to the output layer (Fig. 13). The hidden neurons are associated with the so-called RBF centers, which are points in the *n*-dimensional space.

The output of each hidden neuron is computed by a Gaussian function

$$h_i = \exp\left(-\frac{\left\| \vec{x} - \vec{r}_i \right\|^2}{2\sigma_i^{\ 2}} \right)$$

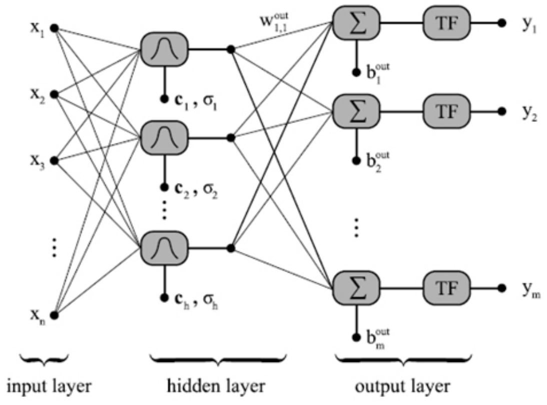


Fig. 13 Radial Basis Function Network Topology

The vector $\vec{x}$ represents the input to the neuron, the vector $\vec{r}_i$ is the RBF center and σ_i is the amplitude of neuron *i*. The $\left\| \ \right\|$ operator computes the Euclidian distance between the input vector $\vec{x}$ and the RBF center $\vec{r}_i$ (Fig. 14). The output of an RBF neuron is thus proportional to the distance between the input and the RBF center. The amplitude σ_i determines the activation range, i.e. the distance over which the neurons are active and have a significant output. The weight factor of a sample decreases with increasing distance from the RBF center. The weighted sum of all outputs of the hidden neurons is the RBF output:

$$Y_i = \sum_{i=1}^{K} \omega_i . h_i + b_i$$

Training of the RBF network consists of finding the RBF centers $\vec{r}_i$, the amplitude σ_i and the weight ω_i for each neuron, such that the error on the prediction of the database the samples is minimal. The resilient backpropagation (RPROP) [6] is used to train the RBF.

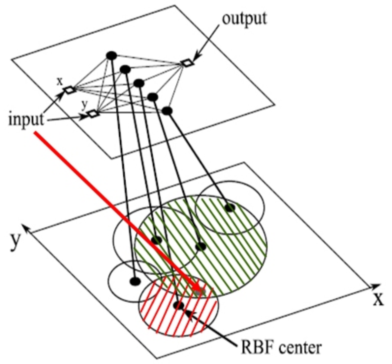
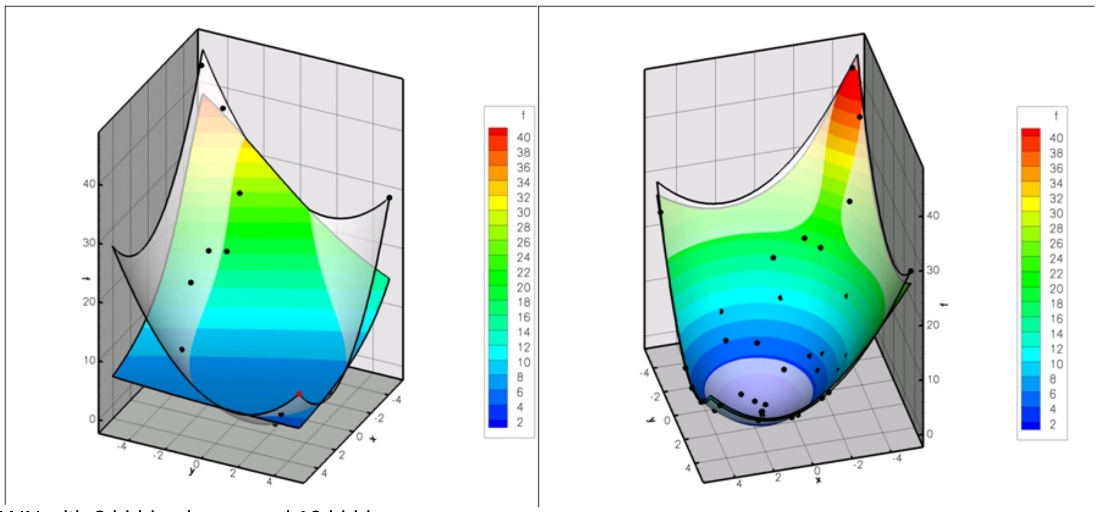


Fig. 14 A 2D RBF interpolation (network mapped on RBF space)

The accuracy of the Metafunctions (ANN and RBF) depends on the information (number of samples and distribution of samples) stored in the database and on the structure of the networks (number of hidden layers and nodes per layer) as illustrated in Fig. 15.



