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

Evolutionary Algorithms & Low-Cost Evolutionary Optimisation, with Applications

Dr. Varvara Asouti, Adjunct Lecturer NTUA

Athens, March 2026

EURO Greece

Scope

Introduction to evolutionary algorithms and how to make them attractive for real-world applications

<u>We will talk about:</u>

- **Population-based stochastic-based optimisation methods; Generalised Evolutionary Algorithm (EA)**
- **Cost reduction techniques in EAs**
- **Industrial applications**

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 MOO, with a single run.
- ✚ Handle constraints in the simplest possible way: through (escalated) penalties.
- ✚ Are amenable to parallelisation (simultaneous independent evaluations).

Cons:

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

The PCOpt/NTUA EA: The EASY Platform

All theory and computations presented below are based on the s/w EASY developed by the PCOpt/NTUA, using a (μ,λ)EA as the background optimisation method.

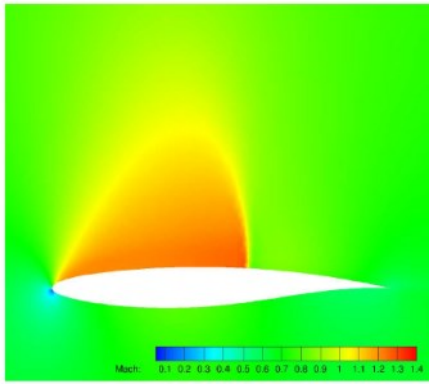
The Evolutionary Algorithm SYstem
http://velos0.ltt.mech.ntua.gr/EASY
http://147.102.55.162/EASY

Terminology - Minimisation or maximisation, etc.

- Single-Objective Optimisation, SOO
- Multi-Objective Optimisation, MOO

A two-objective problem with a Multi-Objective Optimisation (MOO) method:

max C_L min C_D	min $-C_L$ min C_D	min $1/C_L$ min C_D	max C_L max $-C_D$



Reformulate a SOO problem to be solved with a SOO method:

min $C_D + w/C_L$	min $C_D + 1/C_L$	min $C_D + 10/C_L$

Downgraded to a Constrained SOO problem:

min C_D subject to: $C_L=1.2$	min C_D subject to: $C_L>1.2$

Constrained Optimisation Problem Statement

$$\min \vec{F}(\vec{b}) = \min\{f_1(\vec{b}), \dots, f_{M_o}(\vec{b})\}$$
$$\text{subject to } c_j(\vec{b}) \leq 0, \text{j} = 1, M_c$$
$$\vec{b} \in \mathbb{R}^N$$

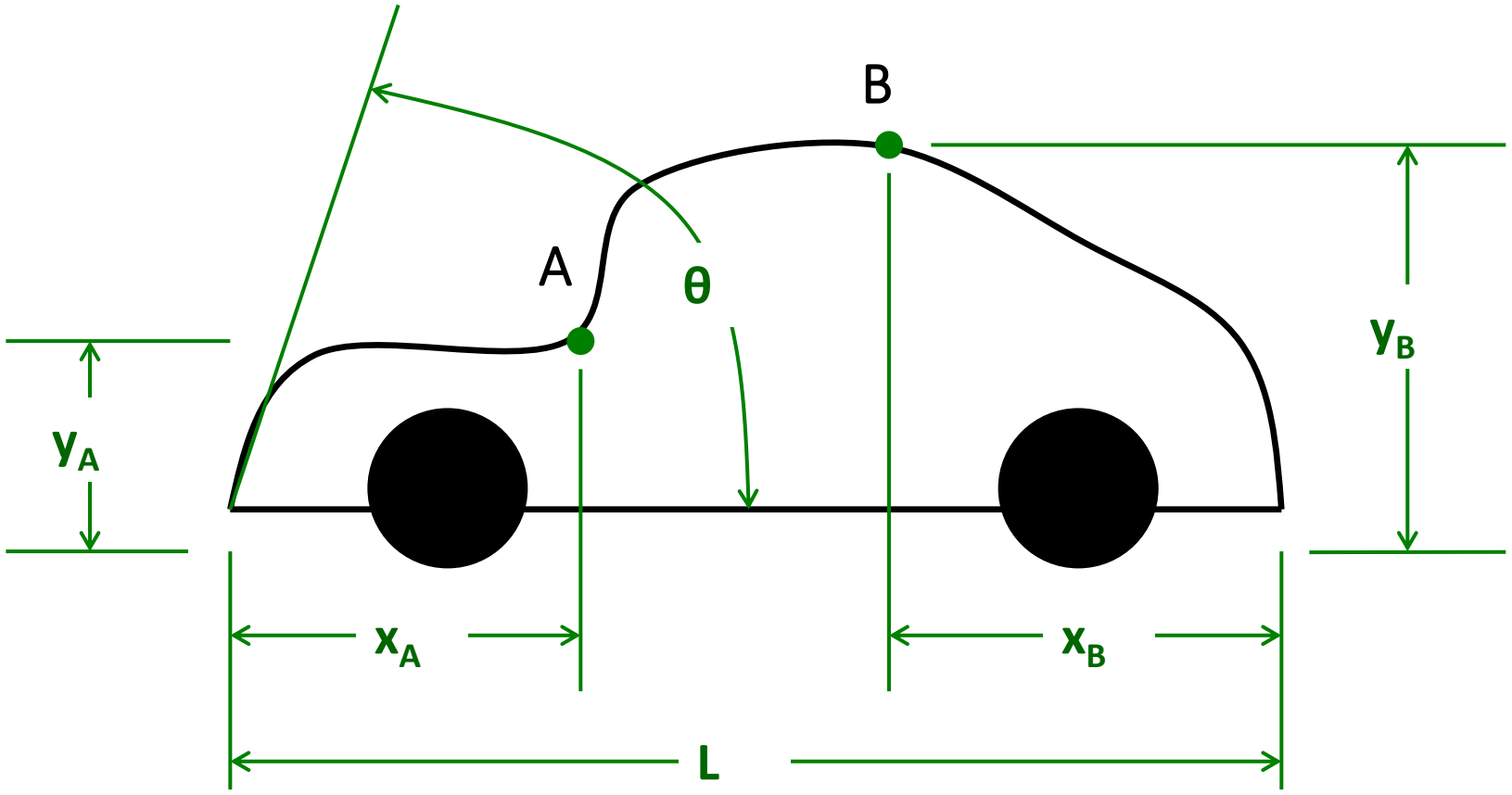
- **N design (optimisation) variables**
- **M_o objectives (SOO or MOO)**
- **M_c constraints**

<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)**

Gradient-Free Optimisation – The (μ,λ) EA for Single-Objective Optimisation (SOO).

Very simple Working Example



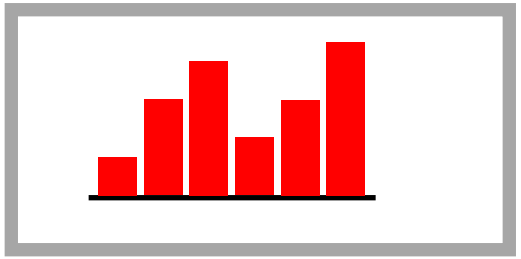
N=6 degrees of freedom (DOFs), design or optimisation variables.
Objective (cost) function: min. (drag)
Evaluation s/w: a CFD solver

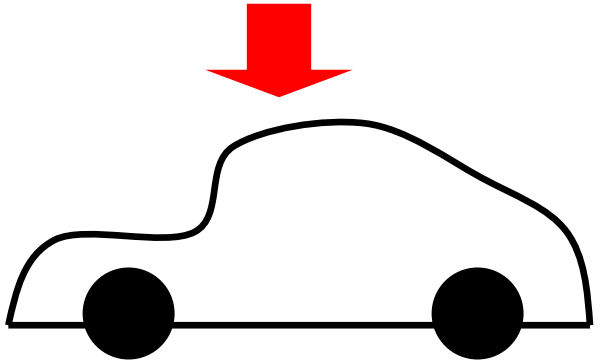
The evaluation S/W

L	x_A	y_A	x_B	y_B	θ

$\vec{b}$ =

b_1=…	b_2=…	b_3=…	b_4=…	b_5=…	b_6=…



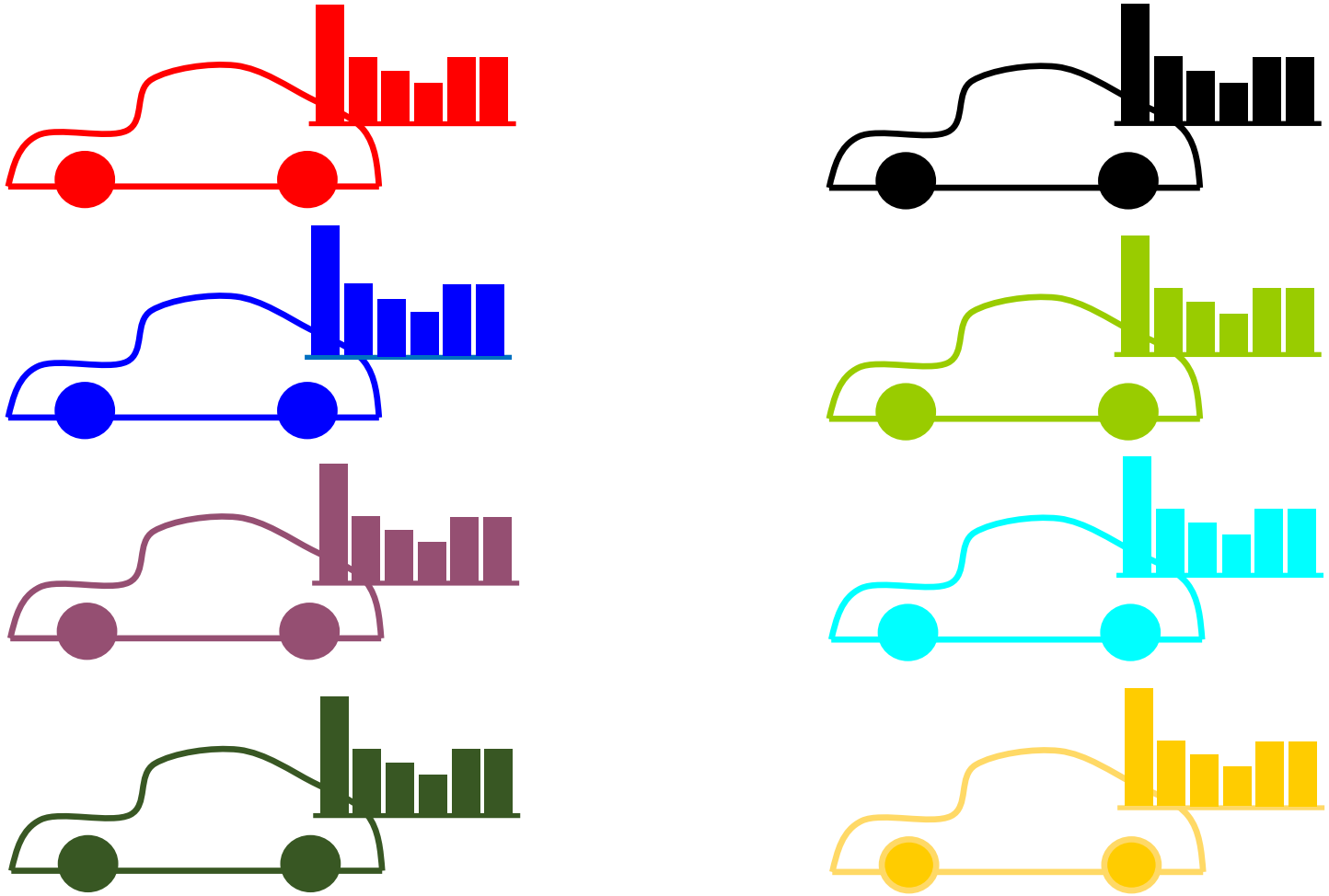


Evaluation script

$F=C_D$=…

No access to the source code is needed!!!

Very simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (1/8)



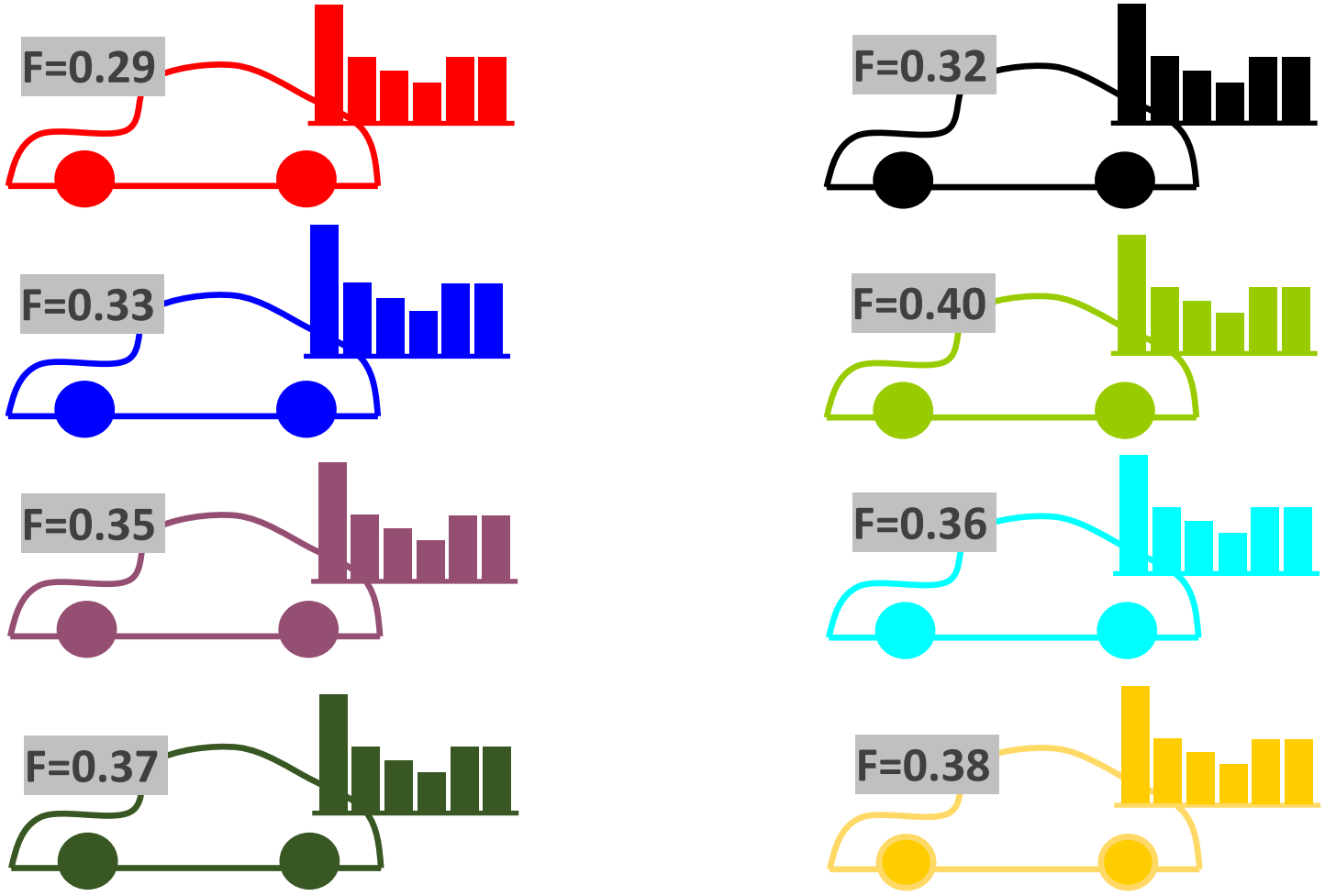
(μ,λ) Evolutionary Algorithm in a Minimisation problem (min. F)

Start by λ randomly selected offspring.

