NATIONAL TECHNICAL UNIVERSITY OF ATHENS (NTUA)
SCHOOL OF MECHANICAL ENGINEERING
PARALLEL CFD & OPTIMIZATION UNIT (PCOpt/NTUA)

Cost-Reduction in Gradient-Free Optimisation

Dr. Kyriakos C. Giannakoglou, Professor NTUA

Dr. Varvara Asouti, Adjunct Lecturer NTUA

Contributors:

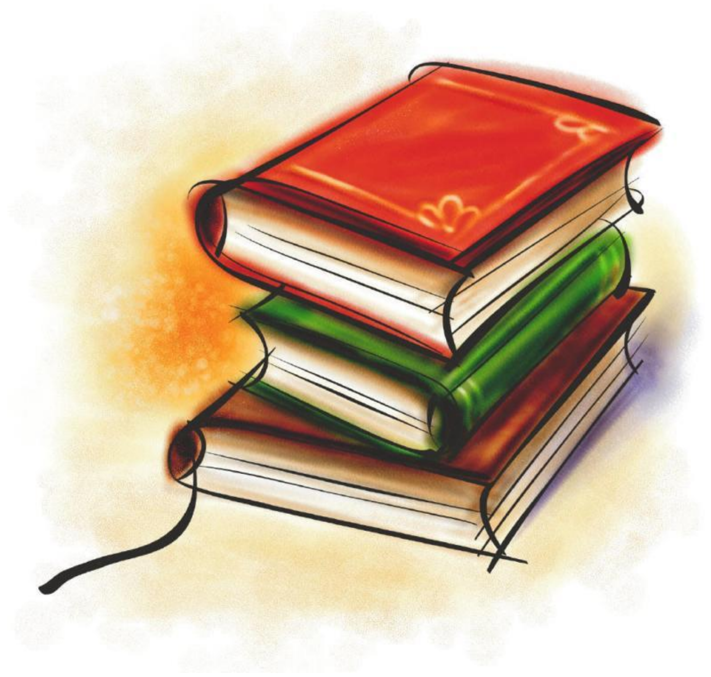
Dr. A. Giotis

Dr. M. Karakasis

Dr. I. Kampolis

Dr. S. Kyriacou

Dr. D. Kapsoulis



Introduction – Recap

Recap - Constrained Optimisation Problem Statement

$$\min \vec{F}(\vec{b}) = \min\left\{f_1(\vec{b}), \dots, f_{M_o}(\vec{b})\right\} \qquad \vec{b} \in \mathbb{R}^N$$

$$\text{subject to } c_j(\vec{b}) \le 0, \mathrm{j} = 1, M_c$$

N design (optimisation) variables
M_o objectives (**SOO** or **MOO**)
M_c constraints

<u>In practice:</u> To compute F_i or c_j, calls to the evaluation script are needed.
The computational cost of each evaluation is one time unit (TU); this is used to measure the cost of the optimisation loop as a whole, and compare different algorithms.

Optimisation Cost = # calls to the evaluation script * Cost per evaluation (1TU)

Recap – (μ,λ) EA Notations and Key Terms

Notations:

μ = parent population size

λ = offspring population size

e = elite population size

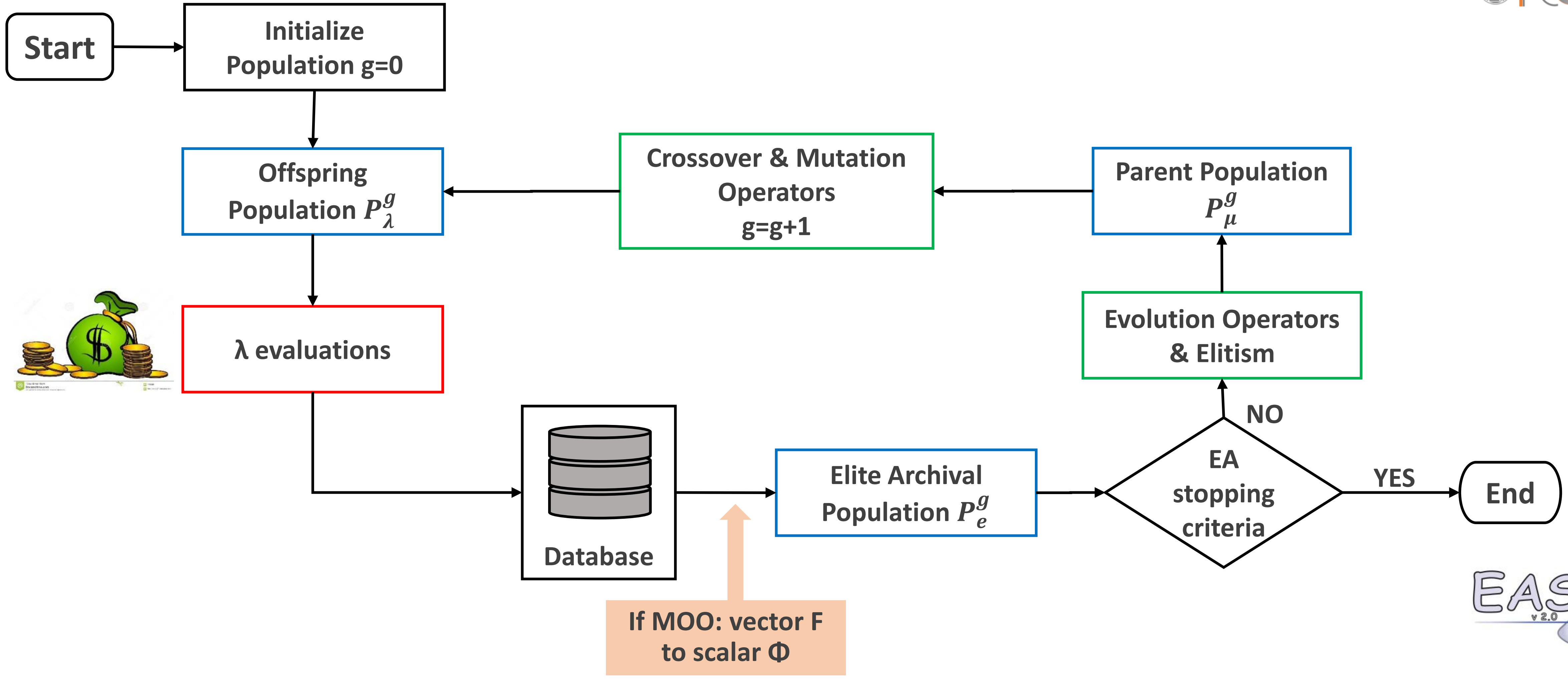
g = generation index

$$P_\mu^g,\, P_\lambda^g,\, P_e^g$$

Key Terms:

Individual:	**candidate solution**
Genes:	**design variables**
Chromosome (genotype):	**set of genes of an individual**
Objective function (phenotype):	**'physical'/observable characteristics of an individual**

Recap – Flowchart of a (μ,λ) EA


Start
Initialize Population g=0
Offspring Population P_λ^g
Crossover & Mutation Operators g=g+1
Parent Population P_μ^g
λ evaluations
Evolution Operators & Elitism
NO
Database
Elite Archival Population P_e^g
EA stopping criteria
YES
End
If MOO: vector F to scalar Φ

Stochastic, Population-Based Optimisation Methods. Pros & Cons

Pros:

- **Readily accommodate any analysis-evaluation software/script (as a black-box), to compute the cost or fitness function value(s) of candidate solutions.**
- **Gradient-free search.**
- **Compute (Pareto) front of non-dominated solutions, in many-objective optimisation (MOO), via a single run.**
- **Handle constraints in the simplest possible way: through penalties.**
- **Are amenable to parallelisation (simultaneous independent evaluations).**

Cons:

- **Require a great number of (costly/CFD) evaluations: many calls to the evaluation script!**

Possible ways to Reduce the Computational Cost of EAs

◆ Use **Metamodels** (or Surrogate Evaluation Models) to reduce the number of calls to the expensive **Problem Specific Model (PSM)** (CFD, CSM, CAA, etc. s/w) → **Metamodel-Assisted EAs (MAEAs)**.

◆ Perform Distributed search → **Distributed EAs or DEAs or DMAEAs**.

◆ Consider Dimensionality Reduction to overcome the «curse of dimensionality»→ **PCA-driven EAs or MAEAs**.

◆ Perform Hierarchical or Multilevel search → **Hierarchical EAs (HEAs or HMAEAs)**.

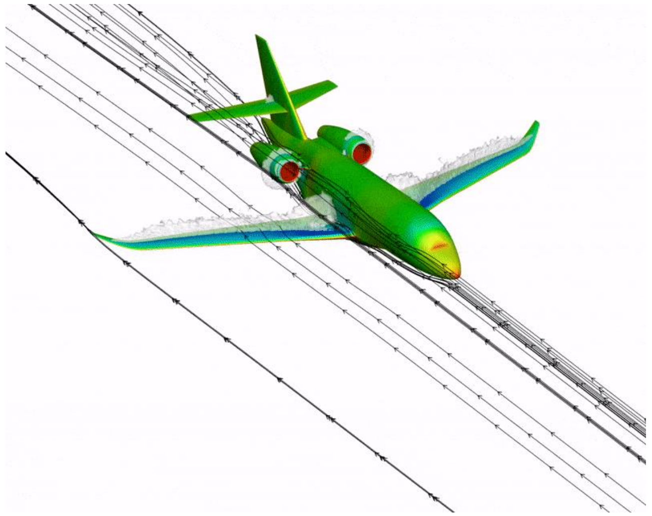
◆ Hybridise EAs with Gradient-based optimisation (GFM & GBM) → **Hybrid/Memetic Schemes**.

◆ Overcome the generation synchronisation barrier → **Asynchronous EAs**.

◆ Combine all/some of the above!

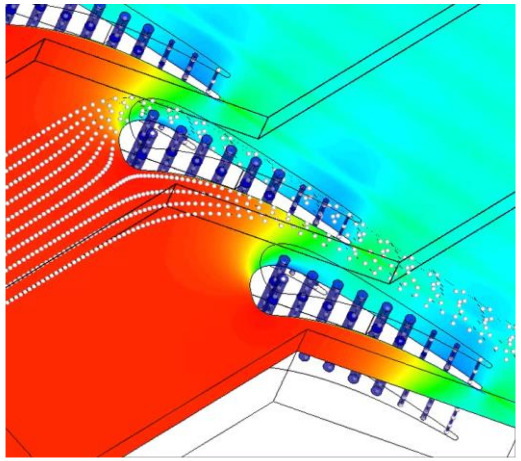
Problem Specific Model (PSM)

Computational
Fluid
Dynamics

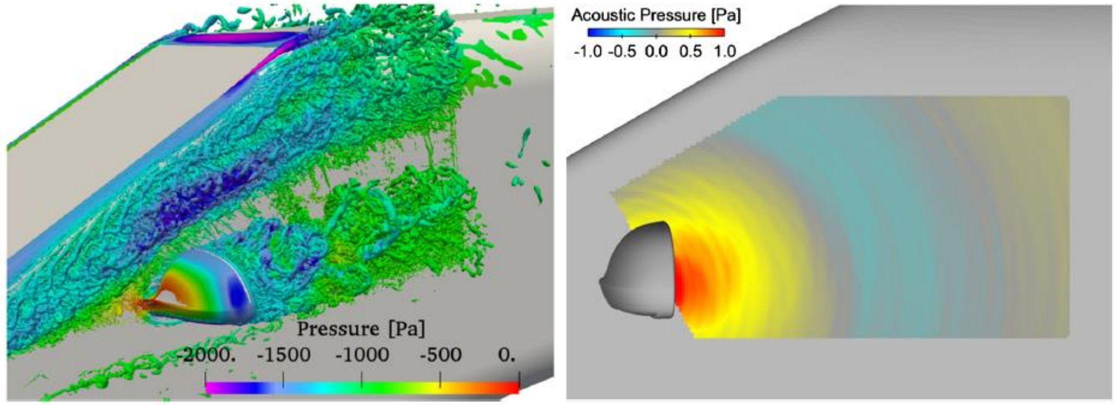


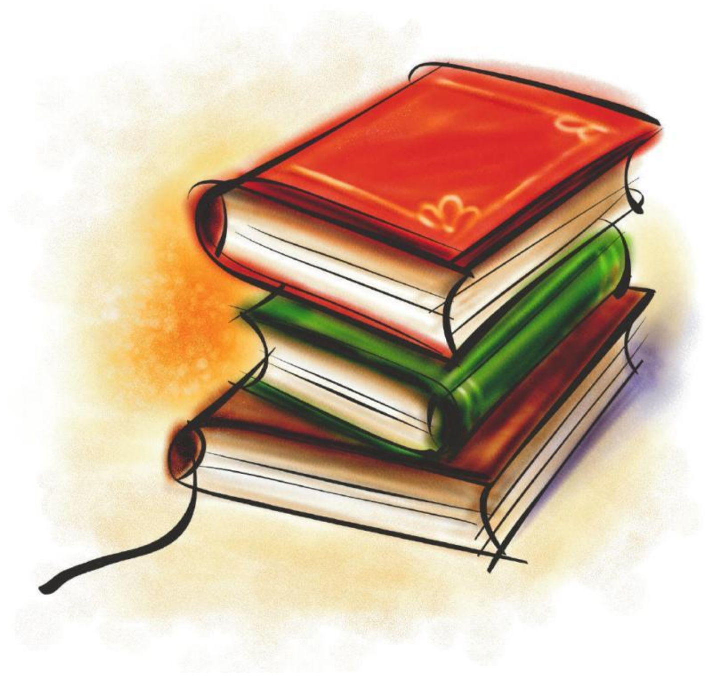
Computational
Structural
Mechanics

Conjugate
Heat
Transfer



Computational Aero-Acoustics





Metamodel-Assisted EAs (MAEAs)

Metamodel-Assisted EAs (MAEAs) – The Concept

Having already evaluated a number of candidate solutions (individuals; all gathered in the EA DB), build a model-agnostic/black-box (metamodel or surrogate evaluation model) to approximate the objective or constraint function values of new individuals generated during the evolution and avoid the use of the costly PSM tool, as much as possible.

Valid for both SOO & MOO. Different implementation might, though, be needed!

Questions:

- **How to select training patterns for the metamodel?**
- **When and how to train the metamodel(s)?**
- **Which metamodel (polynomial regression, neural networks, other?)?**
- **How to use the cost/fitness function approximation provided by the metamodel?**

MAEAs: Ways to Implement Metamodels

MAEAs with Off-Line Trained Metamodels:

A global metamodel, for the whole design space, trained 'outside' the evolution; the EA search relies exclusively on it. The computed "optimal" solution must be re-evaluated on the PSM and the optimisation restarts, if needed, by also updating/upgrading the metamodel.

MAEAs with On-Line Trained Metamodels:

Local metamodels are trained during the evolution; coordinated use of metamodels and the PSM during each generation.

MAEA with Generational Control:

Metamodel (local/global) trained during the evolution; selective use of the metamodel or the PSM in each generation.

Sampling Methods Supporting MAEAs with Off-line Trained Metamodels

Design of Experiments (DOE):

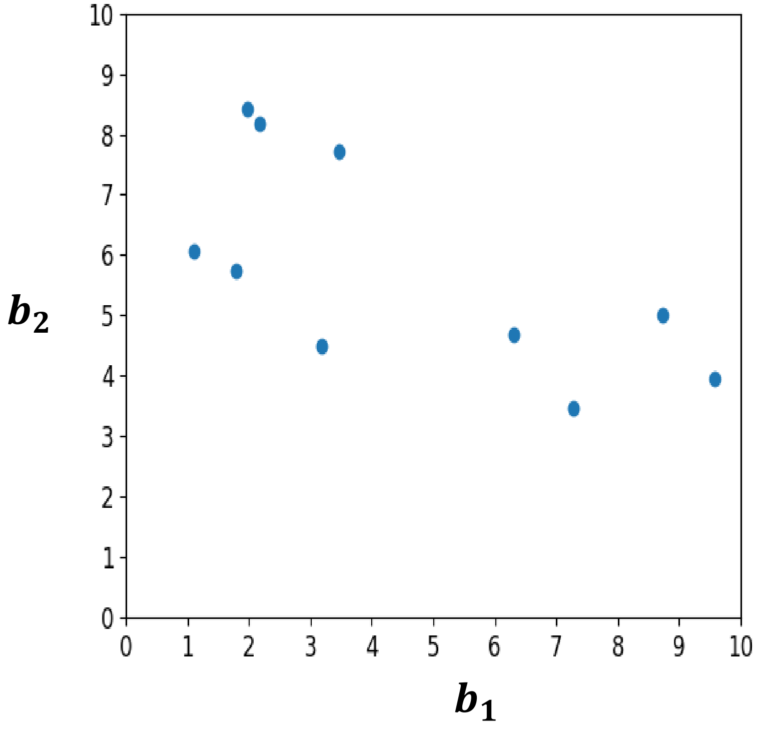
A systematic series of tests performed to measure the effect of changes in input variables on a system's response. Valid for both physical processes and computer simulation models.

Fisher, R. A., The Design of Experiments, Oliver & Boyd, 1935.

DOE Techniques:

- **Random Sampling**
- **Latin Hypercube Sampling**
- **Factorial Sampling**

Random Sampling



$0 \leq b_1 \leq 10$

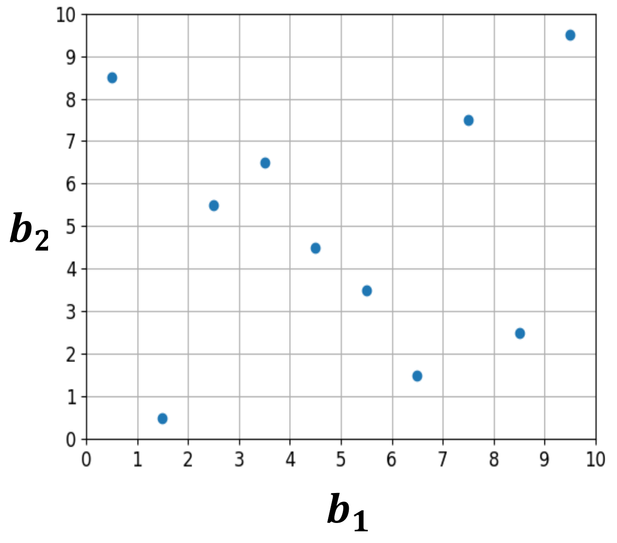
$0 \leq b_2 \leq 10$

$n_{DOE} = 10$

Latin Hypercube Sampling (LHS) (1/2)

Latin Hypercube Sampling (LHS):

The generalisation of the 2D Latin square to the N-dimensional hypercube.
2D Latin square: a square grid with only one sample in each row and each column.



$$0 \leq b_1 \leq 10$$

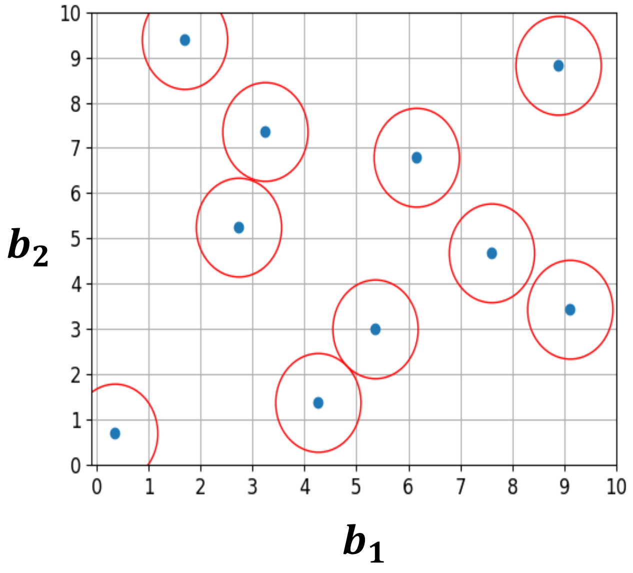
$$0 \leq b_2 \leq 10$$

$$n_{DOE} = 10$$

Samples are centered within each hypercube.

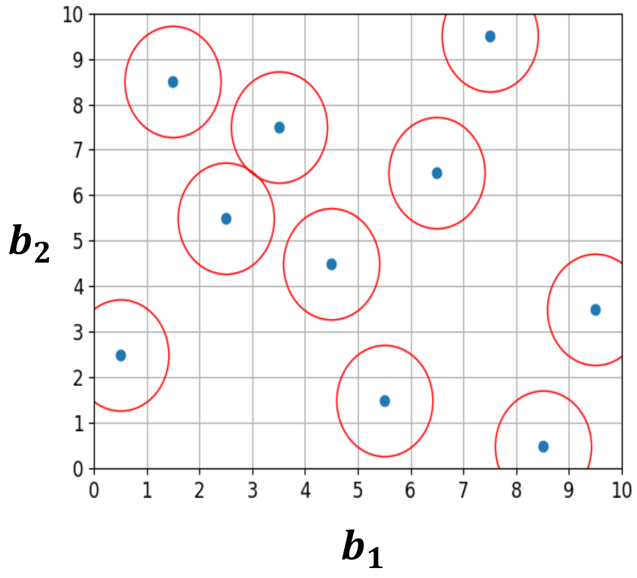
Latin Hypercube Sampling (LHS) (2/2)

Maximin



Maximize the minimum distance between samples

Maximin Centered



Maximin samples → centered within each hypercube.

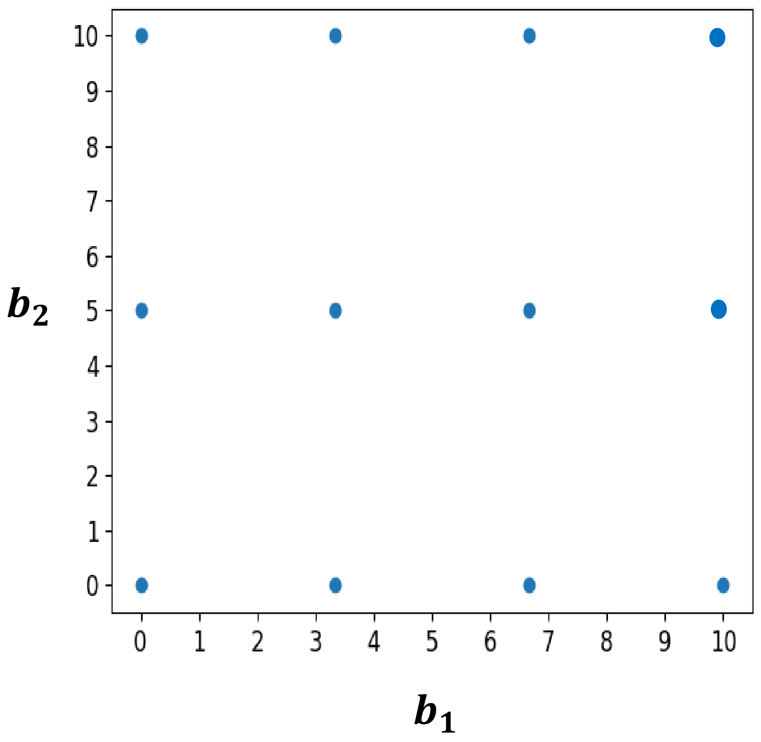
$$0 \le b_1 \le 10$$

$$0 \le b_2 \le 10$$

$$n_{DOE} = 10$$

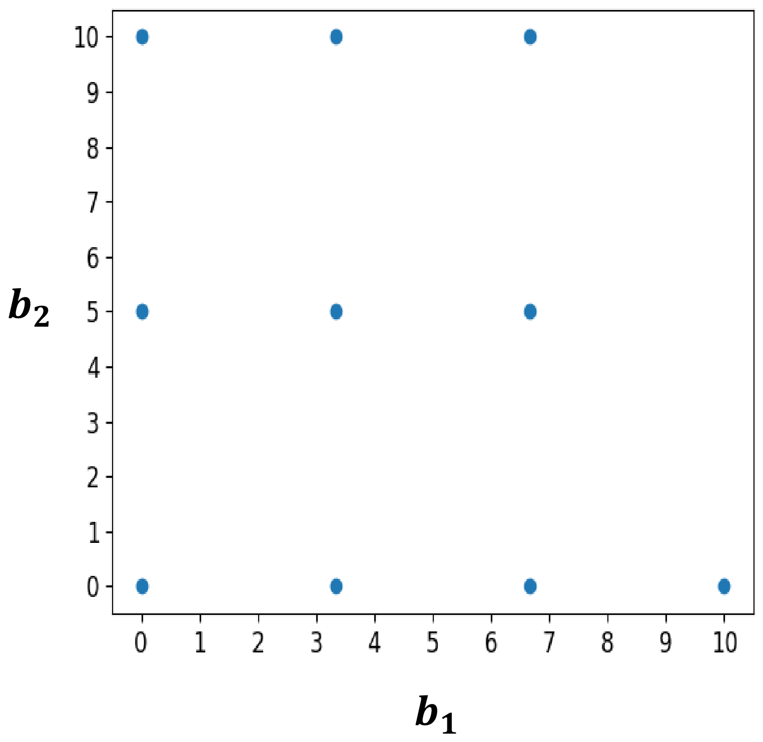
Factorial Sampling

Full Factorial Design



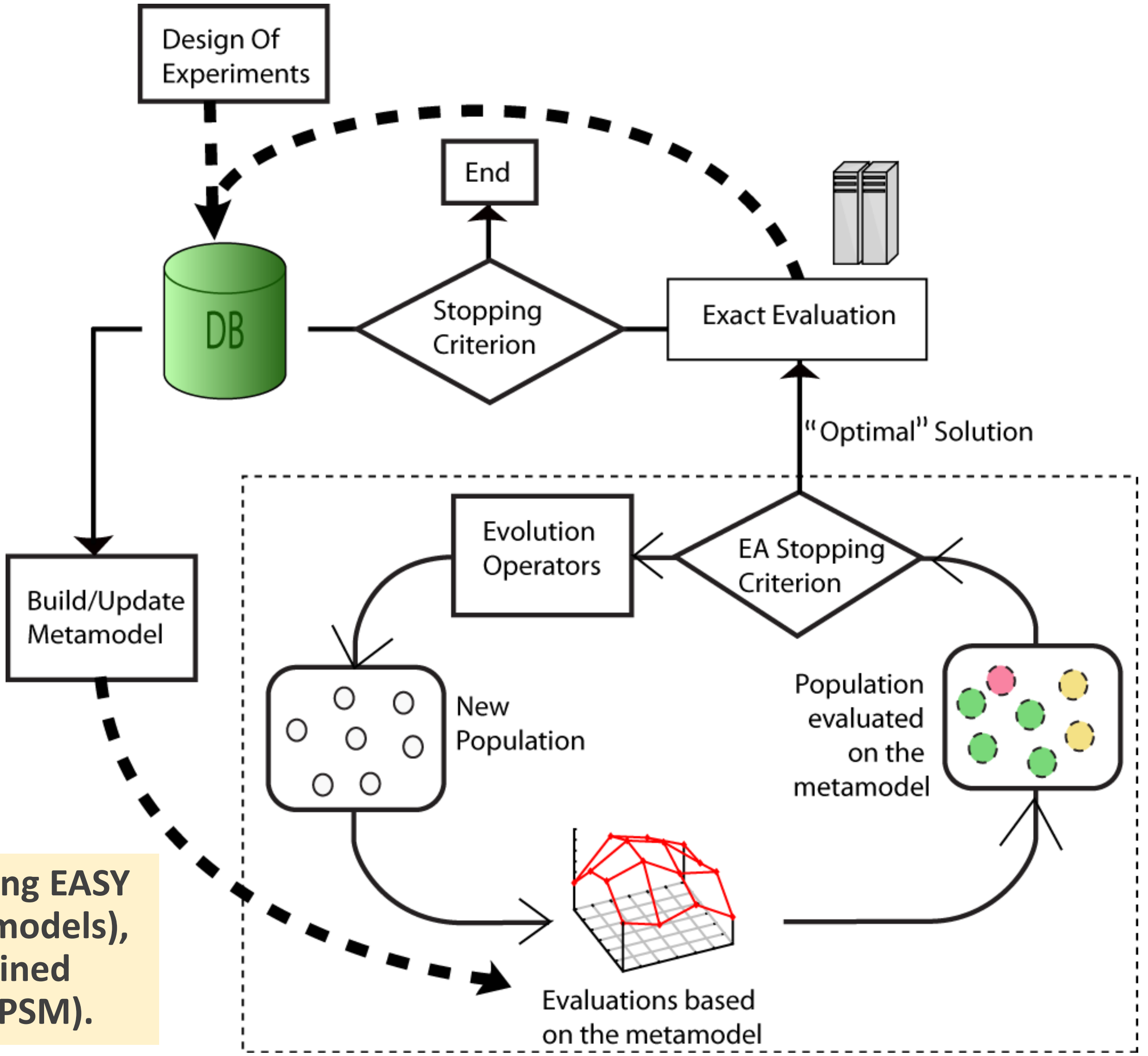
All possible combinations of factor (design variables) levels

Partial Factorial



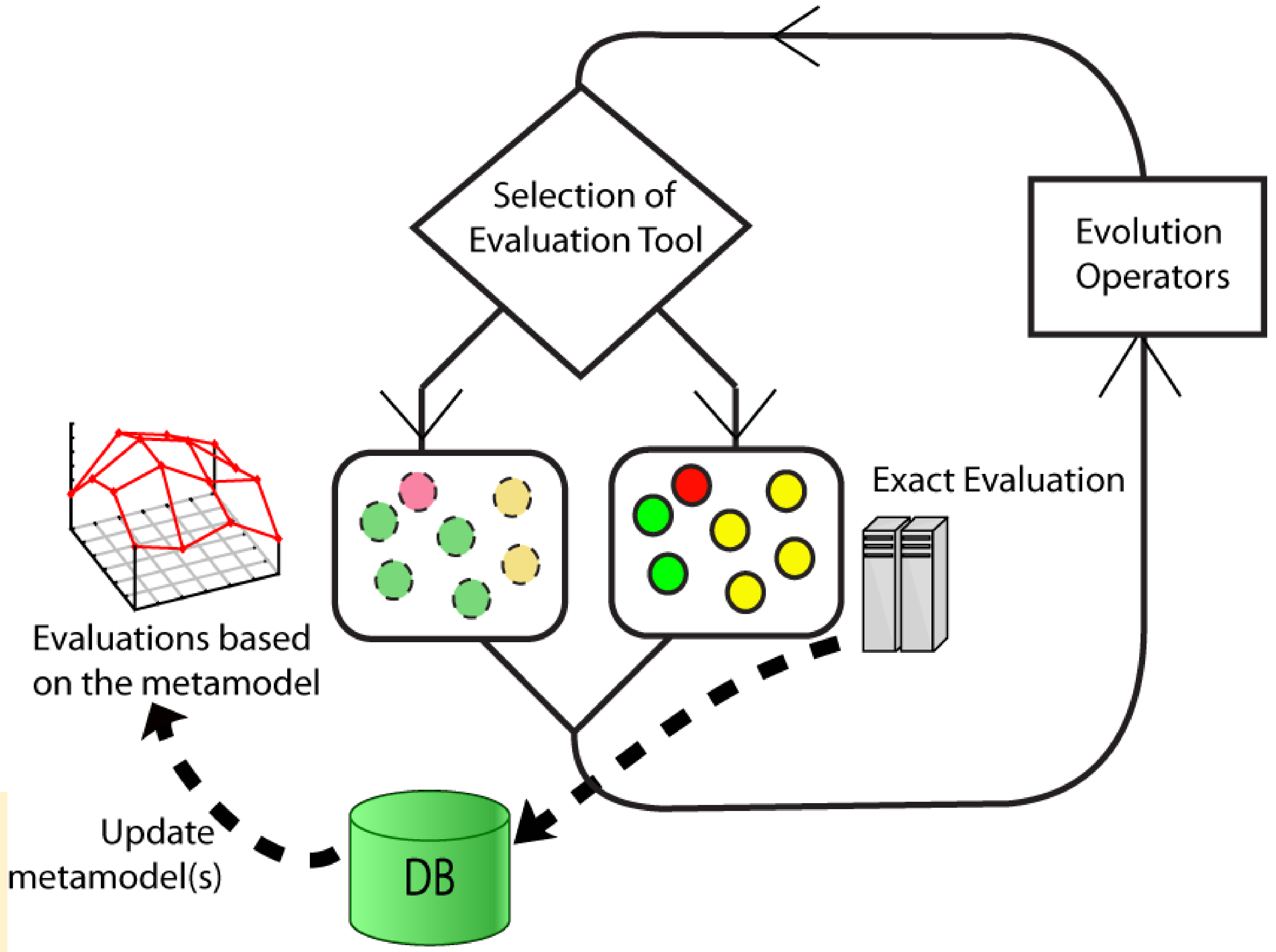
Carefully chosen subset from full factorial

A MAEA with Off-Line Trained Metamodels



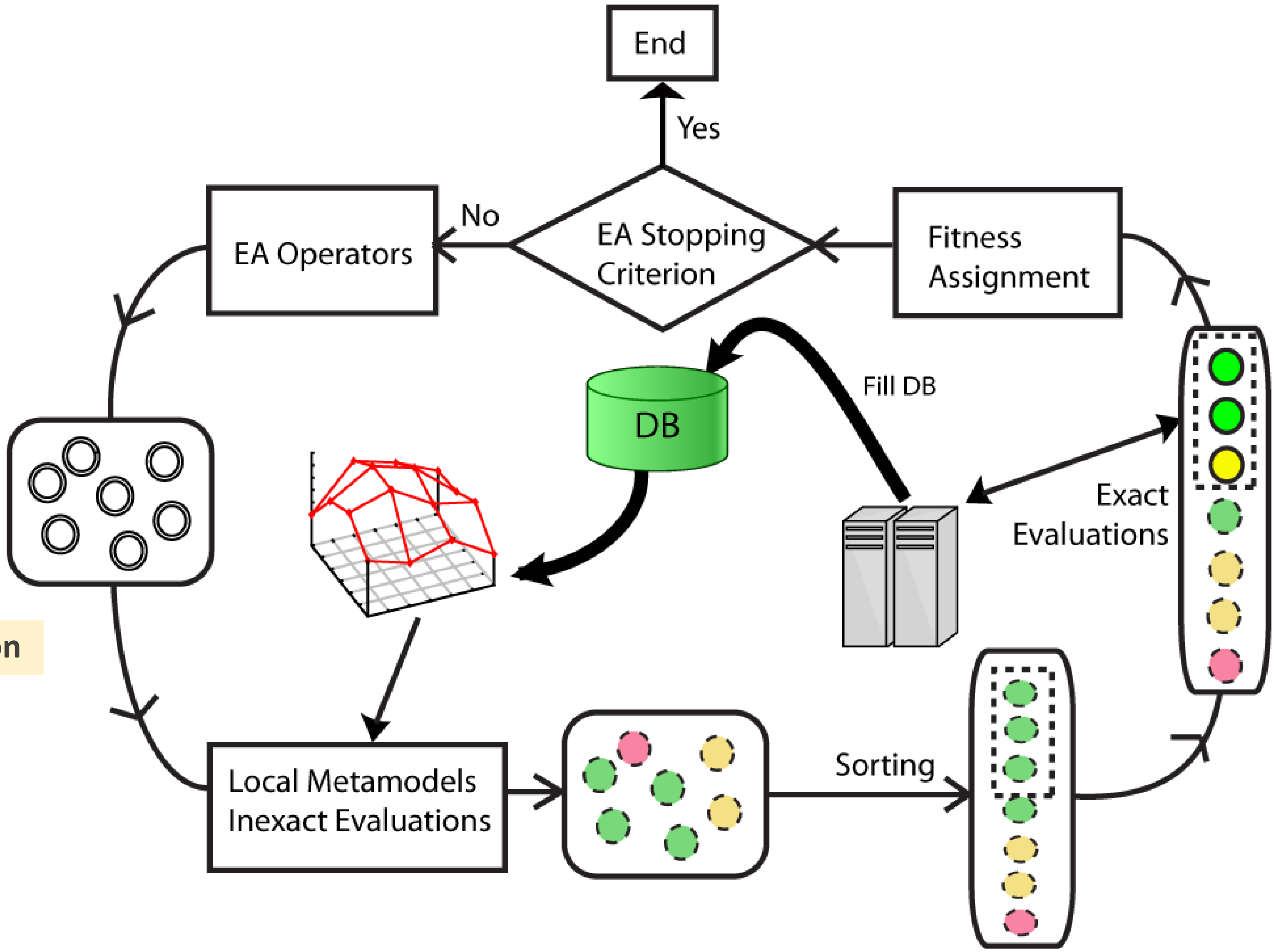
This type of MAEA may, also, run using EASY (without activating its built-in metamodels), by linking an external off-line trained metamodel (replacing calls to the PSM).

A MAEA with Generational Control


Selection of Evaluation Tool
Evolution Operators
Exact Evaluation
Evaluations based on the metamodel
Update metamodel(s)
DB

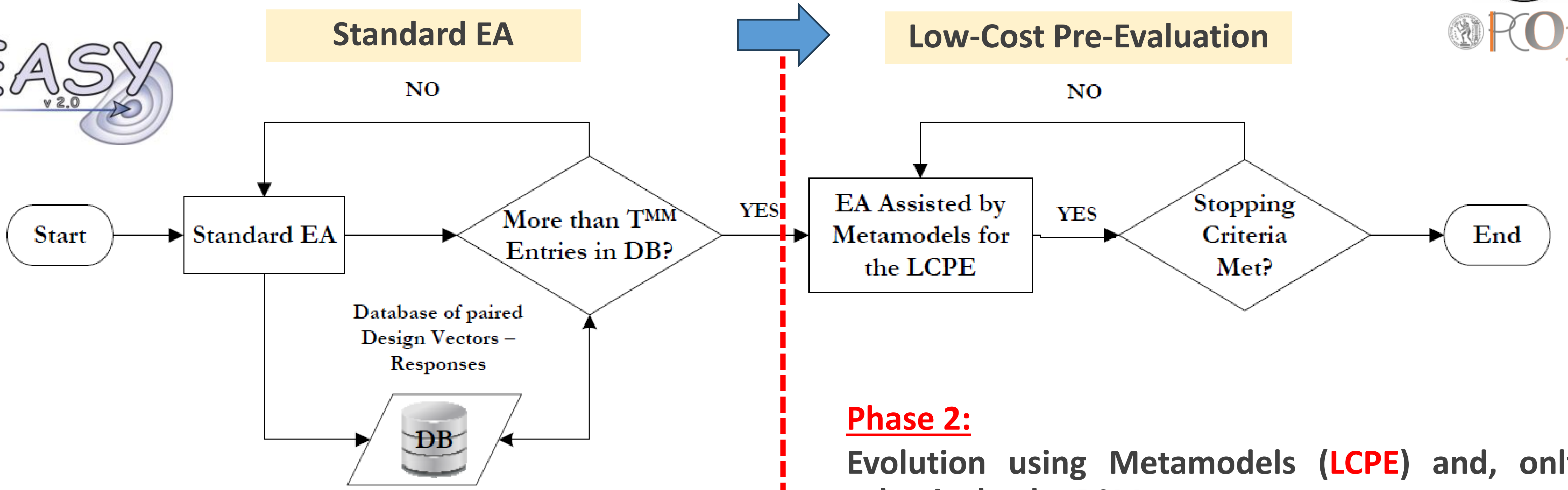
Each generation of offspring is selectively evaluated on the PSM or approximated by the metamodels.