ANN with 2 hidden layers and 10 hidden neurons RBF with 1 hidden layer and 5 hidden neurons

Fig. 15 Comparison between ANN and RBF predictions of De Jong 2D test function

Both the ANN and the RBF have been used for the multi-objective optimization of the cooling holes in a HP turbine [1] where the comparison between the approximate and the accurate predictions are shown on Fig. 34). Both optimizations are run for 30 iterations, after which a synchronization of the databases is made i.e. all existing samples are put together in one unified database. An additional 20 optimization iterations are then performed restarting with this extended database.

Figure 16 shows a different convergence for both metafunctions during the first 30 iterations and after merging. The RBF (open circles) shows for the first 30 iterations an almost straightforward convergence (red line) to an optimum whereas the ANN shows a more scattered convergence to a better optimum (lower value of the *OF*). Continuing the optimization on the extended database (iteration 31-50) shows no further improvement for the ANN (same *OF*). However the RBF shows a further improvement of the optimum to the same level as the ANN one. It is believed that this is the consequence of a more monotonic convergence of the RBF towards the optimum by giving more weight to the database geometries that are closer (within the activation range) to the geometry for which the performance (lifetime and mass flow) has to be predicted. This results in a more accurate prediction in the region near to known solution which is similar to what happens in a local gradient method. However this increases the risk of getting stuck in a local optimum. Extending the database by adding the more randomly distributed geometries defined by the better converged ANN reduces that risk and allows reaching a better optimum.

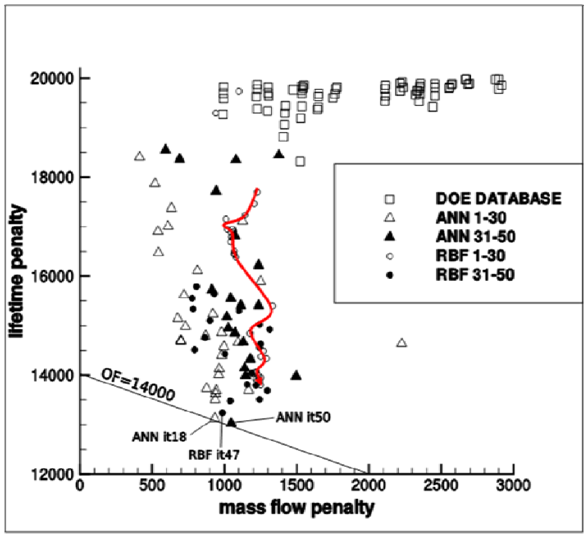


Fig. 16 Pareto front

Kriging is a model developed by geologists to estimate the concentration of minerals based on very scarce data that are available [7, 8]. The main advantage of this technique is the estimation of the uncertainty of the prediction as illustrated on Fig. 17 This allows a better judgment of the geometry proposed by the optimizer. A detailed evaluation is recommended for optimal geometries (low *OF*). The same applies for geometries with a high *OF* and high uncertainty because it could well be that they are not as bad as predicted by the metamodel.

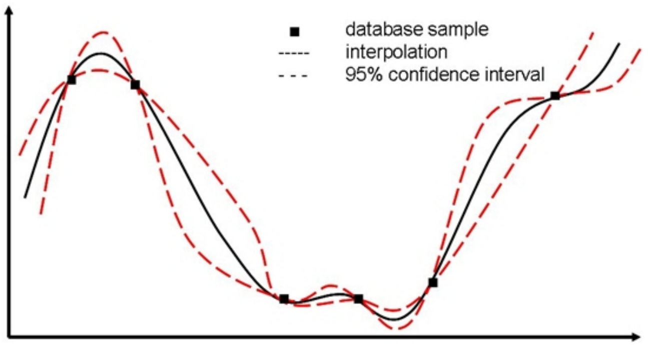


Fig. 17 1D Kriging interpolation with exact prediction of data-points and uncertainty between them

4.0 DATABASE

The main purpose of the DATABASE is to provide information to the metafunction about the relation between the geometrical changes and performance. The more general and complete this information, the more accurate may be the metafunction and the closer the first optimum geometry, defined by the GA, will be to the real optimum. Hence a good database may considerably speed up the convergence to the optimum.

Making a database is an expensive operation because it requires a large number of expensive performance analyses (NS, FEA etc.). Hence one is interested in making the smallest possible database containing the maximum amount of information about the whole design domain. This means relevant information with a minimum of redundancy, including the impact of every design parameter, but only once.