(The b_n values of the initial population have been randomly selected (between user-defined lower and upper bounds).

(μ=λ is just a naïve simplification!)

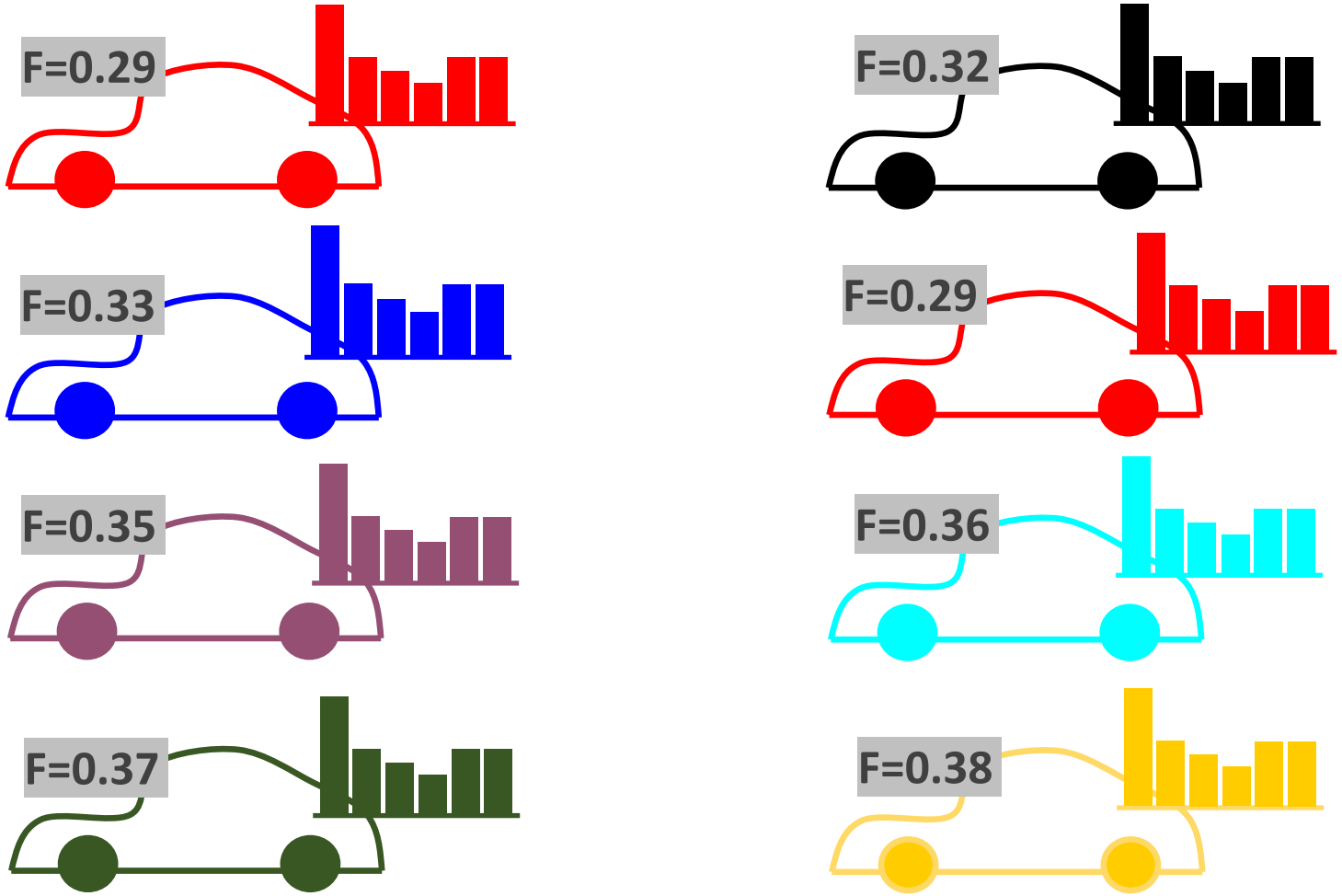
Very simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (2/8)



Evaluation of the offspring population. Cost:
λ (=8) calls to the evaluation script.
$F=C_D$ (to be minimised)

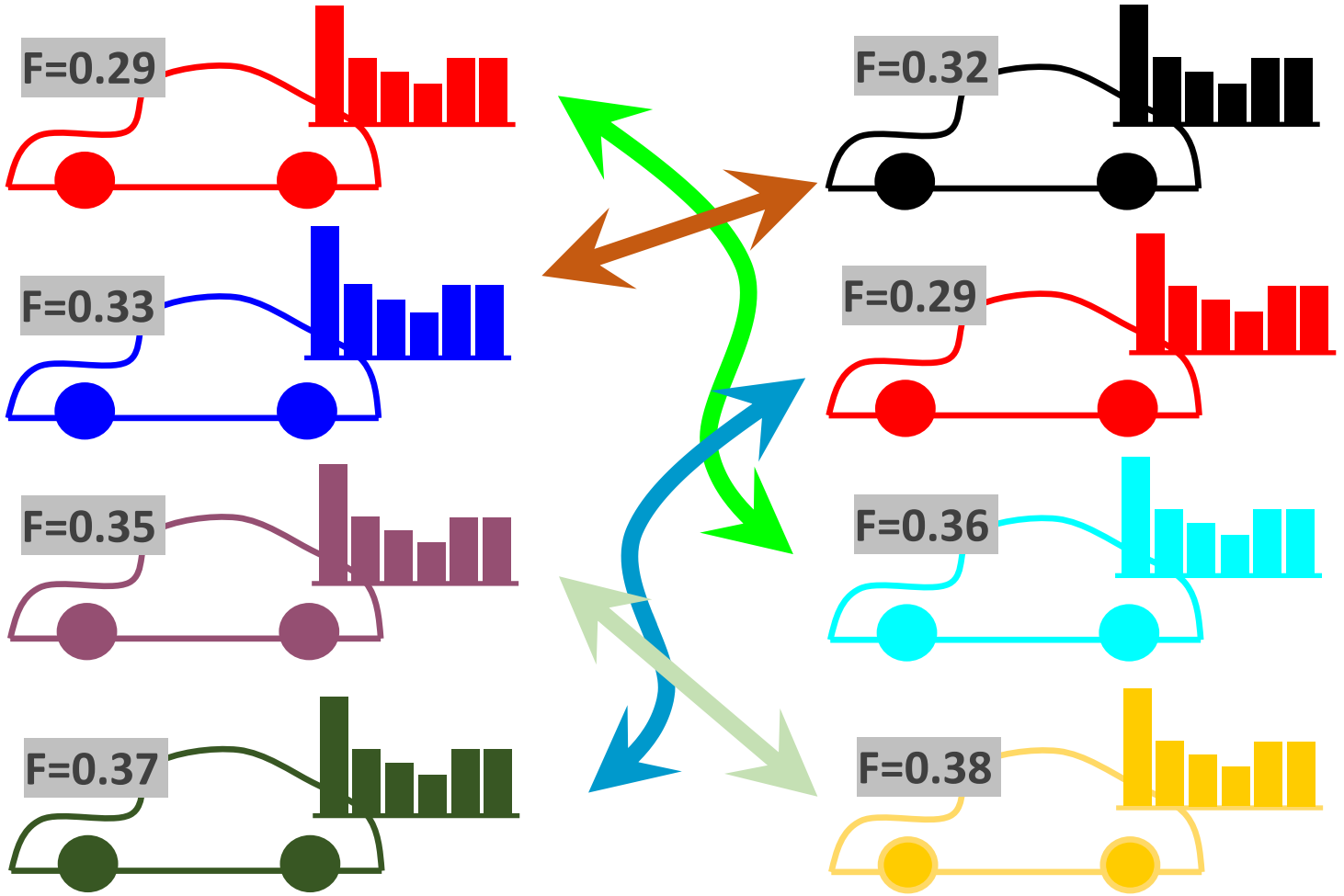
(Concurrent evaluations on a multi-processor system is possible)

Very simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (3/8)



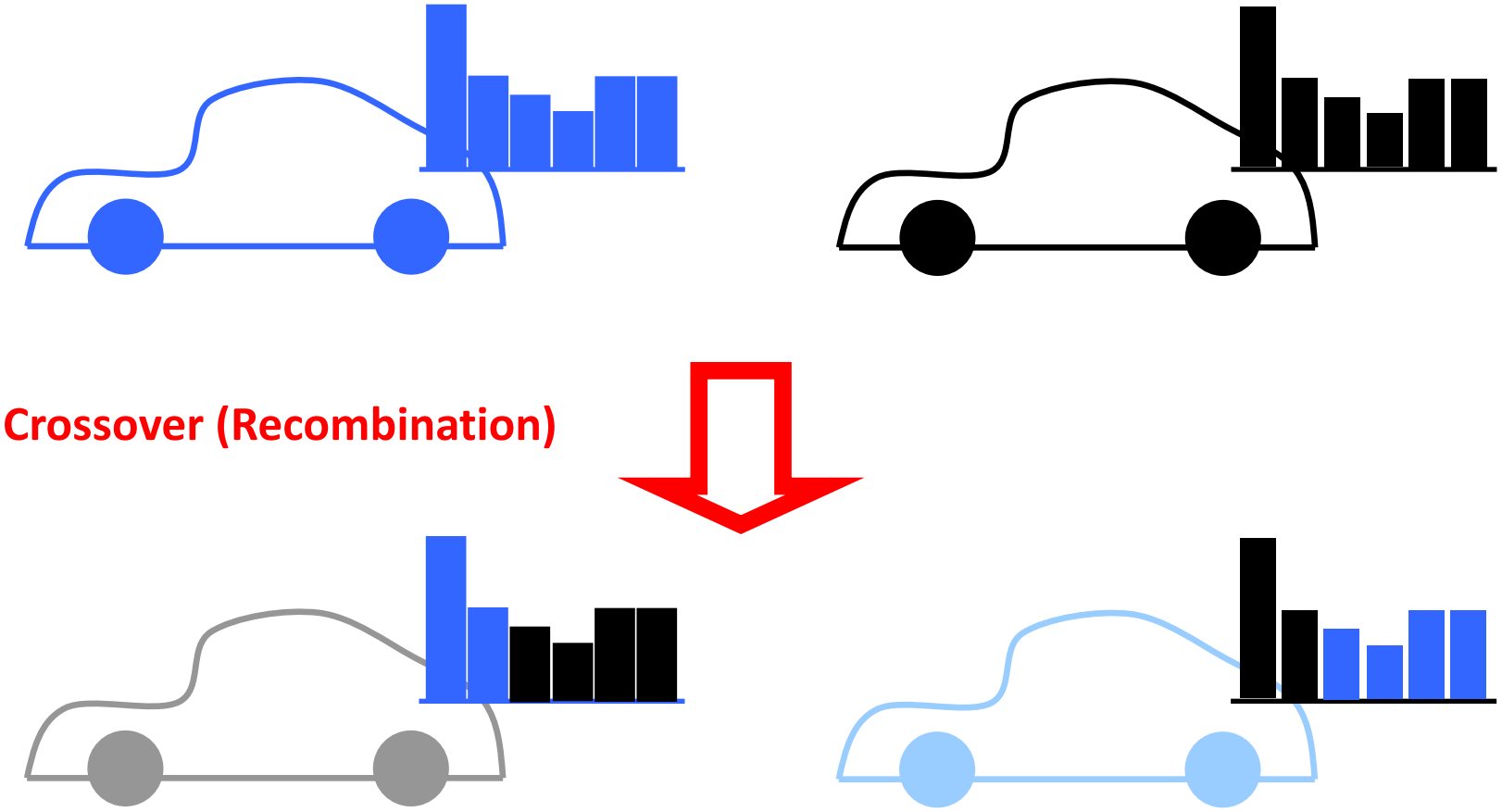
A naïve parent selection scheme:
Replace the worst performing individual with the best.
The population of μ parents will be used to create new λ offspring.

Very simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (4/8)



Randomly **Mate** them.

Very simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (5/8)



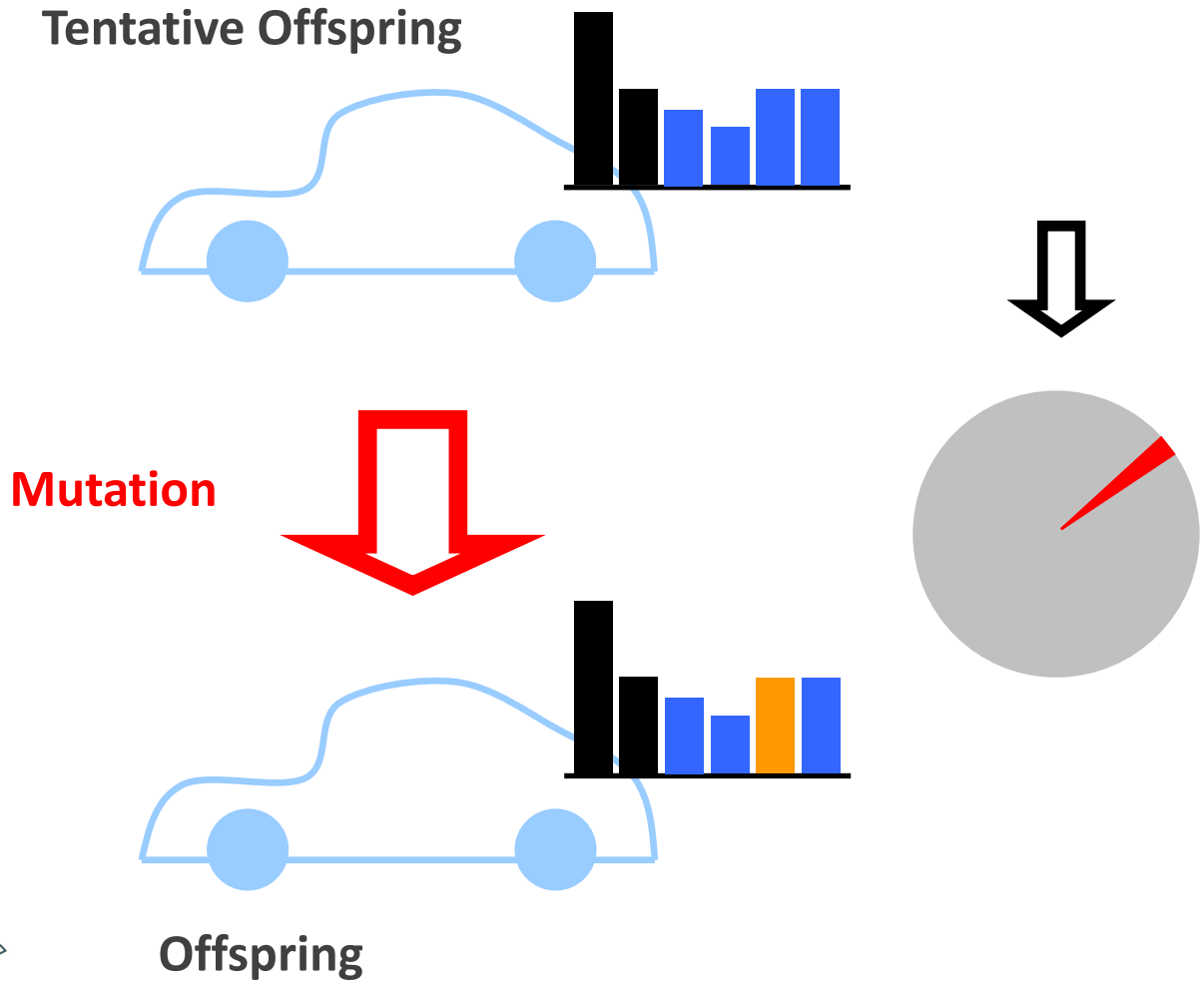
Two (tentative) Offspring

Application of evolution operators on the pairs selected from the parent population.