A MAEA with On-Line Trained Metamodels



LCPE: Low-Cost Pre-Evaluation

A MAEA with On-Line Trained Metamodels


Standard EA
Low-Cost Pre-Evaluation
NO
Start
Standard EA
More than T^MM Entries in DB?
YES
Database of paired Design Vectors – Responses
DB
NO
EA Assisted by Metamodels for the LCPE
YES
Stopping Criteria Met?
End

Phase 1:

Exclusive use of the PSM; no use of Metamodels

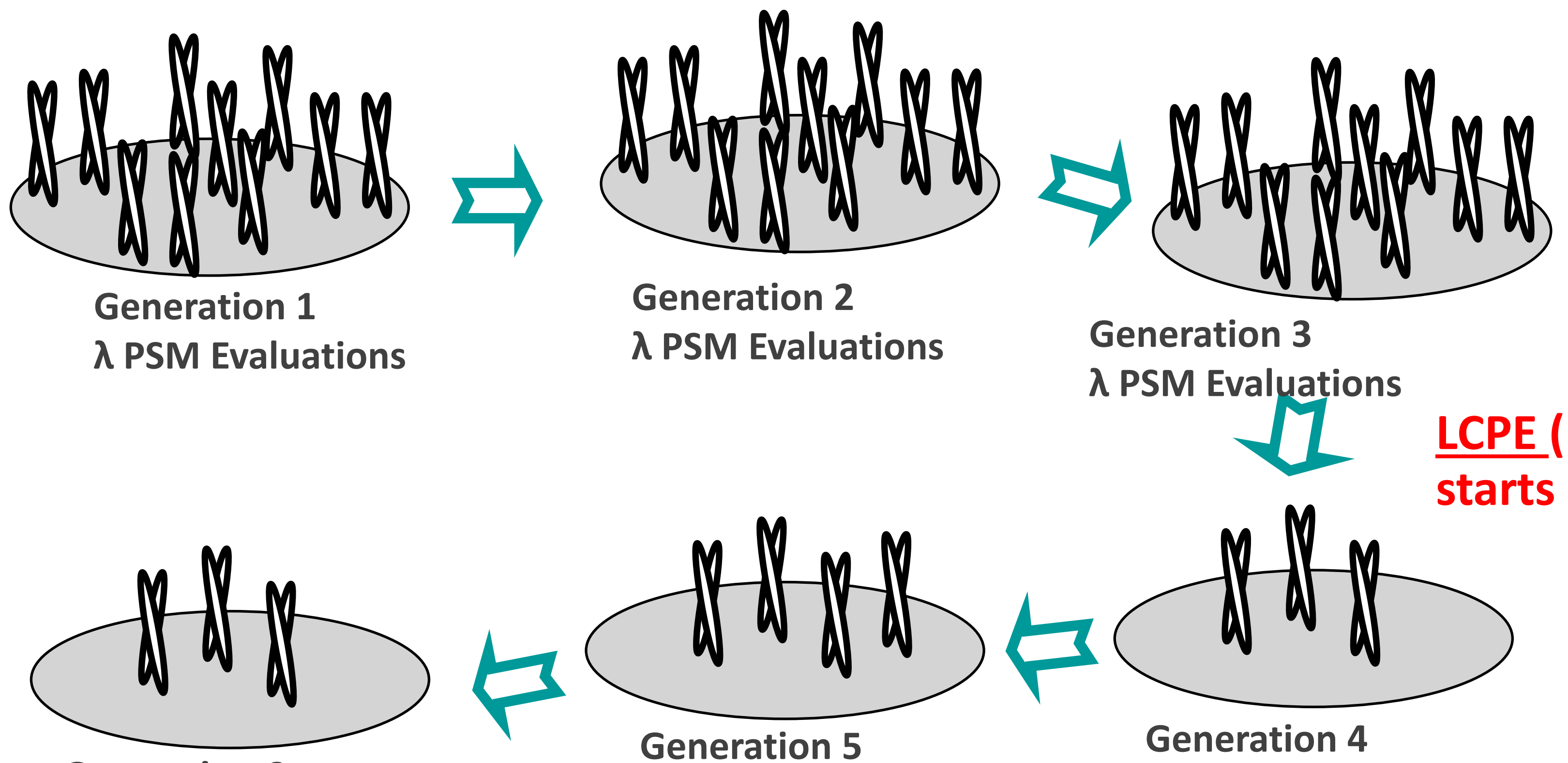
Terminates once T^{MM} evaluated individuals are stored in the DB.

Phase 2:

Evolution using Metamodels (LCPE) and, only selectively, the PSM

- All population members are evaluated on **local/personalised metamodels** trained on neighboring data in the DB.
- The best λ_e($\ll\lambda$, $\lambda_{e,min} \leq \lambda_e \leq \lambda_{e,max}$) of them are re-evaluated on the PSM & stored in the DB; this determines the computational cost of each generation.

A MAEA with On-Line Trained Metamodels – The LCPE Phase at a glance



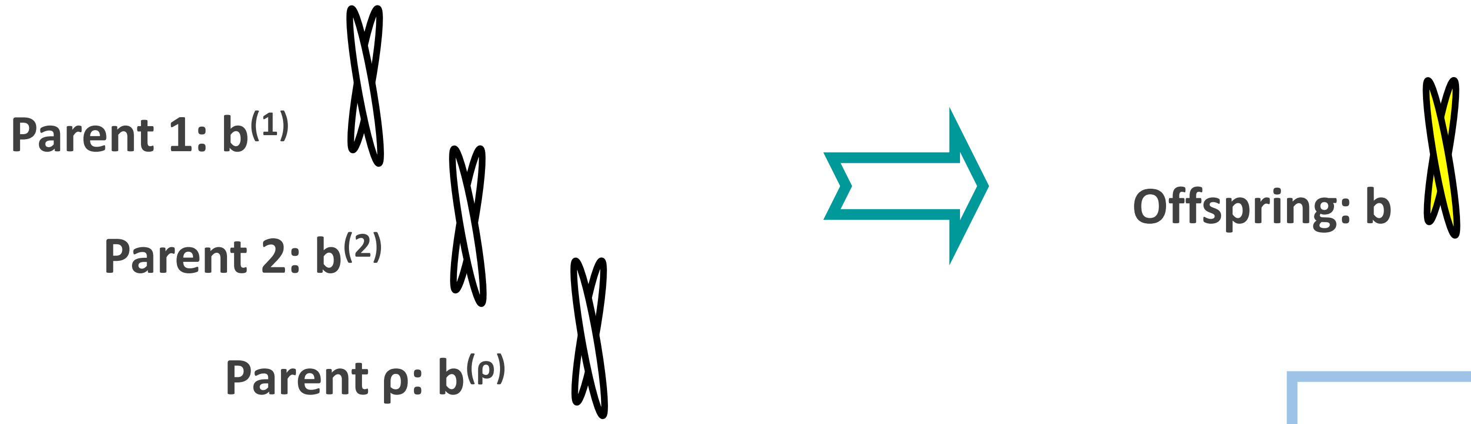
LCPE (Low-Cost Pre Evaluation) starts here!

Int. Review Journal Progress in Aerospace Sciences, 38:43-76, 2002.

Metamodels – Types

- ❑ **Fitness Inheritance**
- ❑ **Polynomial Regression**
- ❑ **Artificial Neural Networks**
  - ▪ **Radial Basis Function (RBF → RBFNs)**
  - ▪ **Multilayer Perceptrons**
- ❑ **Kriging Model**
- ❑ **Support Vector Machines (SVM)**

Fitness Inheritance



Fitness inheritance approximates the fitness of any offspring from the weighted (based on distances D among them, in the design space) fitness values of its ρ parents.

$$\vec{o}(\vec{b}) = \sum_{p=1}^{\rho} \omega_p \vec{f}\left(\vec{b}^{(p)}\right)$$

$$\omega_p = \frac{D_p}{\sum_{i=1}^{\rho} D_i}$$

$$D_p = \prod_{i=1, i\neq p}^{\rho} \left\| \vec{b}^{(i)} - \vec{b} \right\|$$

Polynomial–based Regression Surface Methods (RSM)

Linear RSM:

$$\vec{o}\left(\vec{b}\right) = \alpha_0 + \sum_{n=1}^{N} \alpha_n b_n + \epsilon$$

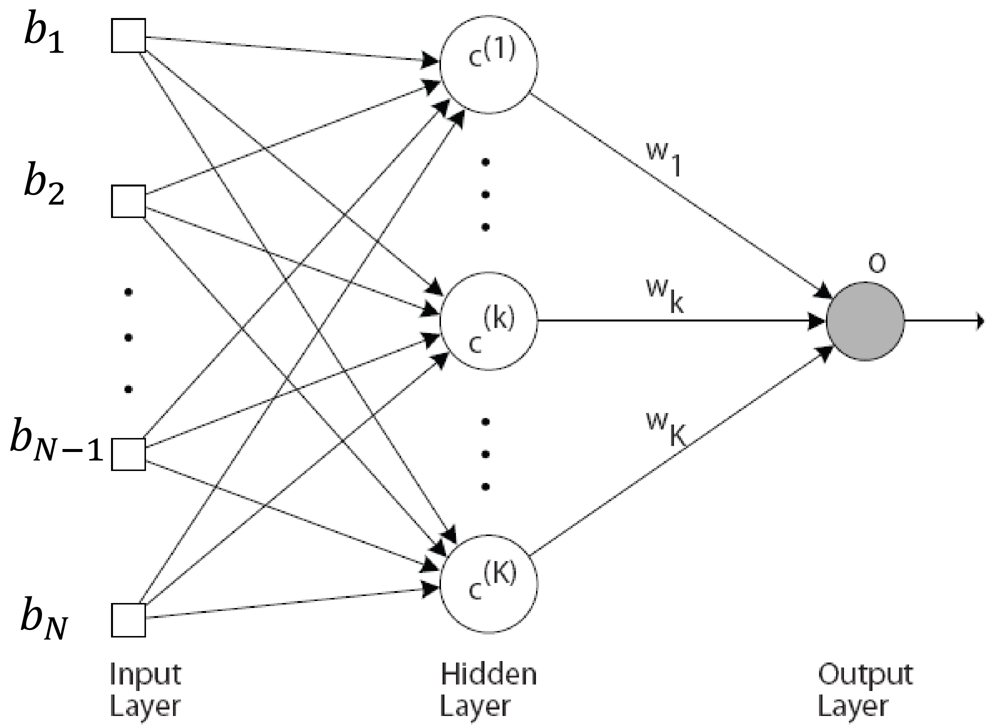
Second-order RSM:

$$\vec{o}\left(\vec{b}\right) = \alpha_0 + \sum_{n=1}^{N} \alpha_n b_n + \sum_{i=1}^{N} \alpha_{ii} b_i^2 + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \alpha_{ij} b_i b_j + \epsilon$$

Least-squares system:

$$\boldsymbol{\alpha} = (\boldsymbol{B}^T \boldsymbol{B})^{-1} \boldsymbol{B}^T \boldsymbol{o}$$

Radial Basis Function Networks (RBFNs) (1/6)



RBFN with N inputs, K hidden units (RBF centers) and a single output (M_o=1).

Performs a nonlinear mapping from the input to the hidden layers, followed by a linear mapping to the output.

Activation functions:

- **Multiquadratic:**

$$G(h) = (h^2 + r^2)^{1/2}$$

- **Inverse Multiquadratic:**

$$G(h) = (h^2 + r^2)^{-1/2}$$

- **Gaussian:**

$$G(h) = \exp\left(-\frac{h^2}{r^2}\right)$$

Network response: $\vec{o}(\vec{b}) = \sum_{k=1}^{K} \omega_k G(\left\|\vec{b} - \vec{c}^{(k)}\right\|_2)$

Haykin, Neural Networks: A Comprehensive Foundation, 1999.

RBFNs – Training (2/6)

Assume K=T, with RBF centers: $\vec{b}^{(t)} \equiv \vec{c}^{(t)}, t \in [1, T]$

$$\sum_{t=1}^{T} \omega_t G\left(\left\|\vec{b}^{(t)} - \vec{c}^{(t)}\right\|_2\right) = \vec{\zeta}^{(t)}$$

$$\begin{bmatrix} G(\left\|\vec{b}^{(1)} - \vec{c}^{(1)}\right\|_2) & \cdots & G(\left\|\vec{b}^{(1)} - \vec{c}^{(T)}\right\|_2) \\ \vdots & \ddots & \vdots \\ G(\left\|\vec{b}^{(T)} - \vec{c}^{(1)}\right\|_2) & \cdots & G(\left\|\vec{b}^{(T)} - \vec{c}^{(T)}\right\|_2) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_T \end{bmatrix} = \begin{bmatrix} \zeta^{(1)} \\ \vdots \\ \zeta^{(\mathrm{T})} \end{bmatrix}$$

T = number of training patterns
K= number of RBF centers (hidden layer neurons)

RBFN with Importance Factors (3/6)

Network response:

$$\vec{o}(\vec{b}) = \sum_{k=1}^{K} \omega_k G\left(\left\|\vec{b} - \vec{c}^{(k)}\right\|_{\cancel{2}}\right)$$

wei

Use *weighted norm*

$$G\left(\left\|\vec{b}^{(t)} - \vec{c}^{(k)}\right\|_{wei}\right) = \sqrt{\sum_{n=1}^{N} I_n \left(b_n^{(t)} - c_n^{(k)}\right)^2}$$

with

$$I_n = \frac{\left|\frac{\vartheta o^b}{\vartheta b_n}\right|}{\sum_{i=1}^{N}\left|\frac{\vartheta o^b}{\vartheta b_i}\right|}$$

and

$$\sum_{n=1}^{N} I_n = 1$$

Gradients are directly/analytically computed by the RBFN.

Inverse Problems in Engineering, 9:389-412, 2001.

Selection of the RBF Centers (4/6)

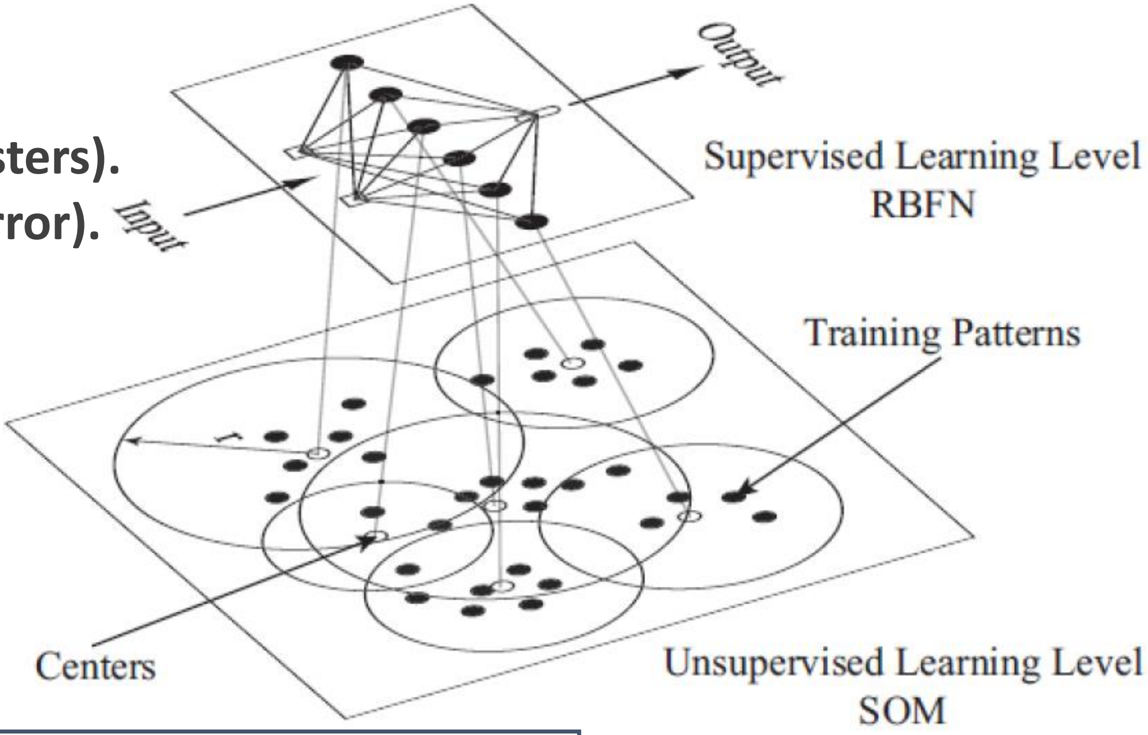
Using less hidden nodes than training patterns (K<T) enhances network's generalisation.

Selection of RBF centers using Self-Organized Maps (SOMs).

Unsupervised neural network algorithm (for clustering) that maps an N-dimensional input data to a 2D space.

Two levels of RBFN training:

- **Unsupervised (classification of training patterns in K clusters).**
- **Supervised (training by minimising the approximation error).**

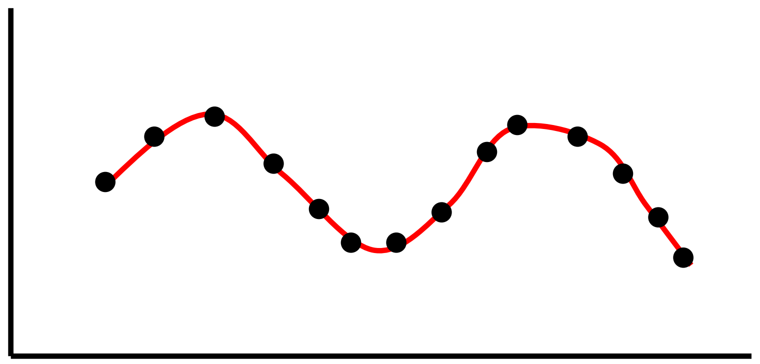


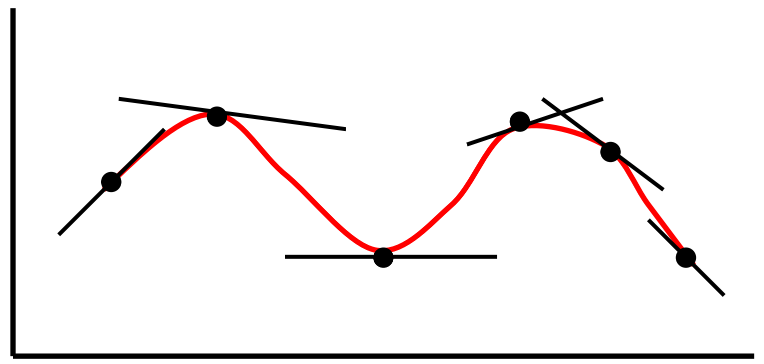
Engineering Optimization, 38:941-957, 2006.

T. Kohonen, Self-Organization and Associative Memory, Springer 1999.

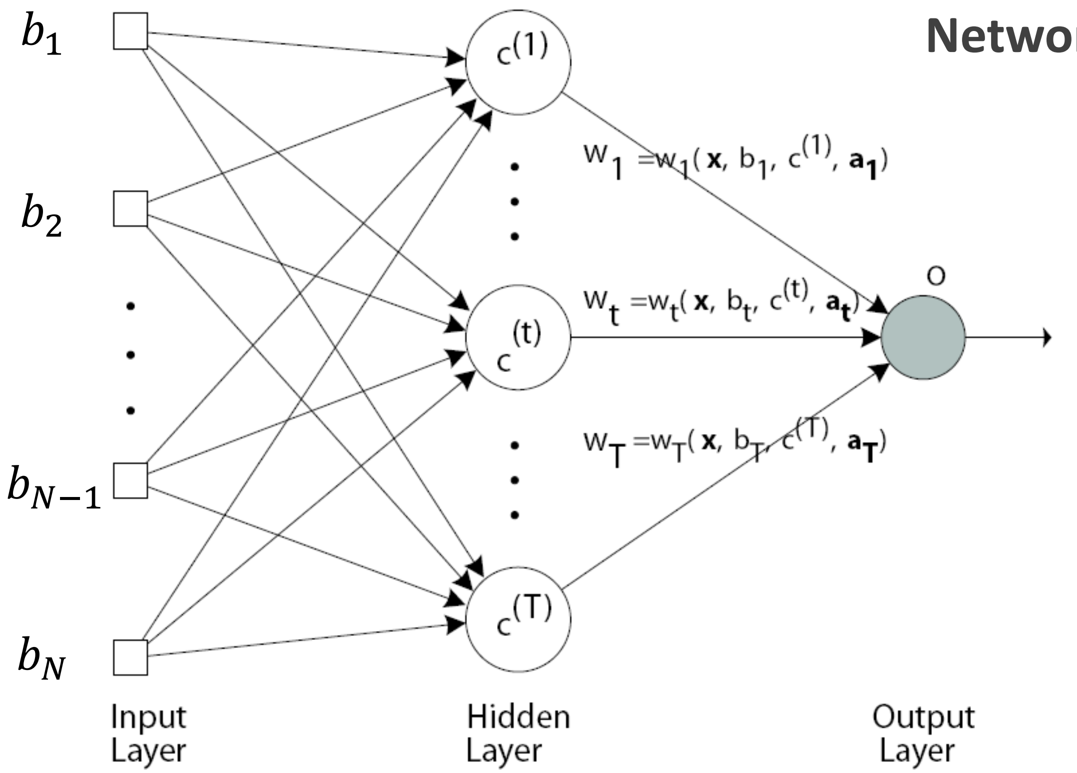
Gradient-Assisted RBFs (GARBFs) (5/6)

- **Concept: Devise and use new metamodels (MLP & RBF), trained on both responses and gradients.**
- **GARBF - Prerequisites:**
  - **New mathematical formulation for the new metamodels.**
  - **Tools to compute gradients of responses (Aerodynamics: Adjoint Method)**
- **Target: The cost for gathering responses & gradients should be less than the cost for gathering MORE responses (in a traditional RBF network). Or, with the same CPU cost, the new gradient-assisted metamodels to give better predictions.**





Gradient-Assisted RBFs (6/6)



Network response:

$$o(\vec{b}) = \sum_{t=1}^{T} \underbrace{\omega_t\left(\vec{b}\right)} G(h_t)$$

$$\omega_t\left(\vec{b}\right) = \beta_t + \sum_{n=1}^{N} \alpha_{t,n}\left[1 - \left(b_n - c_n^{(t)}\right)\right]$$

Training the RBFN involves the computation of coefficients α and β

$$\frac{\vartheta o}{\vartheta b_r} = \sum_{t=1}^{T} \left(2\omega_t\left(\vec{b}\right) G(h_t)'\left(b_r - c_r^{(t)}\right) + \alpha_{t,r} G(h_t)\right)$$

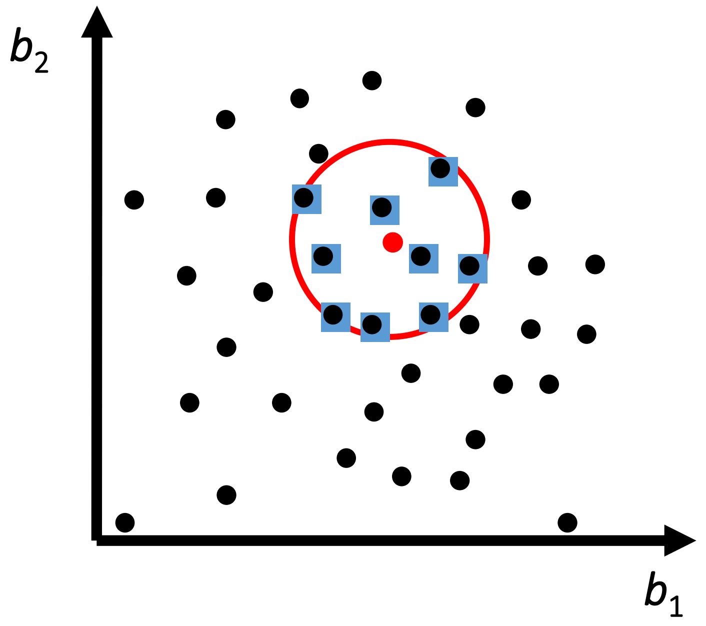
Use the Adjoint method to efficiently compute the gradients

Applied Mathematical Modeling, 28:197-209, 2004.
Computer Methods in Applied Mechanics and Engineering, 195:6312-6329, 2006.
Journal of Inverse Problems in Science & Engineering, Vol. 14(4):397-410, 2006.

Training Pattens Selection (1/5)

For MAEAs using on-line trained metamodels.

A local metamodel is trained for each individual using "neighboring" previously evaluated solutions

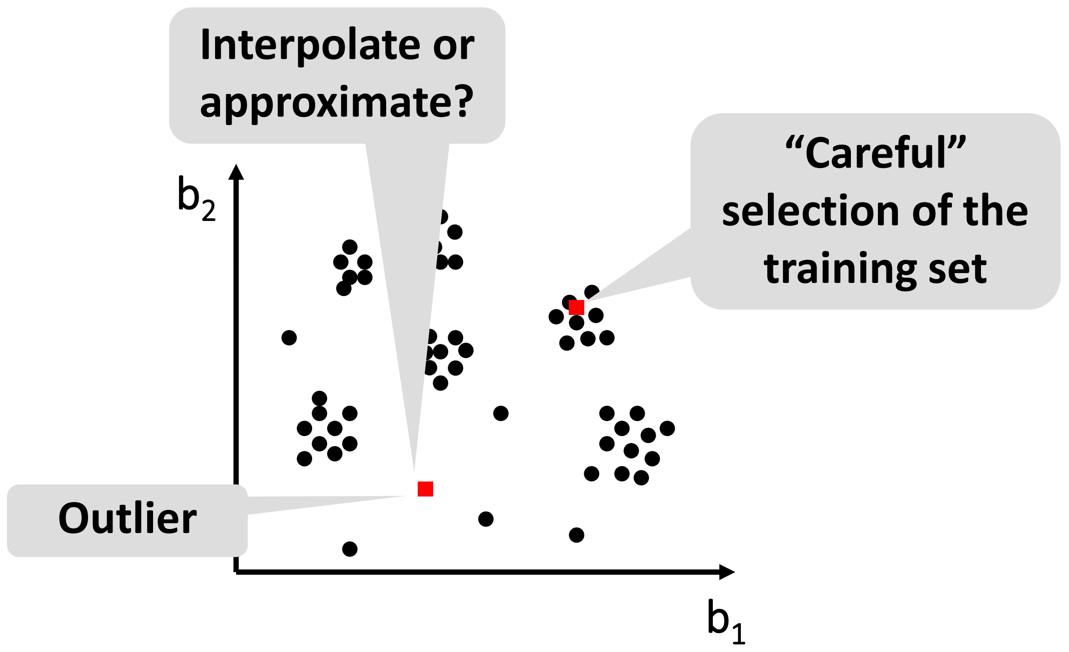


Attention should be paid to:

- **Selection of the training patterns**
- **Detection of outliers – unsafe to trust the metamodel**
- **A criterion to choose between highly specialised metamodels and metamodels with good generalisation capabilities.**

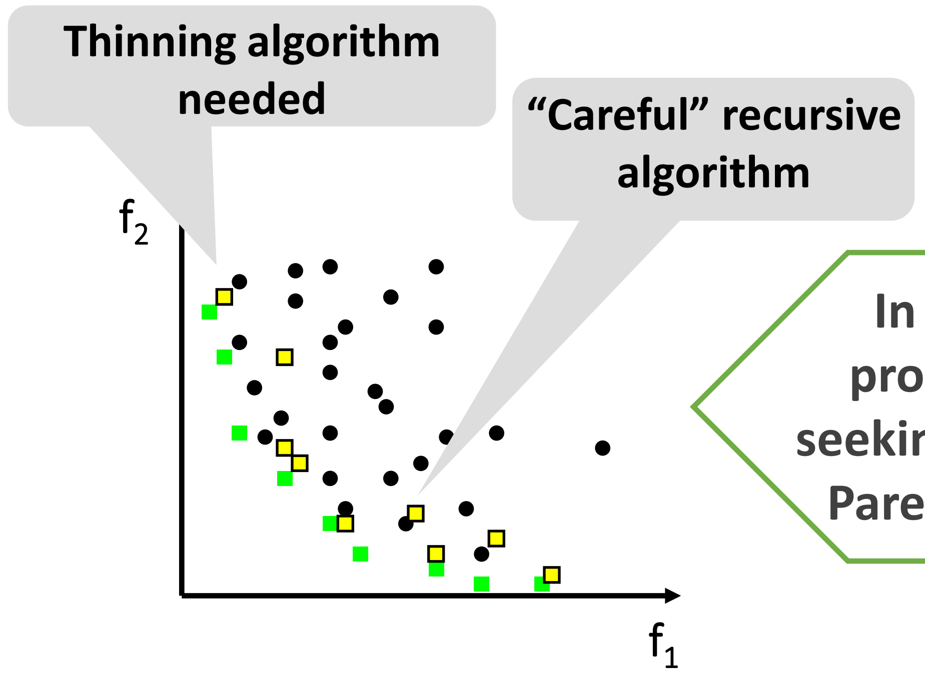
Training Pattens Selection – Difficulties in MOO (2/5)

For MAEAs using on-line trained metamodels.


Interpolate or approximate?
"Careful" selection of the training set
Outlier
b_2
b_1

● Database entries (training patterns)
🟥 New individual

Scattered training patterns.
Many outliers exist.



In MOO problems, seeking for the Pareto front

🟩 Non-dominated generation members
● Dominated generation members

Estimation errors accumulate and interact.
All 🟩 call for exact evaluations?

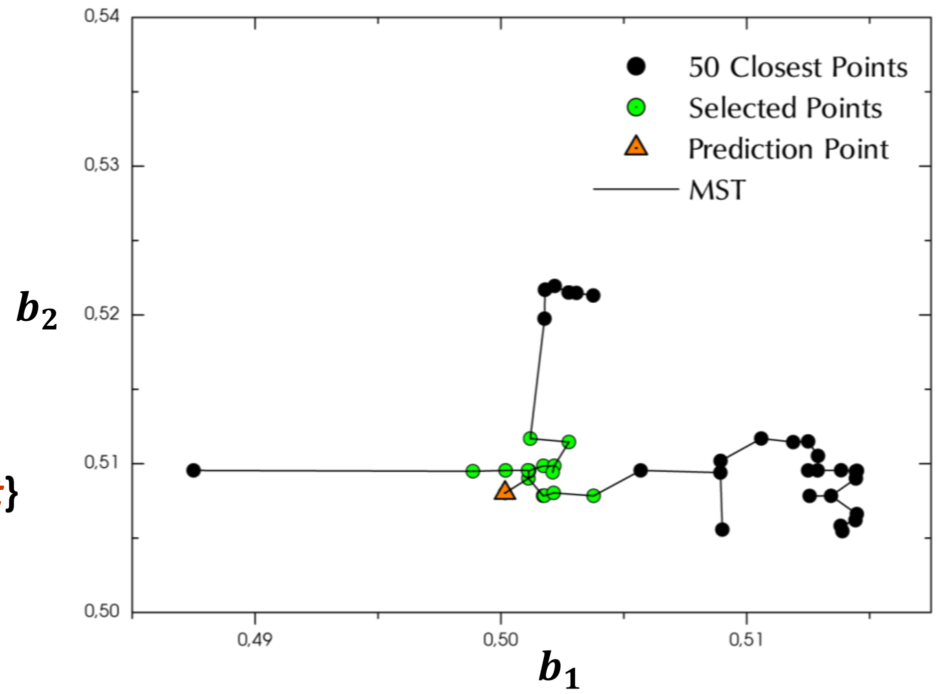
Training Pattens Selection – Minimum Spanning Tree (3/5)

For each new individual:

1. **Select the closest DB entries.**
2. **Construct the Minimum cost Spanning Tree (MST).**
   Undirected graph (edge-weighted) connecting all points without cycles while minimising the total edge weight.
3. **Start "walking" in the MST from the prediction point**
   - **Traverse a branch, if "not too long" (Criterion!):**

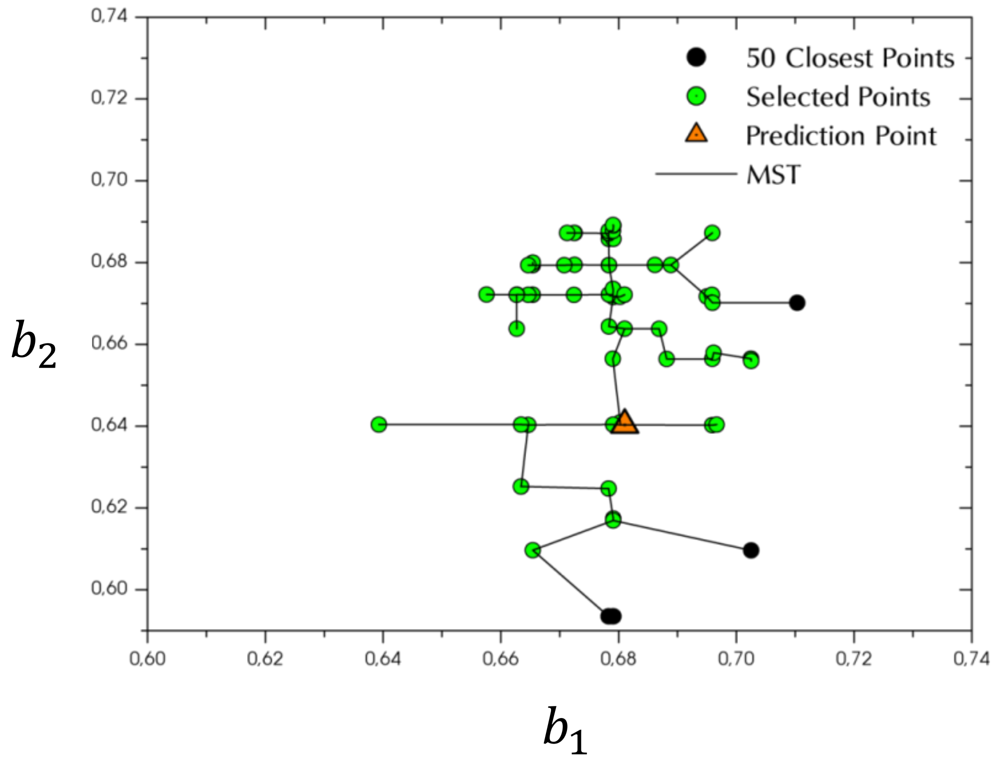
     ***BranchLength*{*TrainSet*↔*Q*} < *AveLength*{*TrainSet*} + K *StdDeviation*{*TrainSet*}**

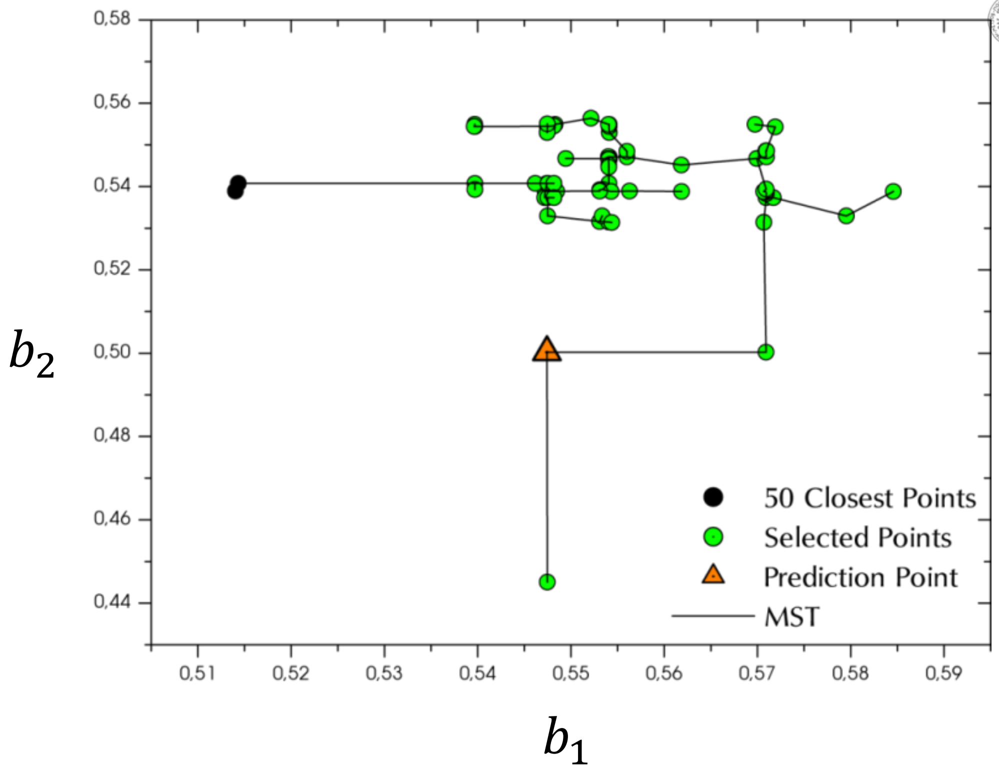
   - **Select only the nodes that are reached as Training Patterns.**



Engineering Optimization, 38:941-957, 2006.
D. Knuth, The Art of Computer Programming, Volume 1: Fundamental Algorithms, 1997.