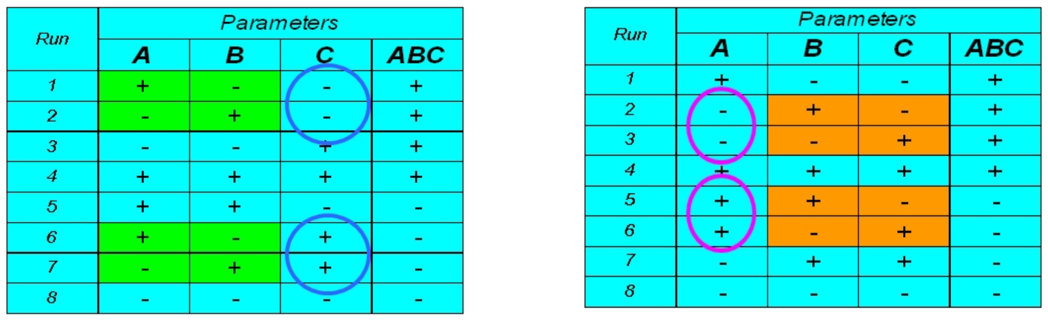
Any information missing in the database may introduce an erroneous metafunction that could drive the GA into a geometry that is not optimum. This is not a problem because the detailed performance analysis of that geometry will provide the missing information when it is added to the database. However the worst case is when an incomplete database results in an erroneous extrapolation by the metafunction predicting a low performance (large *OF*) in that part of the design domain where in reality the *OF* is low. As a consequence, the corresponding geometry will never be selected by the GA and the error may never been detected and corrected. This second shortcoming is more difficult to remediate because no mechanism is build-in to correct for it and the error may persist during the whole optimization. It is therefore important to assure that the initial database covers the whole design domain.

Design Of Experiment (DOE) refers to the process of planning an experiment so that the appropriate data, when analyzed by statistical methods, result in valid and objective conclusions. The advantages of using DOE to construct the database have been evaluated in detail in [9].

4.1 Factorial Design

Factorial designs are widely used in experiments where it is necessary to study the effect of the different factors on a response. The most important one is where each of the *k* factors has only two values corresponding to the “high” or “low” level of a design variable. A complete replicate of such a design requires 2^k observations/analyses and is called full factorial design.

Consider a design with 3 design parameters A, B and C. Each of them can be at two levels, indicated by +1 or –1 in Fig. 18. Evaluating all possible geometries requires $2^3 = 8$ Navier Stokes evaluations. The 2^k factorial designs with each design parameters at two levels, includes k main effects (A, B and C), $\frac{k!}{(k-2)!2!}$ two-factor interactions (AB, BC and AC), and one *k*-factor interaction (ABC) .



Run	Parameters			
	A	B	C	ABC
1	+	-	-	+
2	-	+	-	+
3	-	-	+	+
4	+	+	+	+
5	+	+	-	-
6	+	-	+	-
7	-	+	+	-
8	-	-	-	-

Run	Parameters			
	A	B	C	ABC
1	+	-	-	+
2	-	+	-	+
3	-	-	+	+
4	+	+	+	+
5	+	+	-	-
6	+	-	+	-
7	-	+	+	-
8	-	-	-	-

Fig. 18 Full 2^3 factorial design

For an increasing number of factors, the number of analyses required for a complete replicate of the design, rapidly outgrows the resources of most designers. It reaches 128 or even more than 10^7 runs for respectively the 7 and 27 parameter design space. Information on the main effects and low-order interactions can be obtained by running only a fraction of the complete factorial experiment if one can assume that certain high-order interactions are negligible or redundant with lower order interactions. This is illustrated on Fig. 18 showing that the same combination of A and B are repeated for different value of C. Idem for the combinations B and C but for different values of A. Hence some lines can be eliminated without losing much information.

Reducing the number of samples in the database reduces the information stored in it and hence the accuracy of the ANN based on that information. In what follows one will evaluate the loss of information by comparing the ANN predictions, based on different fractional factorial designed databases, with the exact values for the following test function with 6 variables:

$$R = 1 - 0{,}001\,(A-D)^3 + 0{,}002\,(C+E)(F-B) - 0{,}06\,(A-F)^2 + (F+C)(E+A)$$

The results of the databases defined by DOE are compared to those of a database in which the geometrical parameters are randomly generated between the prescribed boundaries. The values, attributed in present test to each of the 6 parameters A, B, C, D, E, and F are listed in Table 1.