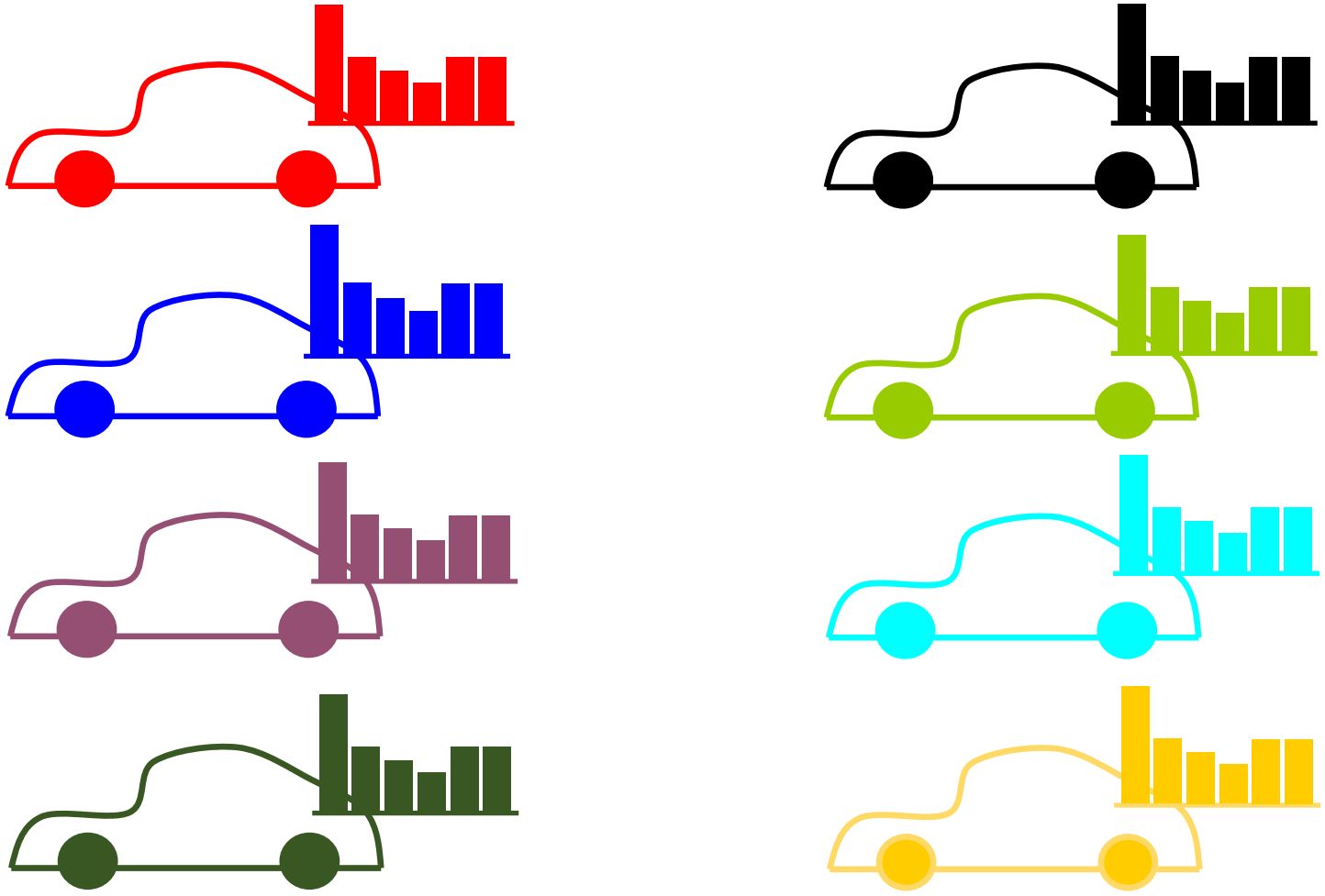
(Based on a Random Number Generator or RNG)

(Using 2 parents to create 2 offspring is not necessarily the best option)

Very simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (6/8)



Very simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (7/8)



The new offspring population with λ offspring has been created.
A new generation starts here!

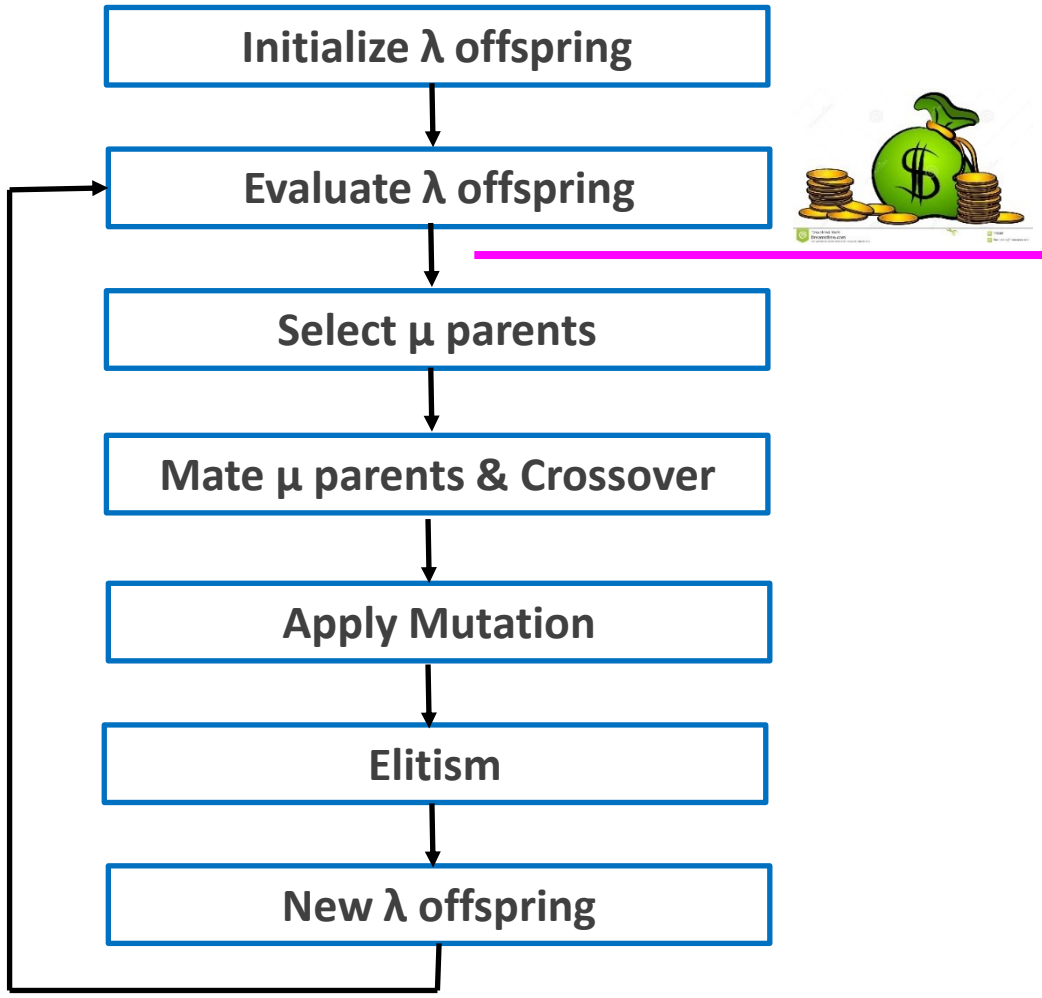
A data base (DB) has kept all evaluated individuals paired with their objective function values, to avoid repeating the same evaluations in the future generations.

Very simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (8/8)

Termination Criteria: Terminate the evolution when:

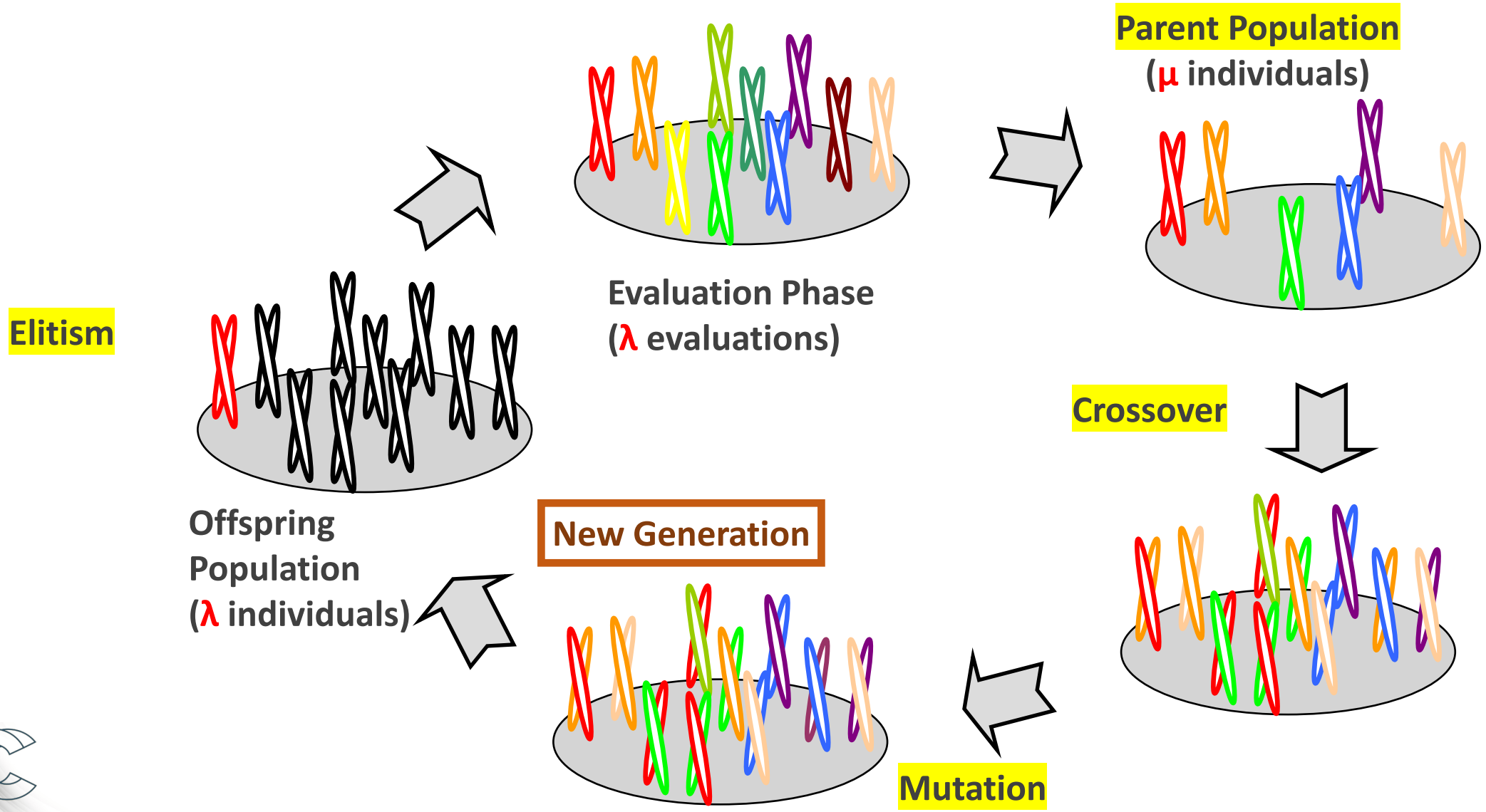
- **The user-defined max. number of evaluations has been reached (budget!)**
- **During the last κ (user-defined) evaluations, there is no improvement in the objective function value.**

The (μ,λ) EA for SOO



Rule of the Thumb:
λ= (2…4) · μ

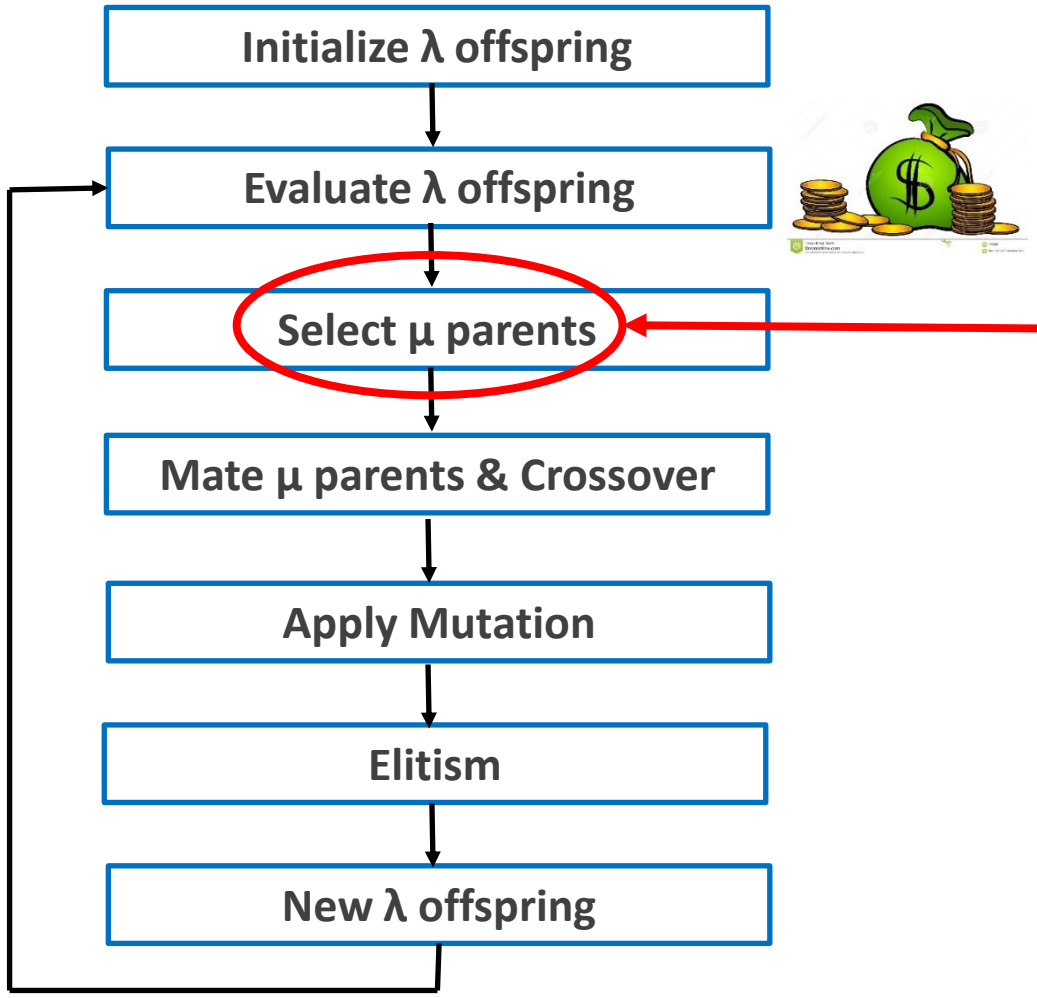
The (μ,λ) EA



Dr. Varvara G. Asouti, vasouti@mail.ntua.gr

The (μ,λ) EA for Multi-Objective Optimisation (MOO)