Training Patterns Selection – Interpolate or Approximate? (4/5)





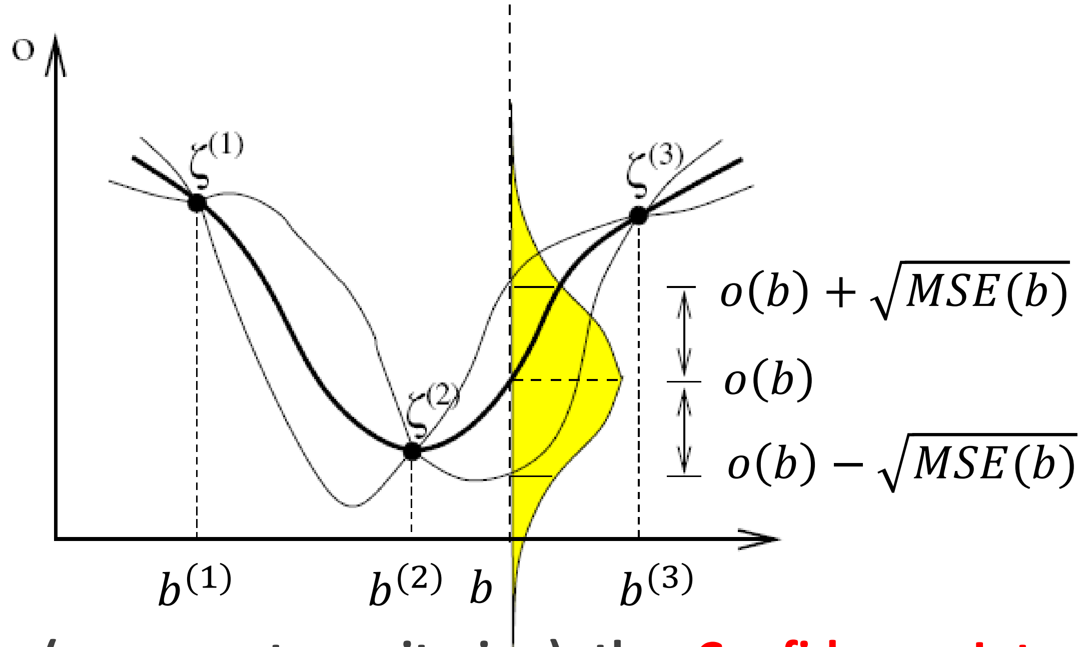
Error in f_1 : 0.0109% Error in f_2 : 0.0882%	**Interpolation**	Error in f_1 : 74.7% Error in f_2 : 0.371%
Error in f_1 : 0.887% Error in f_2 : 3.94%	**Approximation**	Error in f_1 : 1.44% Error in f_2 : 0.0606%

Training Patterns Selection – Lessons Learned (5/5)

- **Careful selection of the training patterns is required.**
- **Ways to detect outliers and accordingly select the appropriate training patterns are necessary. Depending on the case the algorithm should be able to flag the individuals for approximation or interpolation using surrogates.**
- **Attention in MOO problems.**

Kriging (1/6)

Kriging (Gaussian Processes) is an interpolation scheme, based on the maximum likelihood hypothesis, used for modeling deterministic computer analyses as the realisation of a stochastic process. Kriging assumes that the response is a random variable that depends on its distance from the training patterns.



Kriging estimates and uses (as an extra criterion) the Confidence Interval, pertinent to the predicted value of the objective or constraint function.

IEEE Transactions on Evolutionary Computation, 10:421-439, 2006.

Kriging (2/6)

Response: $\vec{o}\left(\vec{b}\right) = \hat{\mu} + z(\vec{b})$

Expected value

Covariance function with zero mean and a correlation function C

$$C\left(\vec{b}^{(\kappa)}, \vec{b}^{(\lambda)}, \vec{\theta}\right) = exp\left(-\sum_{n=1}^{N} \theta_n \left(\vec{b}^{(\kappa)} - \vec{b}^{(\lambda)}\right)^2\right)$$

Correlation parameters $\vec{\theta}$ should be computed

Maximise the Probability Density function (PDF) of F(b) to get the known responses at the training patterns:

$$max\left\{\frac{1}{(2\pi)^{N/2}(\sigma^2)^{N/2}\sqrt{|\boldsymbol{C}|}} exp\left[\frac{(\boldsymbol{Z} - \boldsymbol{1}\hat{\mu})^T \boldsymbol{C}^{-1}(\boldsymbol{Z} - \boldsymbol{1}\hat{\mu})}{2\sigma^2}\right]\right\}$$

Kriging (3/6)

where:

$$\boldsymbol{C} = \begin{bmatrix} c\left(\vec{b}^{(1)}, \vec{b}^{(1)}, \vec{\theta}\right) & \cdots & c\left(\vec{b}^{(1)}, \vec{b}^{(T)}, \vec{\theta}\right) \\ \vdots & \ddots & \vdots \\ c\left(\vec{b}^{(T)}, \vec{b}^{(1)}, \vec{\theta}\right) & \cdots & c\left(\vec{b}^{(T)}, \vec{b}^{(T)}, \vec{\theta}\right) \end{bmatrix}, \boldsymbol{Z} = \begin{bmatrix} \zeta^{(1)} \\ \vdots \\ \zeta^{(\mathrm{T})} \end{bmatrix}, \boldsymbol{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

Adopt the generalised least squares estimates for the mean value and variance:

$$\hat{\mu} = \frac{\boldsymbol{1}^{\mathrm{T}} \boldsymbol{C}^{-1} \boldsymbol{Z}}{\boldsymbol{1}^{\mathrm{T}} \boldsymbol{C}^{-1}}$$

$$\sigma^2 = \frac{1}{\mathrm{N}} (\boldsymbol{Z} - \boldsymbol{1}\hat{\mu})^T \boldsymbol{C}^{-1} (\boldsymbol{Z} - \boldsymbol{1}\hat{\mu})$$

Kriging (4/6)

Finally, the minimisation problem to be solved is:

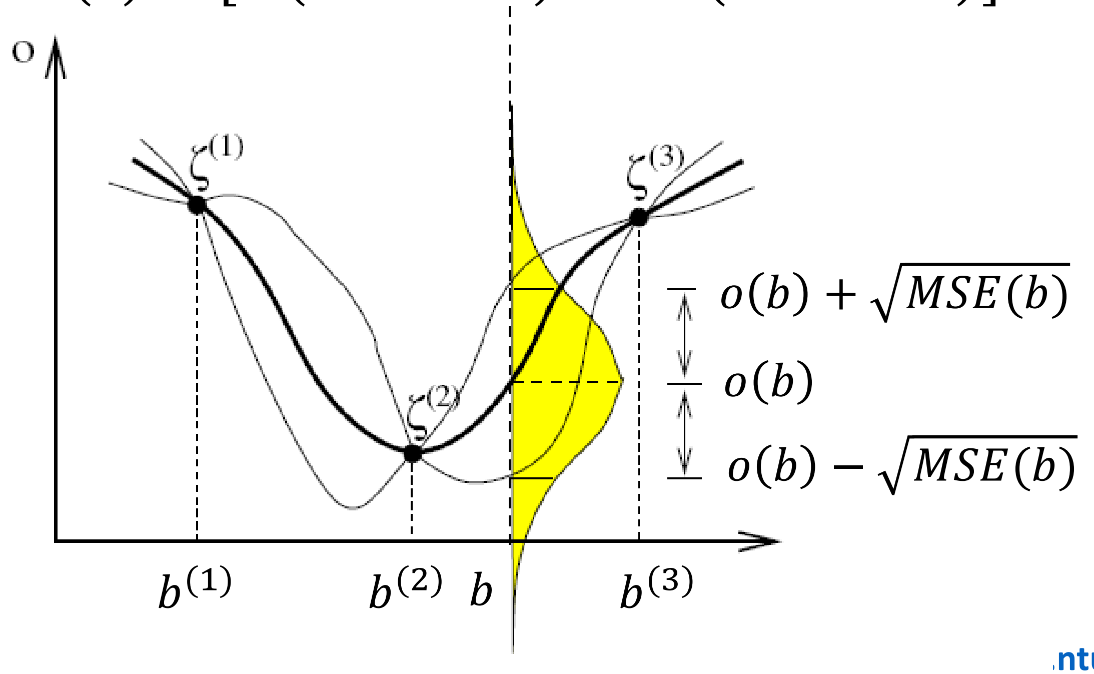
$$\boxed{min\left[N\, ln\left(\sigma^2(\vec{\theta})\right) + ln\left(\left|\boldsymbol{C}(\vec{\theta})\right|\right)\right]}$$

Once $\vec{\theta}$ is known, we can compute:

$$\vec{o}\left(\vec{b}\right) = \hat{\mu} + (\boldsymbol{Z} - \boldsymbol{1}\hat{\mu})^T \boldsymbol{C}^{-1}\boldsymbol{c}(\vec{b})$$

$$MSE\left(\vec{b}\right) = \sigma^2\left(1 - \boldsymbol{c}(\vec{b})^T\boldsymbol{C}^{-1}\boldsymbol{c}(\vec{b})\right)$$

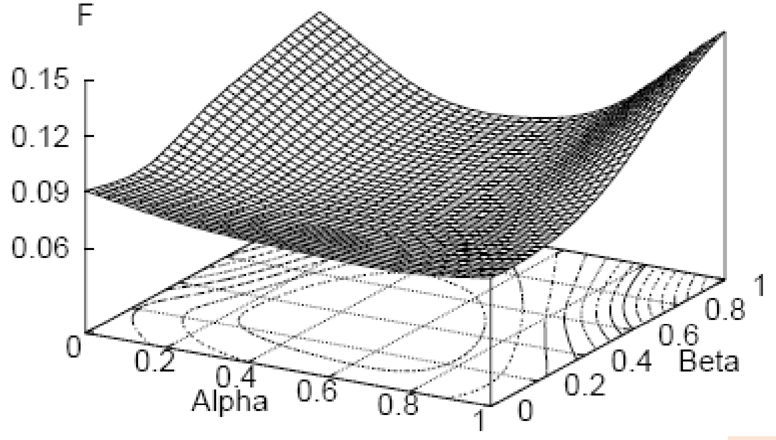
$$\boldsymbol{c}\left(\vec{b}\right) = \left[C\left(\vec{b}\ \ , \vec{b}^{(1)}, \vec{\theta}\right), \dots, C\left(\vec{b}\ \ , \vec{b}^{(T)}, \vec{\theta}\right)\right]^T$$



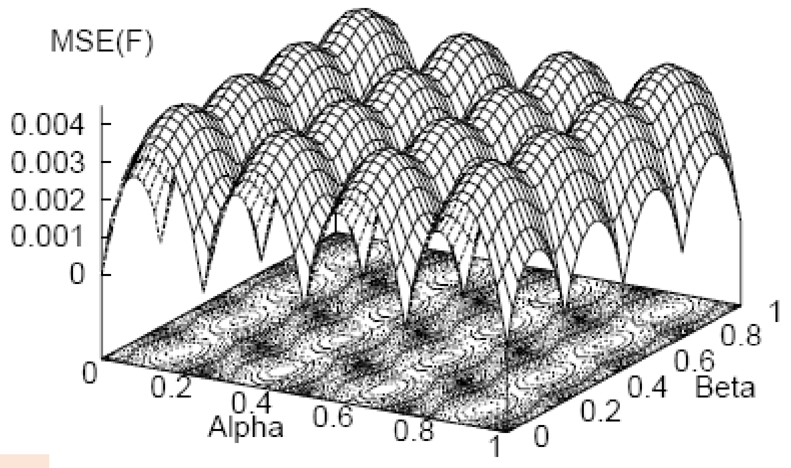
Kriging (5/6)

Predicted

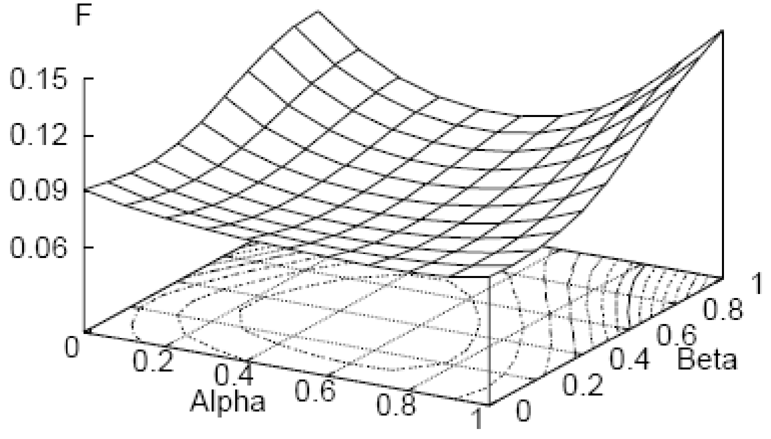
Response



Mean Squared Error (MSE)



Exact/"Real"



How to use Kriging within a MAEA (6/6)

Kriging → $\vec{o}\left(\vec{b}\right)$ **&** $\sigma\left(\vec{b}\right) = \sqrt{MSE\left(\vec{b}\right)}$

Four 'ideas'/criteria for using Kriging in SOO/MOO MAEAs during the LCPE

1. **Most likely Improvement (MI).** **No confidence information; considers the response with max PDF.**

$$MI\left(\vec{b}\right) = \vec{o}\left(\vec{b}\right)$$

2. **Lower Confidence Bound (LCB).** **Increases re-evaluations in promising but less explored regions.**

$$LCB\left(\vec{b}\right) = \vec{o}\left(\vec{b}\right) - \omega\sigma\left(\vec{b}\right), \omega \in [0,3]$$

3. **Probability of Improvement (PoI).** **Promotes equally solutions that are likely to give very small improvements with high probability and those expected to lead to considerable improvement with small probability.**

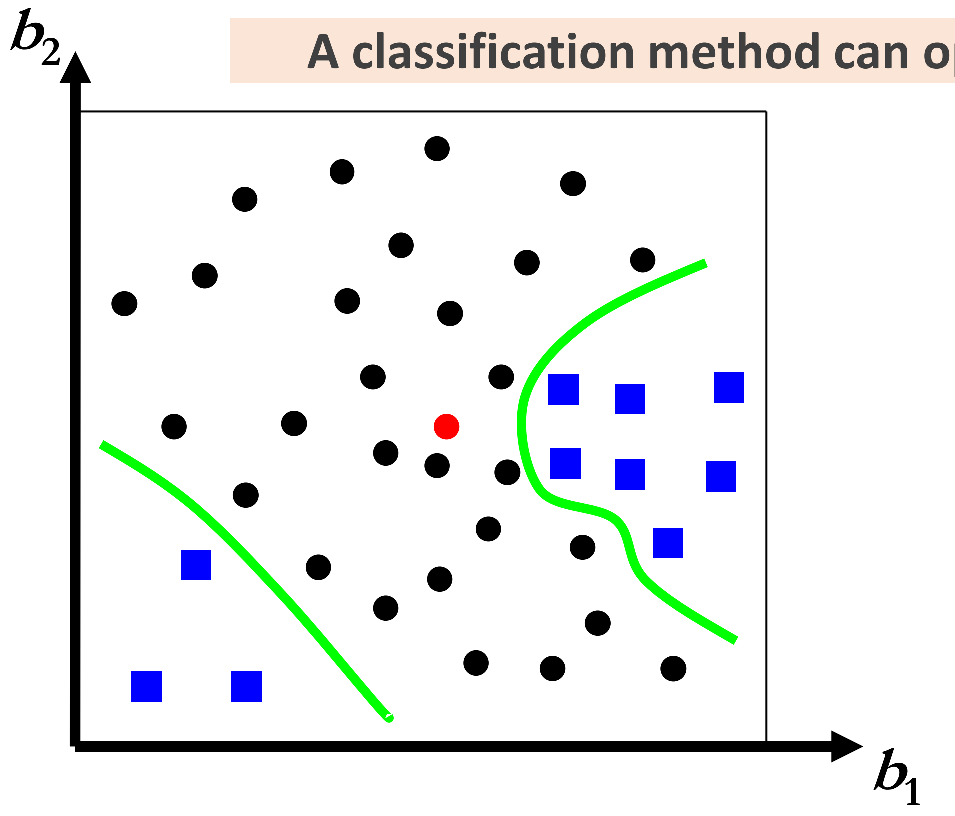
$$PoI\left(\vec{b}\right) = \int_{y=-\infty}^{f_{min}} PDF\left(\frac{y-\vec{o}\left(\vec{b}\right)}{\sigma\left(\vec{b}\right)}\right) dy$$

4. **Expected Improvement (ExI).** **Considers also the quantitative amount of improvement.**

$$ExI\left(\vec{b}\right) = \int_{y=-\infty}^{f_{min}} (f_{min} - y) PDF\left(\frac{y-\vec{o}\left(\vec{b}\right)}{\sigma\left(\vec{b}\right)}\right) dy$$

Use of Classification Methods in Constrained Optimisation (1/4)

A classification method can optionally be used in constrained optimisation!



● New Population Member

● Current DB Entries

■ Promising Neighbors

— Separator of feasible & infeasible solutions

How this information can be used:

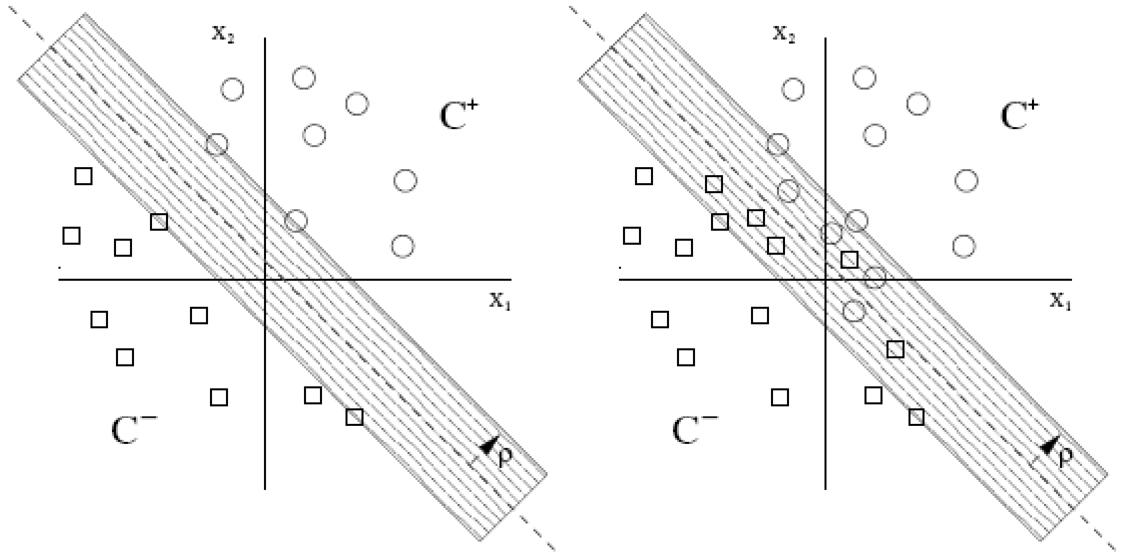
✓ Identify an infeasible solution and avoid using the metamodel to guess its objective function value.

✓ Avoid mixing feasible and infeasible solutions in the same training pattern set.

✓ Replace an interpolation scheme with an approximation one.

Classification using Support Vector Machines (SVM) (2/4)

Separation hyperplane: $\omega^T \vec{b} + \tau = 0$



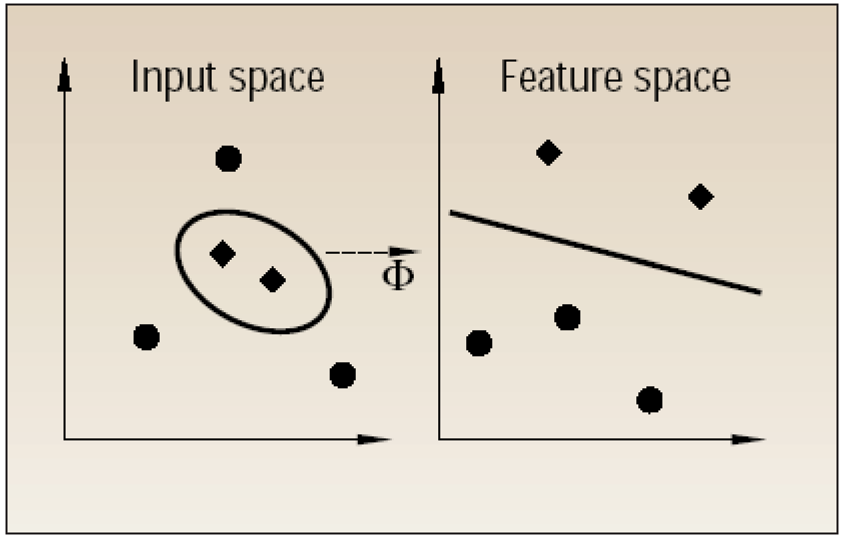
Decision function

$$D\left(\vec{b}^{(t)}\right) = \omega^T \vec{b} + \tau \geq 0, \text{ if } y^{(t)}=+1$$
$$D\left(\vec{b}^{(t)}\right) = \omega^T \vec{b} + \tau \leq 0, \text{ if } y^{(t)}=-1$$

Distance from the hyperplane: $Dist\left(\vec{b}\right) = \frac{D(\vec{b})}{\|\omega\|_2} = \frac{\omega^T \vec{b} + \tau}{\|\omega\|_2}$

$$\begin{cases} D\left(\vec{b}\right) > 0 \Rightarrow \vec{b} \in C^+ \\ D\left(\vec{b}\right) < 0 \Rightarrow \vec{b} \in C^- \end{cases}$$

Classification using Support Vector Machines (SVM) (3/4)



The idea of SVM: Map the training data non-linearly into a higher-dimensional feature space via Φ and construct a separating hyperplane with the maximum margin there. This yields a non-linear decision boundary in input space. By using the kernel function, it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space.

Devise a mapping function : $\Phi : \mathcal{R}^N \rightarrow F$

Perform the linear algorithm in F, through the evaluation of dot products: $k(\mathbf{x},\mathbf{y}) := (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$

Possible choices:

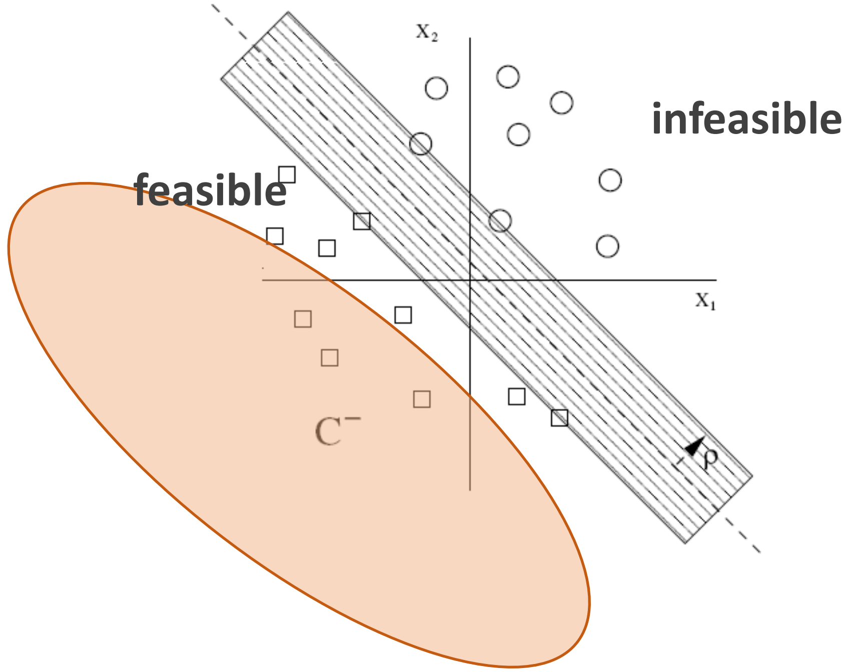
- **The *polynomial* kernel:** $k(\mathbf{x},\mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$
- **The *RBF* kernel:** $k(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2 / (2\sigma^2))$
- **The *sigmoid* kernel:** $k(\mathbf{x},\mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \Theta)$

M. Hearst, Support Vector Machines, IEEE Intelligent Systems, July-August 1998.

Constraint Handling in MAEAs through SVM (4/4)

Before the use of the RBFN in the LCPE phase of the MAEA, train/use an SVM to guess whether the new individual is feasible or infeasible:

- **the RBFN is used only for individuals marked as "feasible".**
- **Individuals marked as "infeasible" are penalised.**

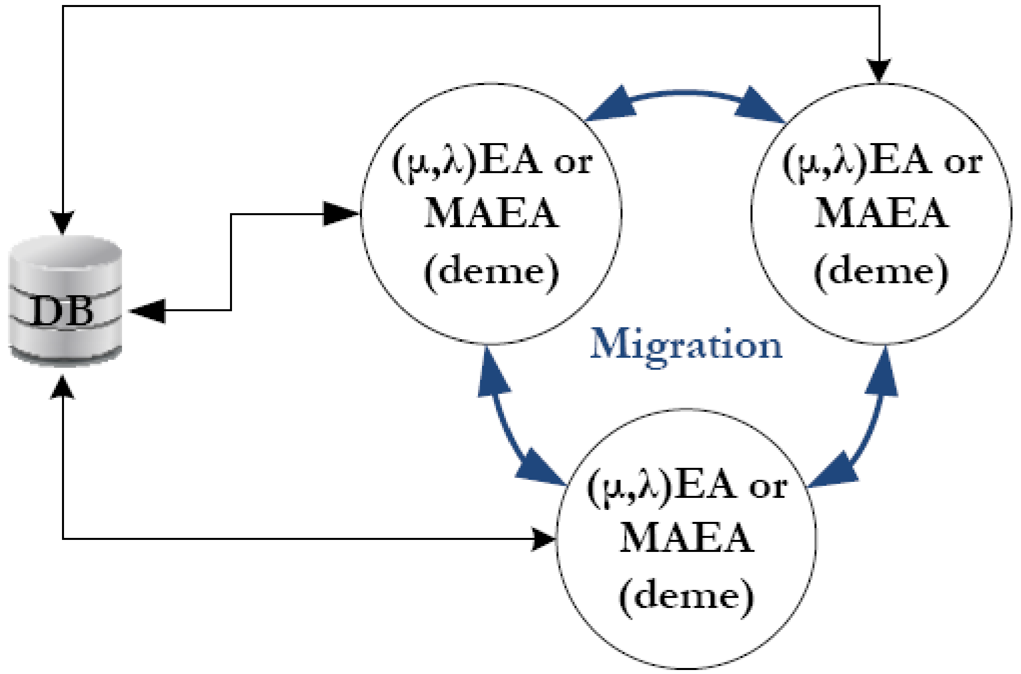


Distributed EAs (DEAs) and MAEAs

Simultaneously run more than one EAs, with different populations evolving in semi-isolation: by regularly exchanging promising etc individuals.

User-Defined Parameters:

● Number of demes or islands
● Communication topology
● Communication frequency
● Migration policy
● EA set-up per deme; exploration/exploitation oriented demes!

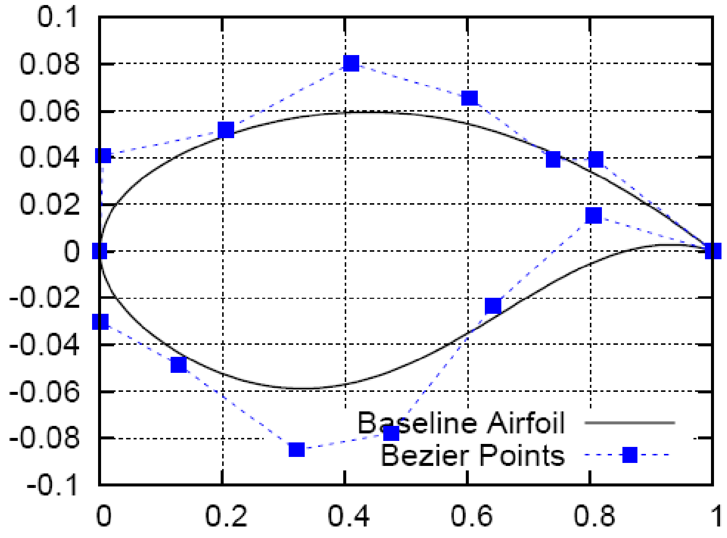


Common DB for all demes.

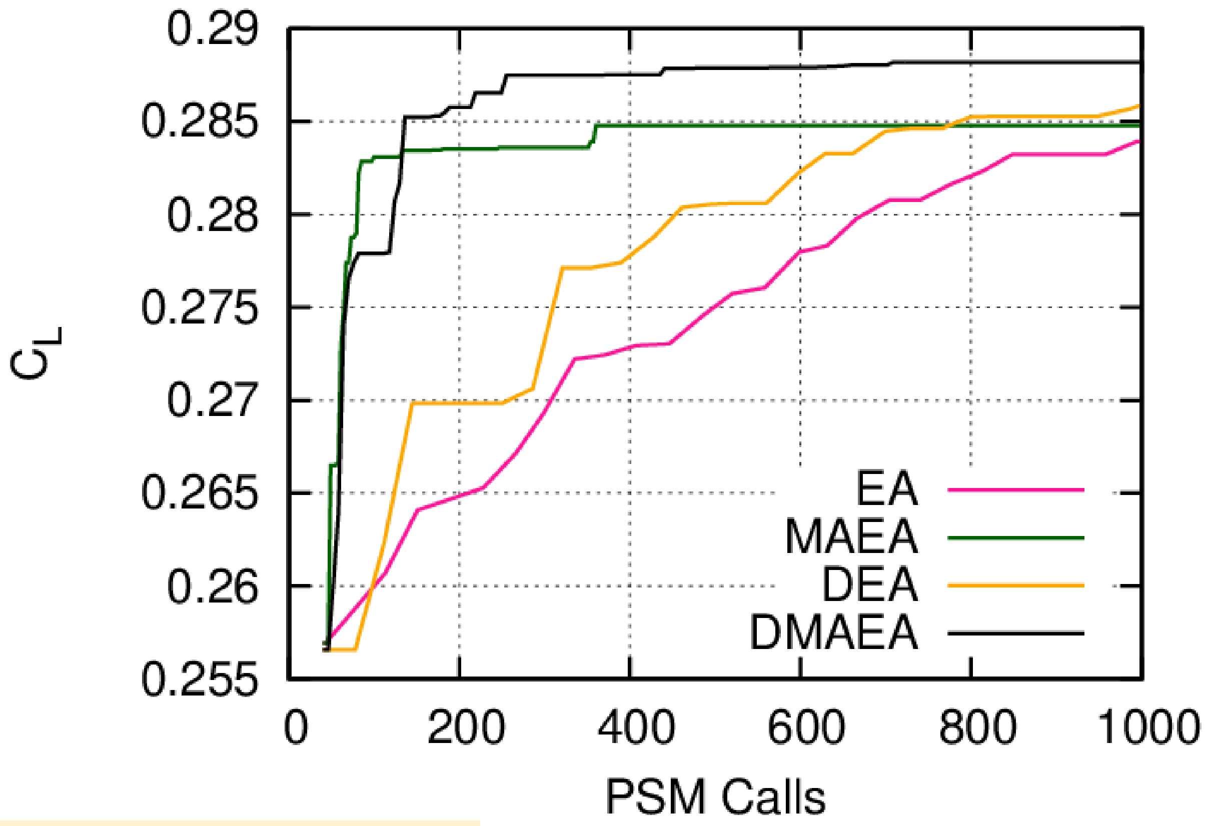
International Journal for Numerical Methods in Fluids, 53:455-469, 2007.

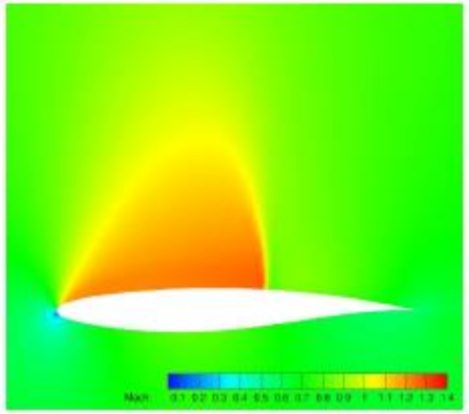
Demo Case A (SOO) – Case Definition

Shape optimisation (ShpO) of an isolated airfoil.
Target: max. Lift (C_L).
Inviscid flow, M_{inf}=0.40, a_{inf}=5°.
Parametrisation: 2 Bezier curves (8 Control Points each).
N=12 DoFs (internal control points moving in the y-direction).
Basis of Comparison: a (20,40)EA.

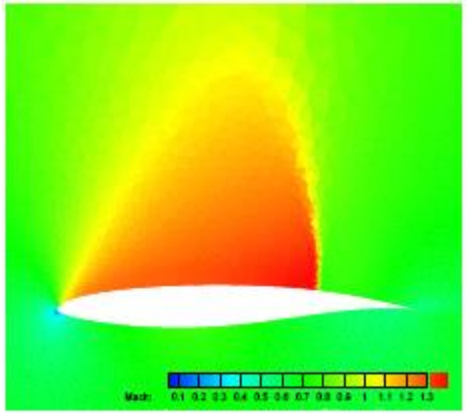


Demo Case A (SOO)





Starting



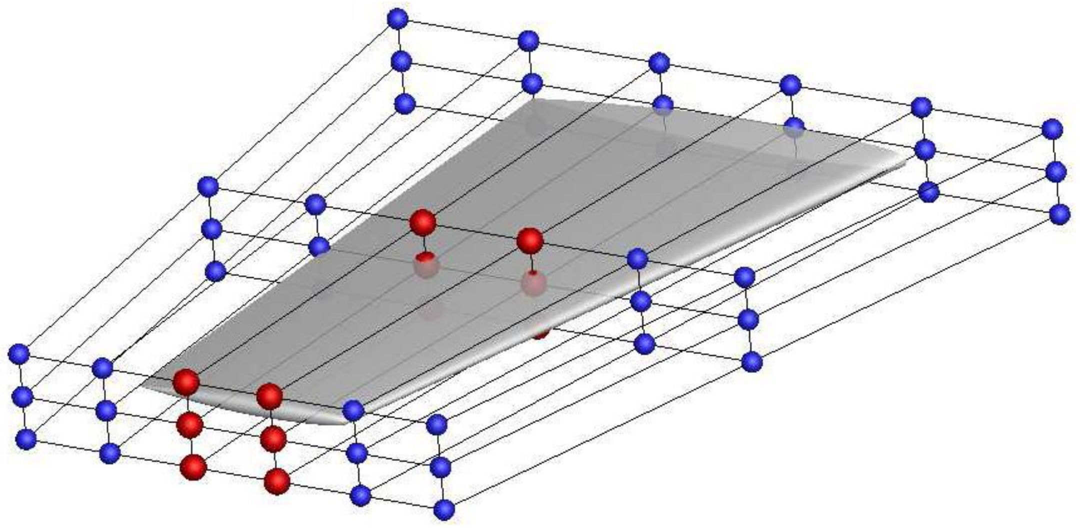
Optimal

The use of metamodels can reduce a lot the optimisation turn-around time or reach a better solution for a given computational cost. Distributed search is also beneficial.

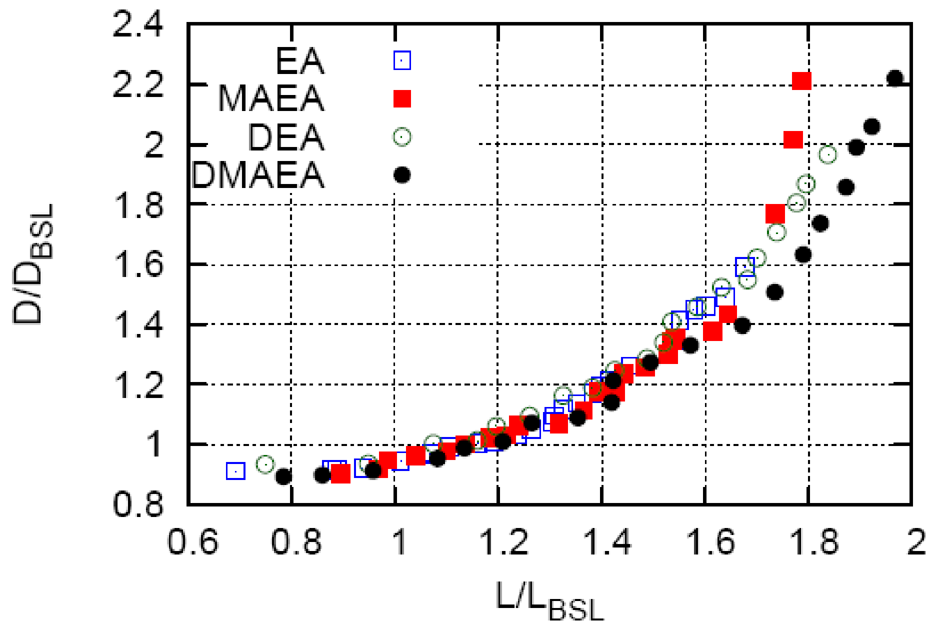
(20,40) EA
(20,40) MAEA, T^{MM}=40, λ_e=3
DEA or DMAEA: two demes (10,20)

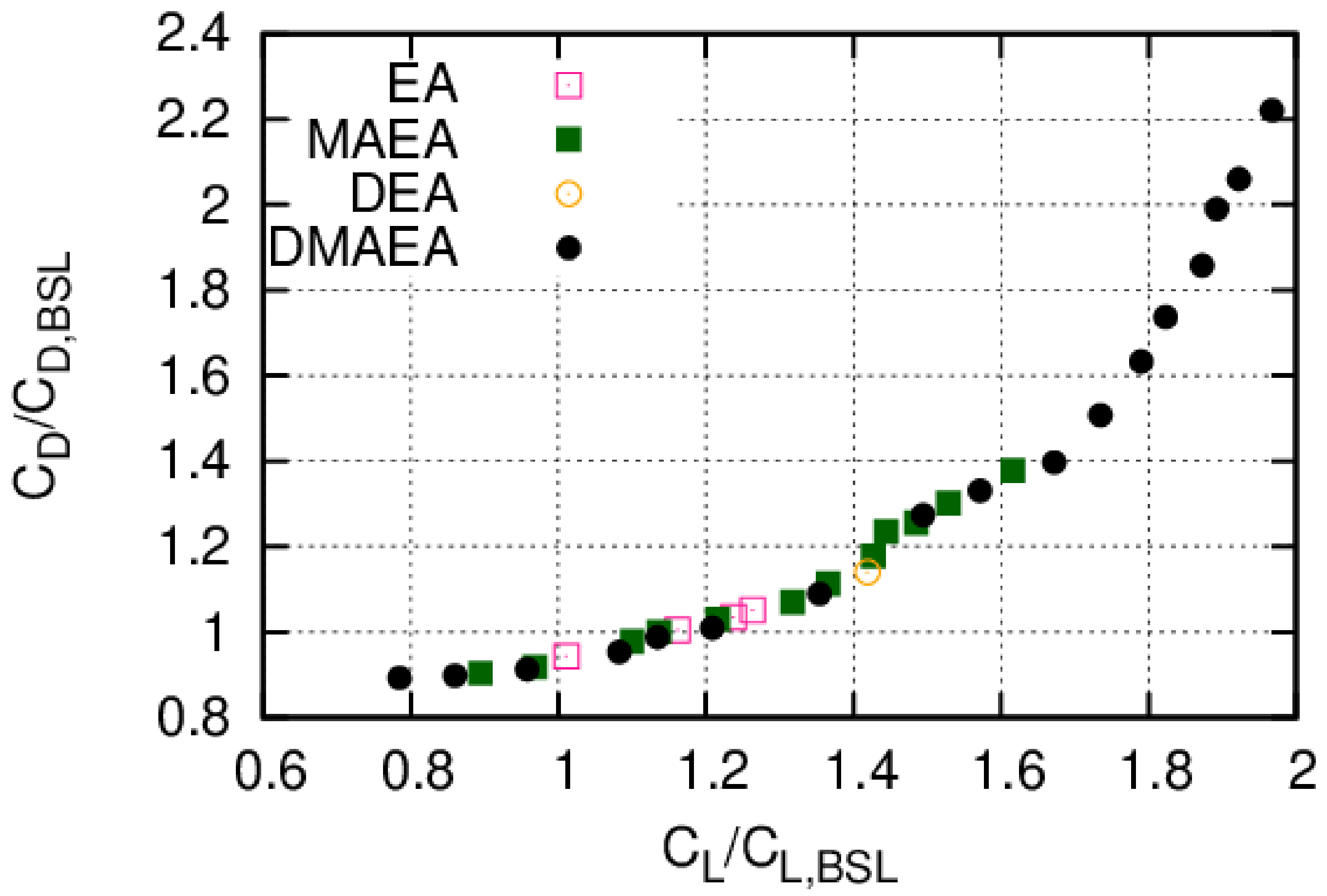
Demo Case B (MOO) – Case Definition

ShpO of an isolated wing (two objectives)
Target: max. Lift (L) and min. Drag (D)
Inviscid flow, M_{inf}=0.8394, $a_{pitch,inf}$=3.06°, $a_{yaw,inf}$=0°.
Parametrisation: 6x3x3 Volumetric NURBS Control Grid.
N=24 DoFs (internal control points moving normal to the planform).
<u>Basis of Comparison:</u> a (10,20)EA.