Table 1 Variables and limits

variable	variable limits	
	high	low
A	5.	1.
B	3.	2.
C	5.	4.
D	4.	3.
E	3.	2.
F	6.	2.

The full factorial design requires $2^6 = 64$ runs to estimate all possible parameter combinations. The loss of information is measured by the following error term, expressing the difference between the exact function and the predictions by an ANN trained on the different databases. The error is defined by the following formula:

$$\text{Global error} = \sum_{i=1}^{n\_samples} \left(\left| \frac{\text{exact value} - \text{predicted value}}{\text{exact value}} \cdot 100 \right| : n\_samples \right)$$

The results of those calculations are shown on Fig. 19 . The number on top of each column indicates the global error obtained with the respective database.

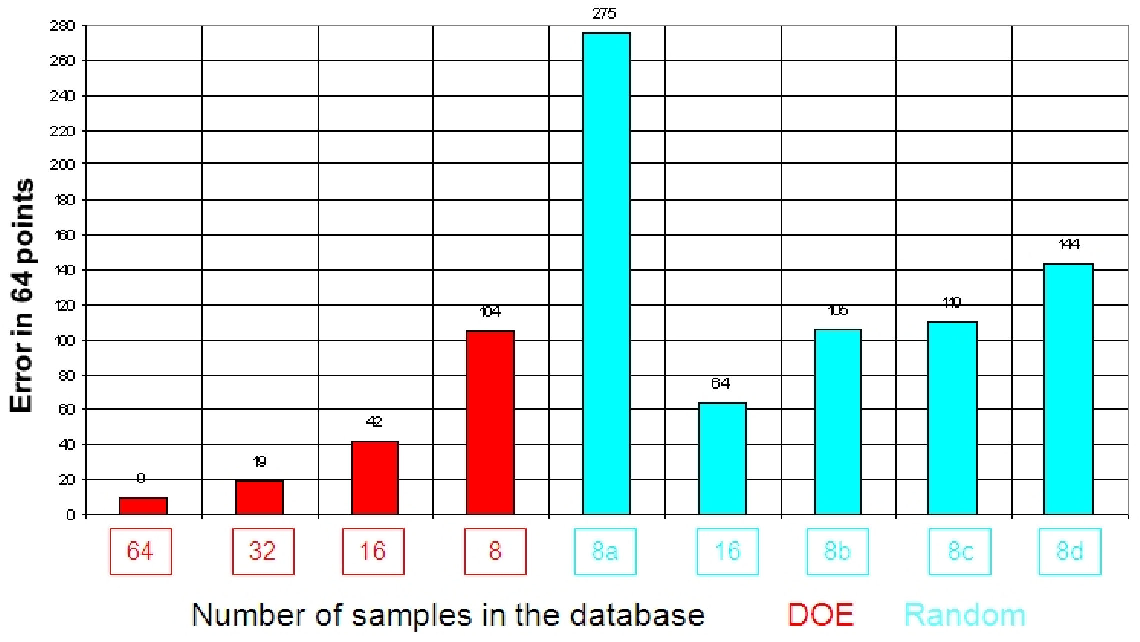


Fig. 19 ANN's global error for different number of training samples

Comparing the error obtained by means of the DOE technique and by means of randomly selected samples, it becomes clear that the DOE based predictions are consistently more accurate for the same number of samples in the database. This is particularly true if the predictions are based on 64 training samples resulting in a very high accuracy. The values obtained from the prediction based on 32 training samples, are still reasonably accurate. Less accurate results are obtained with 16 and 8 samples, respectively.

Randomly generated databases are all different and so is the accuracy of the ANN predictions. The four randomly generated cases with 8 samples in the database, show an error that varies between 105 (the same as with the DOE defined database) up to an error that is almost 3 times larger (First randomly generated 8 sample database).

The database quality can be further improved during the design process by adding geometries selected on the basis of a merit function. This intends to add information where the uncertainty is largest i.e. in those regions where the information is scarce. The merit function $m(x) = f(x) - \rho_m\, d_m(x)$ is minimized, in a way similar to the *OF,* when defining new geometries that should be analysed and added to the database (Fig. 20).