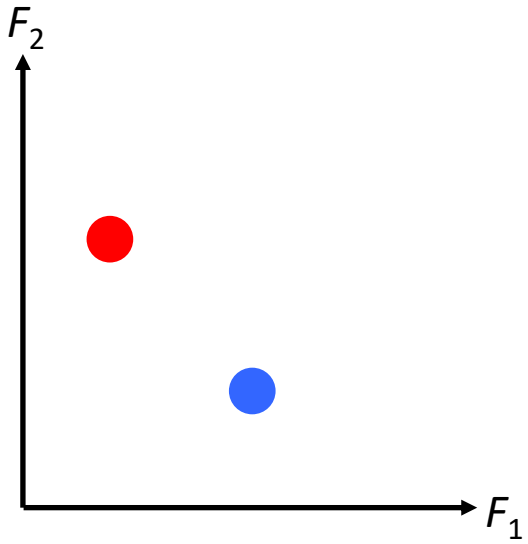
The (μ,λ) EA for MOO



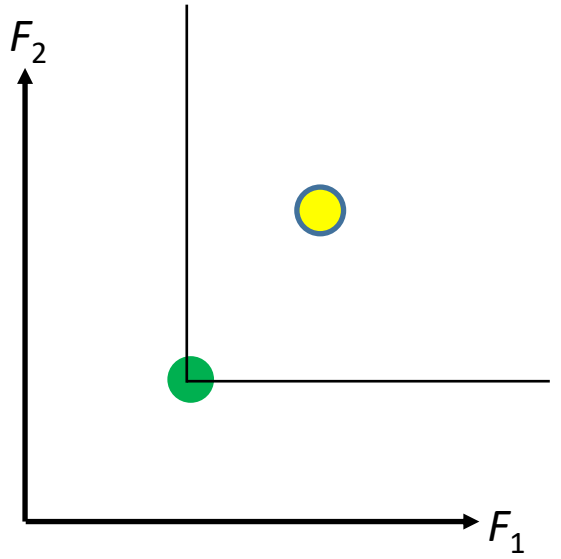
Transform vector (F_1,…, F_{Mo}) to a **scalar utility function Φ (dominance driven)** before selecting μ parents based on **Φ** values

Multi-Objective Optimisation (MOO) – Towards the Pareto Front

Min. F_1
Min. F_2



To improve one objective will imply to introduce a loss regarding the other ones.



The yellow solution is dominated by the green one. Or, a point A dominates B, if A is better or equal in all criteria and strictly better in at least one.

In MOO applications, a GFM can compute the Pareto Front or Front of Non-Dominated Solutions.

MOO – The Pareto front

Bugatti Veyron
€1100000
400 km/h

VW Golf
€25000
220 km/h

Simca 1100
€2000
60 km/h

F_2
Price

Min. F_1
Min. F_2

(Top Speed)$^{-1}$ F_1

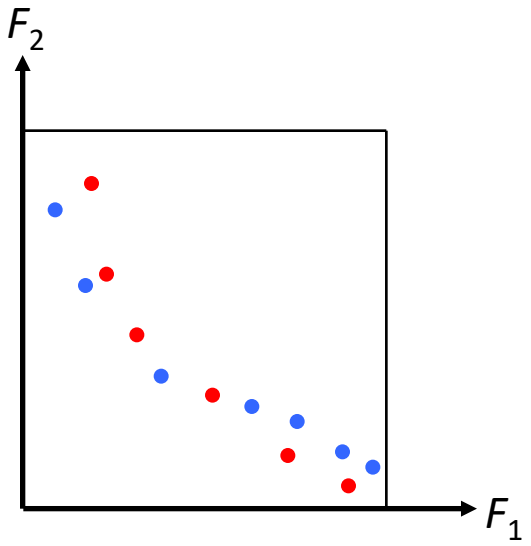
The Pareto front in the space of objective functions, in MOO, stands for a set of solutions that are non-dominated to each other but are superior to all other solutions in the search space. By moving along the front, one could reduce car price at the expense of lower top speed or increase top speed at the expense of higher car price, but cannot improve both at once!

Use the reciprocal or negative of a Quantity of Interest (QoI) as objective function, to switch between min. and max.o

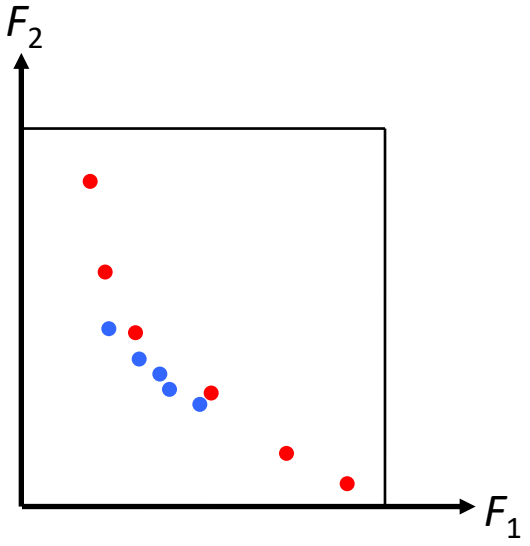
Comparing Fronts of Non-Dominated Solutions

Min. F_1
Min. F_2

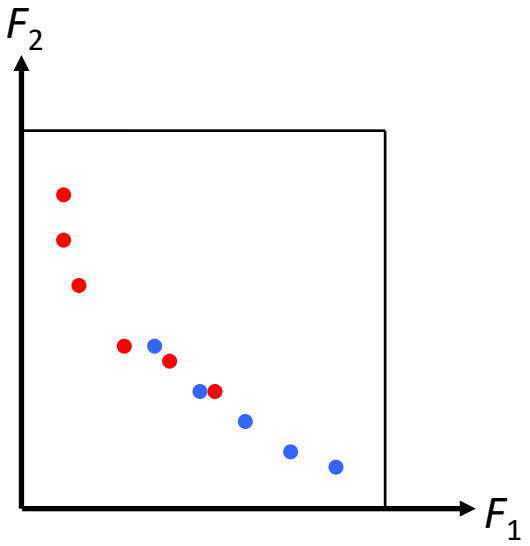
Case A



Case B



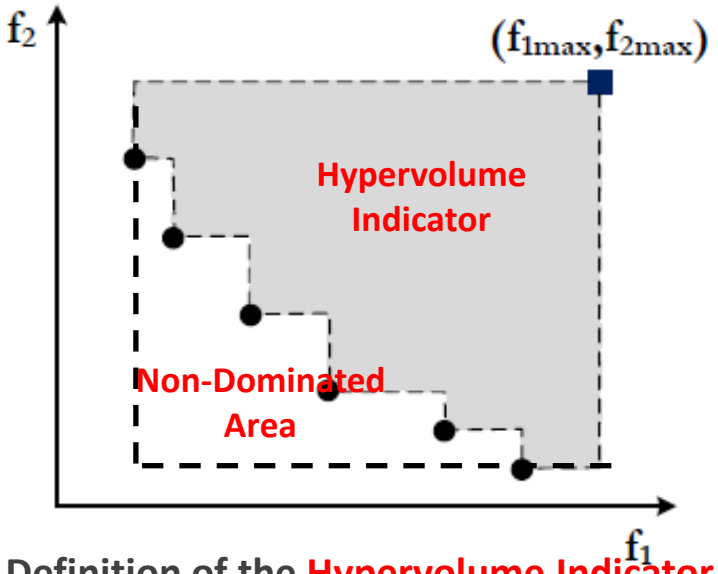
Case C



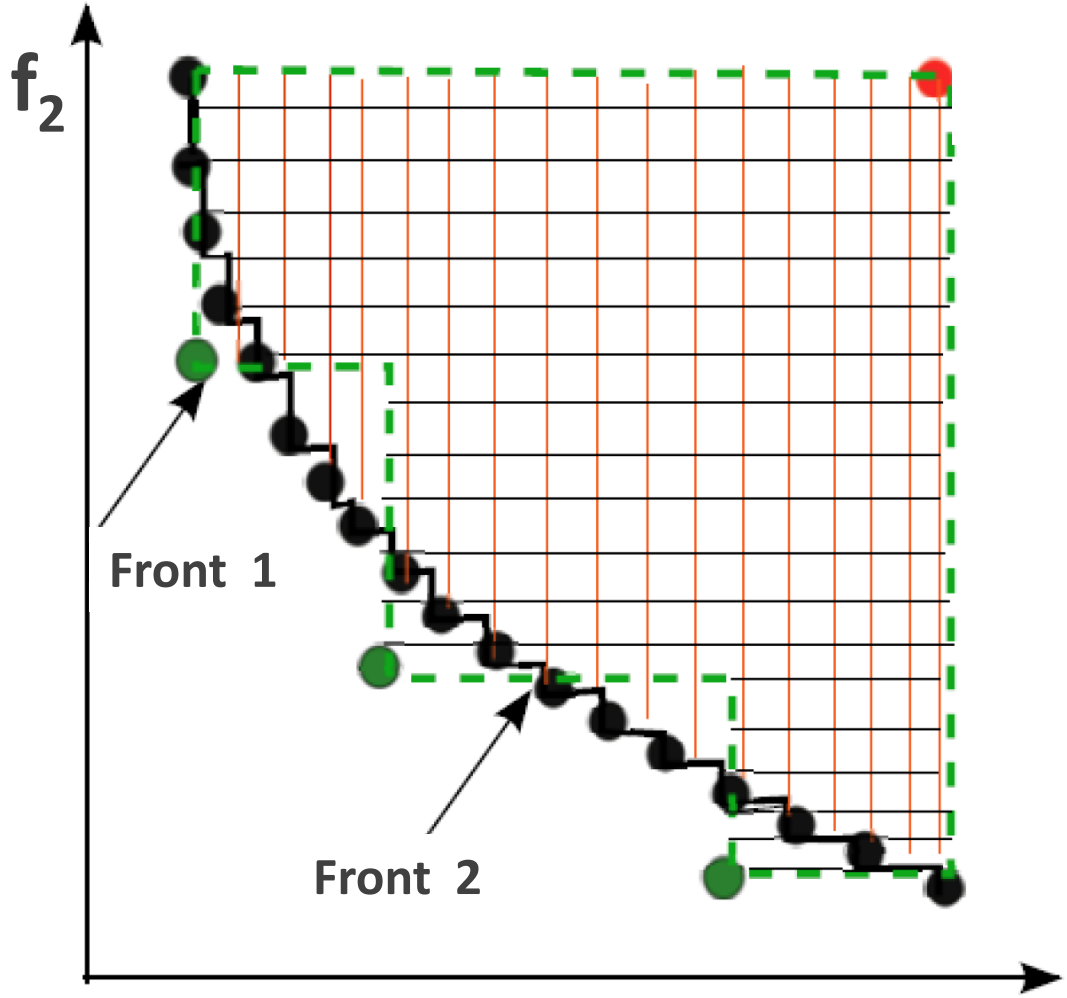
Comparison of computed fronts of non-dominated solutions
Visualisation/interpretation issues if M_o>3.
Decision Making!

The Hypervolume Indicator (HI)

Min. F_1
Min. F_2



Definition of the **Hypervolume Indicator** (for min. f_1 & min. f_2).



The (μ,λ) EA – Parameters' Coding & Basic Operators

(μ,λ) EA: Notations – Key Terms

Notations:

μ = parent population size

λ = offspring population size

e = elite population size

g = generation index

ρ = number of parents to form an offspring

$$P_{\mu}^{g}, P_{\lambda}^{g}, P_{e}^{g}$$

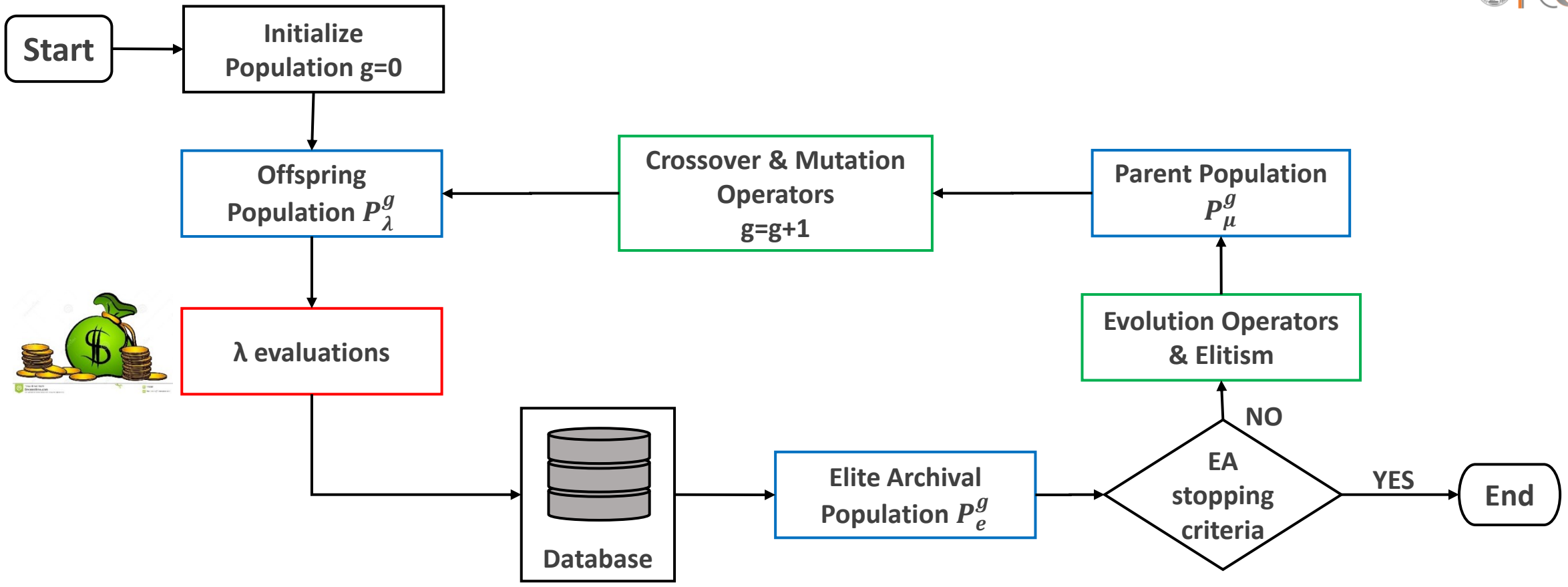
Key Terms:

Individual: Candidate solution

Genes: parameters (design variables) characterizing an individual

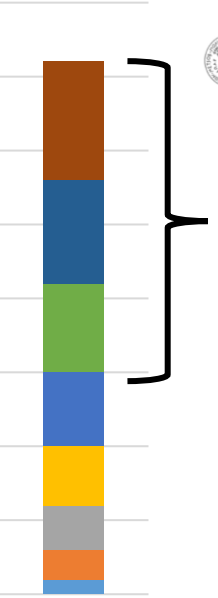
Chromosome (genotype): the set of genes of an individual

Flowchart of Generalised (μ,λ) EA



Parent Selection & Elitism (1/2)

$$P_\mu^g \leftarrow (P_\mu^{g-1} \cup P_\lambda^g)$$



Linear Ranking: $P_\mu^{g-1} \cup P_\lambda^g$ **individuals are sorted based on their fitness value. The probability of selecting an individual depends on its rank in a linear manner, i.e. practically the μ best members are selected.**

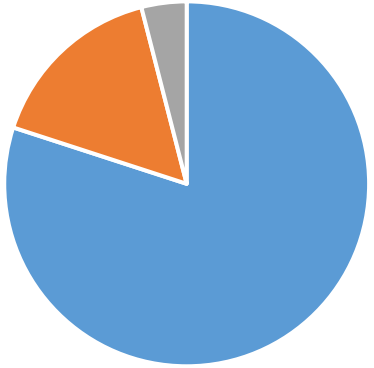
Proportional Selection: $P_\mu^{g-1} \cup P_\lambda^g$ **individuals are associated with a probability based on their fitness value (smaller values correspond to higher selection probability). A roulette wheel is considered, in which each individual is associated with a slot with angular width proportional to its selection probability; this should be turned μ times to select μ parents.**

Parent Selection & Elitism (2/2)

Tournament Selection: Randomly select k individuals from $P_{\mu}^{g-1} \cup P_{\lambda}^{g}$ and sort them based on their fitness value. Select the best among them with a probability p, the second best with p(1-p) and so on and so forth. Repeat this procedure μ times.

k: tournament size

p: tournament probability



k=3

p=0.8

80% for the blue, 16% for the orange and 4% for the grey to be selected as parent

Elitism:

Replace the worst members of P_{λ}^{g} with a few elite individuals from P_{e}^{g} →

Practically increase the probability of an elite member to be selected as parent.