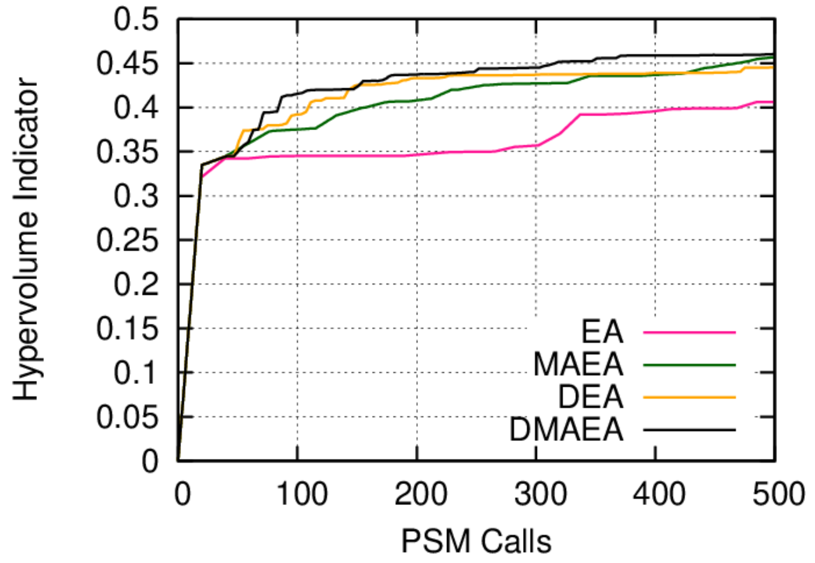
Demo Case B (MOO)

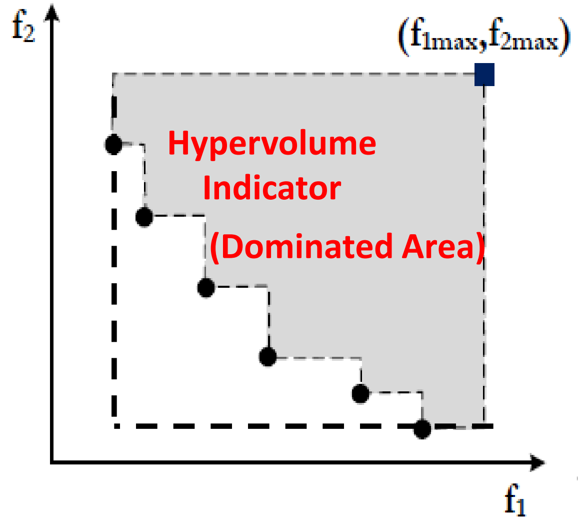




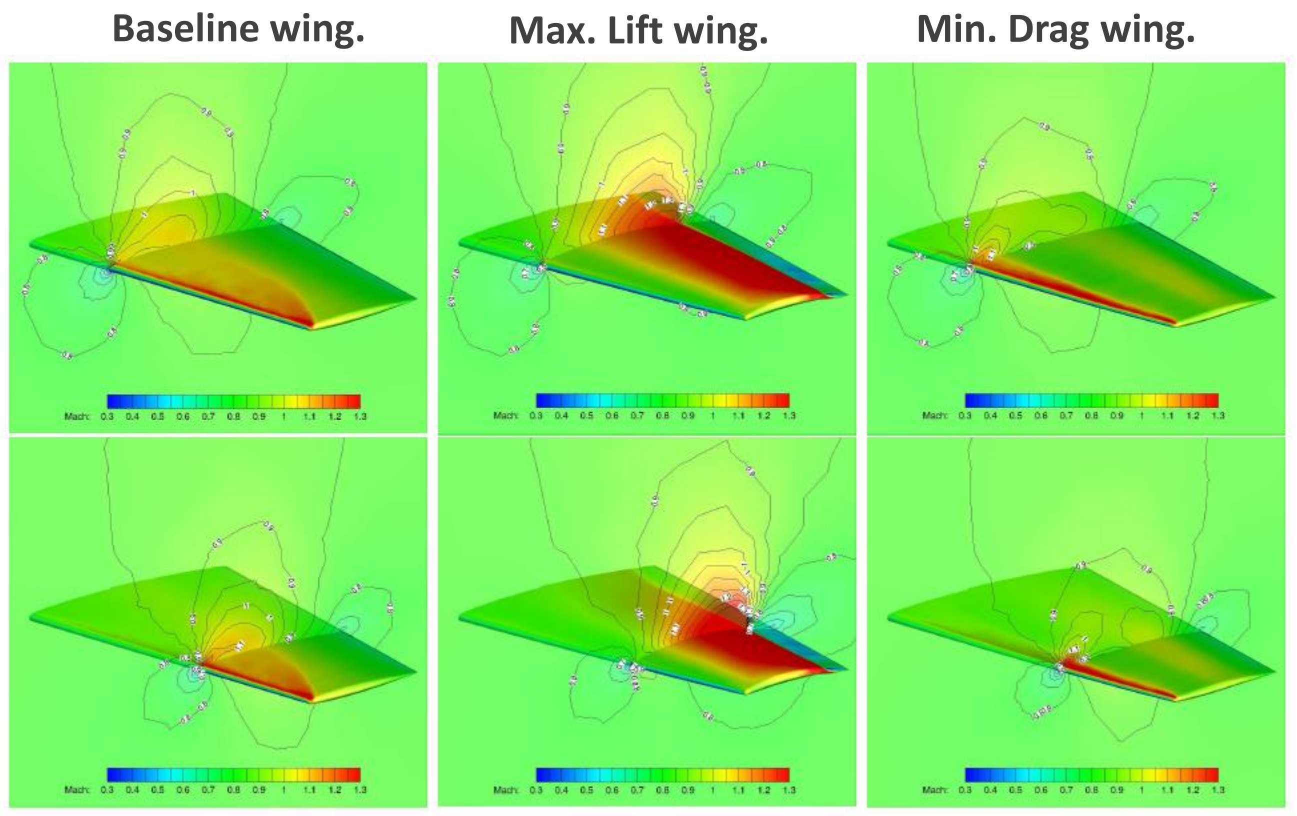
MOO problems may also benefit from the use of metamodels and distributed search.

Demo Case B (MOO)

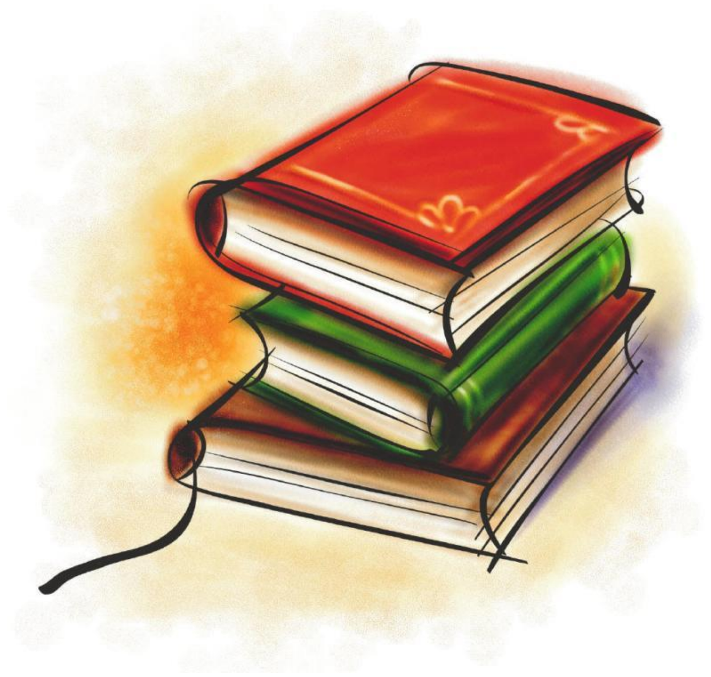




T^{MM} =30 evaluations on the PSM, before the LCPE phase.



Mach number fields on three geometries



MAEAs and the “Curse of Dimensionality”

(MA)EAs & the Curse of Dimensionality

Why EAs and MAEAs become very costly in problems with N>>?

- EAs: The evolution slows down in the presence of many DoFs (N>>).
- MAEAs: Difficult to build dependable metamodels. Need for many training patterns. Consequently:
  - The cost for training metamodel(s) increases a lot.
  - The start of the LCPE phase must be delayed.
  - The quality of metamodel-based predictions is not as good as it should.

Possible Remedies:

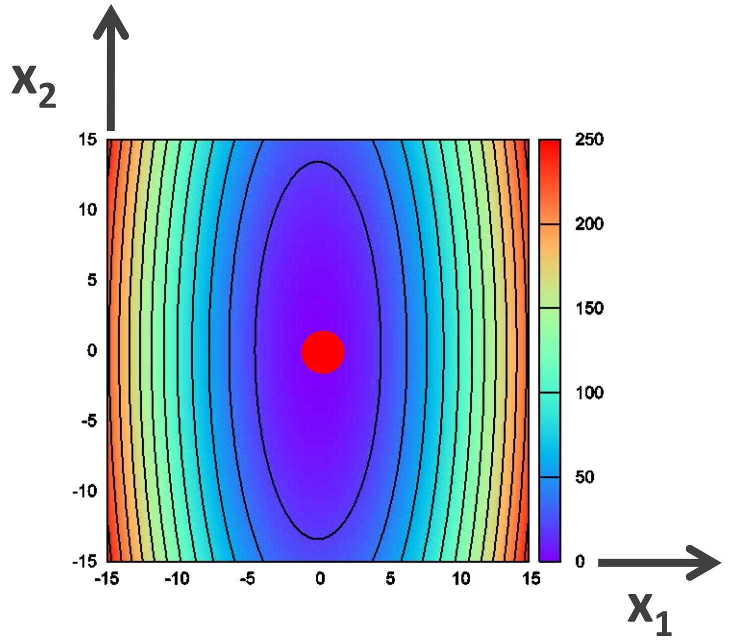
- Artificially transform the problem into a "more separable one".
- Perform the evolution in a different/transformed design space (feature space).
- Identify and exclude less-important DoFs from the metamodel training process.

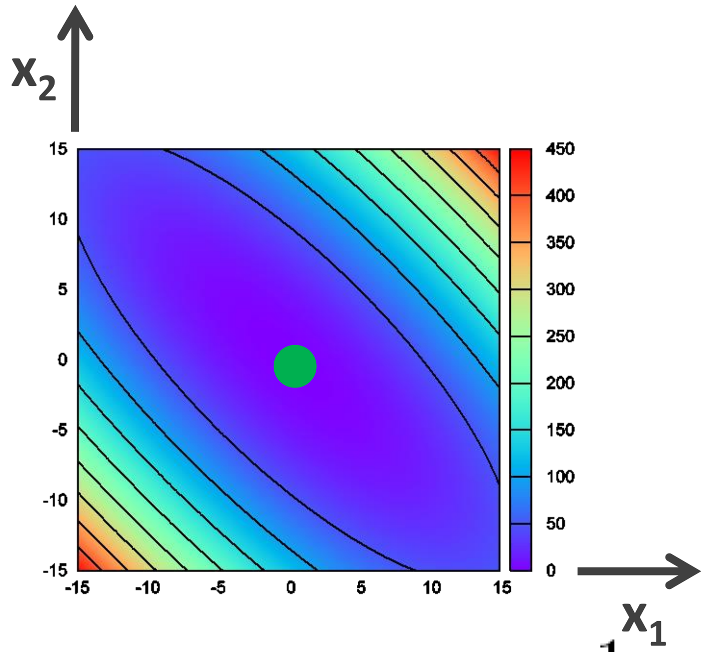
Engineering Optimization, 46:895-911, 2014.

Optimisation in Separable Problems

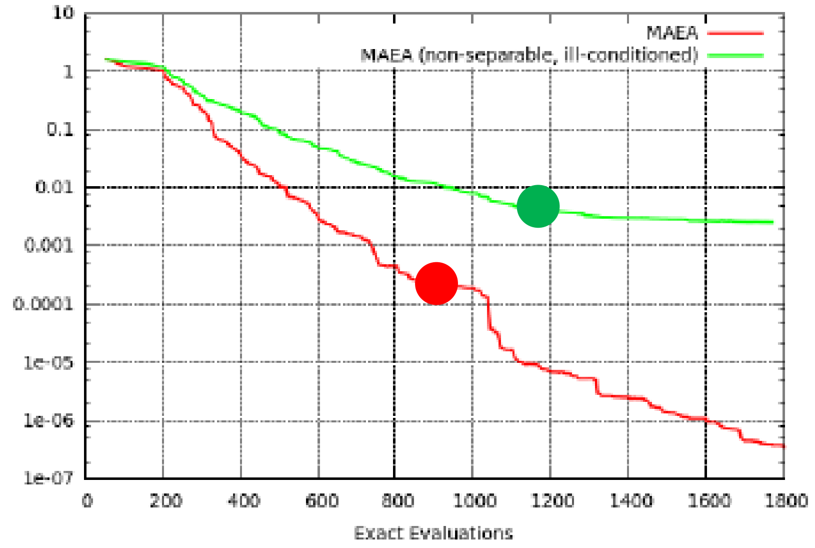
Separable (●) vs. Non-Separable (●) Problems:

$$G(x_1, x_2, \ldots, x_N) = g_1(x_1)g_2(x_2)\ldots g_N(x_N)$$





Example with 10 DoFs

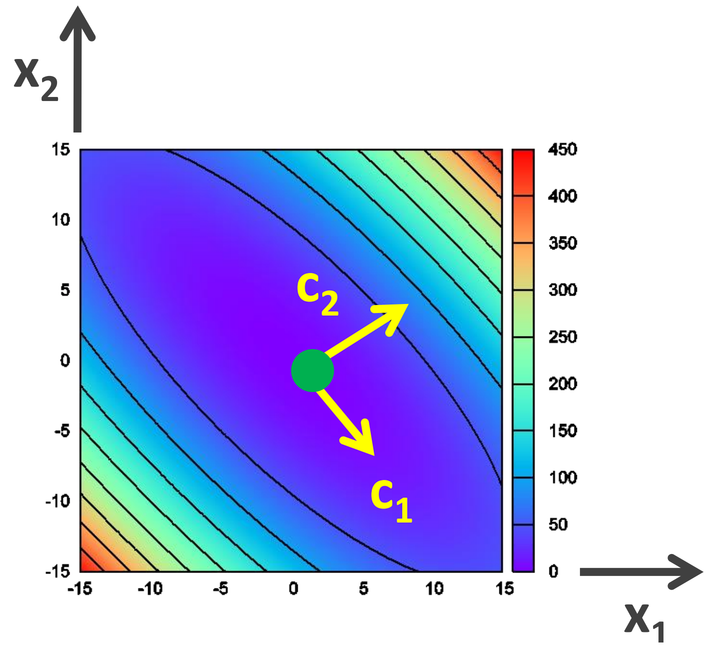


● $$F(\vec{x}) = x_1^2 + \frac{1}{9}x_2^2$$

● $$F(\vec{x}) = \left[x_1 cos(-\frac{\pi}{4}) - x_2 sin(-\frac{\pi}{4})\right]^2 + \frac{1}{9}\left[x_1 sin(-\frac{\pi}{4}) + x_2 cos(-\frac{\pi}{4})\right]^2$$

How can we efficiently solve Non-Separable Optimisation Problems?

Perform a "rotation" (how??) and make the EA solve for (c_1,c_2), instead of (x_1,x_2).
Principal Component Analysis (PCA) can help us!



$$F(\vec{x}) = \left[x_1 cos(-\frac{\pi}{4}) - x_2 sin(-\frac{\pi}{4})\right]^2 + \frac{1}{9}\left[x_1 sin(-\frac{\pi}{4}) + x_2 cos(-\frac{\pi}{4})\right]^2$$

(MA)EAs & the Curse of Dimensionality – Towards MAEA-PCA (Linear PCA)

- **Let X be a set of M observations $\vec{x} \in \mathbb{R}^N$ of possibly correlated variables, formed by the λ (M=λ) members of the current offspring population.**
- **Make them have zero mean and unit standard deviation:**

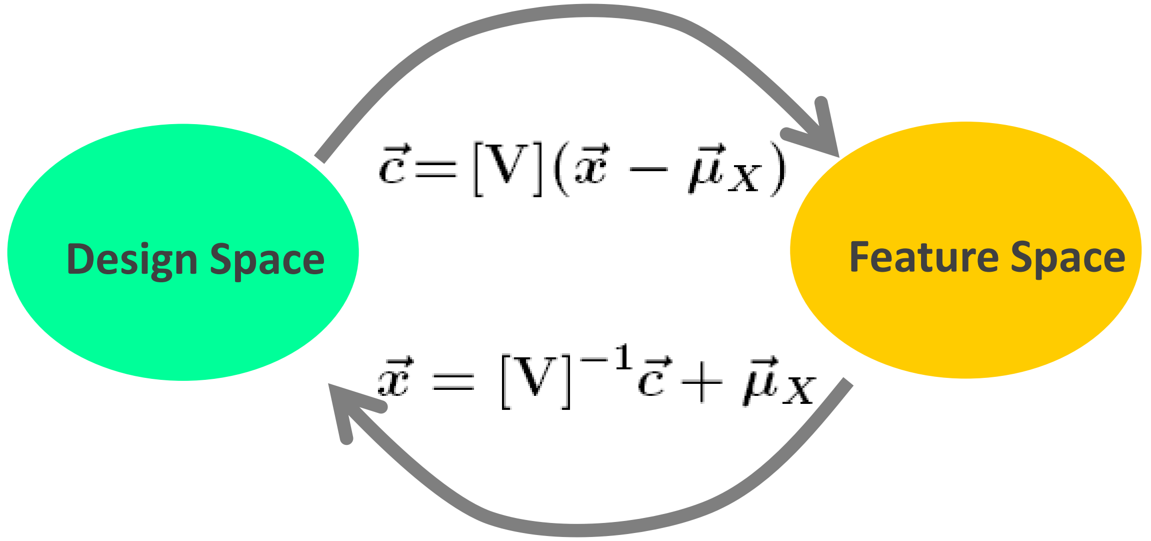
$$E[[X]] = 0 \qquad E[[X]^2] = 1$$

- **Compute the covariance matrix:**

$$[P]_{N\times N} = [P] = \frac{1}{M}[X][X]^T$$

- **Eigen-Decomposition:**

$$[P] = [V][\Lambda][V]^T$$



$$\vec{c} = [V](\vec{x} - \vec{\mu}_X)$$

$$\vec{x} = [V]^{-1}\vec{c} + \vec{\mu}_X$$

where the eigenvectors are the principal components defining the feature space and the eigenvalues are their variances.

(MA)EAs & the Curse of Dimensionality – Towards MAEA-PCA (Kernel PCA)

- **Transformation from R^N to R^L, through a non-explicitly defined mapping** $\phi : \mathbb{R}^N \rightarrow \mathbb{R}^L$
- **Compute the (new) covariance matrix:**

$$P_{\tau\sigma} = \frac{1}{M} \sum_{i=1}^{M} \phi_\tau(\vec{x}^i)\phi_\sigma(\vec{x}^i), \;\; \tau, \sigma \in [1, L]$$

- **Overcome the definition of ϕ by the kernel trick:**

$$K_{ij} = k(\vec{x}^i, \vec{x}^j) = \vec{\phi}^T(\vec{x}^i)\vec{\phi}(\vec{x}^j) = \sum_{p=1}^{L} \phi_p(\vec{x}^i)\phi_p(\vec{x}^j)$$

or

$$[\mathrm{K}] = [\Phi]^T [\Phi]$$

where

$$k(\vec{x}^i, \vec{x}^j) = exp\left(- \frac{||\vec{x}^i - \vec{x}^j||_2^2}{2\sigma^2} \right)$$

(MA)EAs & the Curse of Dimensionality – Towards MAEA-PCA (Kernel PCA)

- **Eigen-Decomposition:** $$[\mathrm{P}] = [\mathrm{V}][\Lambda][\mathrm{V}]^T$$

- **Assume that:** $$[\mathrm{V}] = [\Phi][\mathrm{A}]$$

and, thus: $$[\mathrm{K}][\mathrm{A}] = M[\mathrm{A}][\Lambda]$$

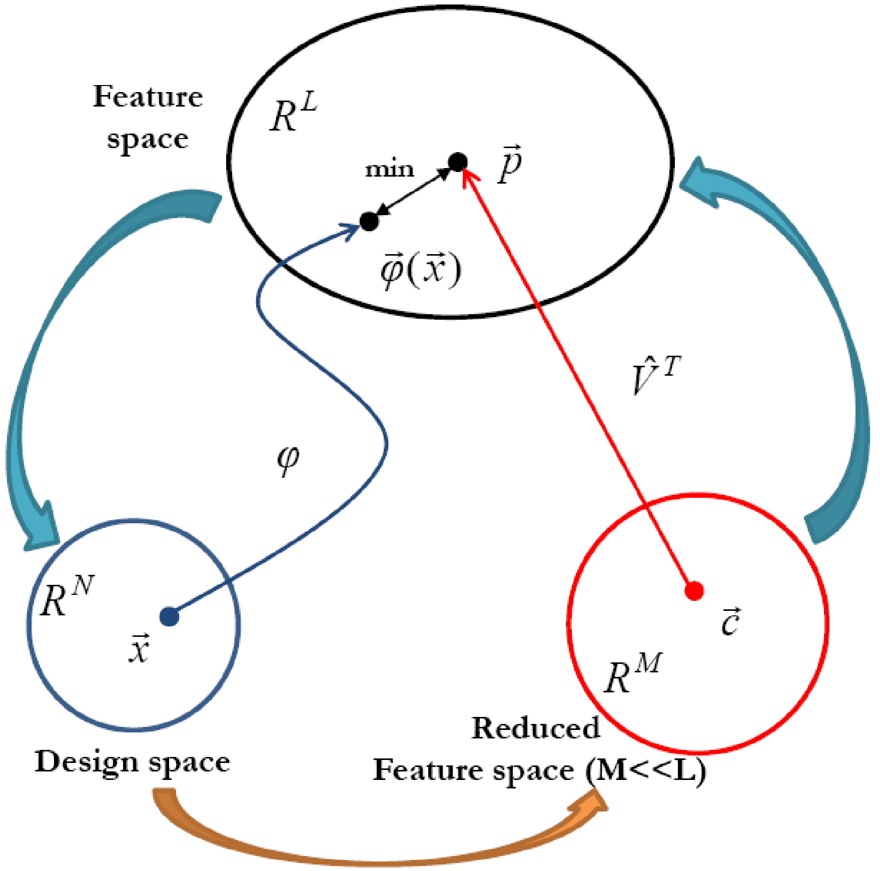
- **Matrices A and L are truncated (M elements are retained; M is user-defined) and a smaller eigen-problem is solved (here M=λ<<L).**
- **In the new feature space:**

$$\vec{c}(\vec{x}) = [\hat{\mathrm{V}}]^T \vec{\phi}(\vec{x}) \Leftrightarrow c_i(\vec{x}) = \sum_{m=1}^{M} a_{im} exp\left(- \frac{||\vec{x} - \vec{x}^m||_2^2}{2\sigma^2} \right)$$

Applied Soft Computing, *64:1-13, 2018.*

(MA)EAs & the Curse of Dimensionality – Towards MAEA-PCA (Kernel PCA)

Transformation from the design to the feature space and vice-versa:



$$\vec{x}^{new} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{M} c_i a_{ij} exp\left(-\frac{||\vec{x}^j-\vec{x}^{old}||_2^2}{2\sigma^2}\right)\vec{x}^j}{\sum_{i=1}^{M}\sum_{j=1}^{M} c_i a_{ij} exp\left(-\frac{||\vec{x}^j-\vec{x}^{old}||_2^2}{2\sigma^2}\right)}$$

D/MA/EA-PCA: Demo Case B (Revisited)

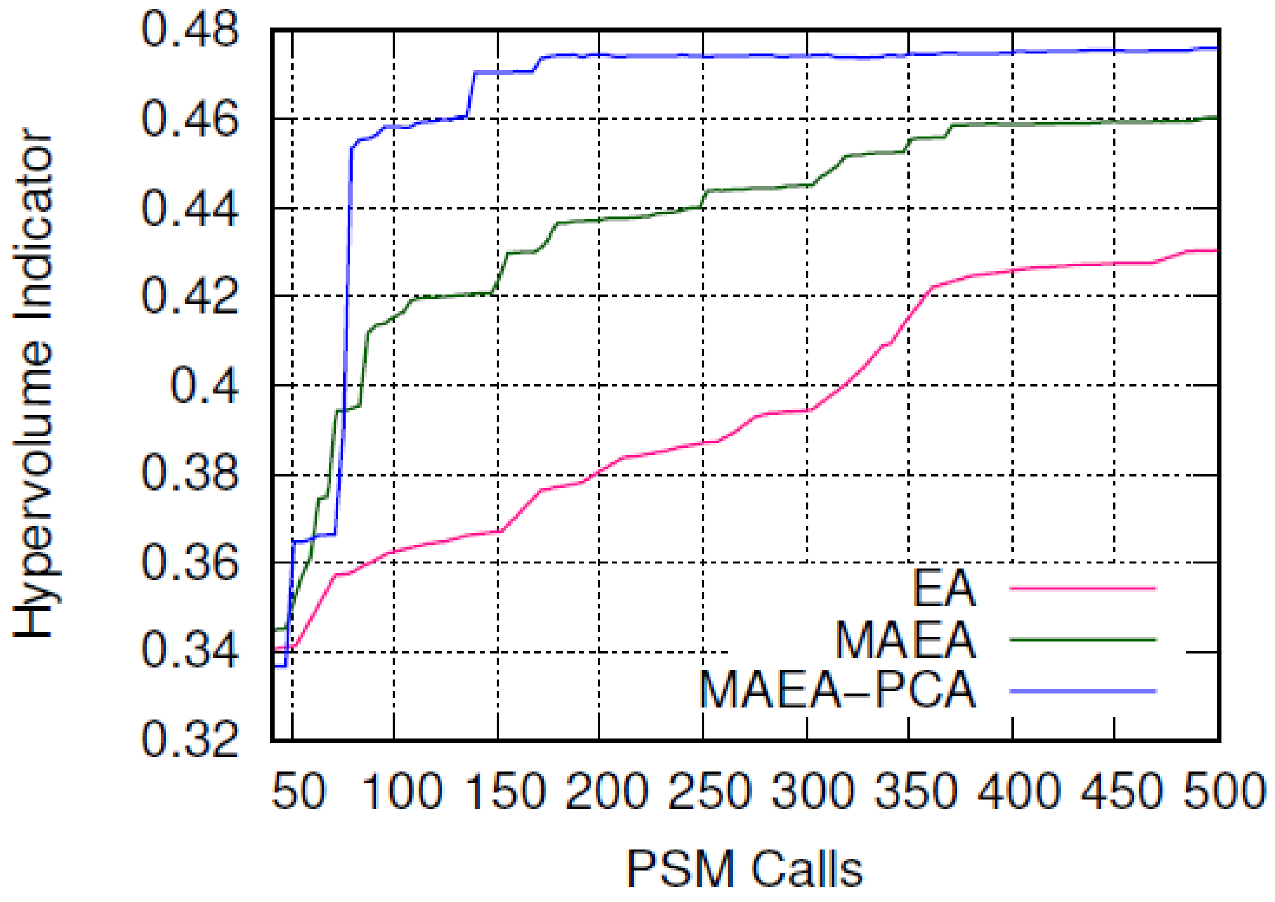
ShpO of an isolated wing

Target: max. Lift (L) and min. Drag (D)

New Flow Conditions: M_{inf}=0.8395, $a_{pitch,inf}$=3.06°, $a_{yaw,inf}$=0°.

N=24 DoFs

Basis of Comparison: a (10,20)EA.



There is additional benefit from the use of PCA in an already fast MAEA.

D/MA/EA-PCA: Demo Case C (SOO) – Case Definition

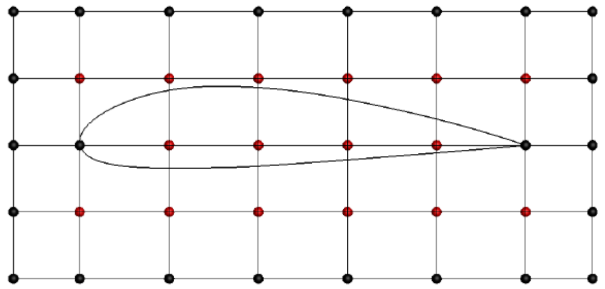
SOO & MOO of the NACA4318 isolated airfoil.

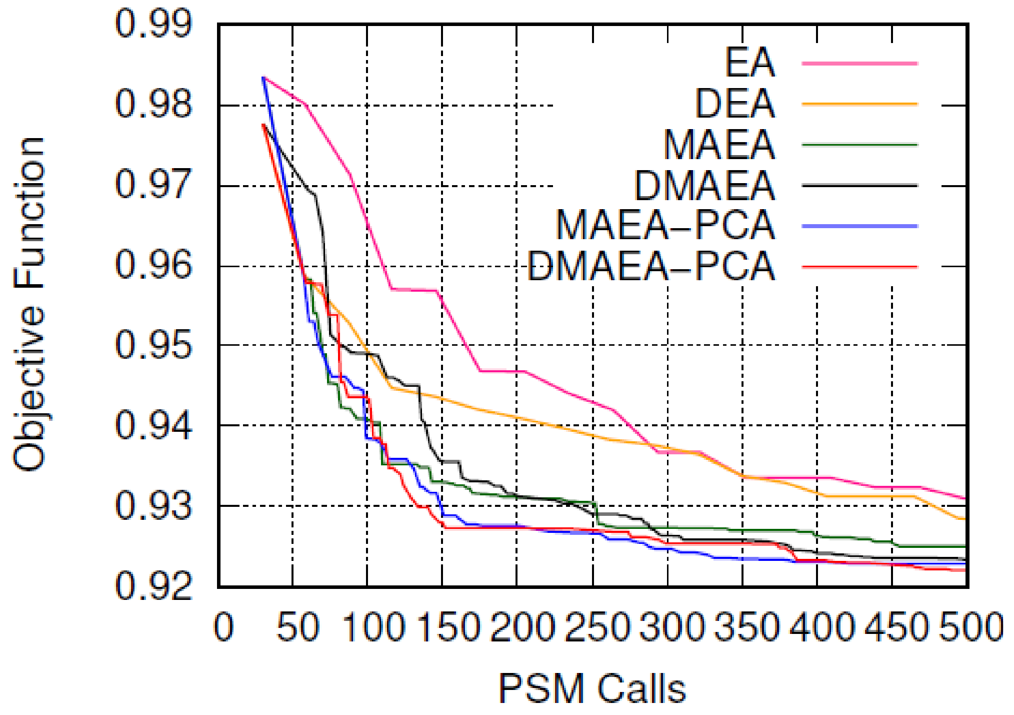
M_∞=0.13, α_∞ =2.2°, Re=3.8 × 10^6.

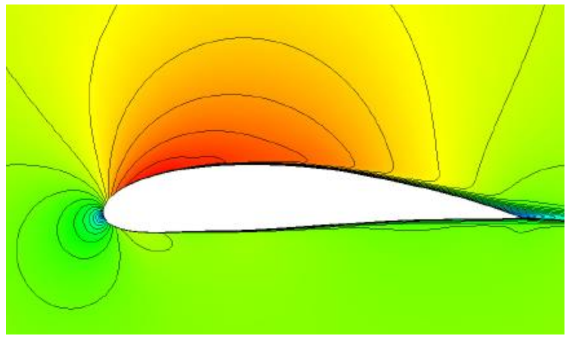
Controlled by the 8×5 control box. N=32 DoFs.

Targets: C_L & C_D. Weighted sum in the SOO.

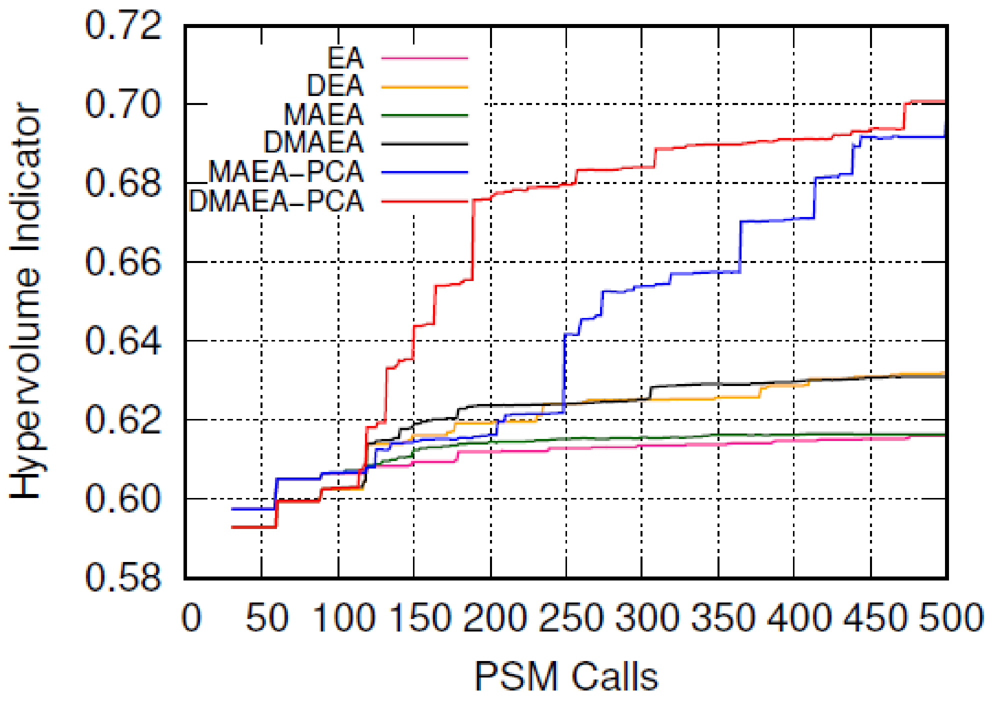
Basis of Comparison: a (10,30)EA.

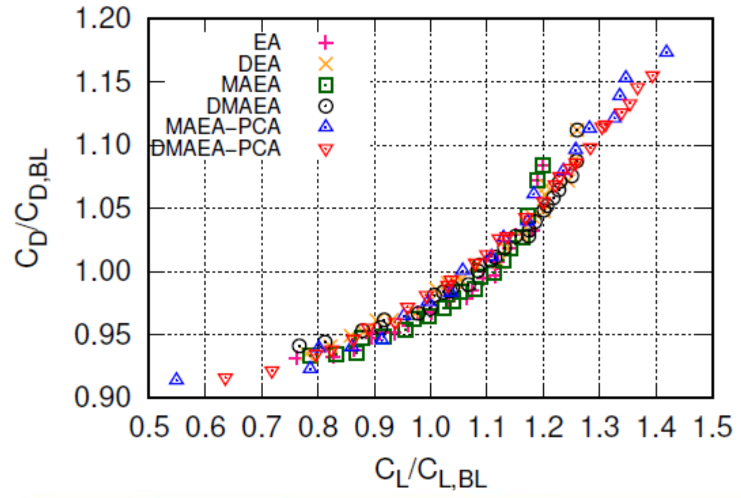


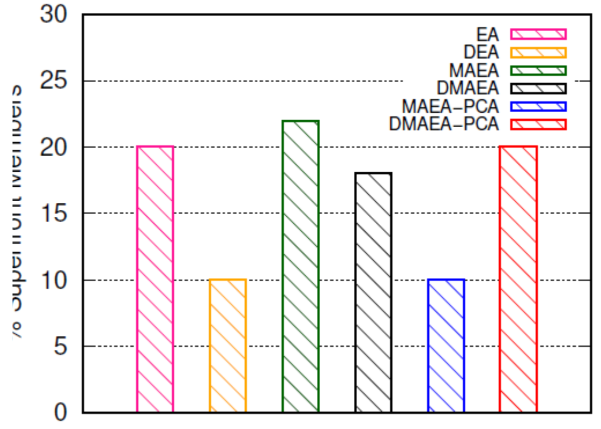




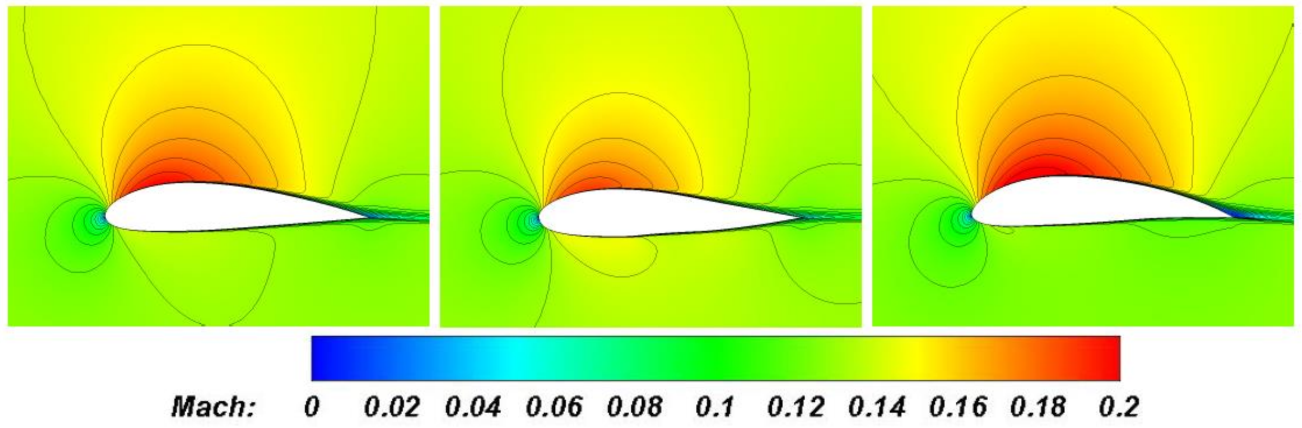
D/MA/EA-PCA: Demo Case C (MOO)



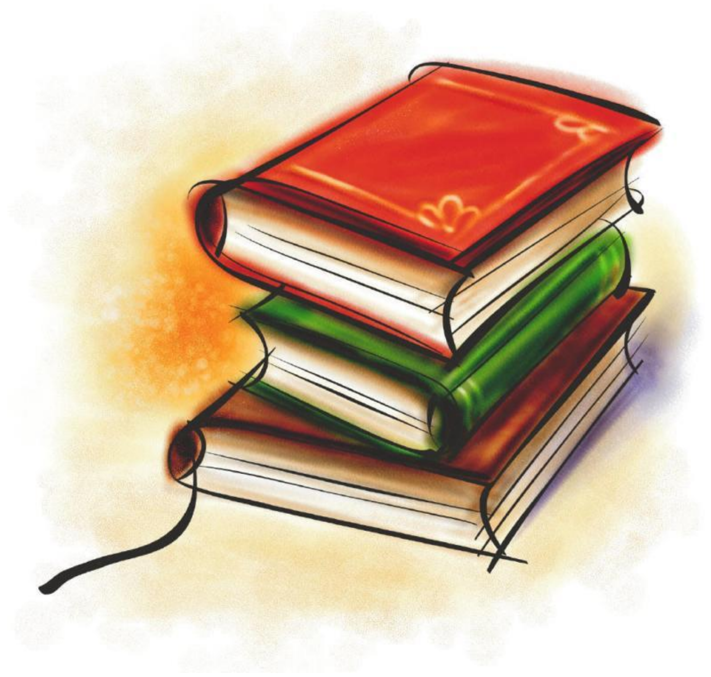




Reconfirming the additional benefit from the use of PCA in the already fast MAEA in MOO problems.