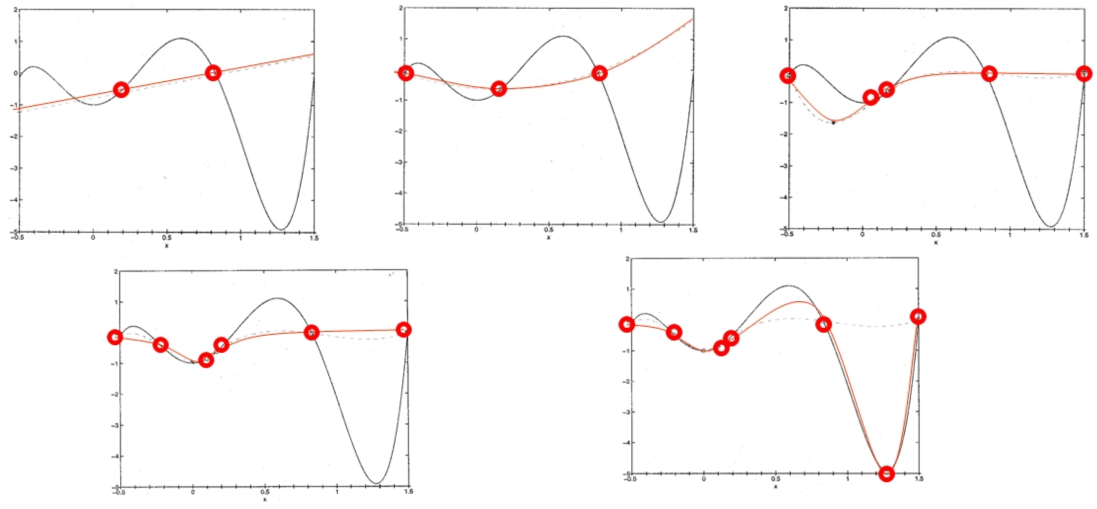


Fig. 20 Definition of new geometries based on merit function

5.0 CODING

Binary coding mimics the natural evolution whereby the digits can be assumed to stand for the genes. Other methods such as Multi-Objective-Differential-Evolution directly work with the discrete digital values. One possible disadvantage of binary coding is the fact that design parameters that are almost the same may show very large changes when binary coded (Fig. 21). The GA could consider them as very different geometries although they are very similar. Other coding techniques, such as gray coding, allow avoiding this. Experiments with both coding techniques did not show a noticeable difference between the two in terms of convergence.

Binary coding

Value	Code
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Gray coding

Value	Code
1	001
2	011
3	010
4	110
5	111
6	101
7	100

Fig. 21 Binary versus gray coding

6.0 CONCLUSIONS

It is shown how the computational effort of an optimization can be reduced by a tuning of the parameters of the GA and selecting an appropriate metafunction trained on a representative database and that this can be achieved without compromising on the quality of the optimum.

The availability of powerful parallel computers allows handling large computational effort and one could conclude from it that these acceleration techniques are less relevant. However one should keep in mind that the optimization jobs become more and more complex (multistage optimization) with an increasing number of design parameters. Any gain in computer capacity is quickly absorbed by the increasing complexity of the problems and an improved convergence remains attractive.

BIBLIOGRAPHY

[1] Thevenin D. and Janiga G., Optimization and Computational Fluid Dynamics, Springer, ISBN 978-3-540-72152-9.

[2] Van den Braembussche R.A., Global Optimization Methods - Theoretical Aspects & Definitions, in: Strategies for Optimization and Automated Design of Gas Turbine Engines, RTO-MP-AVT-176, paper .

[3] Carroll, D.L. (2001). FORTRAN Genetic Algorithm (GA) Driver version 1.7.1a. Available at URL: http://cuaerospace.com/carroll/ga.html.

[4] J. Harinck, Z. Alsalihi, J. P. Van Buijtenen, and R. A. Van den Braembussche. Optimization of a 3D Radial Turbine by Means of an Improved Genetic Algorithm. In Proceedings of the 6th European Conference on Turbomachinery, pages 1033–1042, Lille, 2005.

[5] Alsalihi Z., Van den Braembussche R. A., Theoretical Investigations, Improvements and Algorithmic Changes in Artificial Neural Networks, VKI IN 128, 2009.

[6] Amaral, S., Verstraete, T., Van den Braembussche, R. A., and Arts, T., 2008. “Design and Optimization of the Internal Cooling Channels of a HP Turbine Blade — Part I, Methodology”. In ASME. Paper No. GT2008-51077.

[7]   D. G. Krige. A statistical approach to some mine valuations and allied problems at the Witwatersrand. PhD thesis, University of Witwatersrand, 1951.

[8]   A. S. Goldberger. Best Linear Unbiased Prediction in the Generalized Linear Regression Model. Journal of the American Statistical Association, 57(298):369–375, 1962.

[9]   K. Kostrewa, Z. Alsalihi, and R. A. Van den Braembussche. Optimization of Radial Turbines by Means of Design of Experiment. In Tech. Rep. VKI-PR-2003-17, 2003.

[10] Price K. and Storm R., Differential Evolution, April 1, 1997.