Chromosome representation

Real coding: Chromosomes are strings of real values, e.g. [3.45, 5.12, -7.68, 9.32, 4.77]

Binary coding: Chromosomes are binary strings, e.g. [01101001110101001 10]

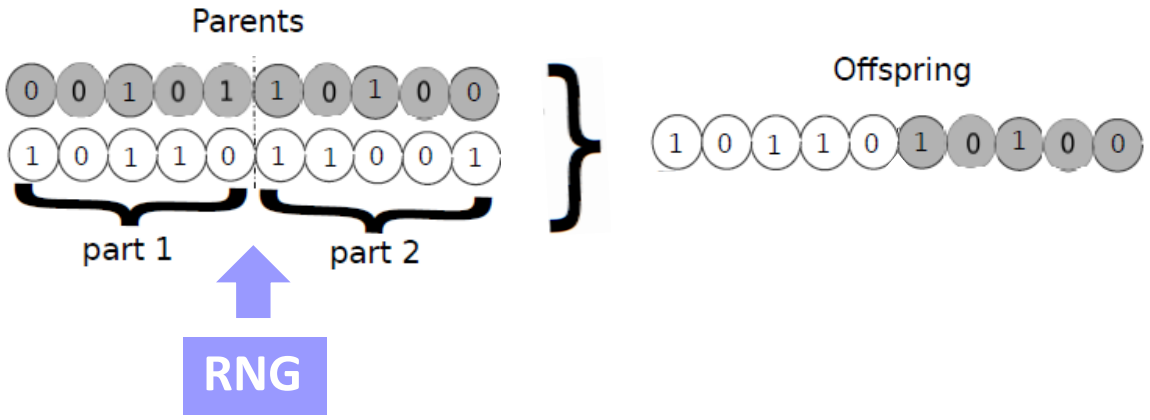
$$b_i \in \left[b_i^{min}, b_i^{max}\right] \qquad \Delta b_i = \frac{b_i^{max} - b_i^{min}}{2^{bits} - 1}$$

Gray of reflected binary coding: ordered binary encoding so that two successive integer values differ in only one bit/binary digit. Why?

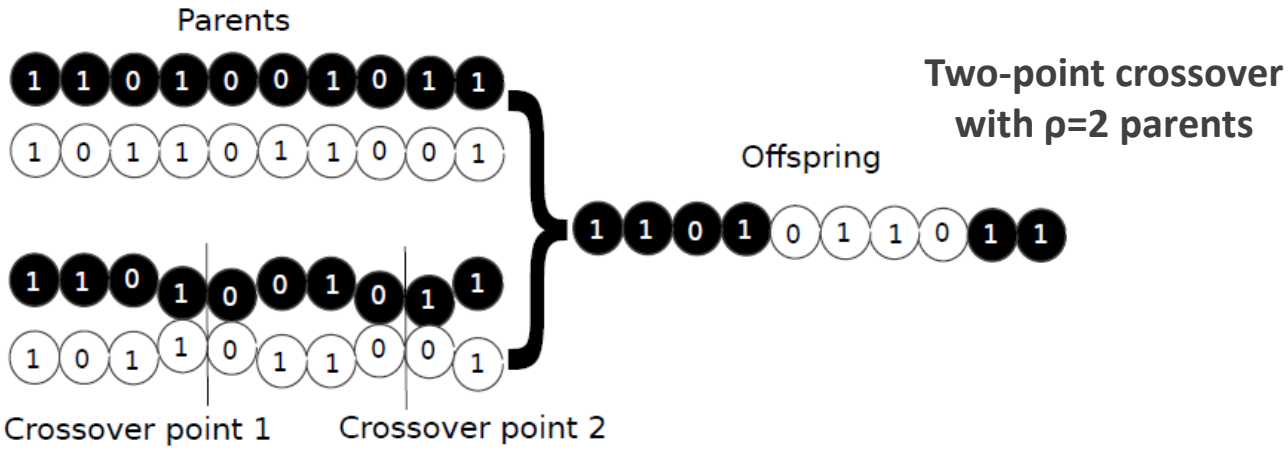
Decimal	Binary	Gray
3	0011	0010
4	0100	0110

Binary Crossover

- **One-point crossover**



- **Two-point crossover**



- **One-point per variable**
- **Two-point per variable**
- **etc.**

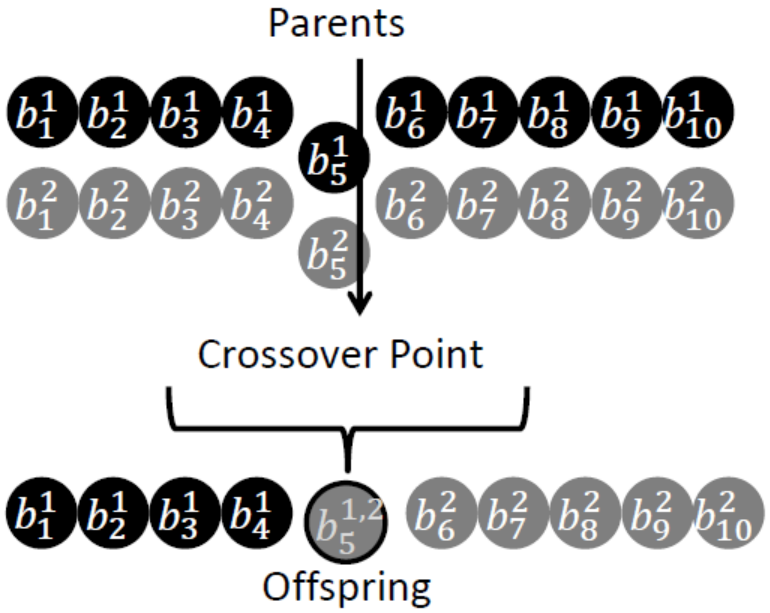
Real Coding: Crossover

- One- or two-point crossover
- Discrete
- Intermediate
- Simulated binary
- etc.

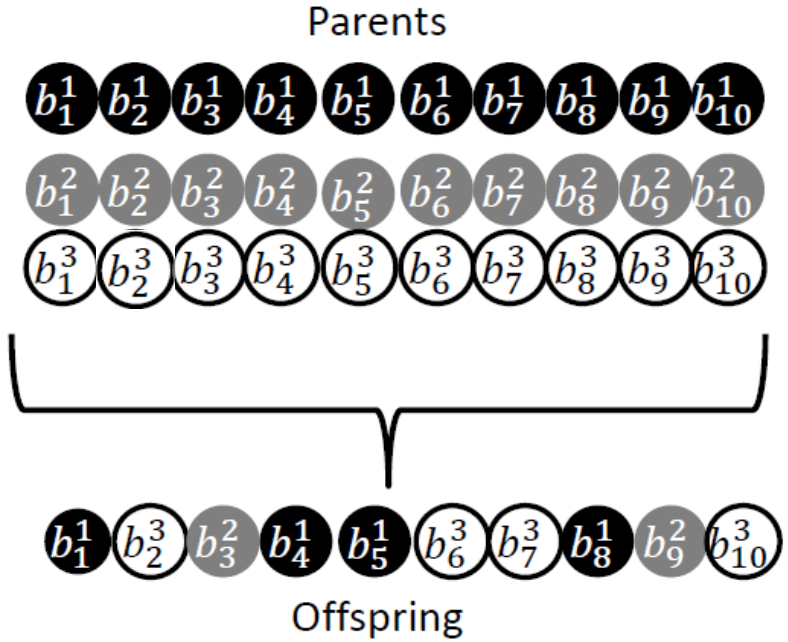
$$\vec{b} = \left[(1+\beta)\vec{b}^{P1} + (1-\beta)\,\vec{b}^{P2}\right] \quad \boldsymbol{\beta} = \begin{cases} (2\boldsymbol{u})^{1/(n+1)} & ,\boldsymbol{u} < \mathbf{0.5} \\ \left(\dfrac{\mathbf{1}}{\mathbf{2(1-u)}}\right)^{1/(n+1)} & ,\boldsymbol{u} \geq \mathbf{0.5} \end{cases}$$

$$\boldsymbol{u} \in [\mathbf{0},\mathbf{1}], \mathbf{n} \in [\mathbf{1}, \boldsymbol{N}]$$

Simulated Binary Crossover (SBX)



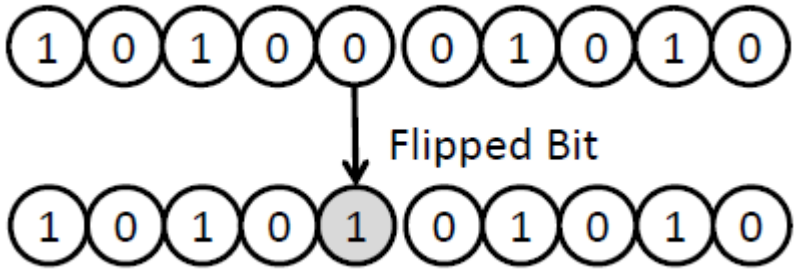
One-point crossover with ρ=2 parents



Discrete crossover with ρ=3 parents

Mutation

Binary/Gray encoding:

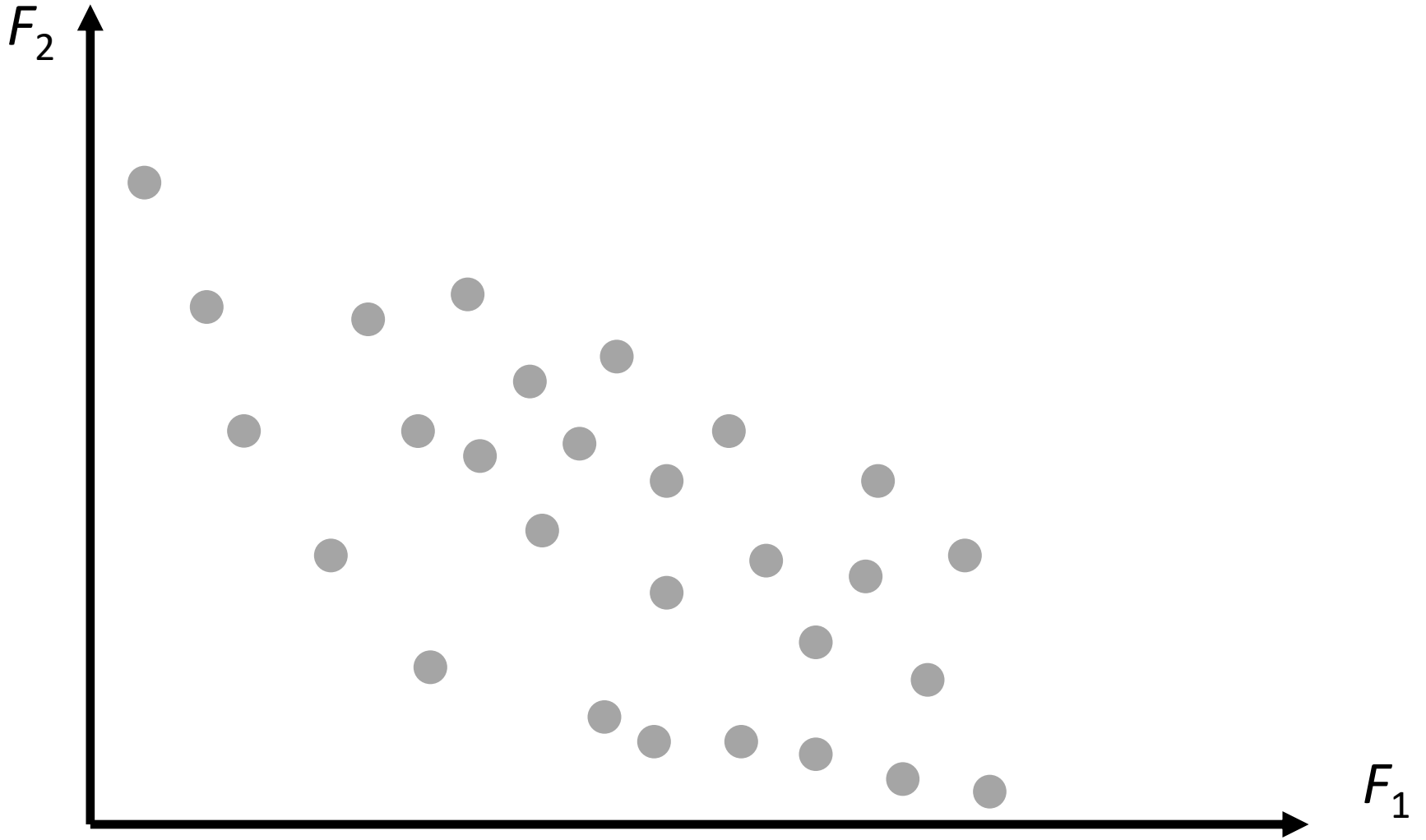


Real encoding:

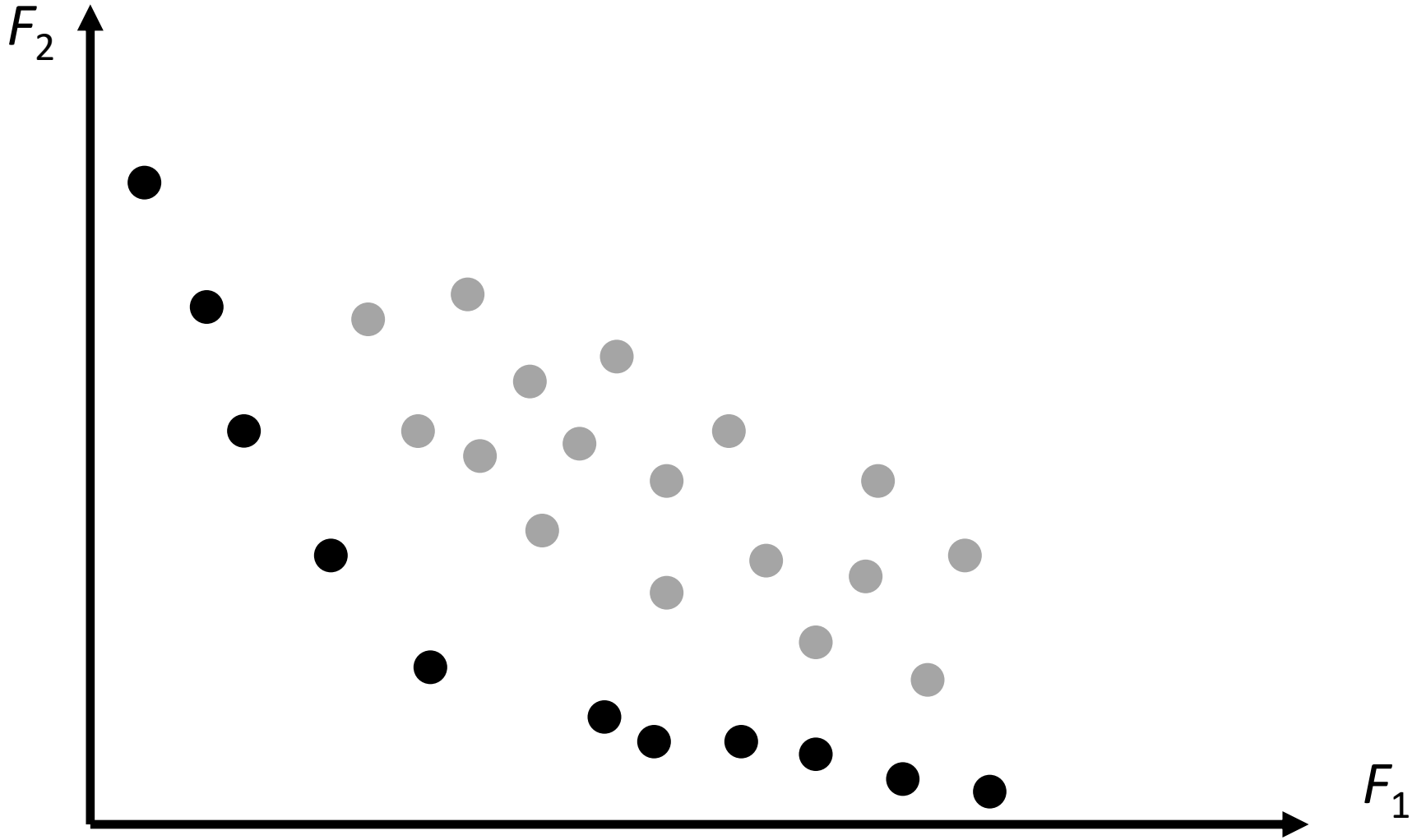
$$\vec{b} = \begin{cases} \vec{b} + D\left(g, \vec{b}^{max} - \vec{b}\right), r > 0.5 \\ \vec{b} - D\left(g, \vec{b} - \vec{b}^{min}\right), r \leq 0.5 \end{cases}$$

D: depends on the current generation and the maximum generations

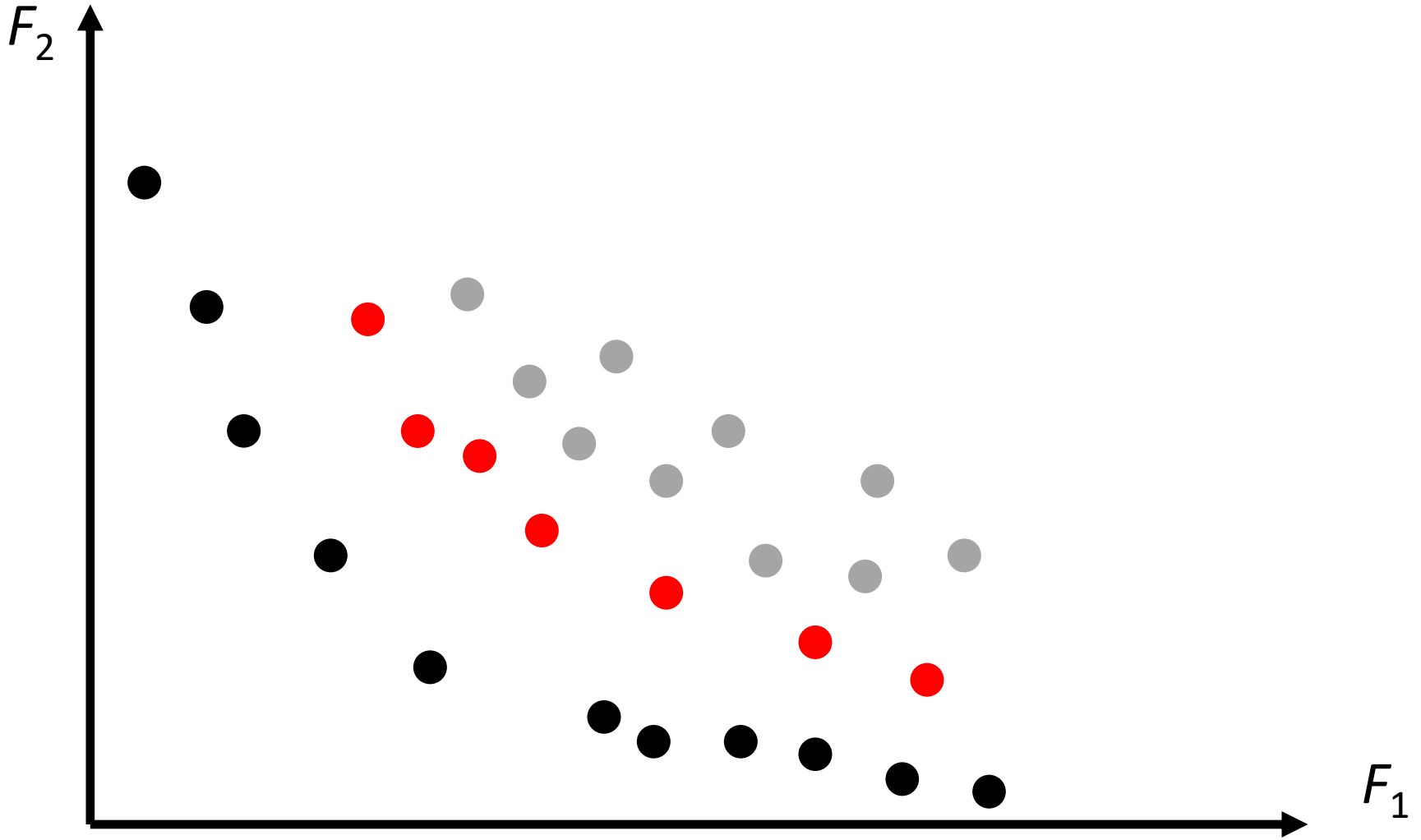
Computing Φ in MOO problems - Front Ranking (1/5)



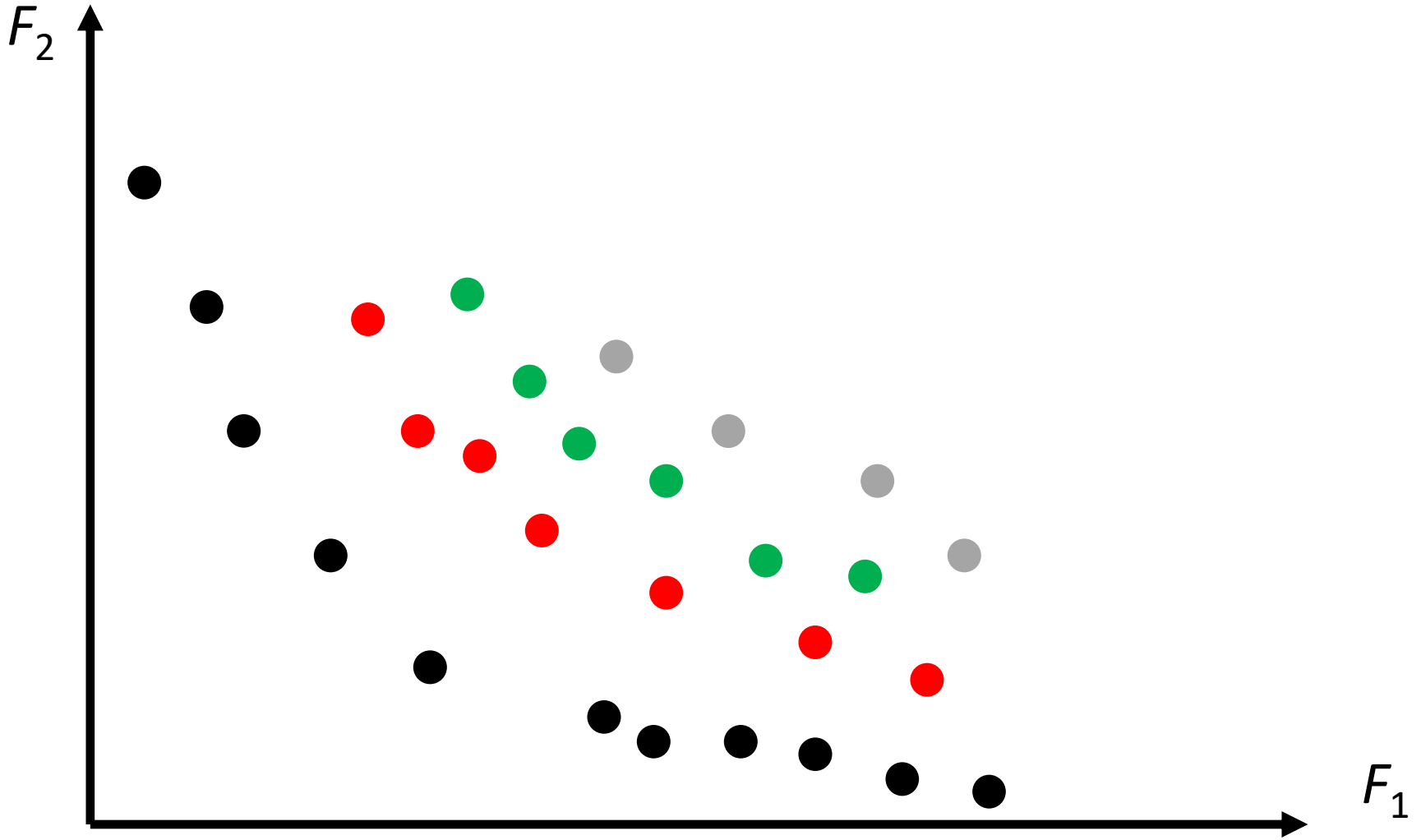
Computing Φ in MOO problems - Front Ranking (2/5)



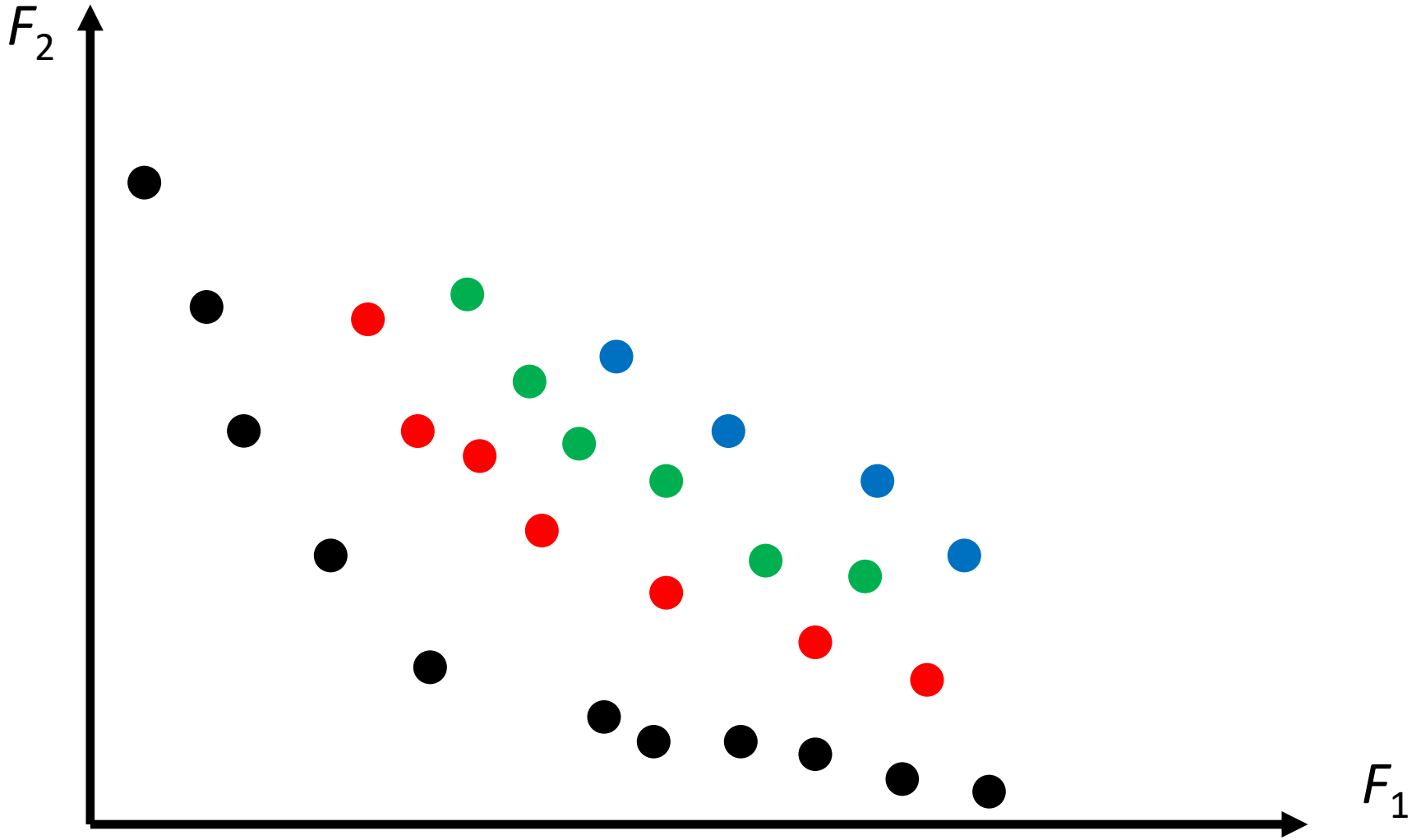
Computing Φ in MOO problems - Front Ranking (3/5)



Computing Φ in MOO problems - Front Ranking (4/5)