Reference shape **Optimised for min. C_D** **Optimised for max. C_L**



Multi-Criteria Decision-Making

Multi-Criteria Decision Making (MCDM) & EAs or MAEAs (1/3)

Multi-Criteria Decision Making (MCDM):

Cost reduction of EAs or MAEAs by directly focusing ("a priori", instead of "a posteriori") to the Decision Maker (DM) preferences.

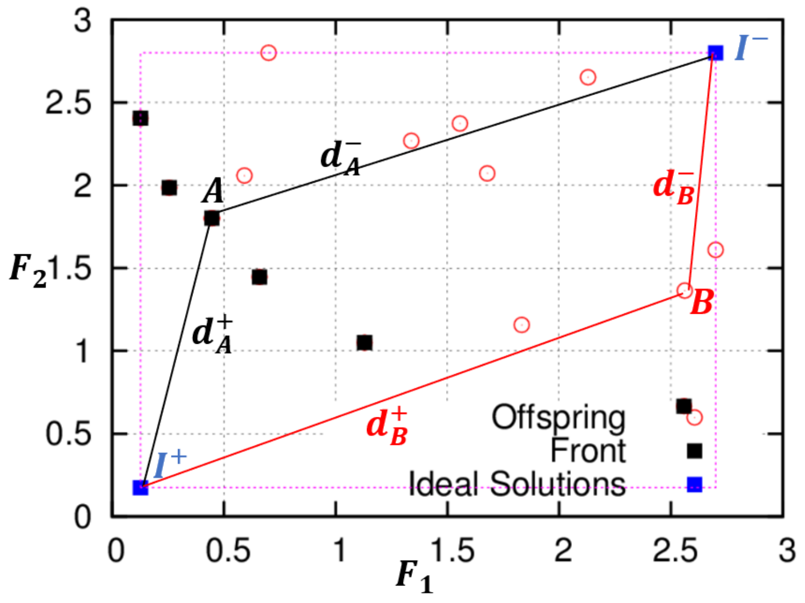
MCDM techniques:

- **TOPSIS: Technique for Order of Preference by Similarity to Ideal Solution.**
- **PROMETHEE: Preference Ranking Organization METHod for Enrichment Evaluations.**
- **WS or WP: Weighted Sum or Weighted Product.**
- **AHP: Analytic Hierarchy Process.**
- **VIKOR: VIseKriteruska Optimizaca I Komoromisno Resenje.**

Multi-Criteria Decision Making (MCDM) & EAs or MAEAs (2/3)

Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) as the MCDM method

1. Construct the Decision Matrix (P); its elements quantify the performance of the i solution of the MOO problem over the j objective function.
2. Construct the normalized decision matrix (r).
3. Construct the matrix V by processing matrix r with the weights articulated by the DM.
4. Compute two ideal solutions, the positive (I^+) and the negative (I^-) one.
5. Compute the Euclidean distances $d^{+/-}$ (in the objective space) of each solution from I^+ and I^-.
6. Compute the relative distance D_i of each solution from I^+ and I^-.



$$P = \begin{bmatrix} p_{1,1} & \cdots & p_{1,M_o} \\ \vdots & & \vdots \\ p_{M_e,1} & \cdots & p_{M_e,M_o} \end{bmatrix}$$

$$v_{i,j} = \frac{p_{i,j}}{\sqrt{\sum_{i=1}^{M_e} p_{i,j}^2}} w_j$$

$$d_i^{+/-} = \sqrt{\sum_{j=1}^{M_o} \left(v_{ij} - v_j^{+/-}\right)^2}$$

$$D_i = \frac{d_i^-}{d_i^- + d_i^+}$$

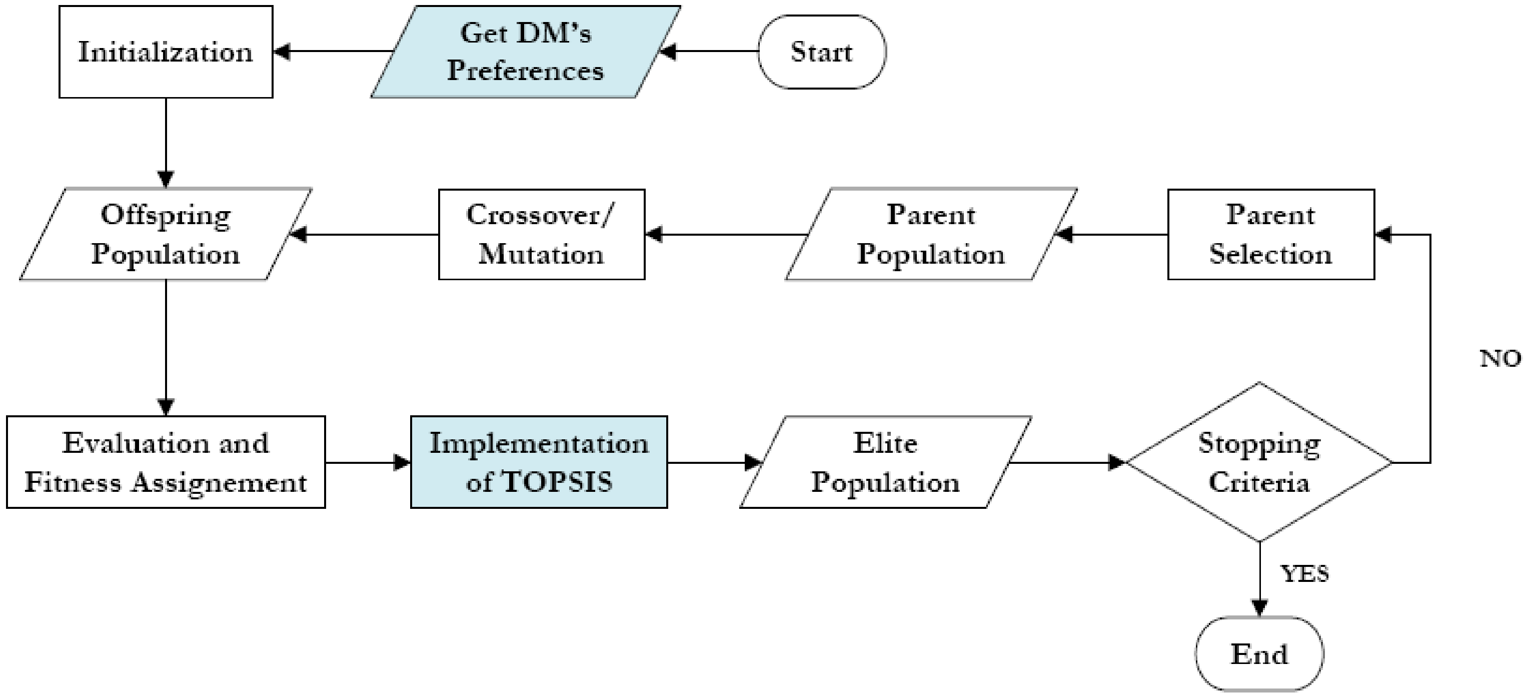
C.-L. Hwang, Multiple Attribute Decision Making: Methods and Applications. Springer-Verlag, New York, 1981.

Multi-Criteria Decision Making (MCDM) & EAs or MAEAs (3/3)

TOPSIS-driven EAs and MAEAs:

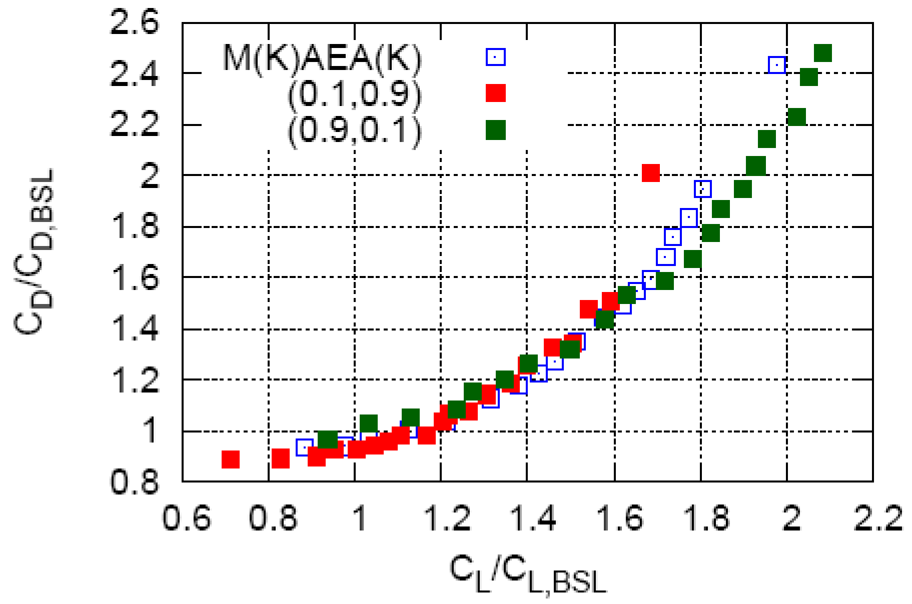
- Based on the a priori articulated DM preferences (i.e. weights per objective), TOPSIS processes the front of elite members in each generation and, through simple mathematical operations, identifies the most appropriate members according to the DM preferences.

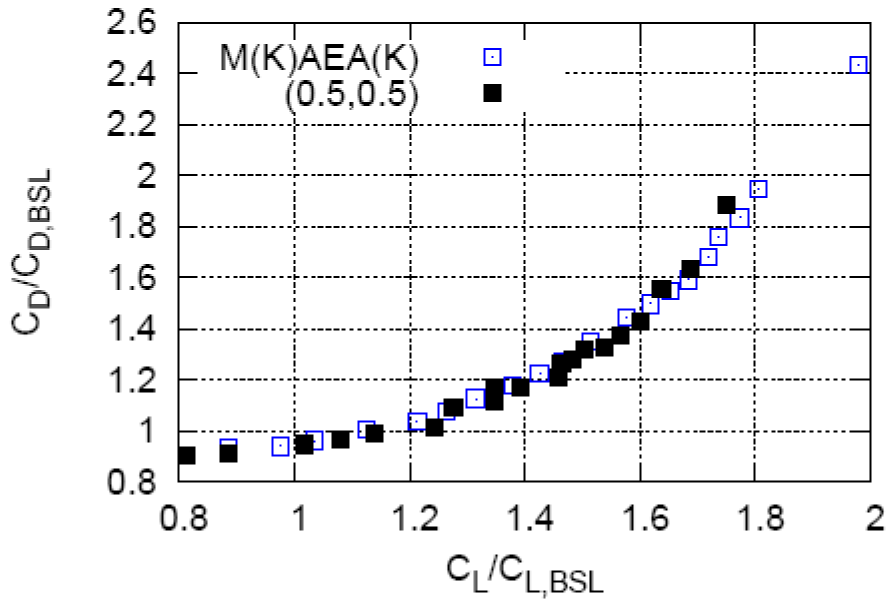
- During the evolution, TOPSIS is used instead of implementing dominance and sharing criteria to rank the offspring population in each generation and compute the scalar utility function.



Applied Soft Computing, 64:1-13, 2018.

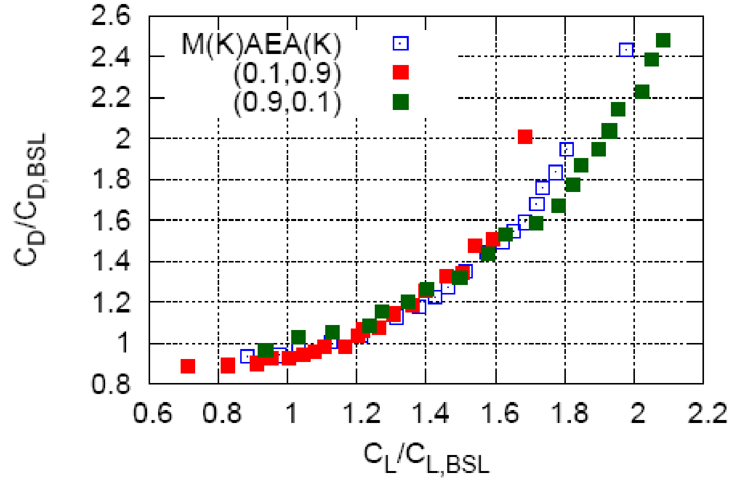
Demo Case B Revisited using the MCDM (TOPSIS) in M(K)AEA(K) (1/3)

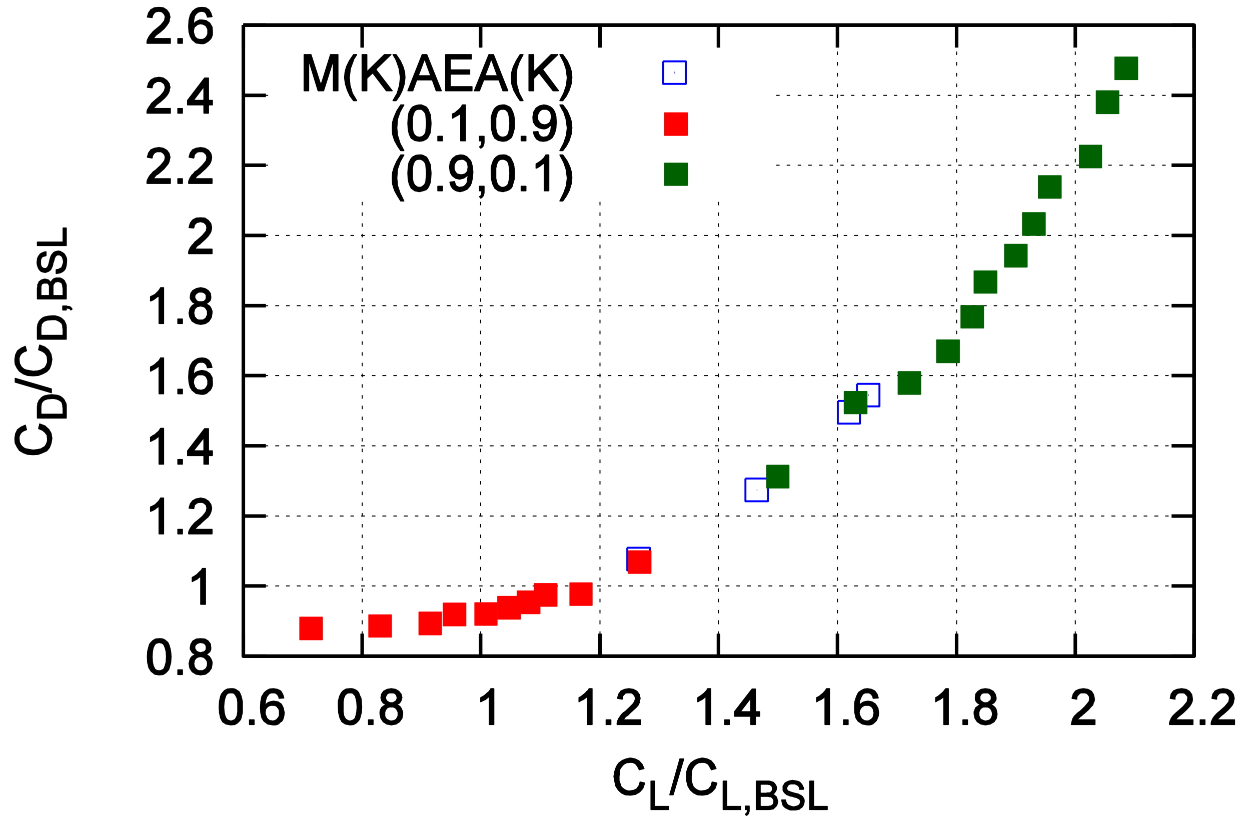




Computations with the same budget.

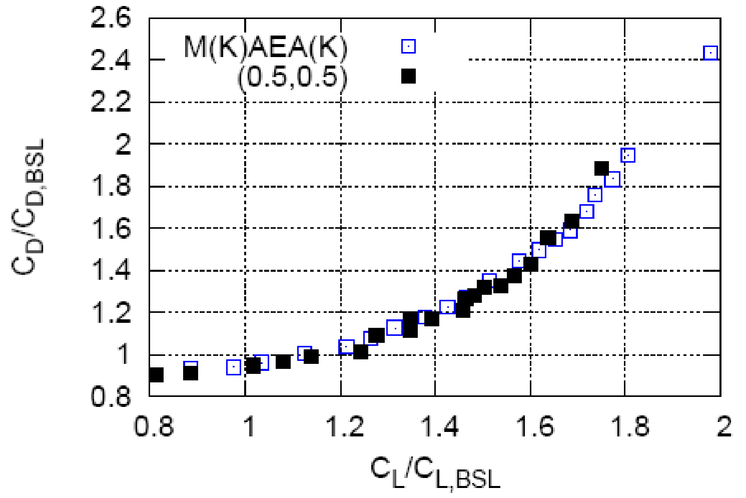
Demo Case B Revisited using the MCDM (TOPSIS) in M(K)AEA(K) (2/3)

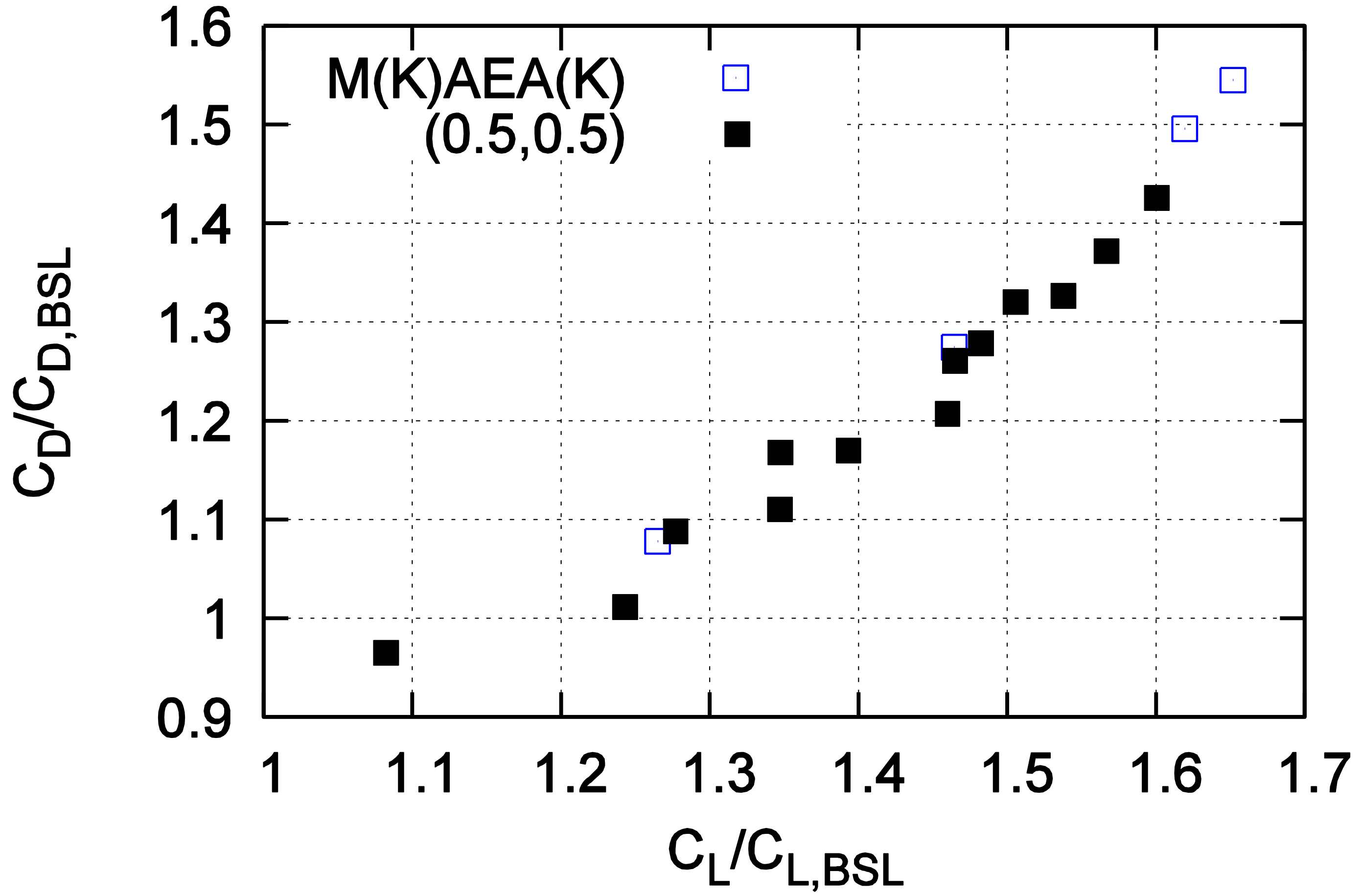


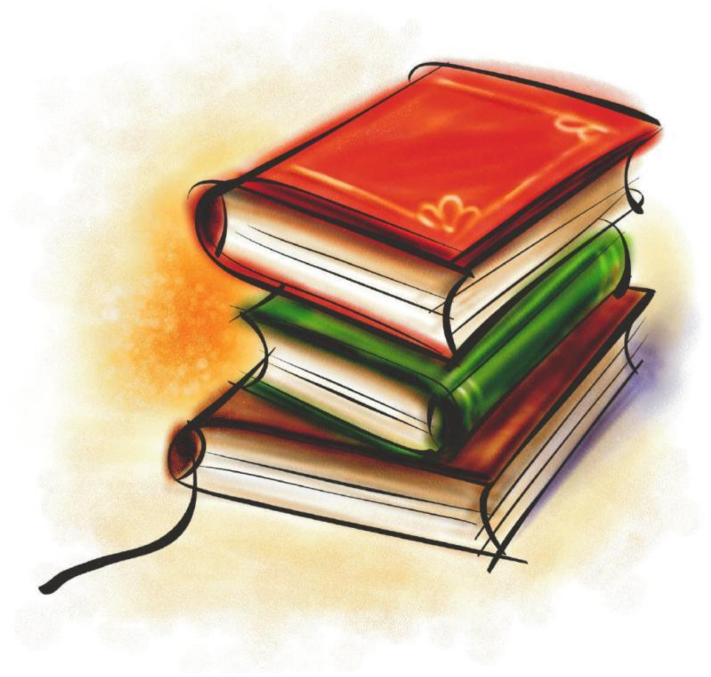


The a priori DM preferences may move the computed Pareto front towards the areas of the objective space indicated by the DM.

Demo Case B Revisited using the MCDM (TOPSIS) in M(K)AEA(K) (3/3)







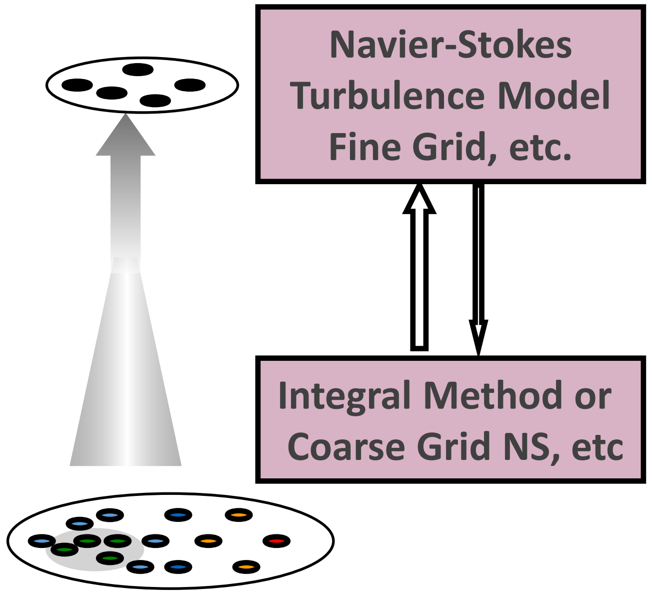
Hierarchical/Multilevel Schemes

Hierarchical/Multilevel Optimisation Schemes

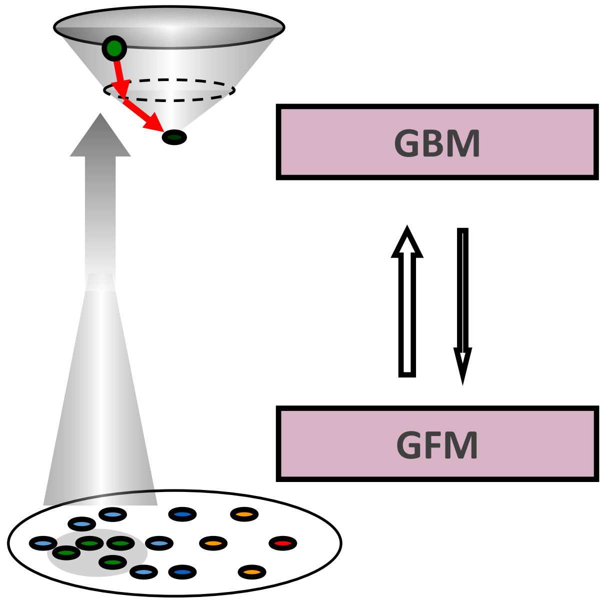
Splitting the optimisation at different (usually two!) levels, allows the combined use of evaluation models of different accuracy and computational cost, different search methods (EAs, MAEAs, DMAEAs, etc. or gradient-based ones) and/or different sets/subsets of design variables.

Exploitation vs. Exploration-based levels.

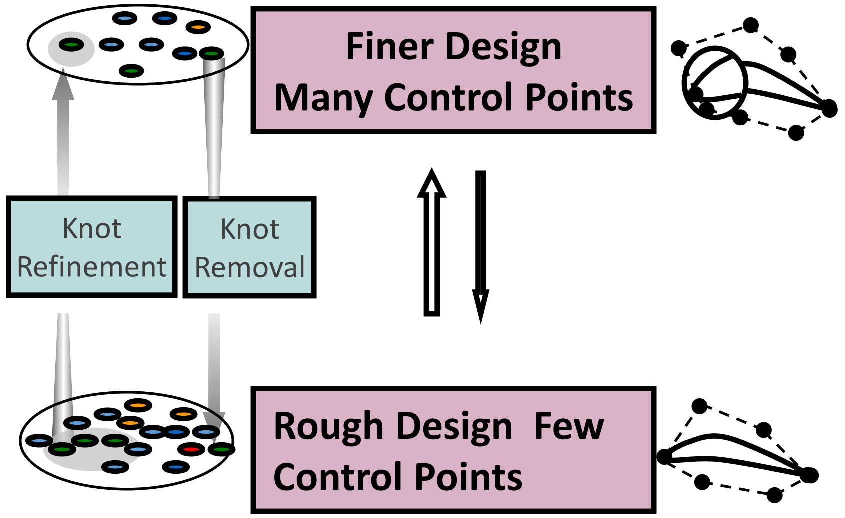
Hierarchical Evaluation



Hierarchical Search

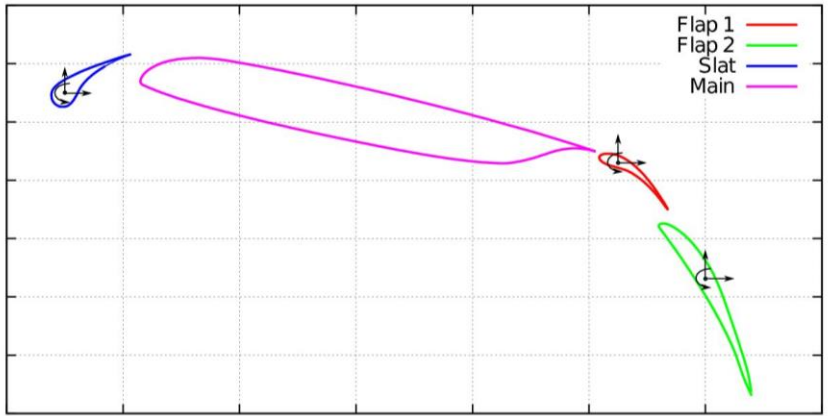


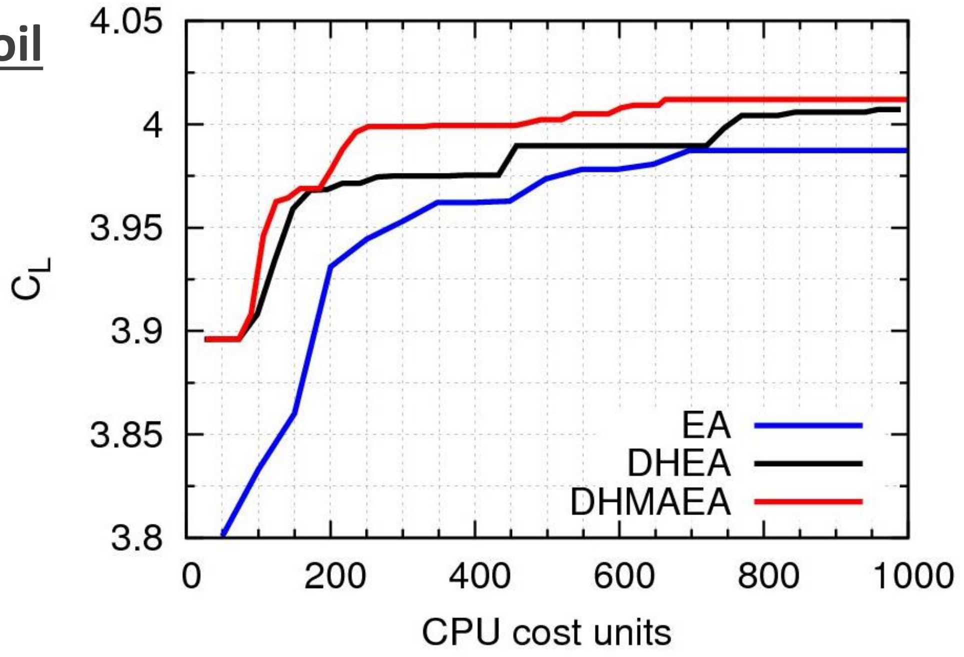
Hierarchical Parametrisation



Hierarchical Evaluation

Example:Optimal Deployment of a 4-Element Airfoil
Target: max Lift.
9 design variables.





Hierarchical EA combining a single- and a double-precision GPU-enabled Navier-Stokes solver (SPA & DPA, respectively).

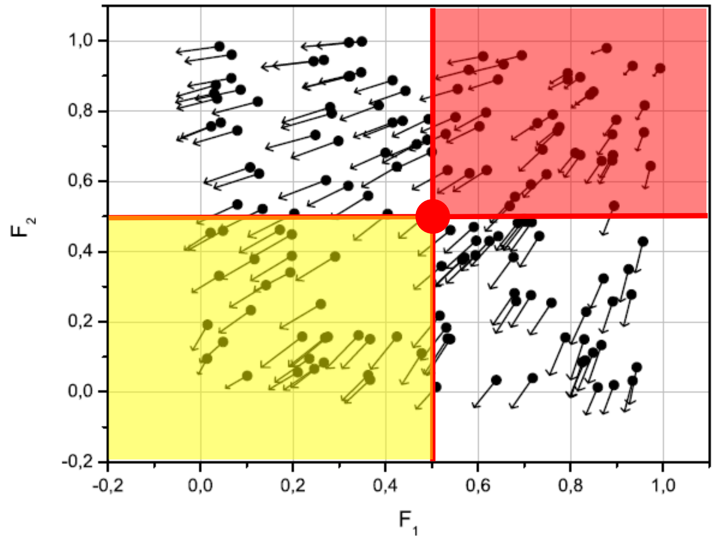
Comparison of:

(EA)= (20,50)EA on DPA.
(DHEA)= 3 demes x (5,15)EA using SPA for the λ=15 offspring & DPA for the top 2 members of each generation.
(DHMAEA)= 3 demes x (5,15)MAEA; during LCPE λ_e=9 members are re-evaluated on the SPA & one is re-re-evaluated on the DPA.

Hybrid GFM/GBM for MOO Problems (1/4)

- **EAs for exploring the design space and adjoint-based GBMs for refining the best solution(s) in each generation.**
- **Question: In which direction should an individual descend in a MOO?**

$$f(\vec{x}) = \sum_{m=1}^{M_o} w_m f_m(\vec{x}) \Rightarrow \frac{\partial f}{\partial x_j} = \sum_{m=1}^{M_o} w_m \frac{\partial f_m}{\partial x_j}$$



Hybrid GFM/GBM for MOO Problems (2/4)

● **Method M1:** Compute the utility function Φ using SPEA2

$$\Phi(\vec{x}) = \Phi(\vec{F}(\vec{x}), \{\vec{F}(\vec{z}) \mid \vec{z} \in \mathcal{P}^g \setminus \{\vec{x}\}\}) \in \mathcal{R}$$

as the sum of a strength (S) and a density (D) function, or

$$\Phi_j = \frac{1}{2+D_j} + \sum_{k \in \mathcal{K}_j} S_k \mathcal{H}(\sum_{i=1}^{M_o} \mathcal{H}(f_i^j - f_i^k) + 0.5 - M_o)$$

where K_j is the pop. set excl. j and $\mathcal{H}$ is the Heaviside function.
The Heaviside function is replaced by a (differentiable) sigmoid function and

$$w_m = \frac{\partial \Phi}{\partial f_m}$$

Computational Methods in Applied Mechanics and Engineering, 197(33-40):2963-2975, 2008.
Engineering Optimization, 44(2):157-173, 2012.

Hybrid GFM/GBM for MOO Problems (3/4)

● **Method M2:** **Computes the descent direction by performing the LPCA of the objective function values of the elite members in each generation.**

After solving the eigen-problem of the covariance matrix, objective function values f_i are transformed into the feature space (c_i):

$$\vec{c} = \vec{c}(\vec{f}) = \mathrm{U}(\vec{f} - \vec{\mu}), \ \ \vec{c} \in \mathbb{R}^{M_o}$$

and:

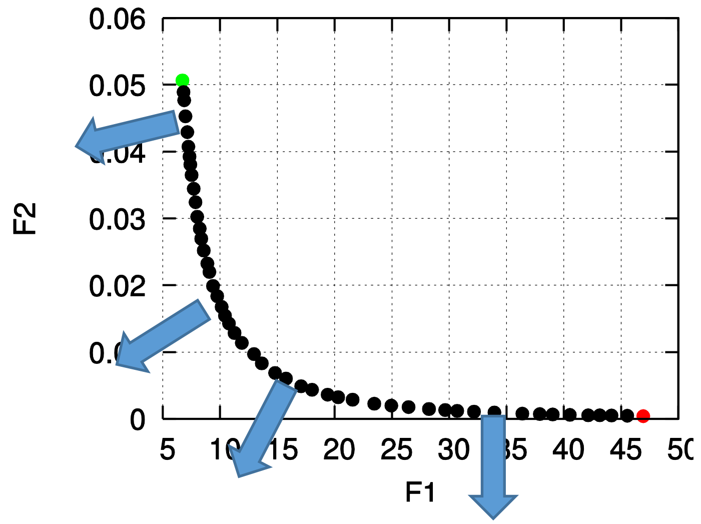
$$\frac{\delta \vec{c}}{\delta \vec{x}} = \mathrm{U} \frac{\delta \vec{f}}{\delta \vec{x}}$$

The M_O-th row of the above matrix is used to upgrade the selected λ_{GB} individuals, by displacing them along the direction with the smallest variance ("normal" to the current front).

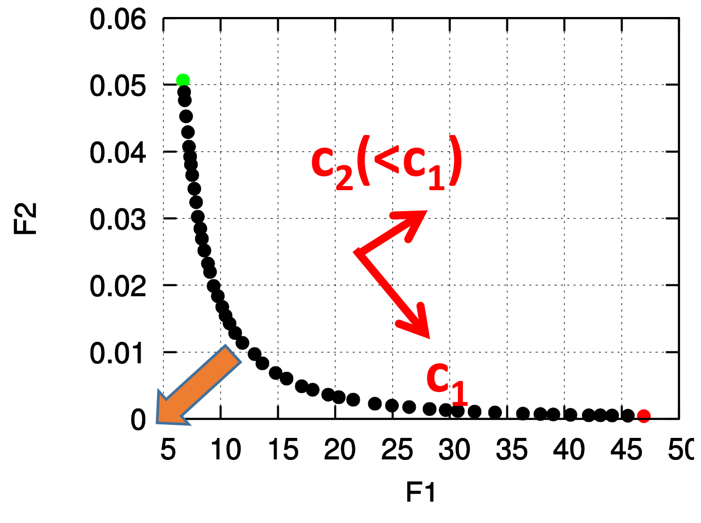
Applied Soft Computing, 64:1-13, 2018.