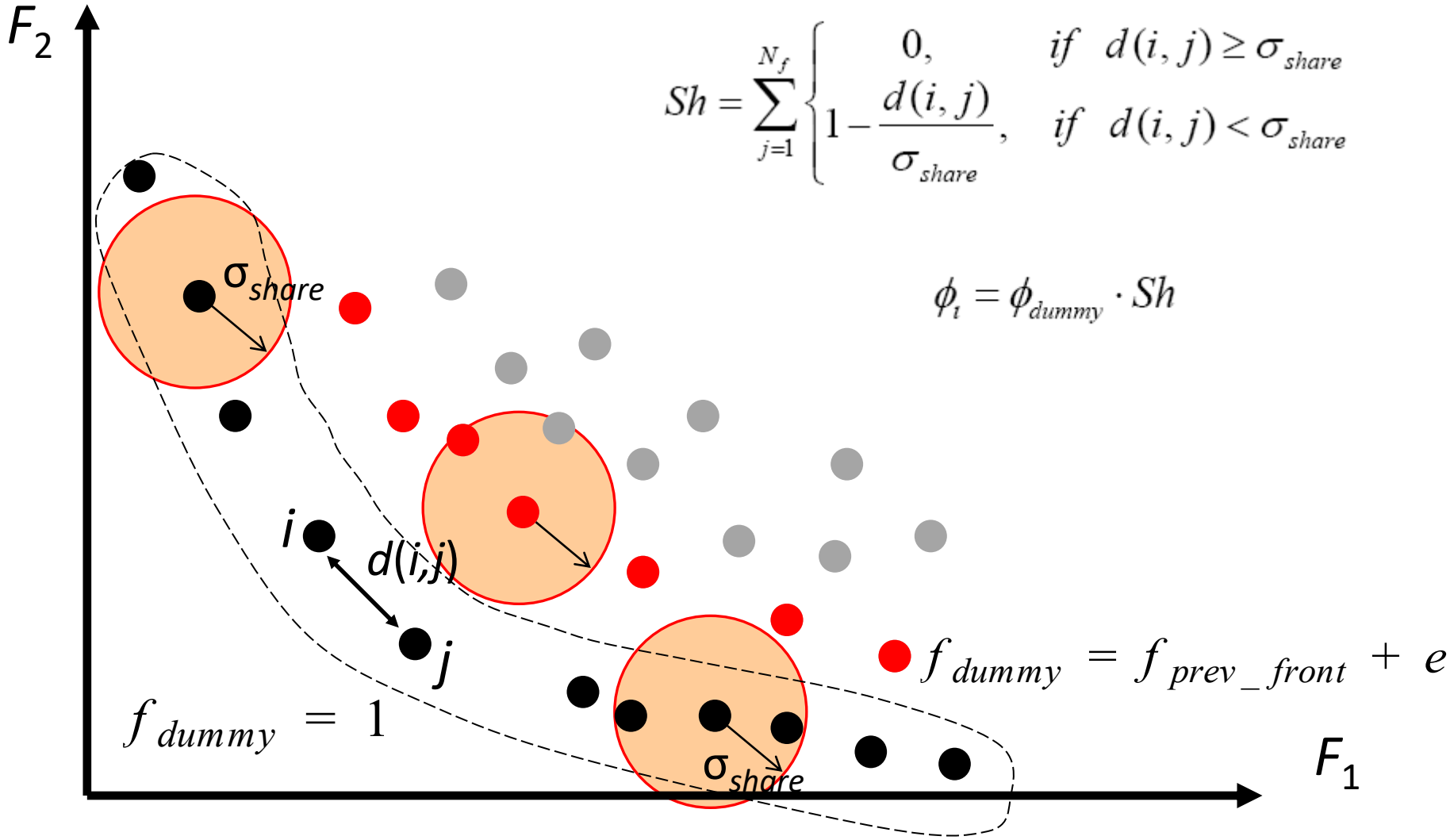
Computing Φ in MOO problems - Front Ranking (5/5)



Computing Φ in MOO problems - NSGA

$$Sh = \sum_{j=1}^{N_f} \begin{cases} 0, & if \quad d(i,j) \geq \sigma_{share} \\ 1 - \dfrac{d(i,j)}{\sigma_{share}}, & if \quad d(i,j) < \sigma_{share} \end{cases}$$

$$\phi_i = \phi_{dummy} \cdot Sh$$



Constraint Handling in EAs

Constraint Handling in EAs – Escalated Penalties

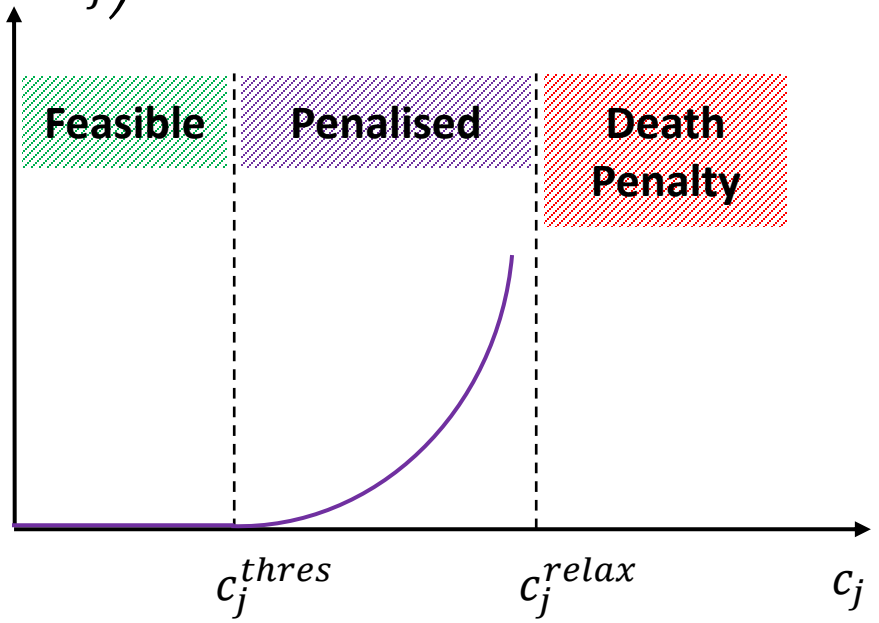
Exponential penalty if

$$c_j^{thres} < c_j(\vec{b}) < c_j^{relax}$$

$$f_m(\vec{b}) \leftarrow f_m(\vec{b}) + \prod_{j=1}^{M_c} \exp\left(a_j \frac{c_j - c_j^{thres}}{c_j^{relax} - c_j}\right) \qquad m \in [1, M_o]$$

Death penalty if

$$c_j(\vec{b}) > c_j^{relax}$$



(* minimisation problem)

In MOO, this technique applies separately to each objective.

Cost Reduction in EAs

PCOpt

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!

EURO Greece

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:

- **When and how to train the metamodel(s)?**
- **Which metamodel ?**
- **How to select training patterns for the metamodel?**
- **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.

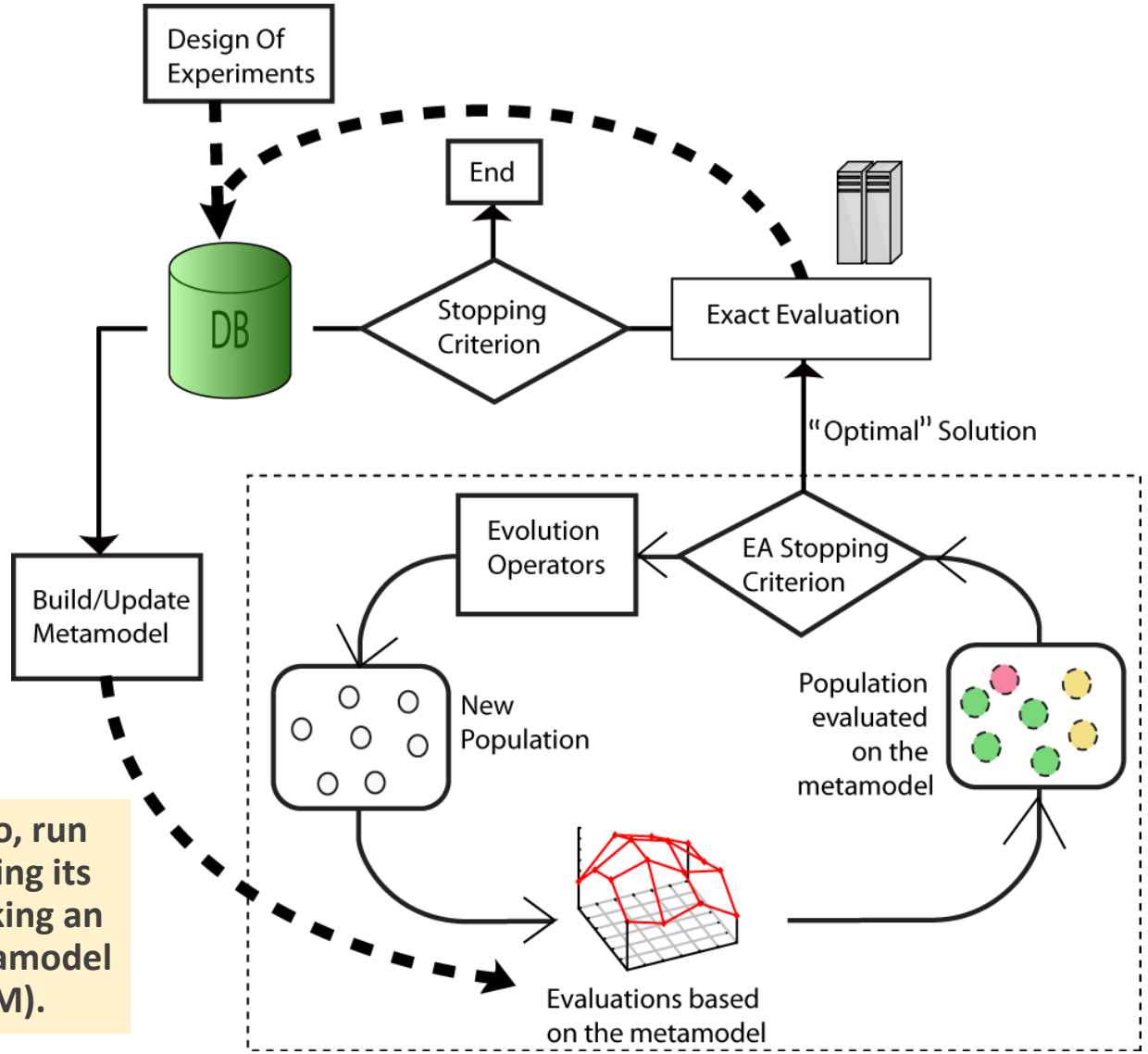
MAEAs with On-Line Trained Metamodels:

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

MAEA with Generational Control:

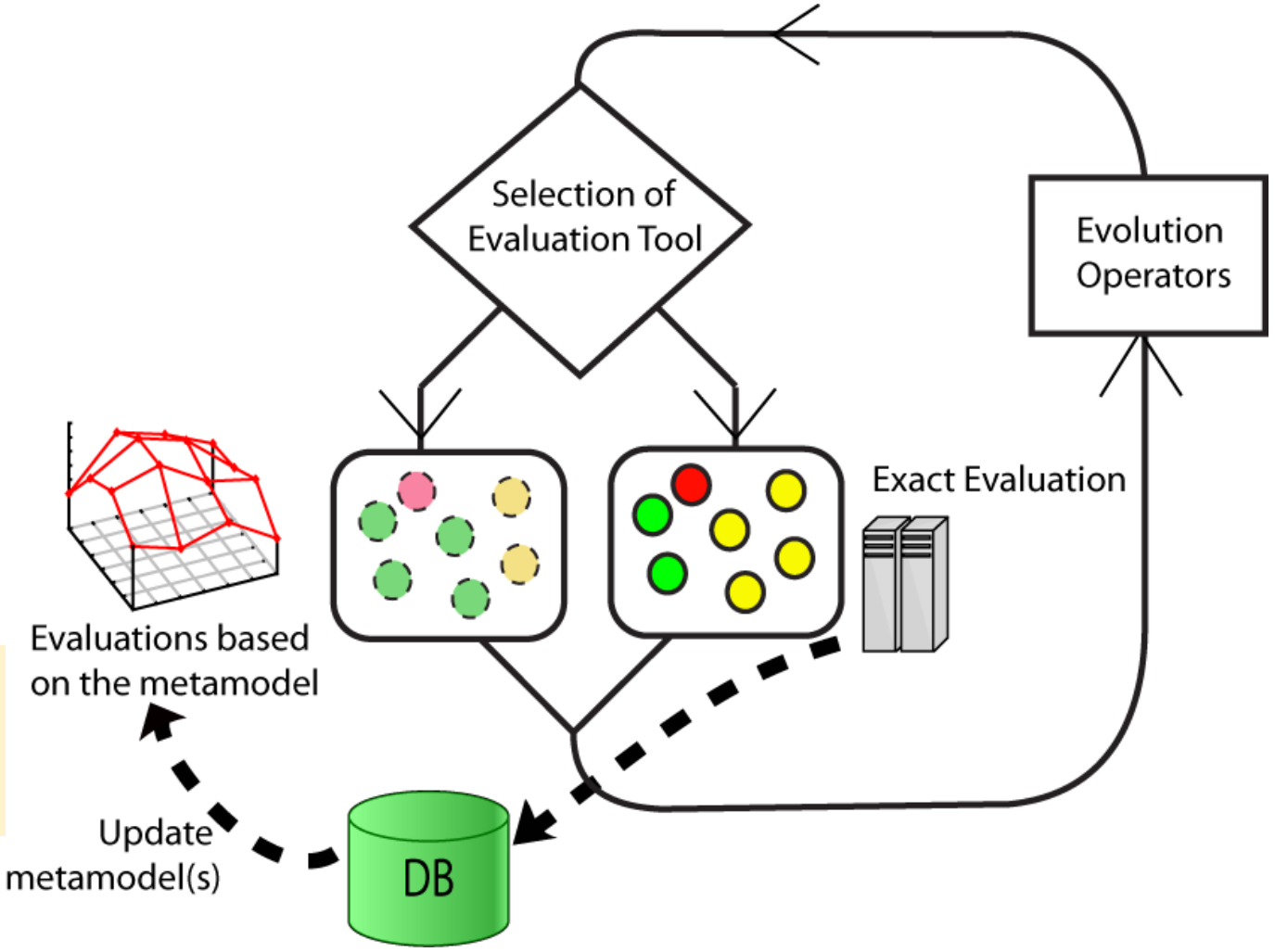
Metamodel (local/global) trained during the evolution; selective use of the metamodel or the PSM in each generation.