Hybrid GFM/GBM for MOO Problems (4/4)

- **Comparison of Methods M1 & M2:**



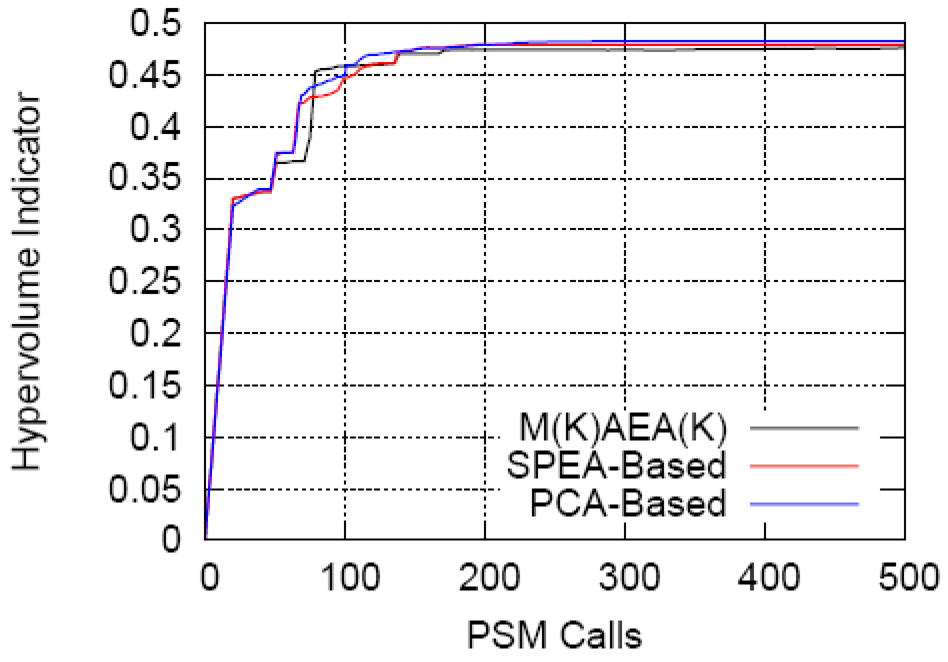
Method M1 (*2008, 2012*)

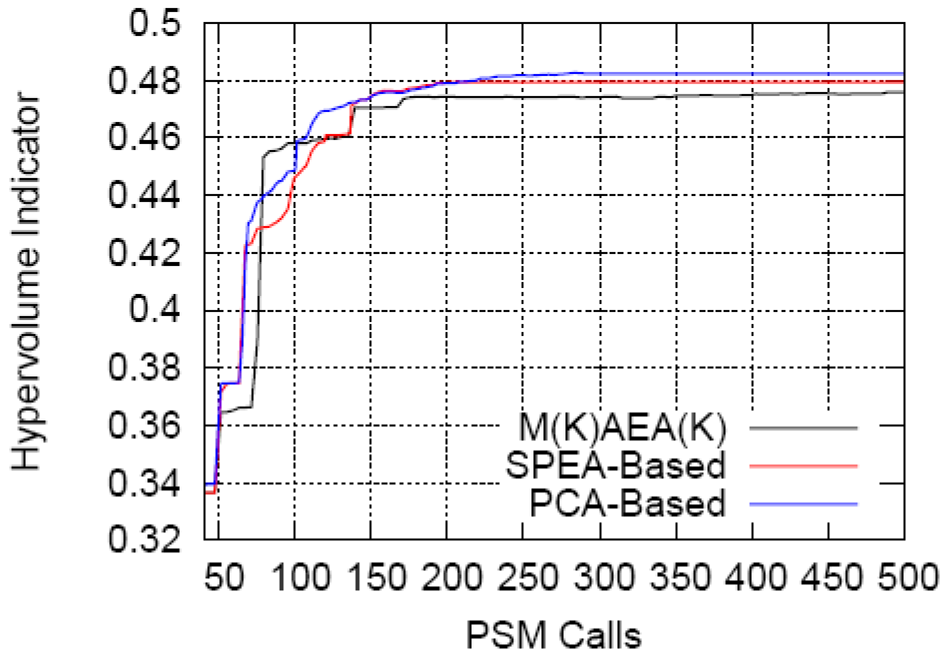


Method M2 (*2018*)

Computational Methods in Applied Mechanics and Engineering, 197(33-40):2963-2975, 2008.
Engineering Optimization, 44(2):157-173, 2012.
Applied Soft Computing, 64:1-13, 2018.

Demo Case B Revisited using the Hybrid Method

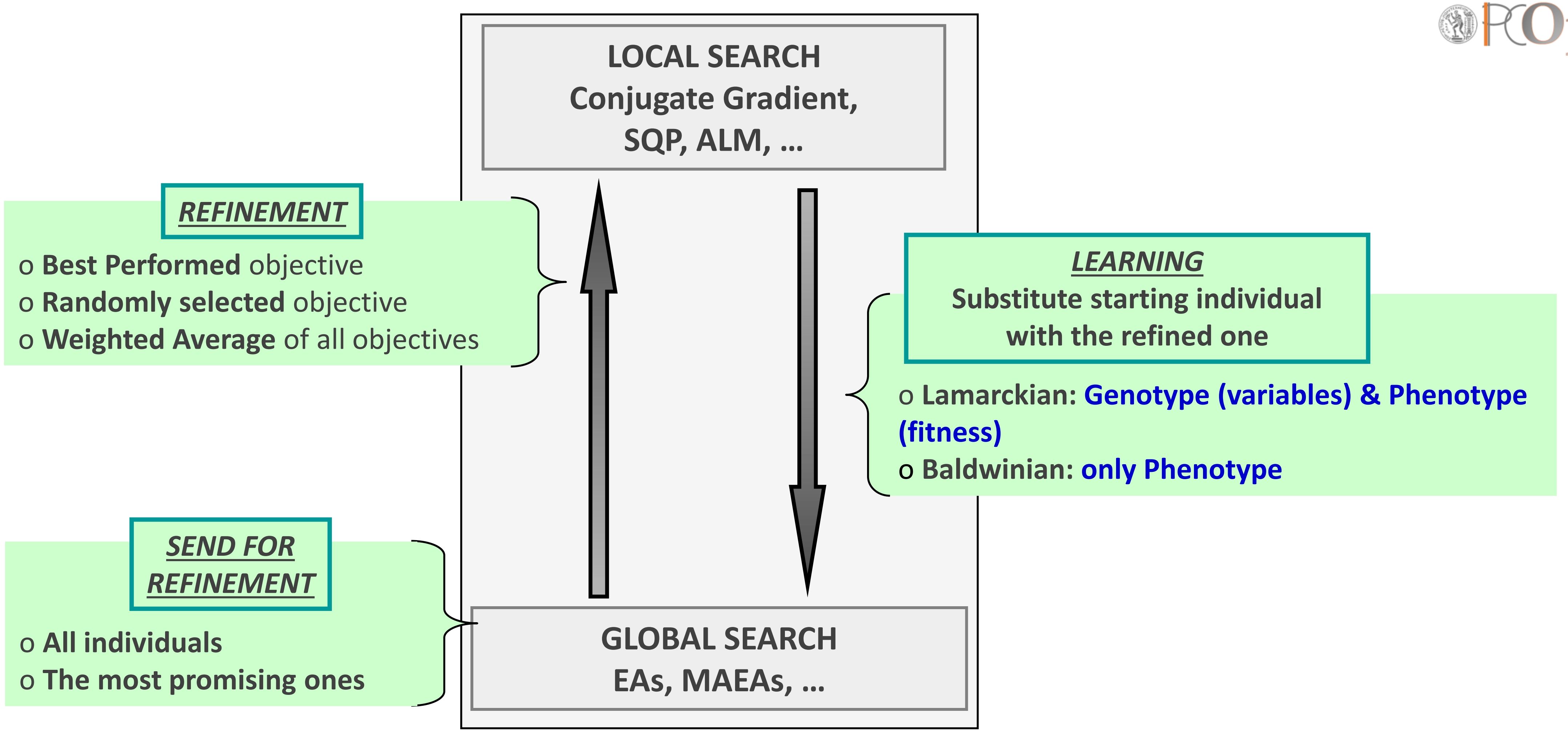




Comparison of Method M1 (SPEA-Based) and Method M2 (PCA-Based) with λ_{GB}=λ_e.

Hybrid methods can efficiently approximate the Pareto front faster than a MAEA

Hybrid/Memetic Schemes – Memetic Algorithms (MA)



REFINEMENT

- o **Best Performed** objective
- o **Randomly selected** objective
- o **Weighted Average** of all objectives

SEND FOR REFINEMENT

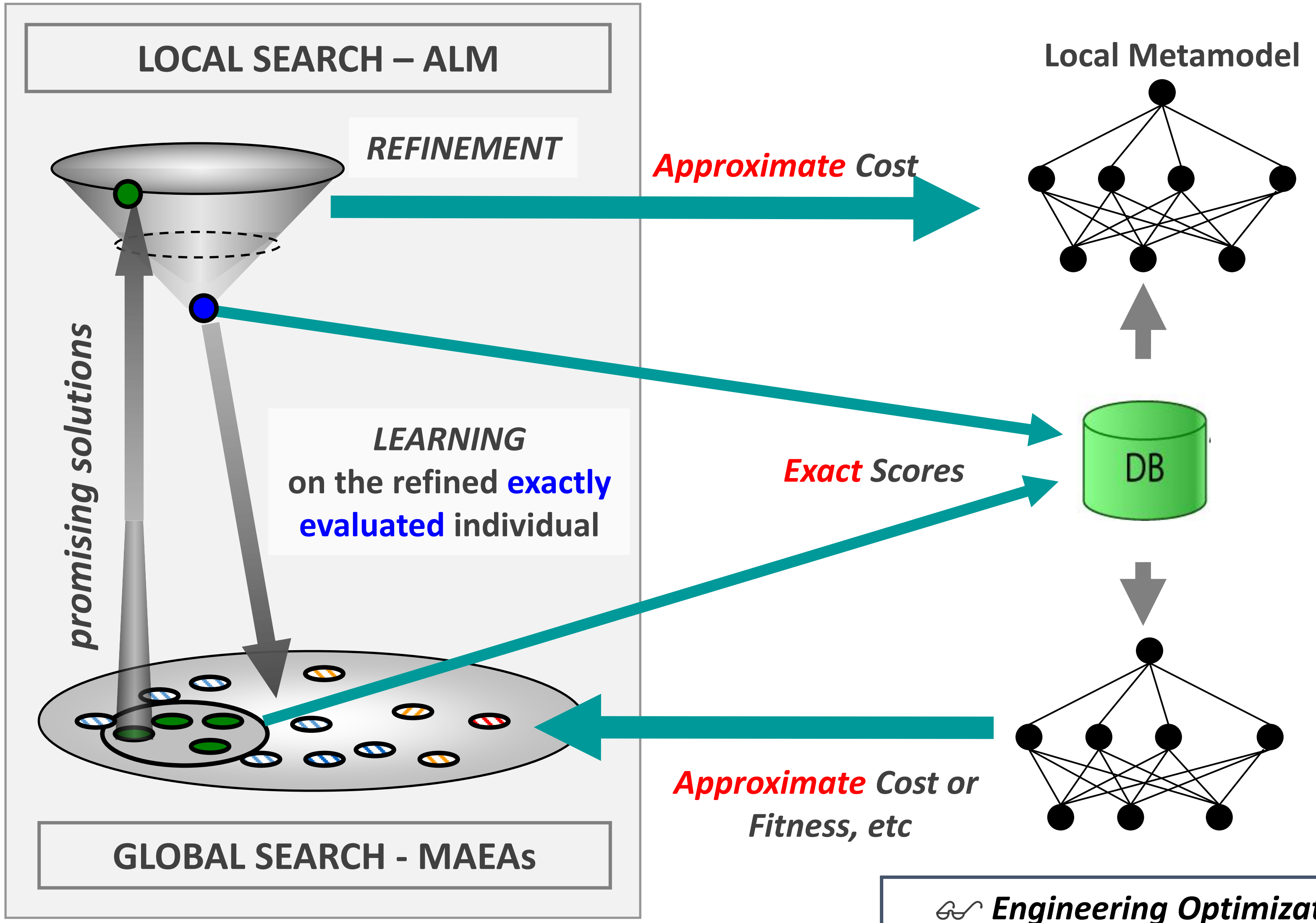
- o **All individuals**
- o **The most promising ones**

LEARNING

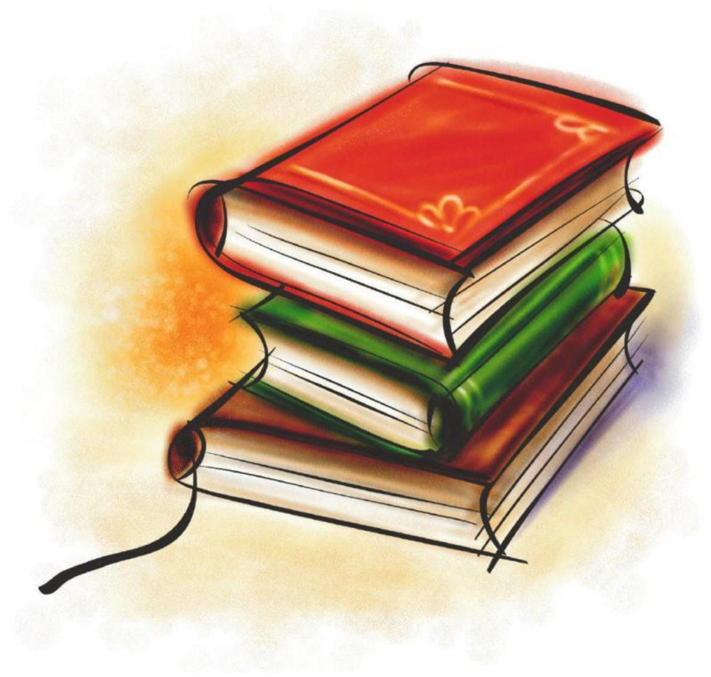
Substitute starting individual with the refined one

- o **Lamarckian: Genotype (variables) & Phenotype (fitness)**
- o **Baldwinian: only Phenotype**

Metamodel-Assisted Memetic Algorithms (MAMA)



Engineering Optimization*, *41:909-923, 2009.



Asynchronous Search

Asynchronous EAs and MAEAs

Main Concept:

- **Maximum exploitation of all available computational resources (processors)**
- **Remove the generation barrier**
- **Once a processor becomes idle, a new offspring is generated on-the-fly and its evaluation is assigned to this processor.**

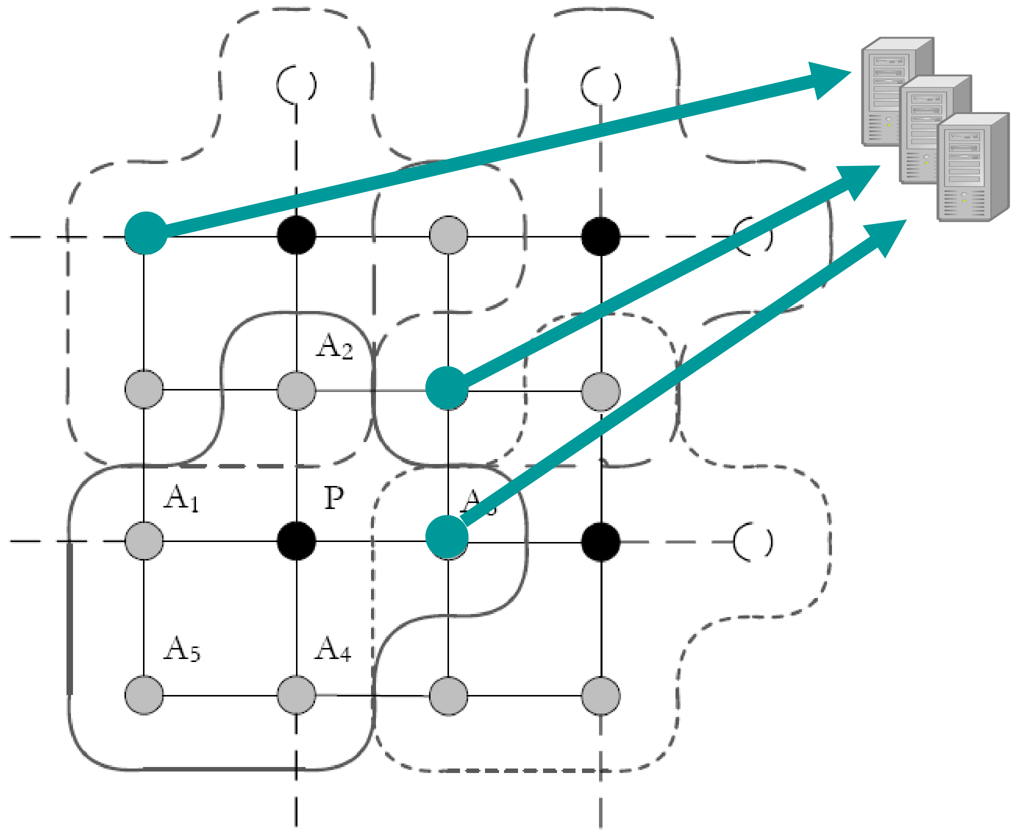
What about metamodels and Asynchronous EAs?

- **Instead of generating a single new member to be evaluated, a few trial members are generated. For each trial member, a local metamodel is trained. The most promising new individual among the trial ones, according to the metamodel (used separately for all of them), is send to the idle processor.**

Engineering Optimization, 41:241-257, 2009.

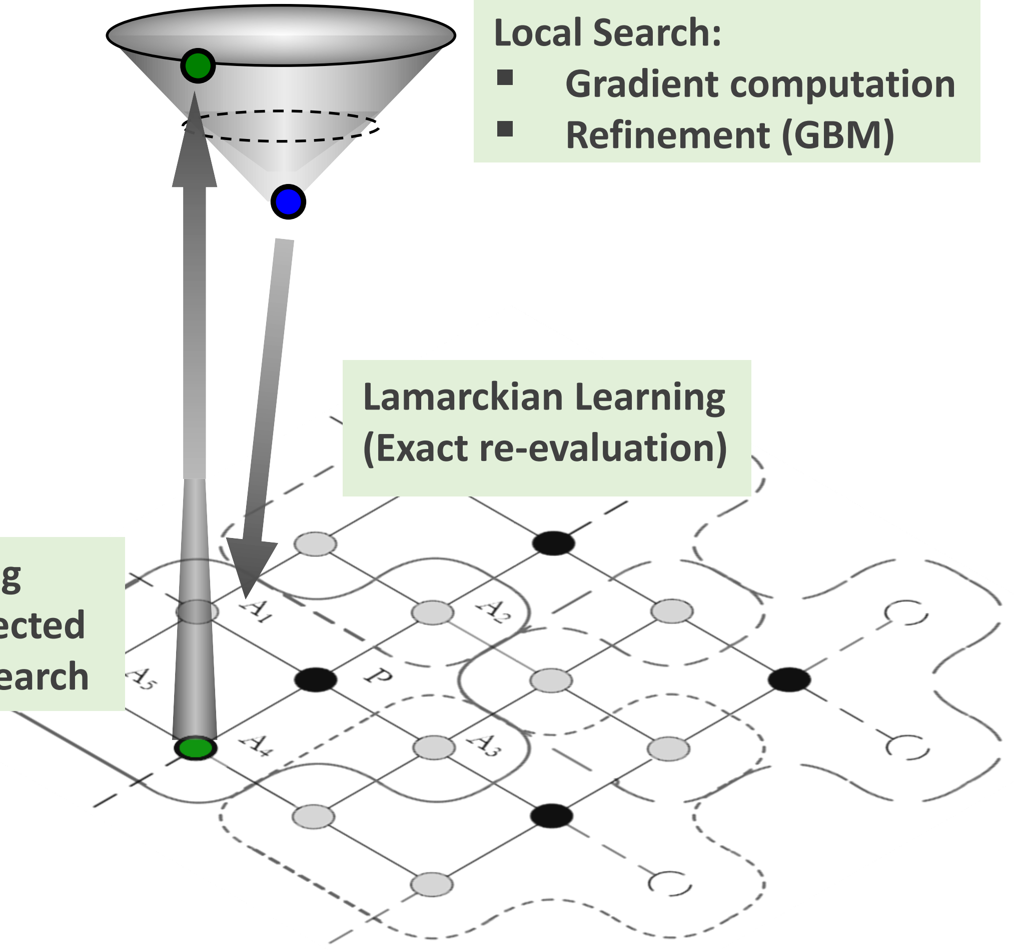
Genetic Programming and Evolvable Machines, 10:373:389, 2009.

Asynchronous EAs



The AEA, controlled by strongly interacting demes ensures max. exploitation of all available processors. At the end of an evaluation, the next agent to undergo evaluation on the idle processor is determined by of intra- and inter-deme processes.

Asynchronous MAs (AMAs)

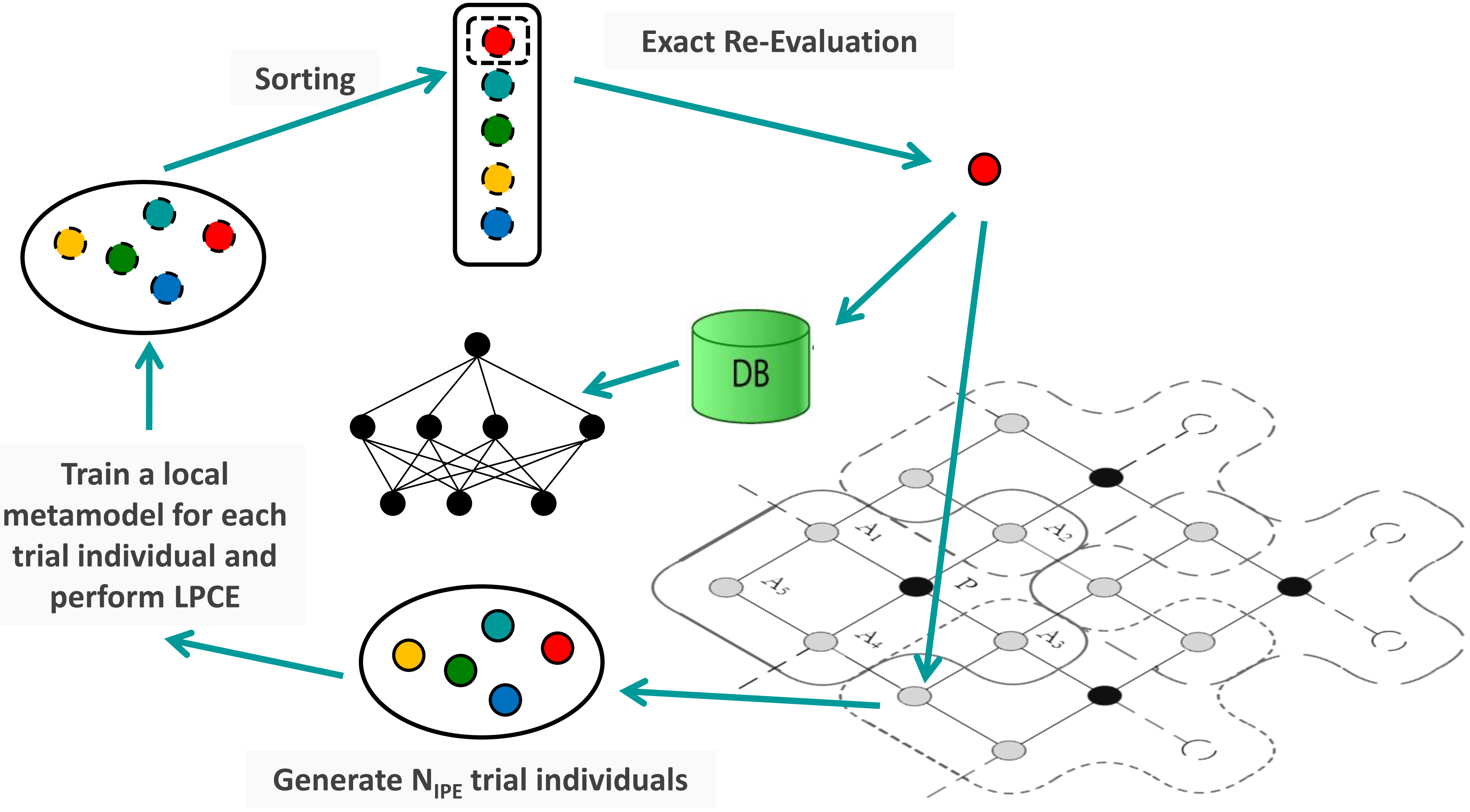


Local Search:
- Gradient computation
- Refinement (GBM)

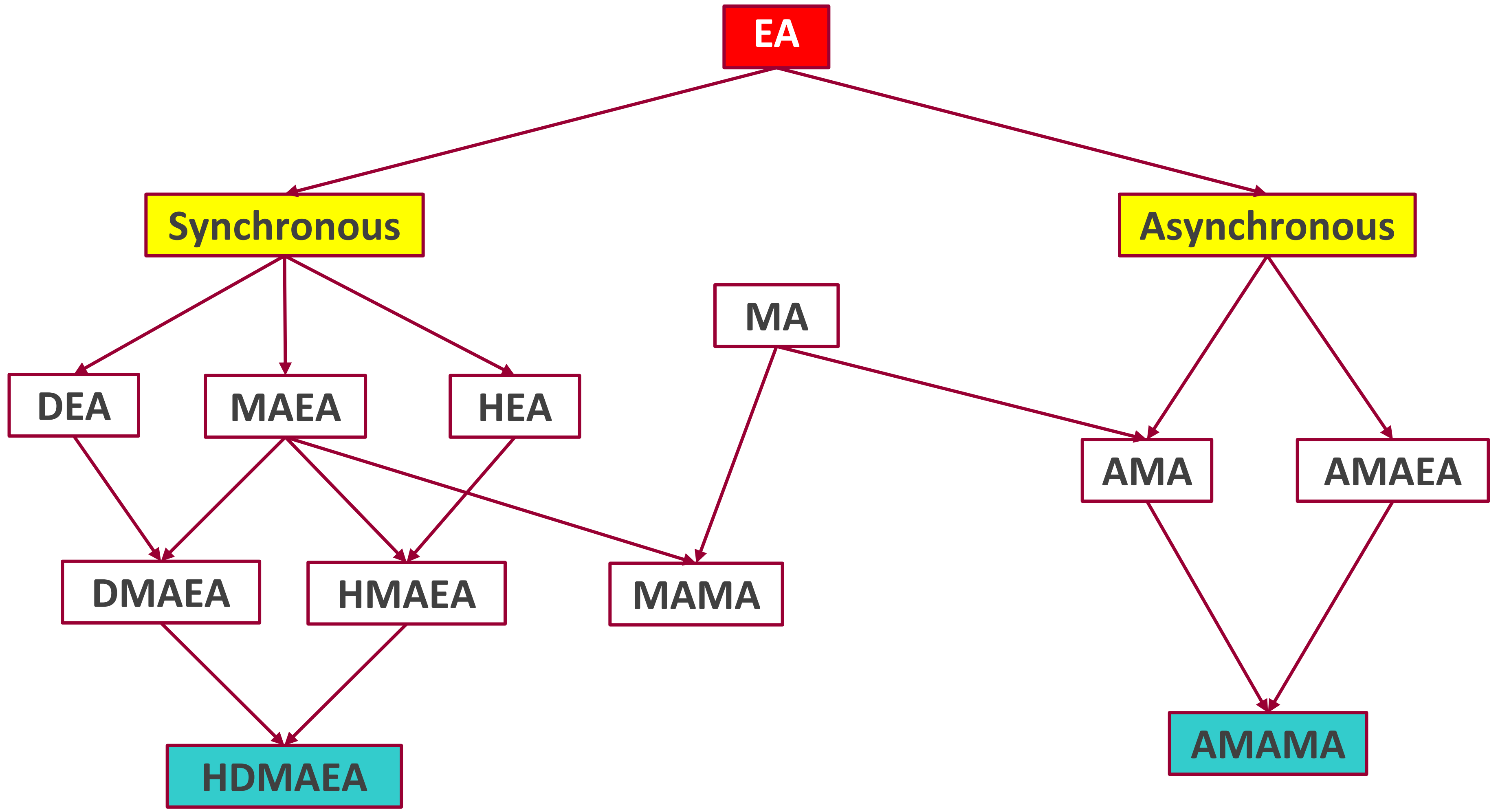
Lamarckian Learning (Exact re-evaluation)

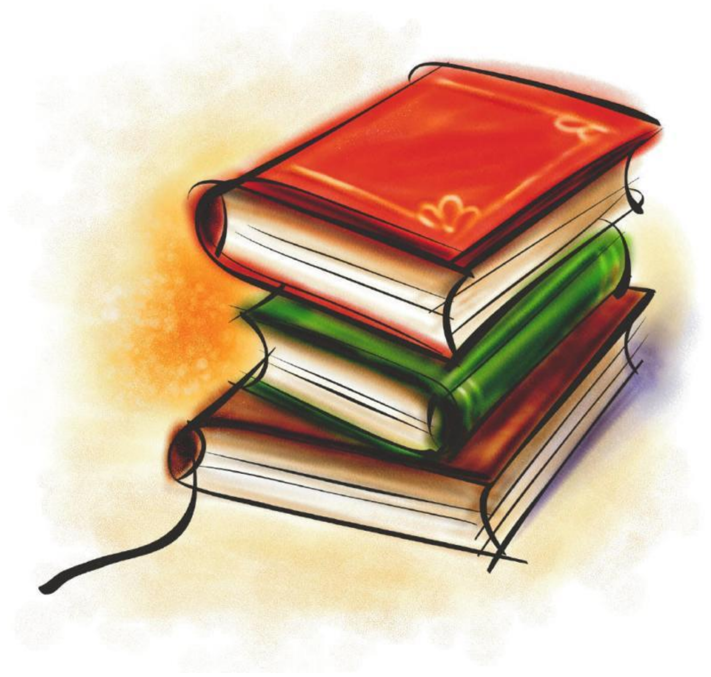
Best performing individuals are selected to undergo Local Search

Asynchronous MAEAS & MAMAs


Sorting
Exact Re-Evaluation
DB
Train a local metamodel for each trial individual and perform LPCE
Generate N_{IPE} trial individuals
A_1
A_2
A_3
A_4
A_5
P

Summary


EA
Synchronous
Asynchronous
MA
DEA
MAEA
HEA
AMA
AMAEA
DMAEA
HMAEA
MAMA
HDMAEA
AMAMA



Industrial Applications

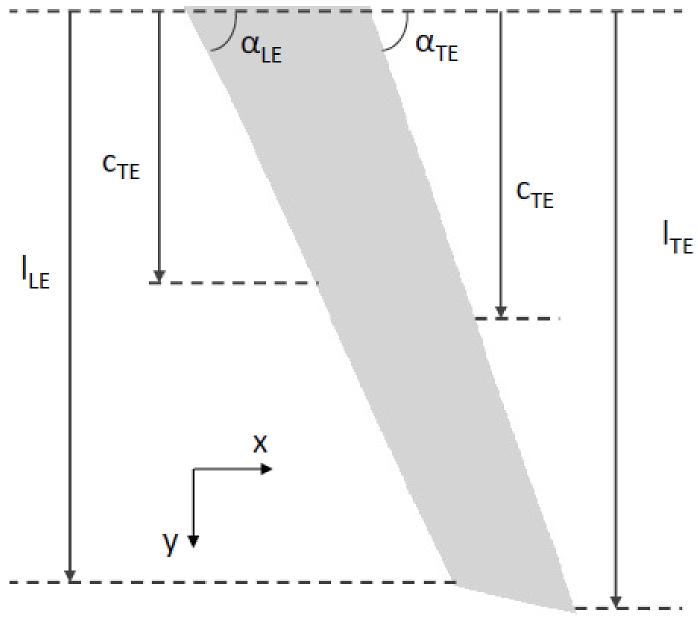
Optimisation of an Aircraft Wing-Body Configuration (1/3)

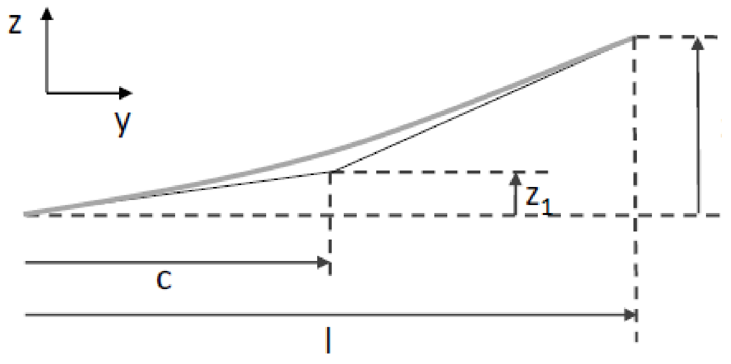
Re-design of an aircraft wing-body configuration, by changing only the wing shape, for max. C_L & min. C_D.

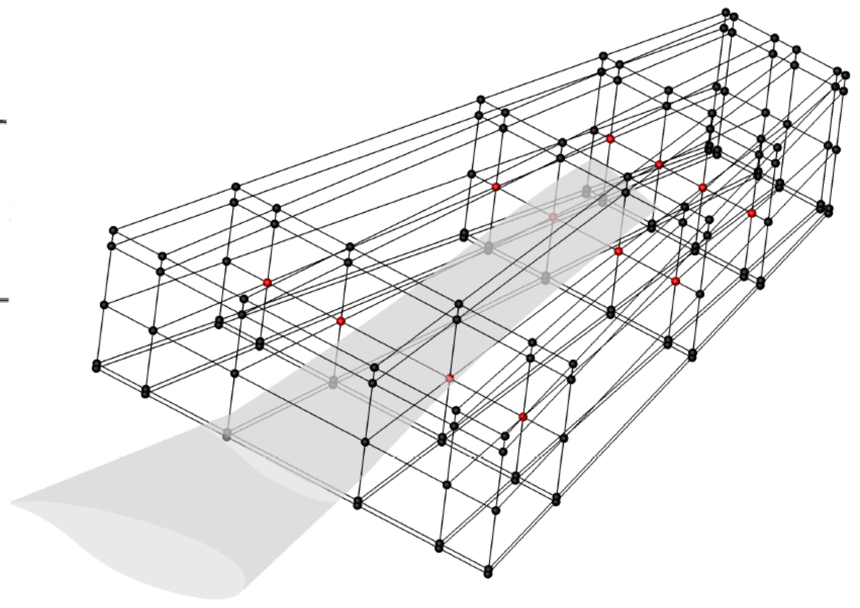
Flow conditions: $Re_c = 10^6$, $M_\infty = 0.75$, $a_\infty = 0^o$.

Cost per PSM call ~12 min. on a NVIDIA V100 GPU.

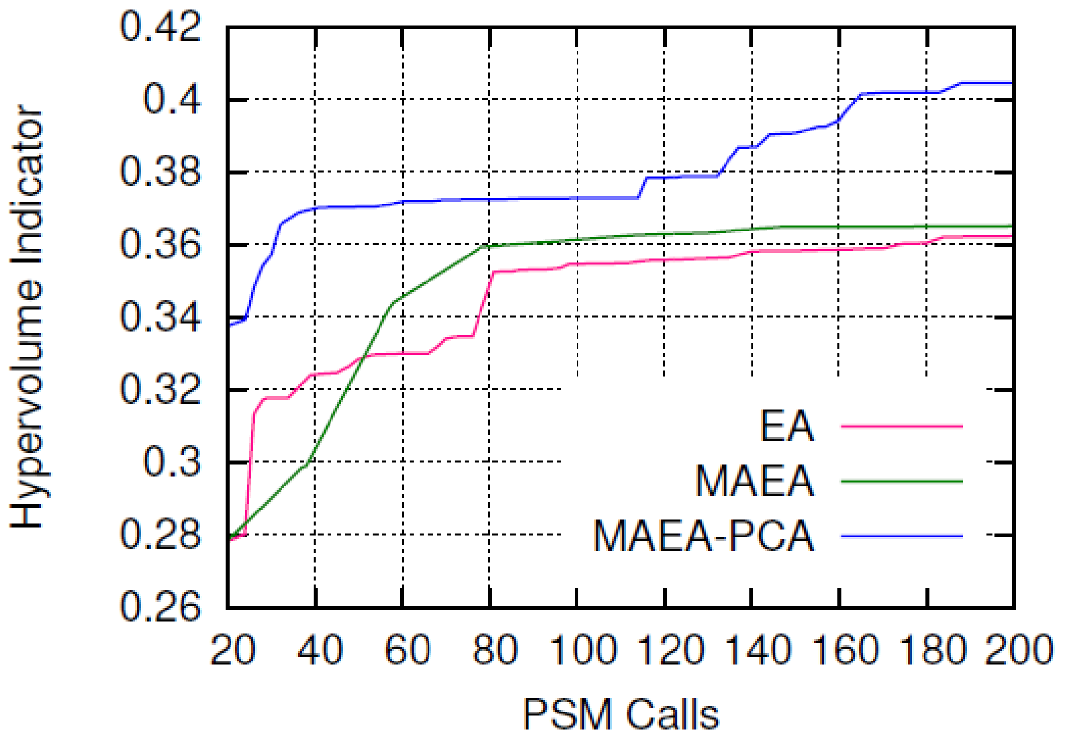
Customized parametrisation (sweep angles, LE/TE curvatures, wing bending & twist) with 8 DoFs, affecting the coordinates of the 3×5×4 NURBS morphing box.

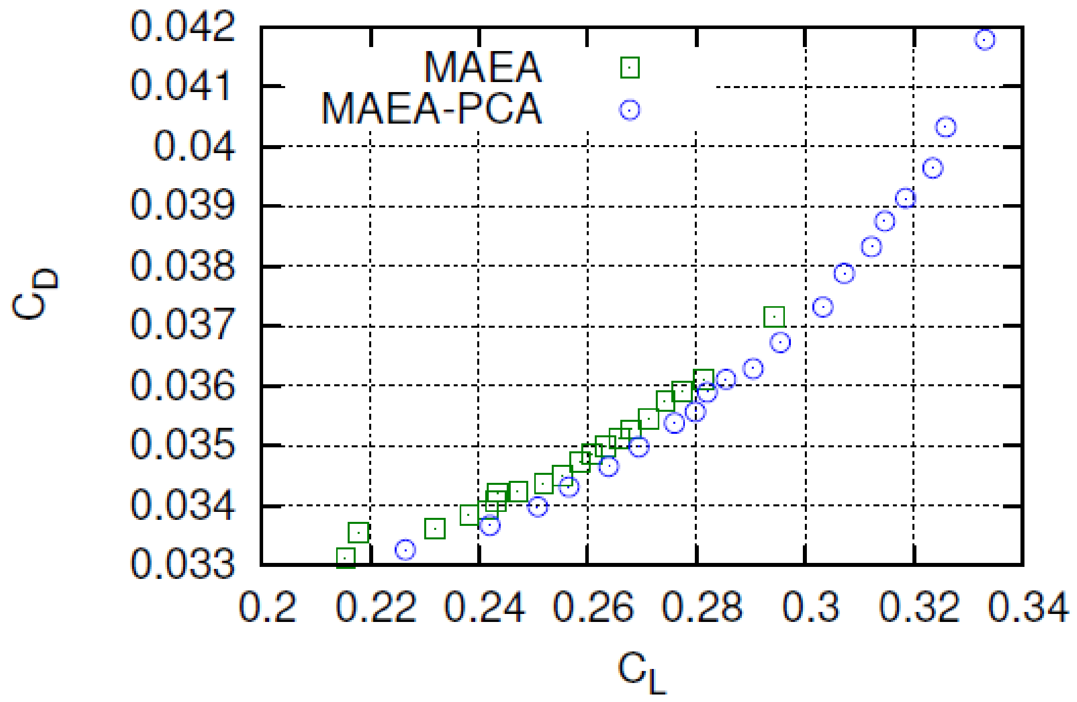






Optimisation of an Aircraft Wing-Body Configuration (2/3)

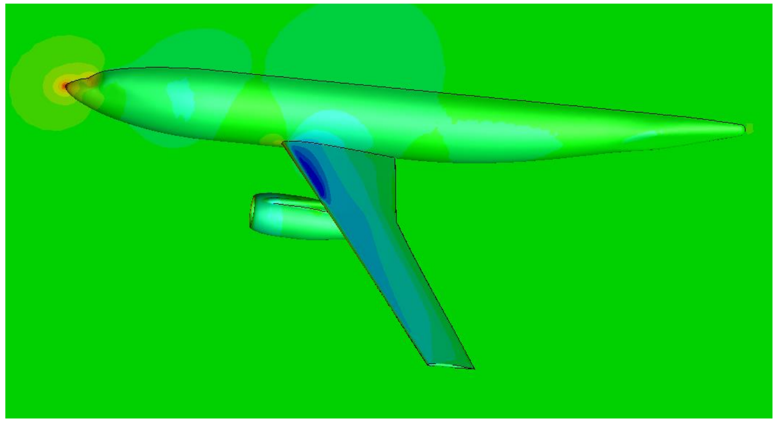




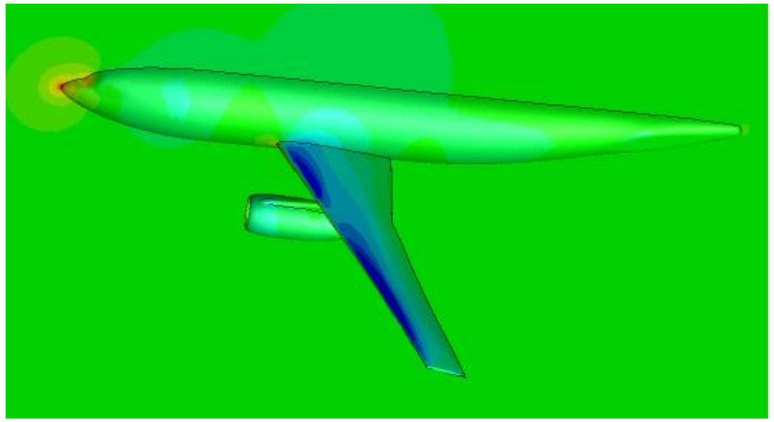
Comparison of (5, 10)EA, MAEA and MAEA-PCA. The LCPE phase starts after T^{MM} = 20 calls to the PSM; λ_e =4. Stopping criterion = 200 calls to the PSM.

Optimisation of an Aircraft Wing-Body Configuration (3/3)

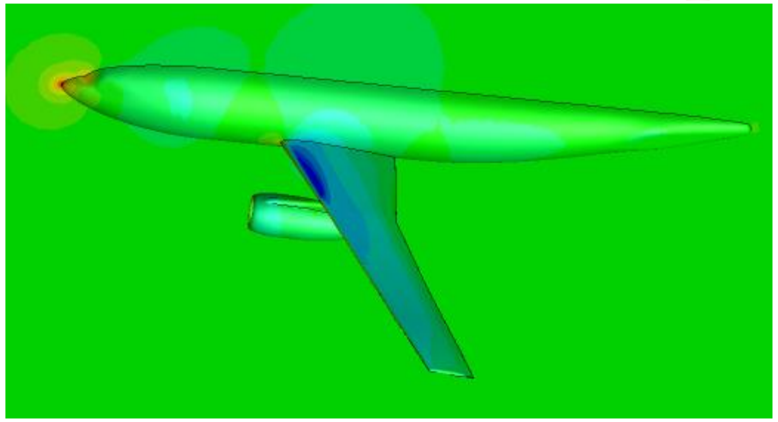
Pressure fields



Reference shape

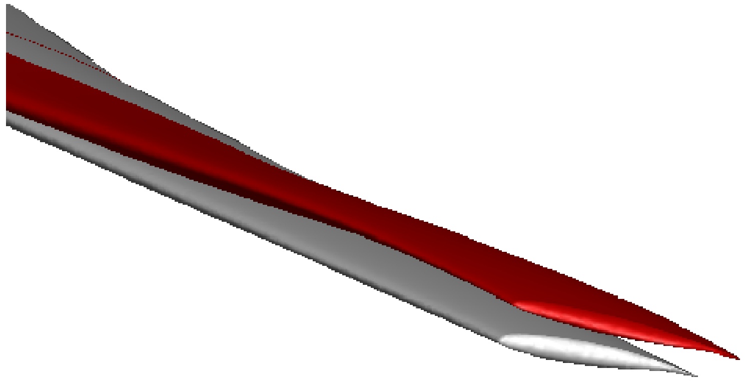


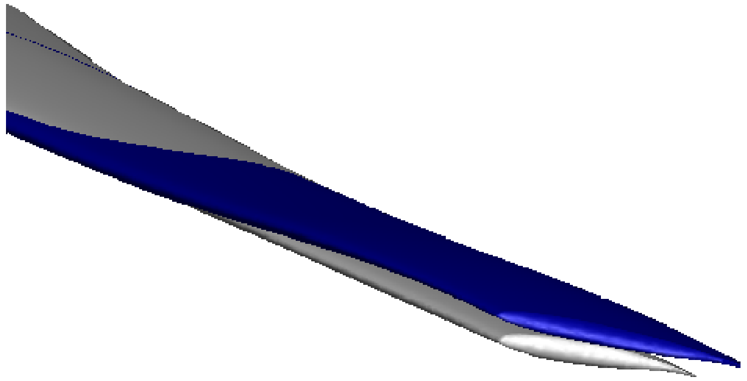
Optimised for max. C_L



Optimised for min. C_D

Reference (grey) & optimised wing shape (in color)



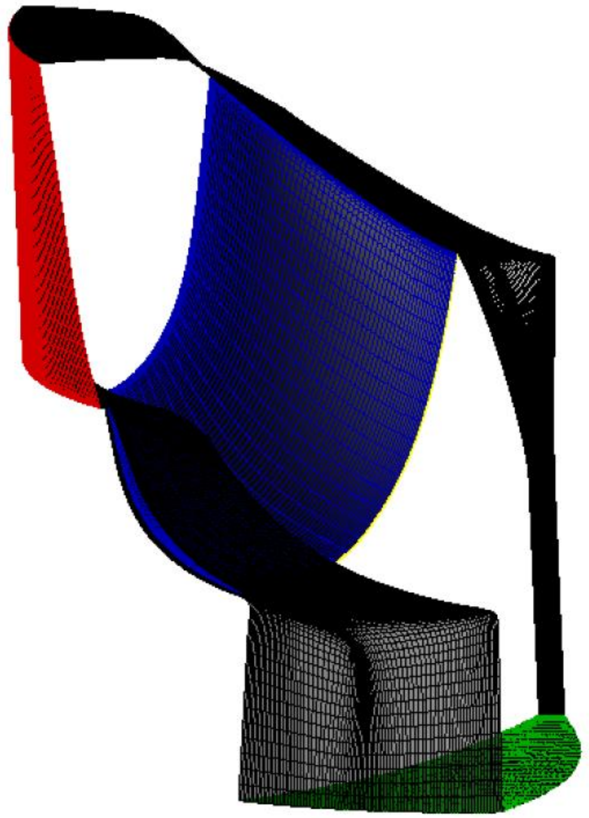


ShpO of a Francis Runner (1/3)

ShpO of a Francis runner with two objectives: (a) max. efficiency & (b) min. cavitation (maximize the min. pressure on the blade surface). Inlet flow conditions: V_{inlet}=8.198 m/s, a_{swirl}=22.36^o & a_{axial}=0^o; outlet static pressure 39900 Pa & rotation speed 117.8 rad/s.

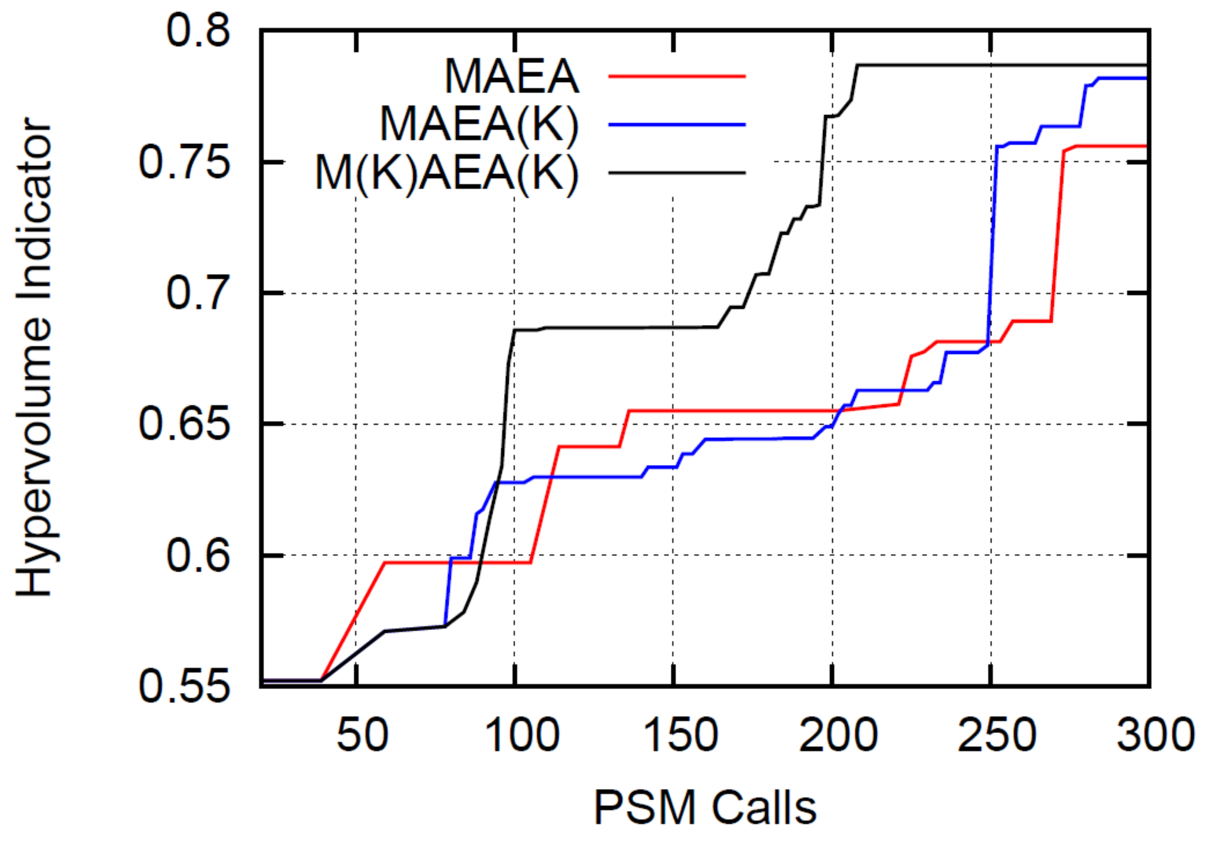
The in-house GMTurbo software is used to parameterize the geometry and provide the 75 design variables for the optimisation. These correspond to the span-wise distributions of quantities parameterizing the camber surface.

Cost per CFD evaluation: ~1 hr on a single K40 GPU.



In collaboration with ANDRITZ Hydro

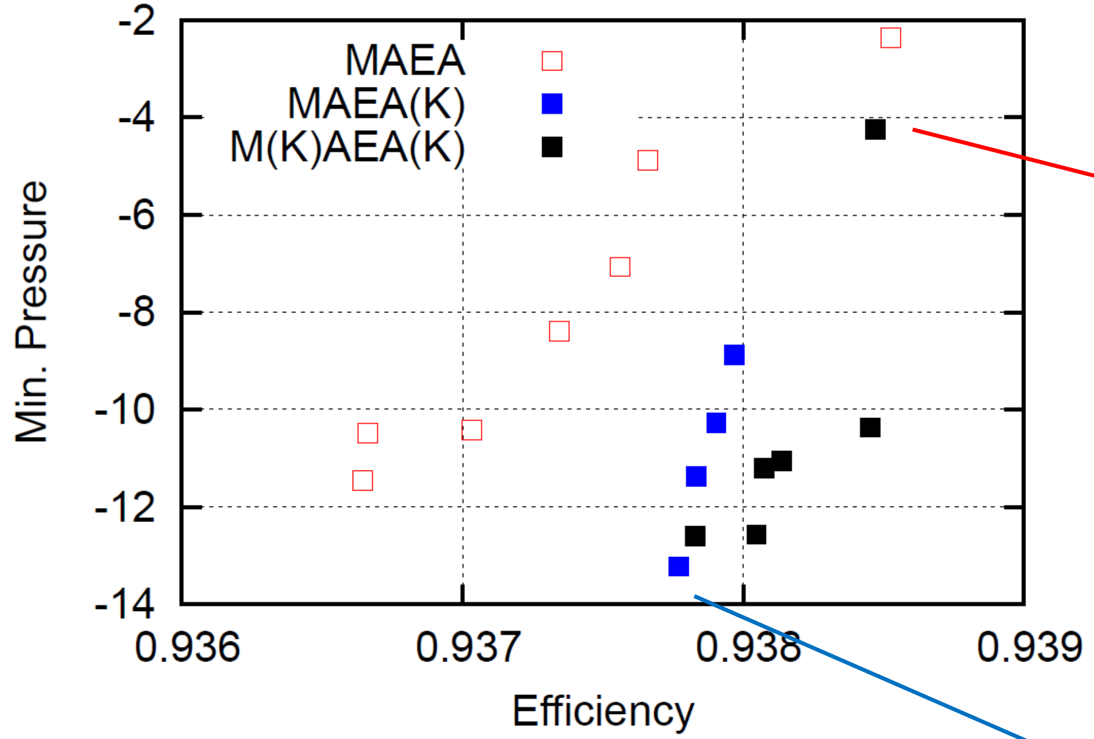
ShpO of a Francis Runner (2/3)

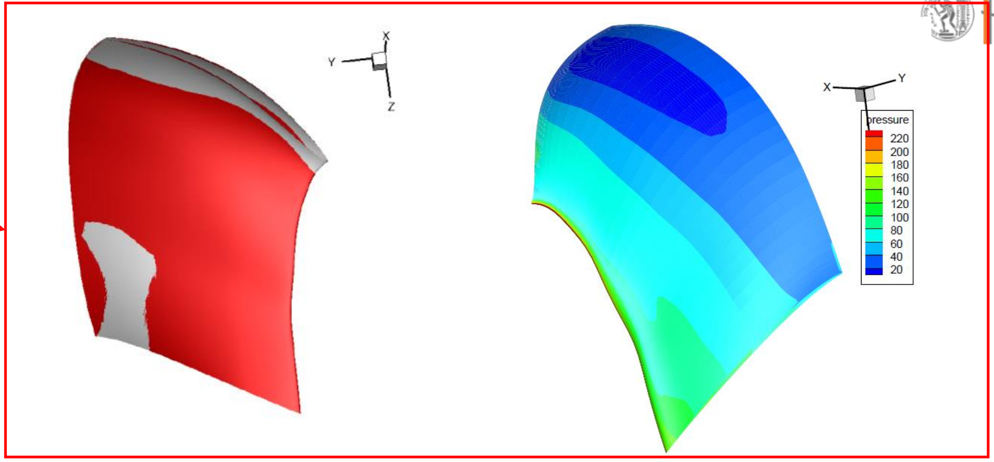


Comparison of the averaged convergence histories for three RNG seeds of the (10,20) EA, MAEA and M(K)AEA(K), in terms of the number of CFD evaluations (PSM Calls). The LCPE phase starts after the first T^{MM}=20 calls to the PSM and the λ_e=2 most "promising" individuals are re-evaluated in each generation. Stopping criterion = 300 CFD evaluations.

In collaboration with ANDRITZ Hydro

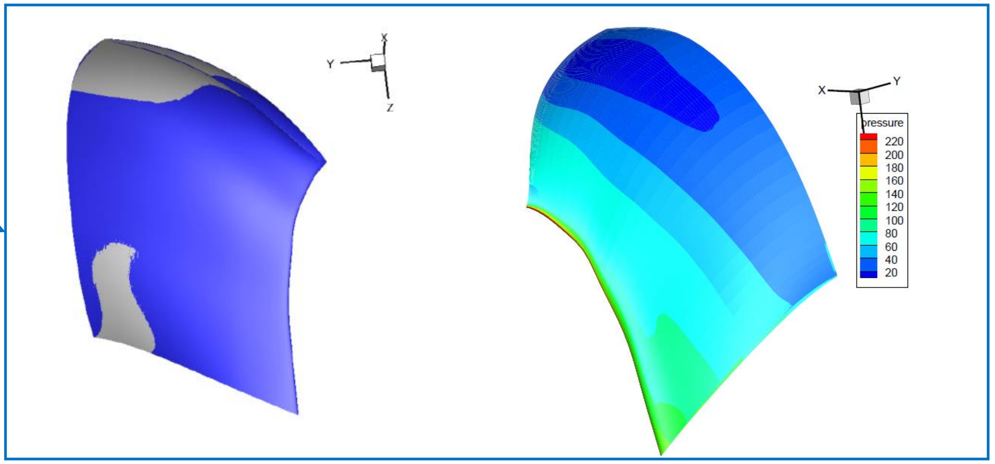
ShpO of a Francis Runner (3/3)





Comparison between the baseline (in grey) and the optimised runner geometry.

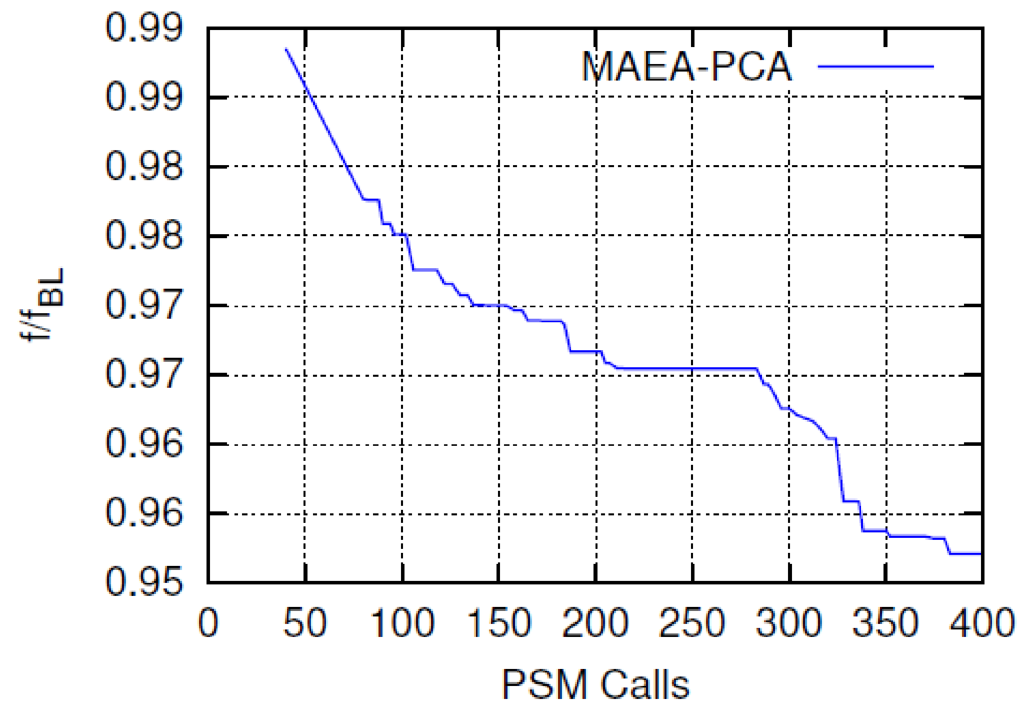
Pressure distribution on the suction side

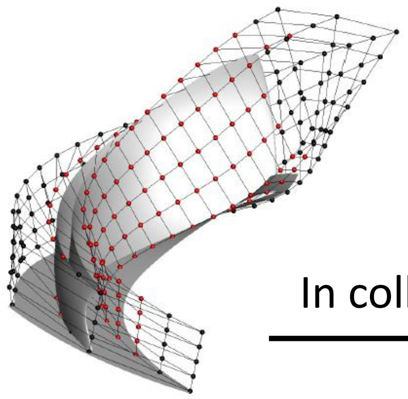


Comparison of the fronts of non-dominated solutions resulted from the (10,20) EA, MAEA and M(K)AEA(K).

In collaboration with ANDRITZ Hydro

ShpO of a Hydraulic Turbine

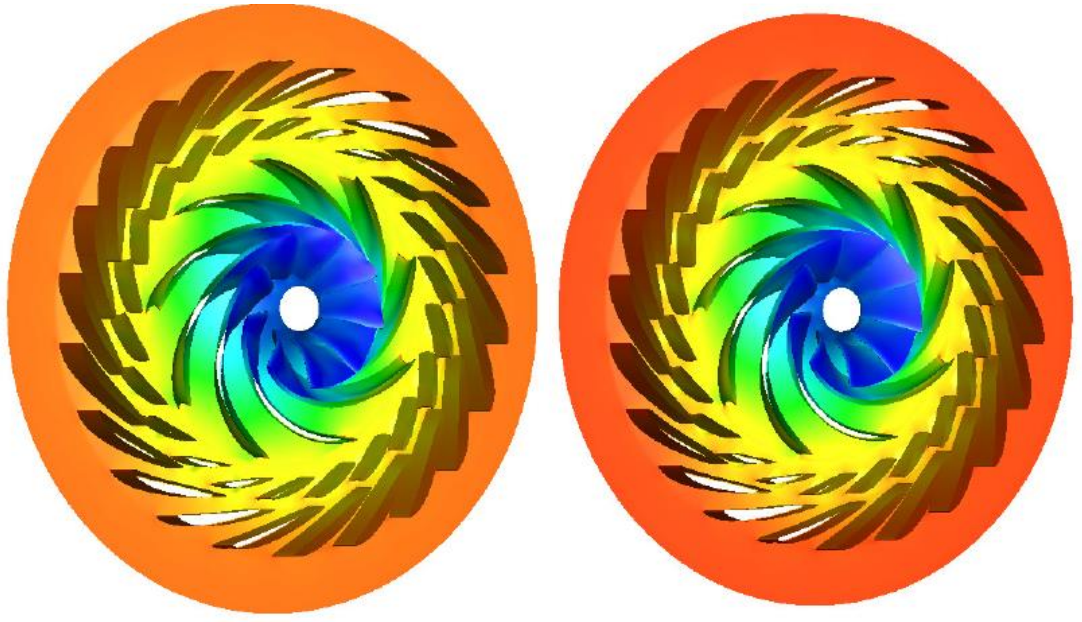




In collaboration with ANDRITZ Hydro

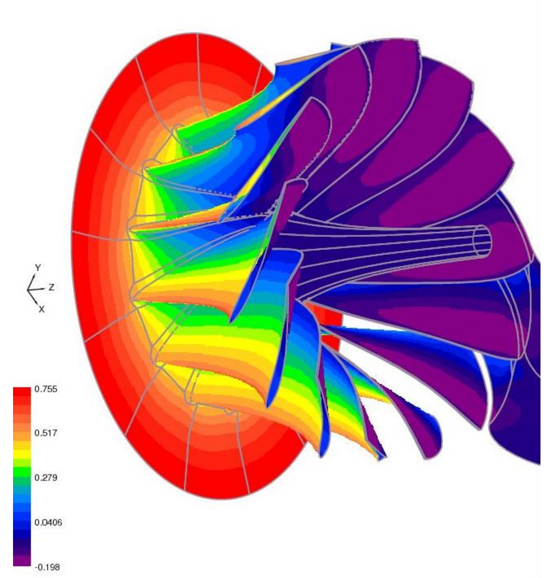
Radial-inflow-axial-outflow turbine, with a stator with two rows (stay and guide vanes) and a runner. Objective: min. strong pressure pulsations due to the interactions of the runner blades with the guide vane wakes. A 21×3×5 NURBS control lattice is used to parameterize the runner blades, with 180 DoFs.

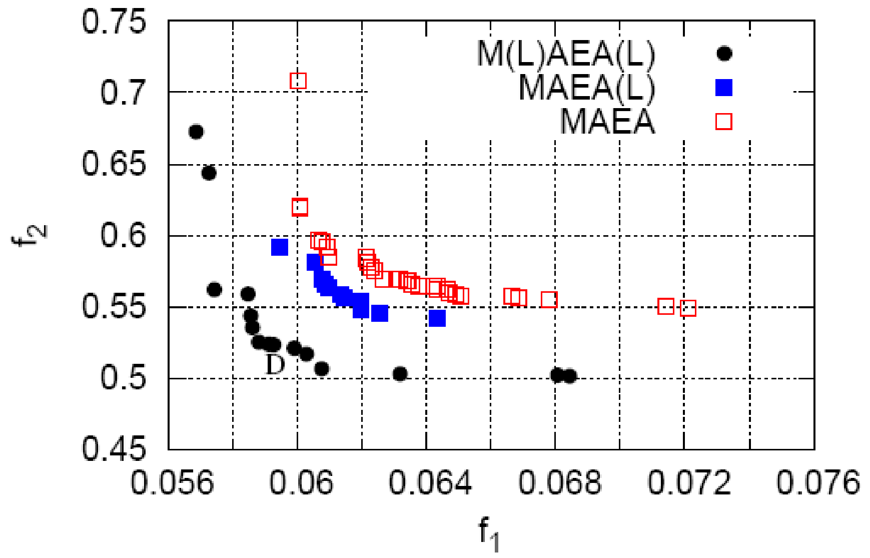
(12,40) MAEA-PCA with T^{MM}=50, λ_e =3. The KPCA is activated after the second generation and metamodels are trained in a truncated design space with 20 inputs only.

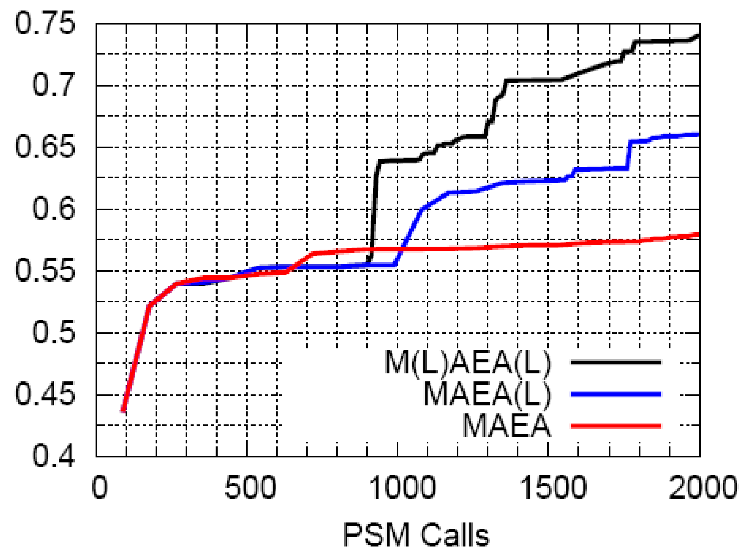


Pressure fields

ShpO of a Francis Runner







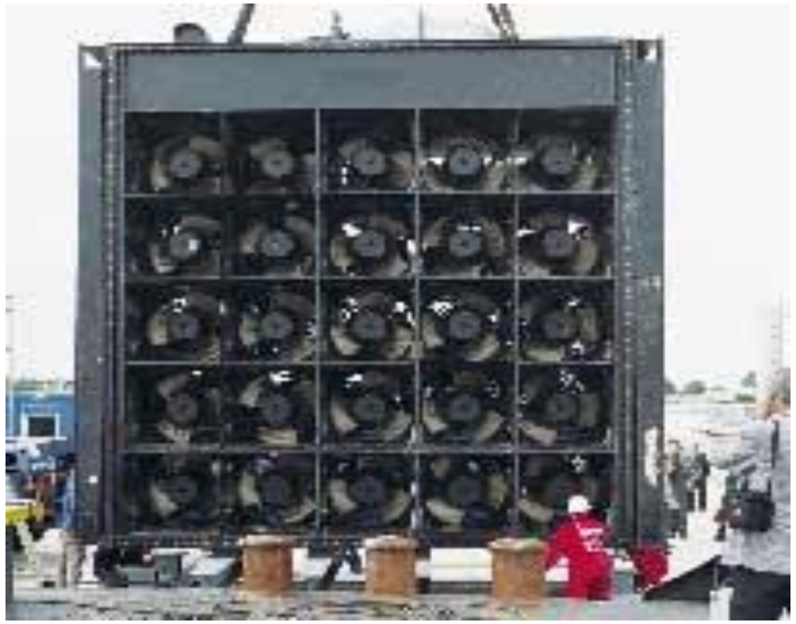
● Design of the Francis runner, at 3 operating points with two objectives: (a) exit velocity profiles' quality and (b) uniformity of the blade loading. Two constraints (head and cavitation). There are 372 DoFs, in total!

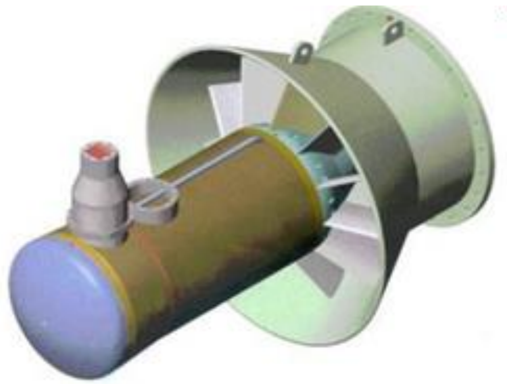
● Comparison of fronts of non-dominated solutions obtained at the same number of evaluations on the PSM (same CPU cost).

● Due to the extremely high problem dimension, the use of M(L)AEA(L) becomes absolutely necessary!

In collaboration with ANDRITZ Hydro

Design of HYDROMATRIX ® (1/3)

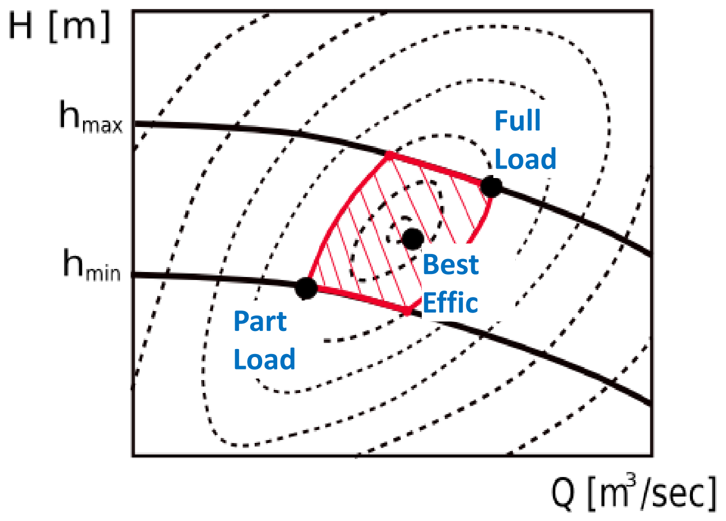




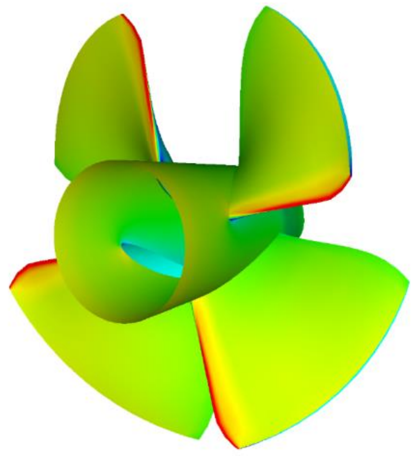
- HYDROMATRIX®: a number of "small", axial flow turbine generator units comprising a factory assembled grid or "matrix".
- Advantages compared to conventional designs (lower cost to power ratio):
- Minimisation of the required civil construction works.
- Minimum time for project schedules, construction and installation.
- Small geological and hydrological risks. Minimum environmental inflict.

In collaboration with ANDRITZ Hydro

Design of HYDROMATRIX ® (2/3)



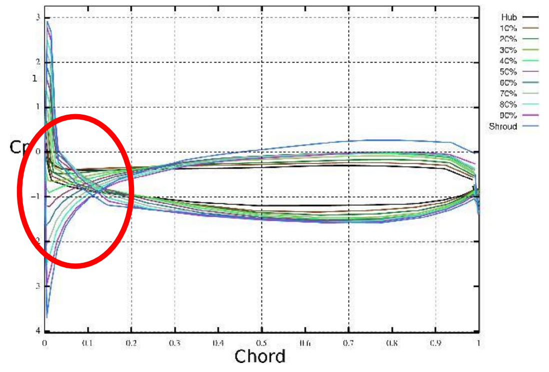
Weights			
	peak point (OP_1)	part load (OP_2)	full load (OP_3)
$w_1^{OP_i}$	1.0	0.0	0.0
$w_2^{OP_i}$	1.0	0.0	0.0
$w_3^{OP_i}$	0.2	0.0	0.0
$w_4^{OP_i}$	1.0	1.0	1.0
$w_5^{OP_i}$	0.0	100.0	100.0
W_{OP_i}	1.0	0.1	0.1



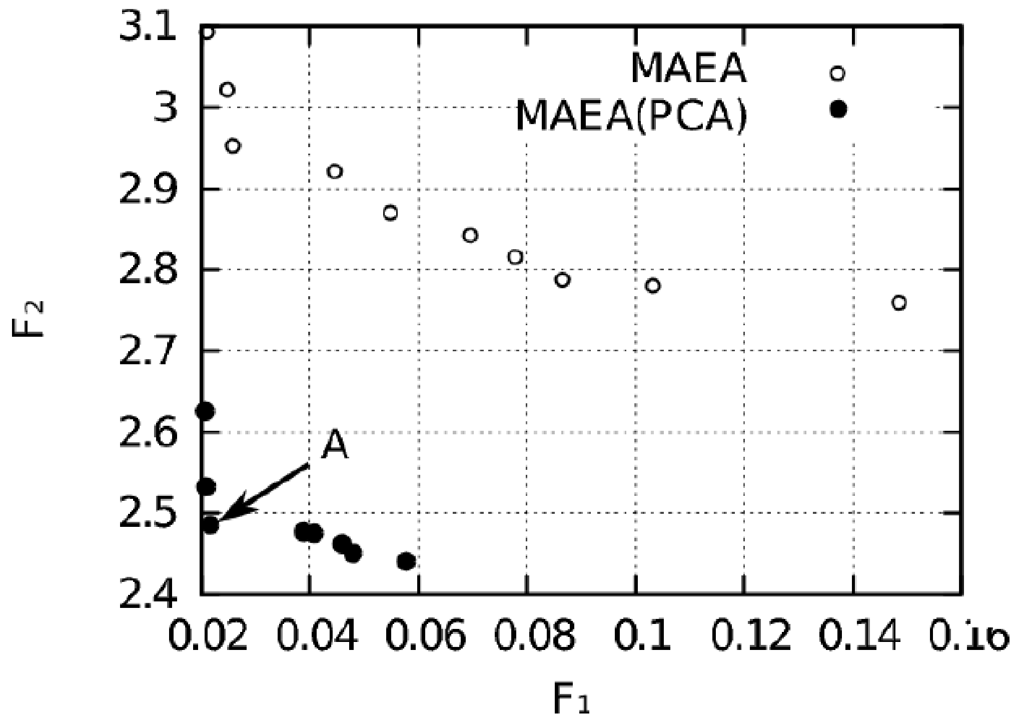
- **52 design variables**
- **Five quantities of interest (metrics): (f_1,f_2) : Given swirl and axial velocity distributions at the exit. (f_3): Uniform loading. (f_4): cavitation index. (f_5): pumping area.**
- **(5 metrics) x (3 operating points) = 15 objectives, cast as a two-objective problem.**
- **min. $F_1(f_1,f_2$, at the 3 OPs) , min. $F_2(f_3,f_4,f_5$, at the 3 OPs)**

In collaboration with

Design of HYDROMATRIX ® (3/3)



Pumping area.



Computations performed with the same budget.

In collaboration with

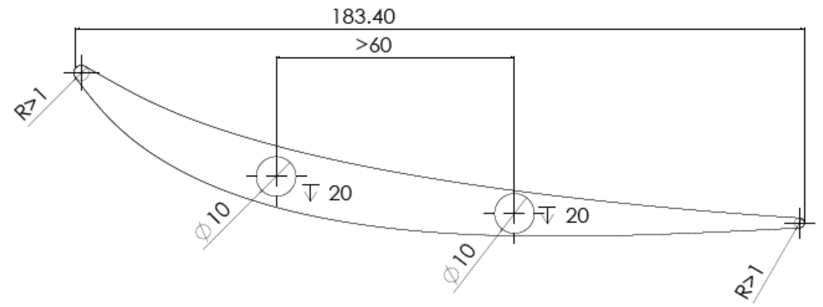
ShpO of theTU Berlin Turbolab Stator (1/2)

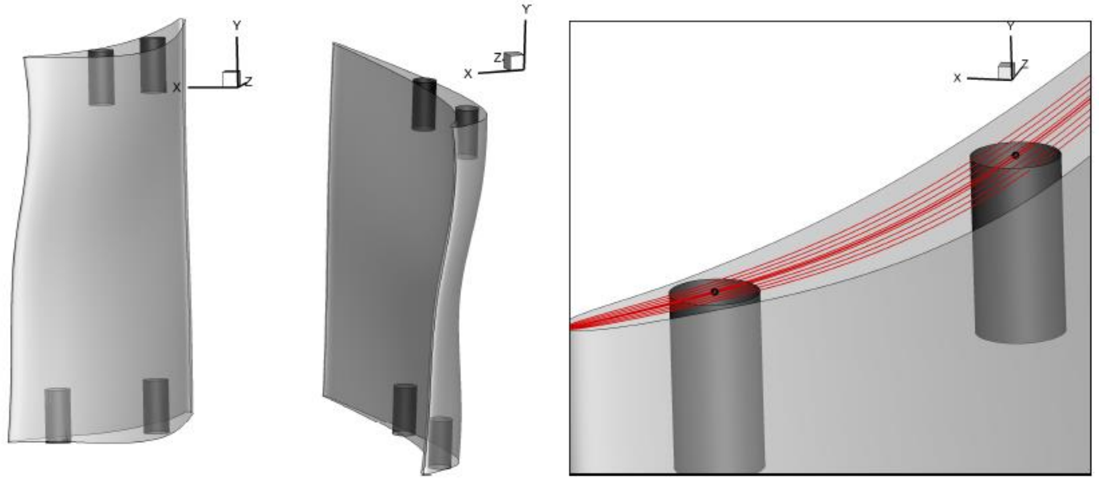
ShpO of the TU Berlis Turbolab stator with two objectives: (a) min. Δp_t (total pressure losses) and (b) min. $\Delta\alpha$ (deviation of the exit angle from the axial direction)

The in-house GMTurbo software is used to parameterize the geometry with 32 design variables corresponding to the spanwise distributions of quantities defining the camber surface.

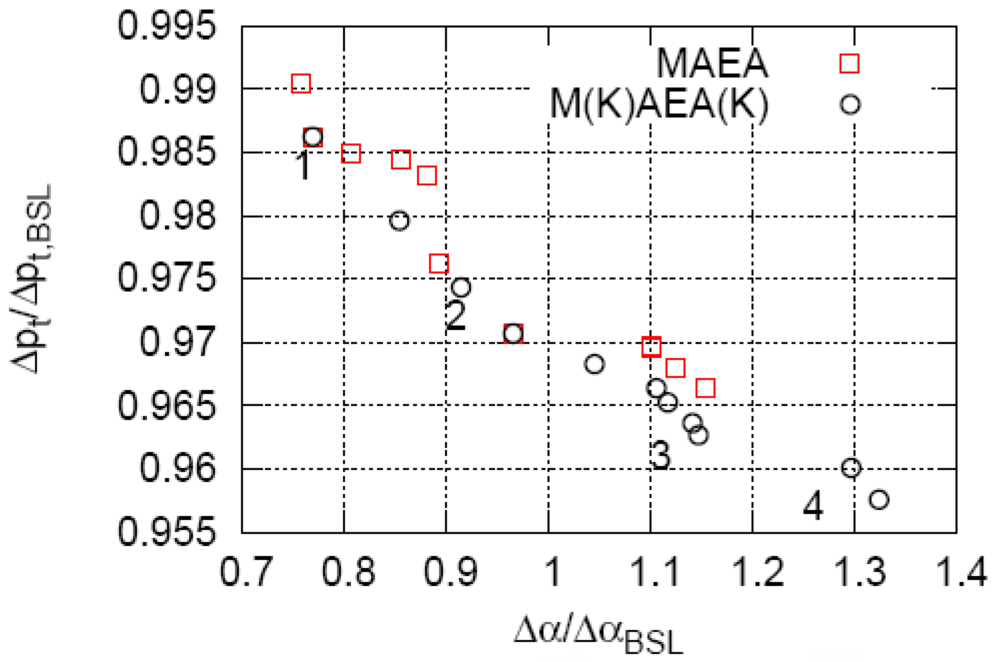
Constraints:

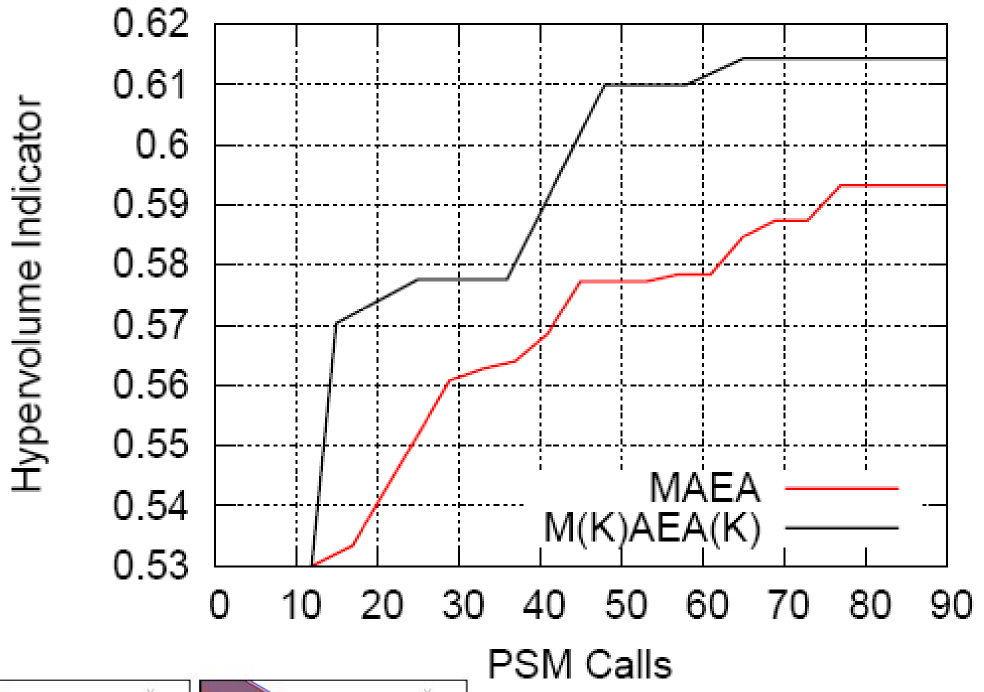
- **Ensure blade mounting in the casing (hub and shroud).**
- **Blade thickness must accommodate cylinders of material of 5mm radius and 20mm depth.**
- **Min. distance of 60 mm between fixture holes.**
- **The first cylinder is fitted by marching along the camber line.**
- **Fitting of the second cylinder follows.**

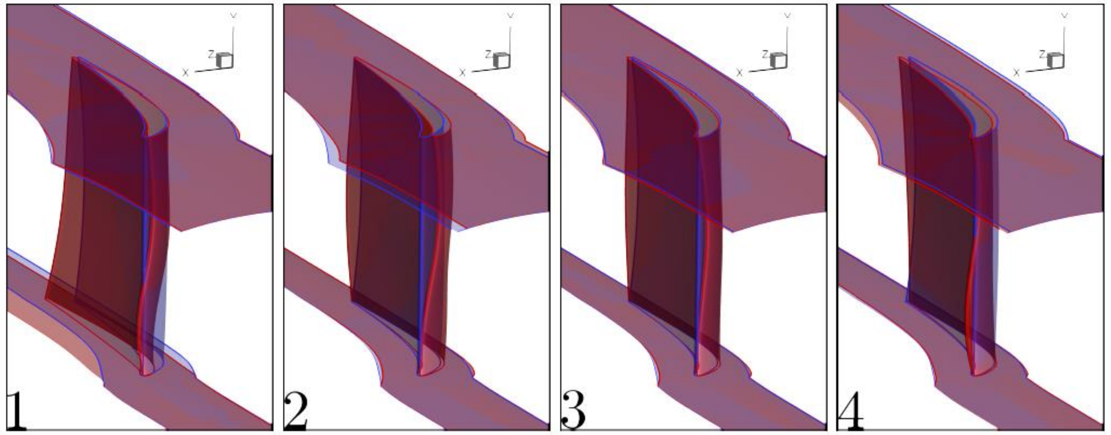




ShpO of TU Berlin Turbolab Stator (2/2)

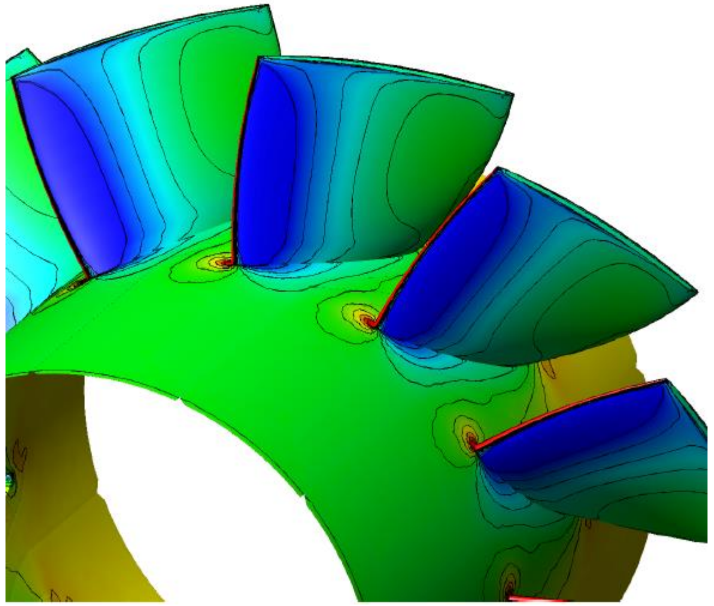


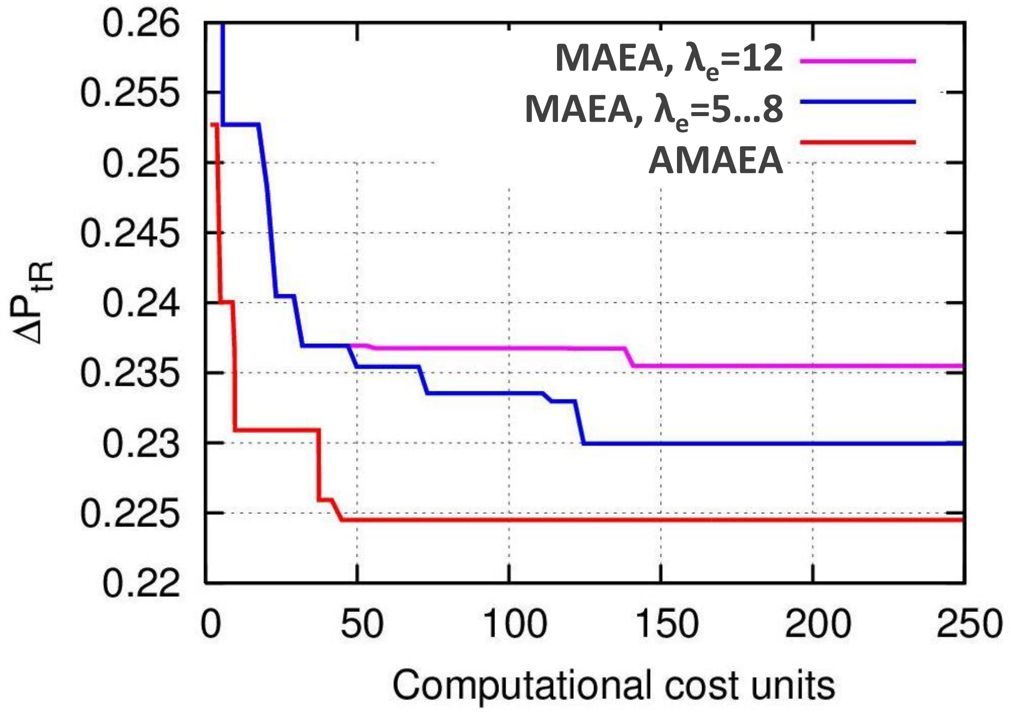




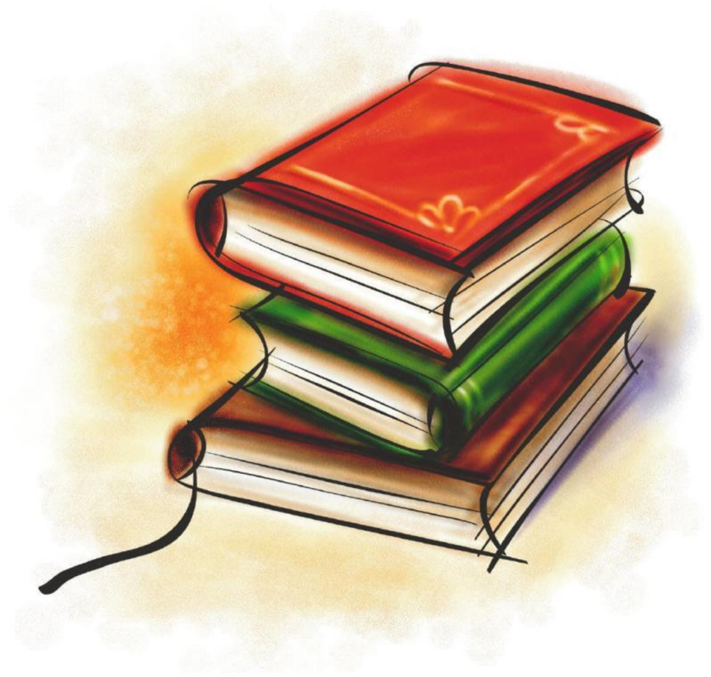
Four **optimised solutions** selected from the computed Pareto front, compared with the **reference blade**.

ShpO of a Compressor Rotor Blading





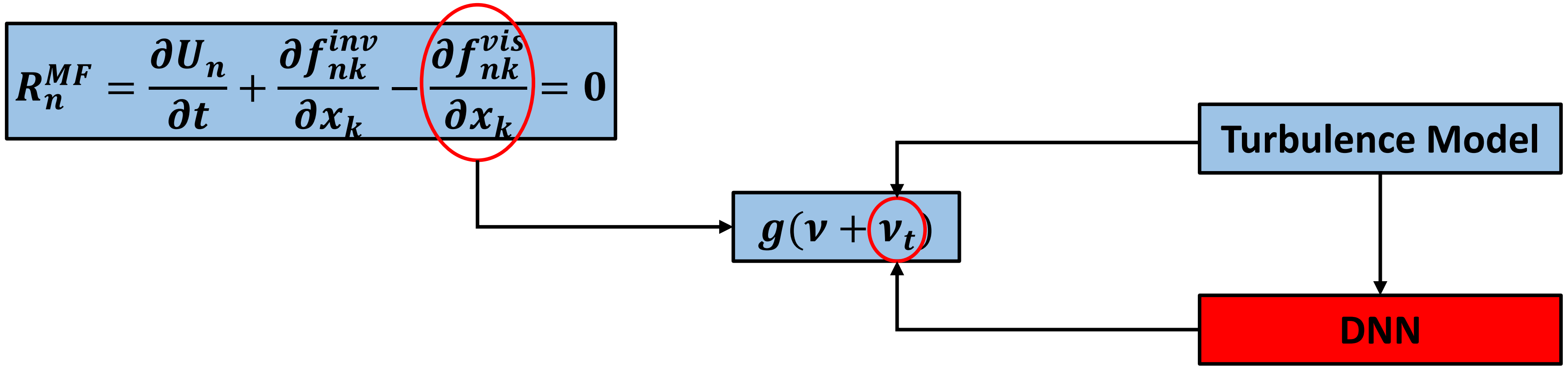
- **Design of a compressor rotor blading for min Δp_{tR}, constrained by fixed Δp_t (same OP).**
- **Axial inflow, v=15.28m/s, ω=1300rpm.**
- **Incompressible Navier-Stokes solver / Spalart – Allmaras turbulence model.**
- **Blade shape parametrisation using 38 design variables.**
- **Comparison of two (16,48) MAEAs & a (8x8) AMAEA with λ_e=8.**
- **12 concurrent evaluations.**



Deep Neural Network as Surrogate for CFD in EAs

DNN Surrogates for RANS – Implementation in EA/MAEA ShpO

- **DNNs may help reduce the cost per evaluation (faster PSM), and have additional gains, over and above to those due to the use of metamodels, PCA, distributed and hierarchical schemes, etc**
- **Replicate turbulence and/or transition models with/without wall-functions in CFD analysis & optimisation; DNN predicts the turbulent viscosity ν_t field.**



$$R_n^{MF} = \frac{\partial U_n}{\partial t} + \frac{\partial f_{nk}^{inv}}{\partial x_k} - \frac{\partial f_{nk}^{vis}}{\partial x_k} = 0$$

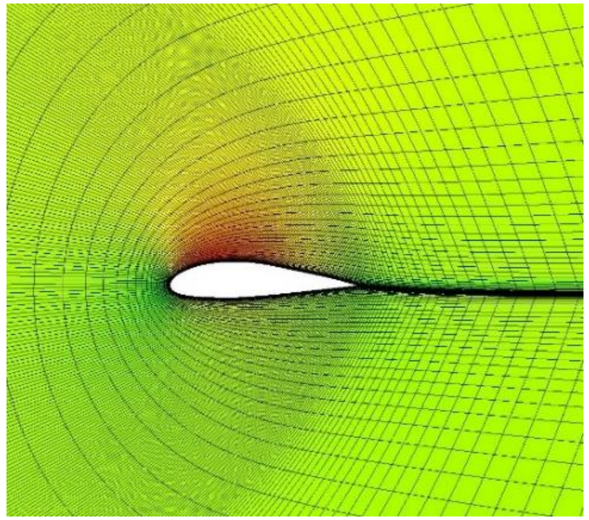
$$g(\nu + \nu_t)$$

Automated Process for finding the Optimal DNN Configuration

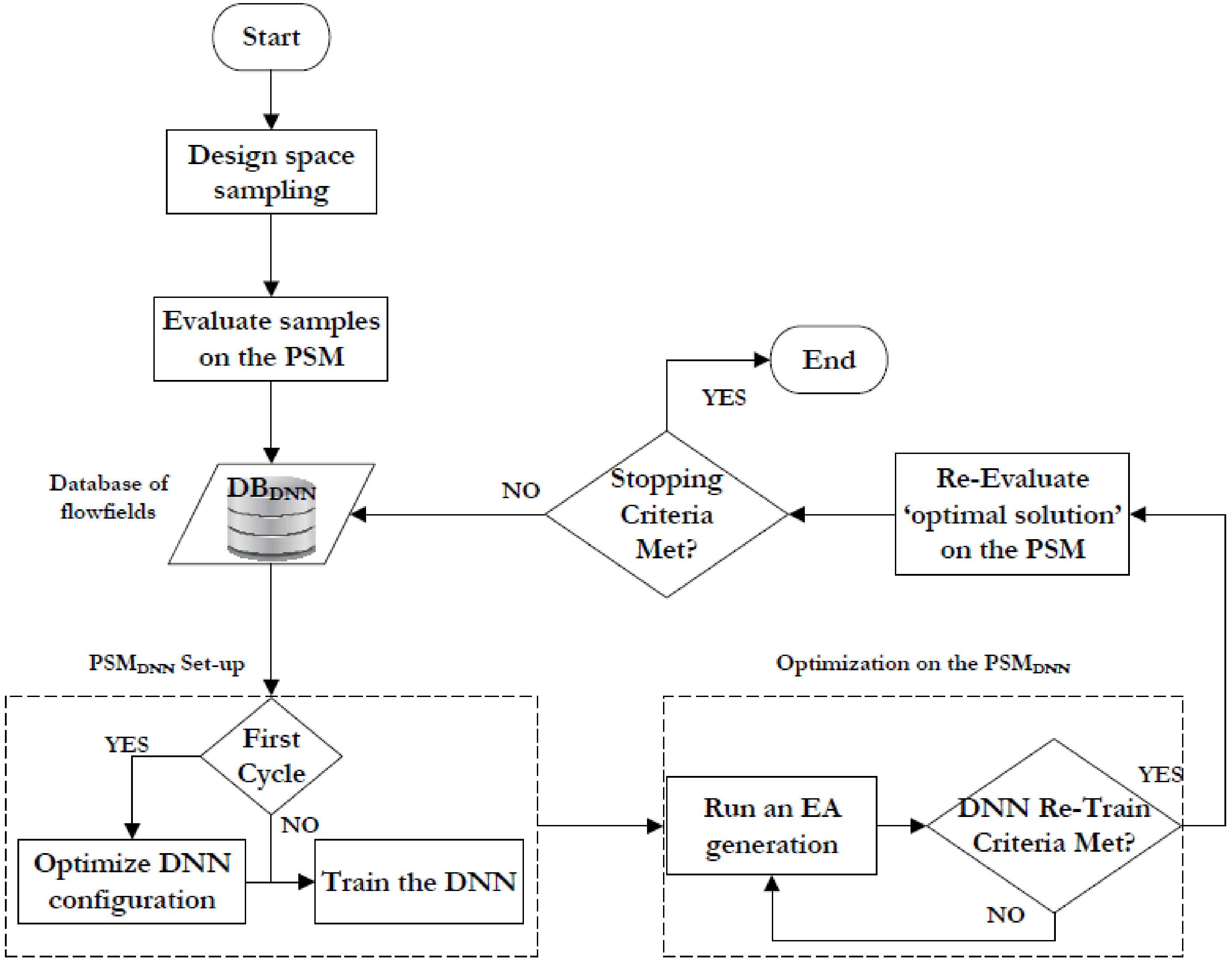
$$\begin{tikzcd}
& \text{} \arrow[dl] \arrow[dr] & \\
\boxed{\text{Which is the DNN architecture?}} \arrow[d] & & \boxed{\text{Which are the DNNs inputs?}} \arrow[d] \\
\boxed{\text{\#layers, \#neurons, activation functions etc}} & & \boxed{\text{Available input data: } \vec{x}, \vec{U}, P, \Omega, S, d \; etc}
\end{tikzcd}$$

Use EAs/MAEAs to find the optimal (minimum prediction error) configuration.

#layers	#neurons	Activation Function	Input Data	Error
6	64,128,256,1024,4096,512	relu/tanh	*x,u,v,Ω,S,d*	0.0038
7	2048,2048,256,32,64,4096,128	tanh/sigmoid	*x,y,u,Ω,P,S,d*	0.0394
5	64,128,256,512,512	relu/tanh	*u,v,Ω,d*	0.1070



DNN Surrogates for RANS – Implementation in EA/MAEA ShpO



DNN Surrogates for RANS – Demo Case C (MOO) Revisited (2/2)

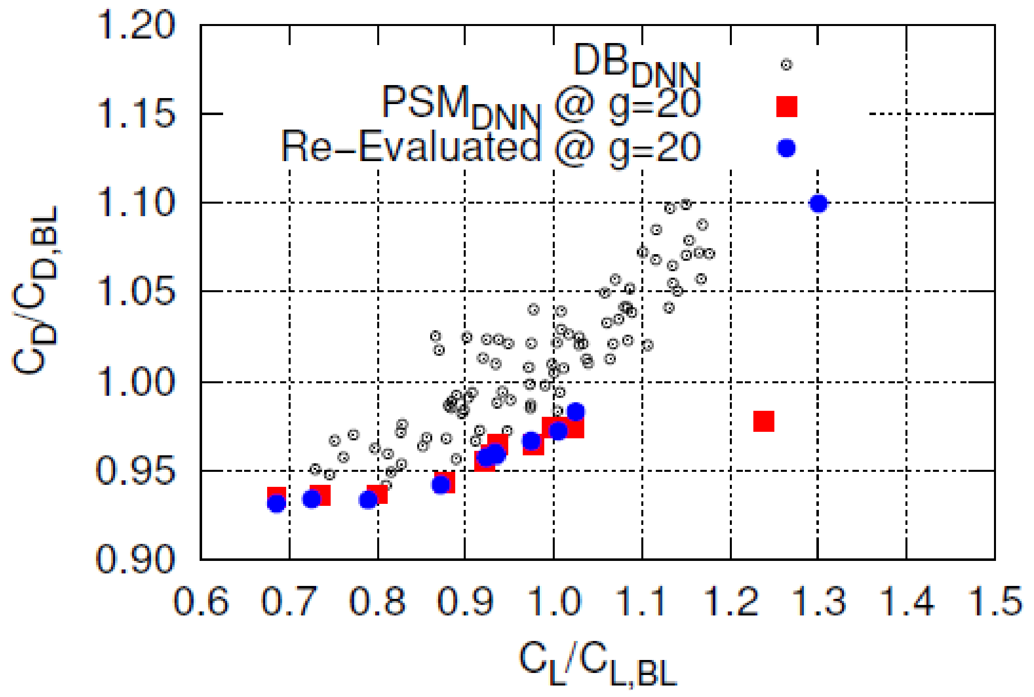
PSM_{DNN} : The PUMA code coupled with DNN for replicating the Spalart-Allmaras turbulence model. Cost=0.80PSM.

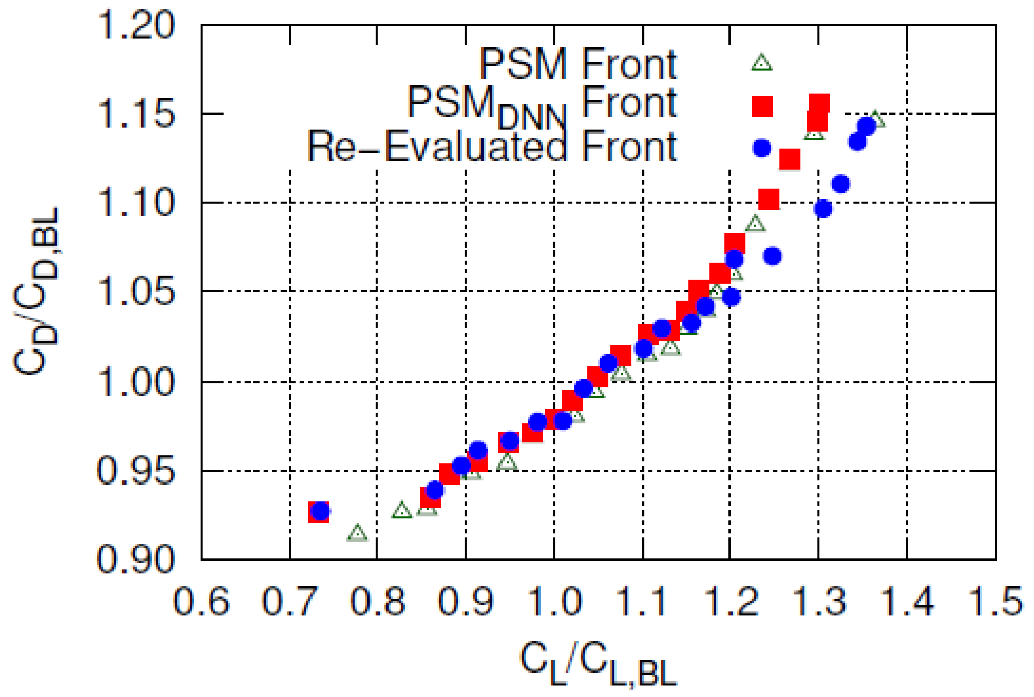
DNN architecture: Fully-Connected (FCNN) with 6 layers

Input: x, u, v, Ω, S, d.

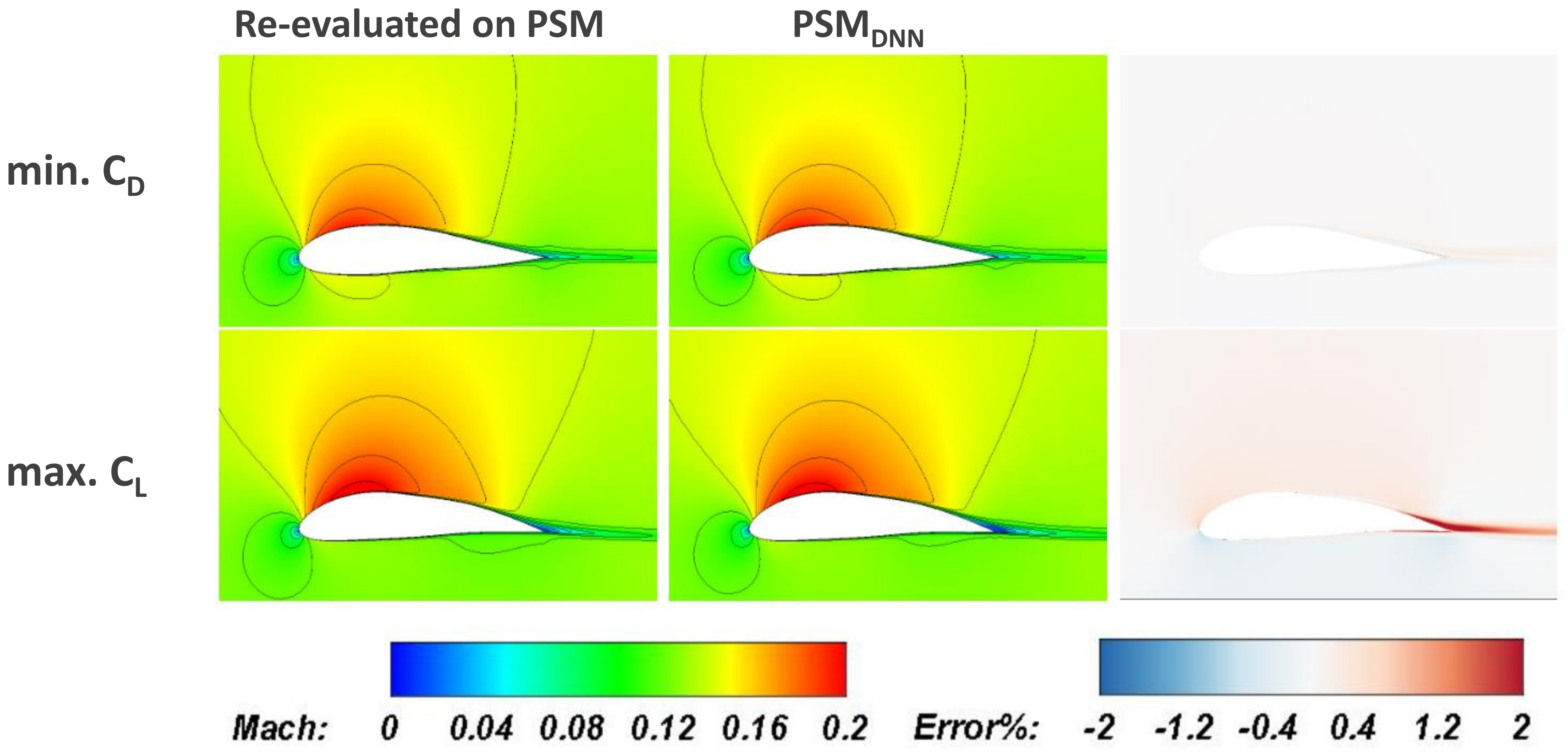
Output: Nodal turbulent viscosity.

DNN training: 75 different geometries; 2h running on 1 NVIDIA A100 GPU.





DNN Surrogates for RANS – Demo Case C (MOO) Revisited (1/2)

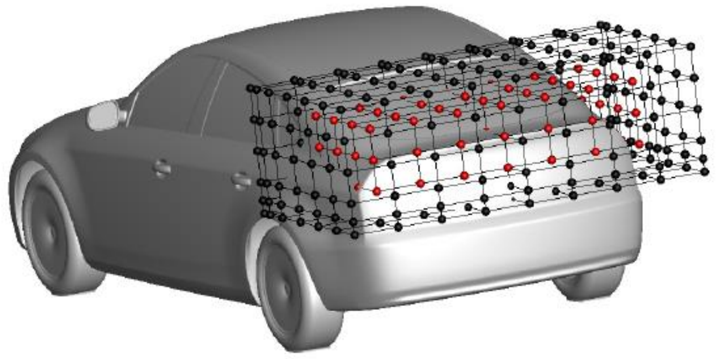


ShpO of the Drivaer Fast-back Car Model (1/2)

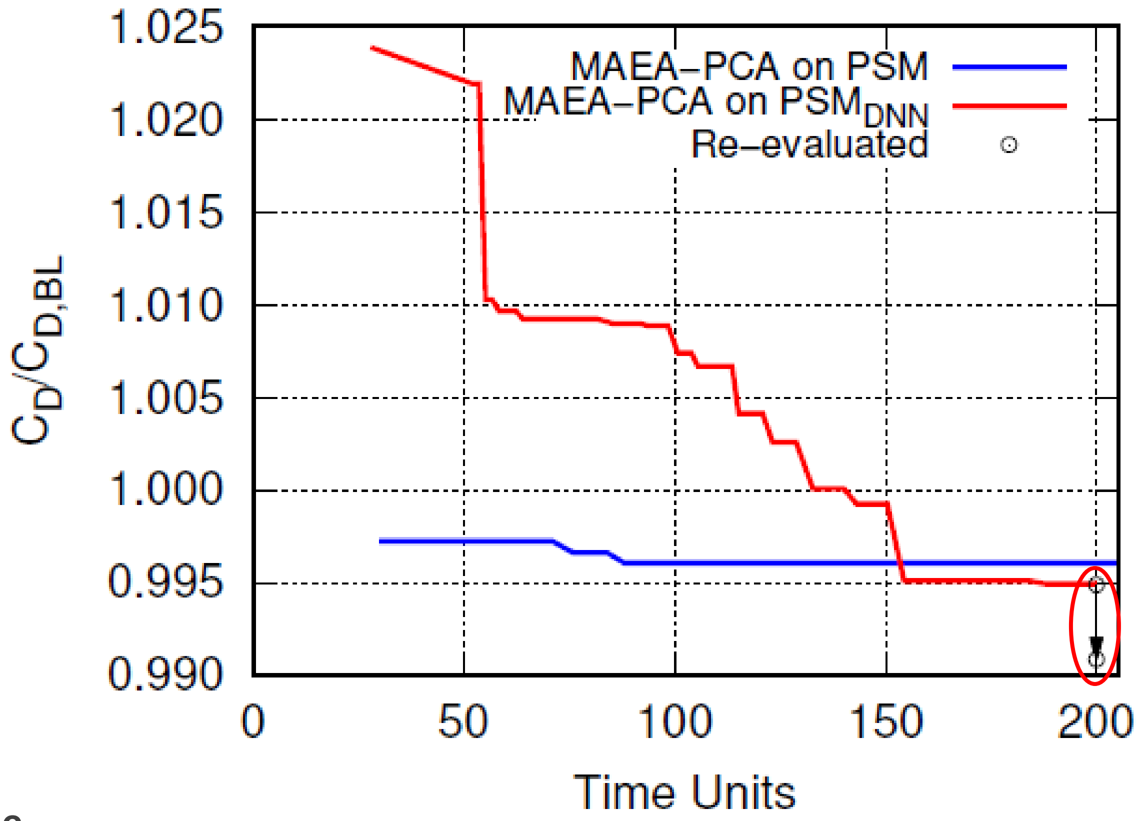
PSM$_{DNN}$: The PUMA code coupled with DNN to replicate the Spalart-Allmaras model. Cost=0.80PSM.

DNN architecture: Fully connected NN (FCNN) with 4 layers

- **Input: x, y, z, u, v, w, Ω, S, d.**
- **Output: Nodal turbulent viscosity.**



A 7×5×6 volumetric NURBS box is used to parameterize the top of the rear part of the car; control points in red are allowed to move by ±25% of their reference position in the longitudinal and ±60% in the normal-to-the ground direction. 96 DoFs. Cost per CFD evaluation ~ 1h on a single NVIDIA A100 GPU. CFD mesh with ~1.3M nodes.

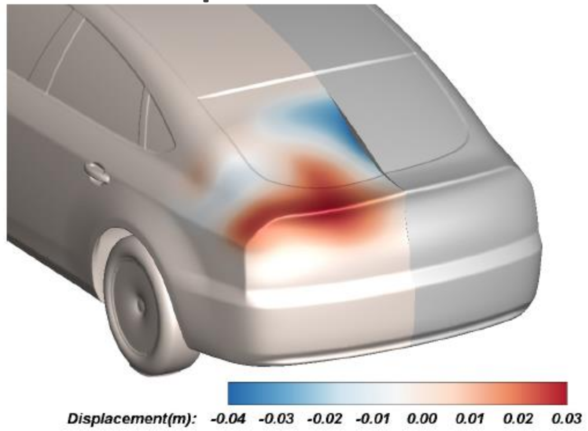


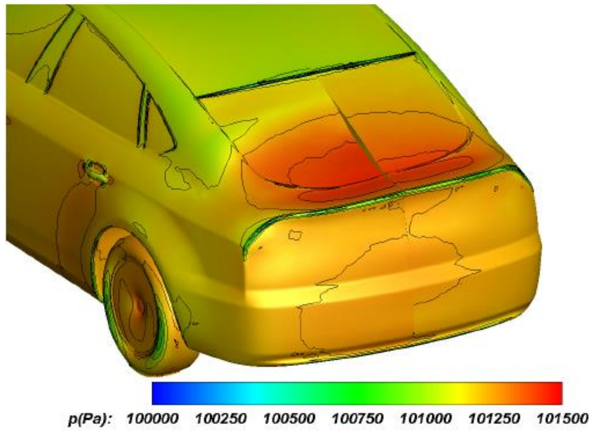
Comparison of (10, 30) MAEA-PCA on the PSM and the PSM$_{DNN}$. T^{MM} = 50, λ_e =2.

ShpO of the Drivaer Fast-back Car Model (2/2)

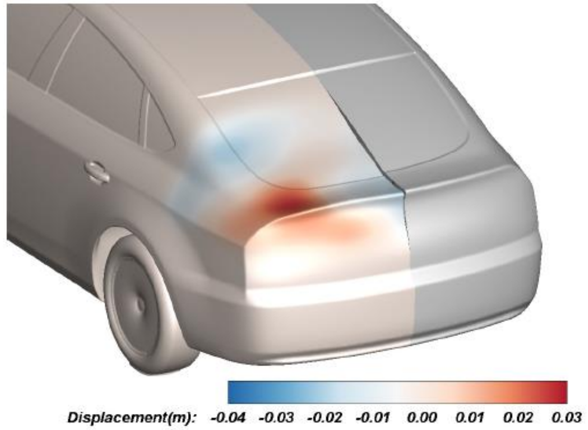
Normal displacement **Pressure distribution**

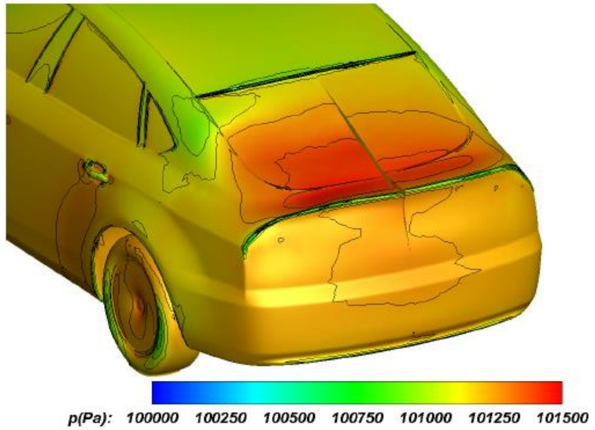
Optimised using PSM


Displacement(m): -0.04 -0.03 -0.02 -0.01 0.00 0.01 0.02 0.03



Optimised using PSM$_{DNN}$


Displacement(m): -0.04 -0.03 -0.02 -0.01 0.00 0.01 0.02 0.03



Comparison with the baseline geometry (starboard side).

Conclusions – Lessons Learned

- ✓ **Assisting EAs or any kind of stochastic population-based algorithm with surrogate models can reduce a lot the optimisation turn-around time by order(s) of magnitude. Cost reduction by even an order of magnitude can be achieved (from EA to MAEA).**
- ✓ **The type the metamodel and the way this is implemented within the search algorithm is critical.**
- ✓ **A well-coordinated distributed search is also beneficial (from MAEA to DMAEA).**
- ✓ **To tackle the curse of dimensionality, smart use of Principal Component Analysis (PCA), either for allowing the application of evolution operators to the "feature space" or for reducing the inputs seen by the metamodels may help a lot. Either linear or kernel PCA can be used (from DMAEA to DMAEA-PCA).**
- ✓ **When a GBM (e.g. an adjoint method in CFD) is available, hybridisation of GFM and GBM may further reduce the computational cost (Hierarchical/Hybrid/Memetic schemes).**
- ✓ **In MOO problems, computing the descent direction that improves the current front of non-dominated solutions is challenging (from DMAEA-PCA to Hybrid DMAEA-PCA).**
- ✓ **Parallelisation and, in particular, asynchronous search may also help a lot (from MAEA to AMAEA).**
- ✓ **Benefits from the above methods are practically superimposed.**
- ✓ **Other techniques, such as DNN-assisted flow solvers, can be incorporated and this is expected to offer additional gain in large-scale applications.**