A MAEA with Off-Line Trained Metamodels



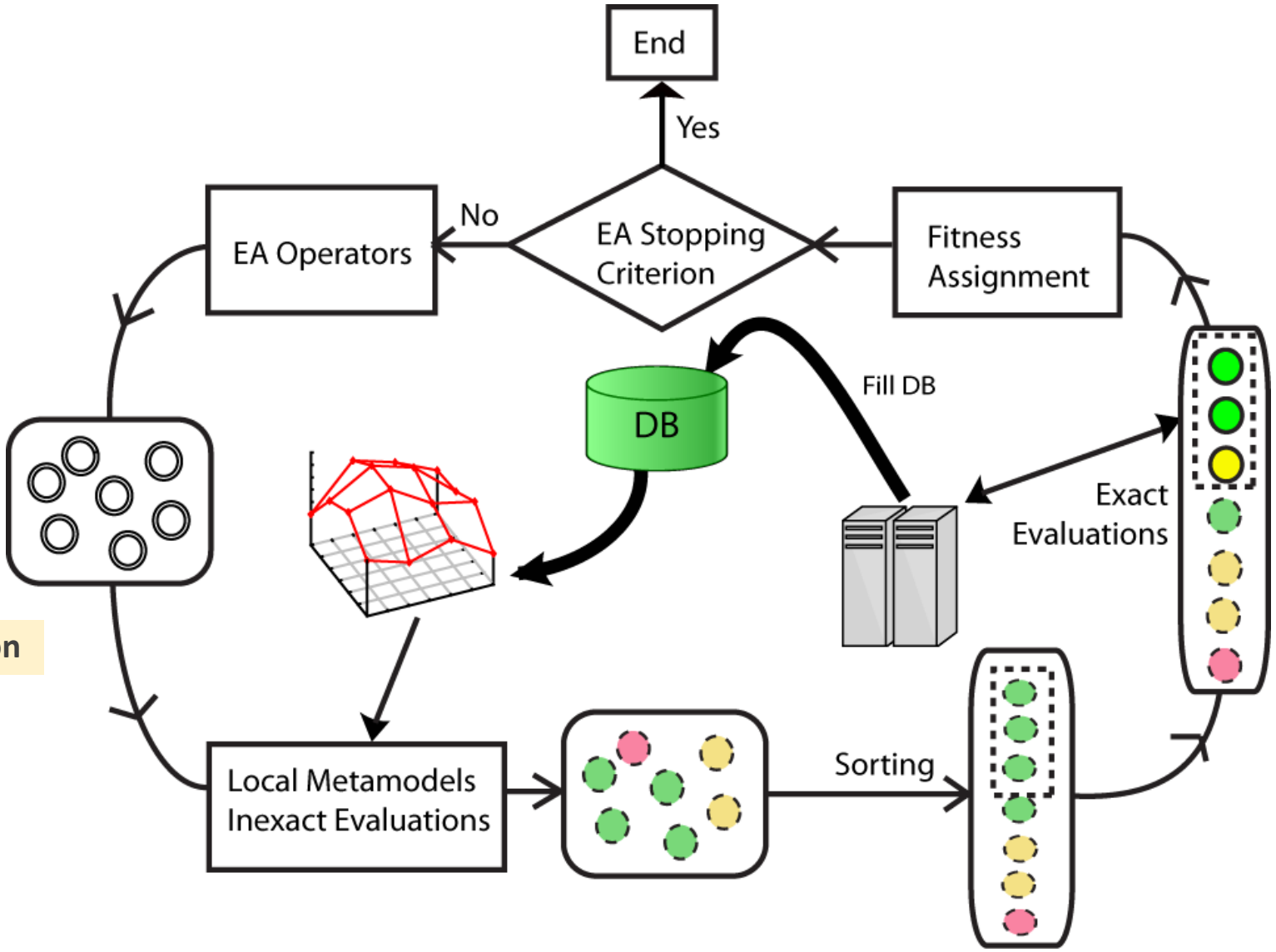
This type of MAEA may, also, run using EASY (without activating its built-in metamodels), by linking an external off-line trained metamodel (replacing calls to the PSM).

A MAEA with Generational Control



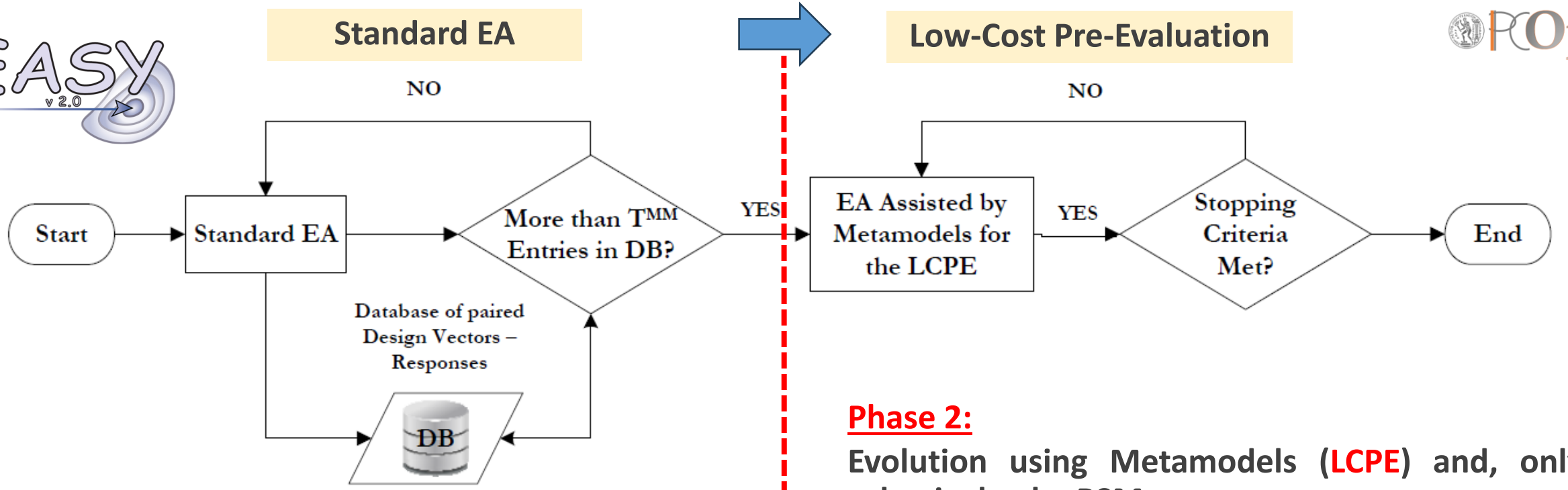
Each generation of offspring is selectively evaluated on the PSM or approximated by the metamodels.

A MAEA with On-Line Trained Metamodels



LCPE: Low-Cost Pre-Evaluation

A MAEA with On-Line Trained Metamodels



Phase 1:

Exclusive use of the PSM; no use of Metamodels

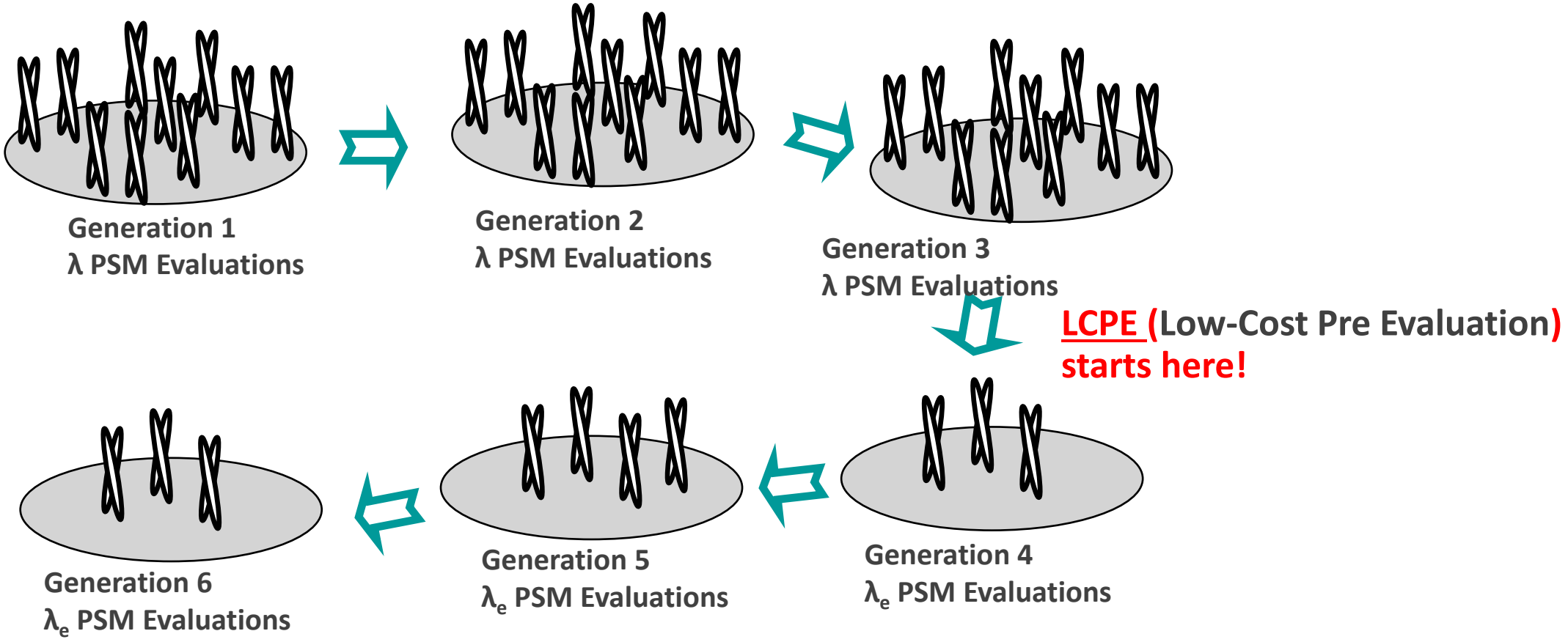
Terminates once T^{MM} evaluated individuals are stored in the DB.

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(«λ, $\lambda_{e,min} \leq \lambda_e \leq \lambda_{e,max}$) 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

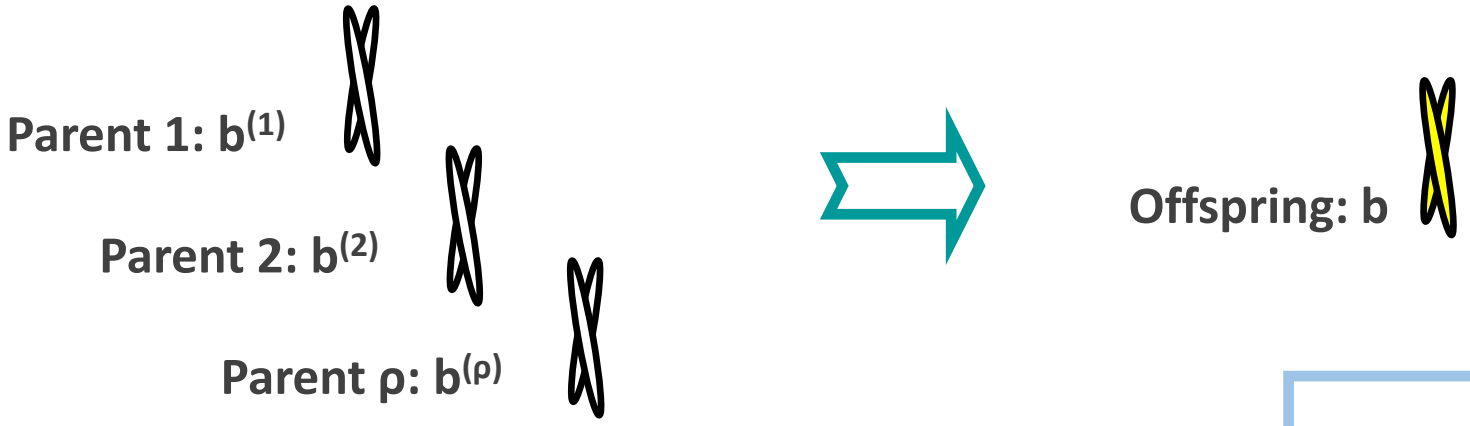


Int. Review Journal Progress in Aerospace Sciences, 38:43-76, 2002.

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.

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

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

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

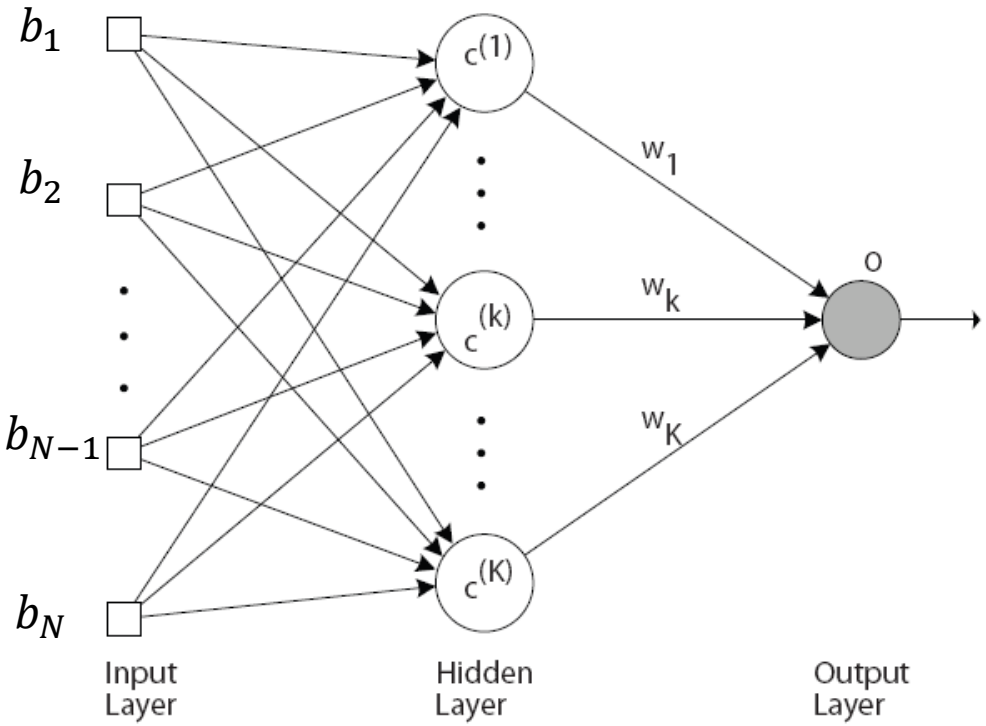
Polynomial–based Regression Surface Methods (RSM)

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

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

Least-squares system: $$\boldsymbol{\alpha} = (\boldsymbol{B}^T \boldsymbol{B})^{-1} \boldsymbol{B}^T \boldsymbol{o}$$

Radial Basis Function Networks (RBFNs) (1/2)



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

Haykin, Neural Networks: A Comprehensive Foundation, 1999.

Radial Basis Function Networks (RBFNs) (2/2)

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

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

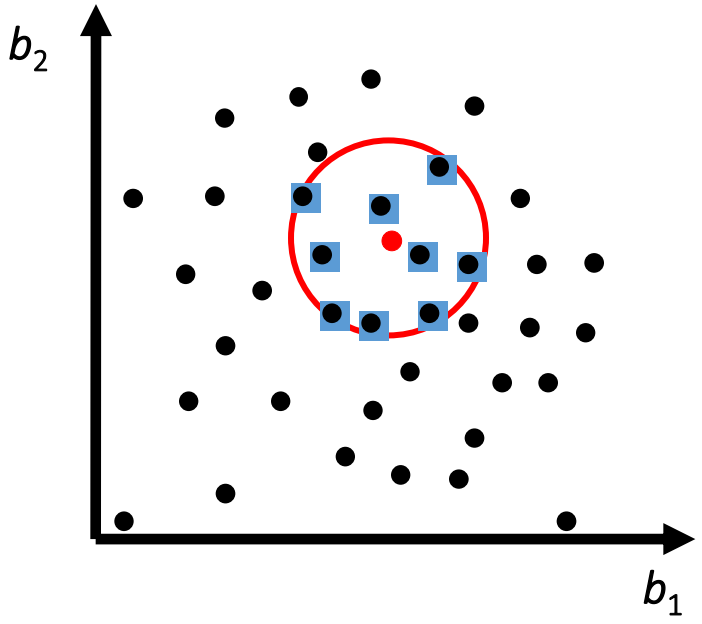
$$\begin{bmatrix} G(\left\|\vec{b}^{(1)} - \vec{c}^{(1)}\right\|_2) & \dots & G(\left\|\vec{b}^{(1)} - \vec{c}^{(T)}\right\|_2) \\ \vdots & \ddots & \vdots \\ G(\left\|\vec{b}^{(T)} - \vec{c}^{(1)}\right\|_2) & \dots & G(\left\|\vec{b}^{(T)} - \vec{c}^{(T)}\right\|_2) \end{bmatrix} \begin{bmatrix} \omega_1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

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

Training Patterns Selection

For MAEAs using on-line trained metamodels.

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

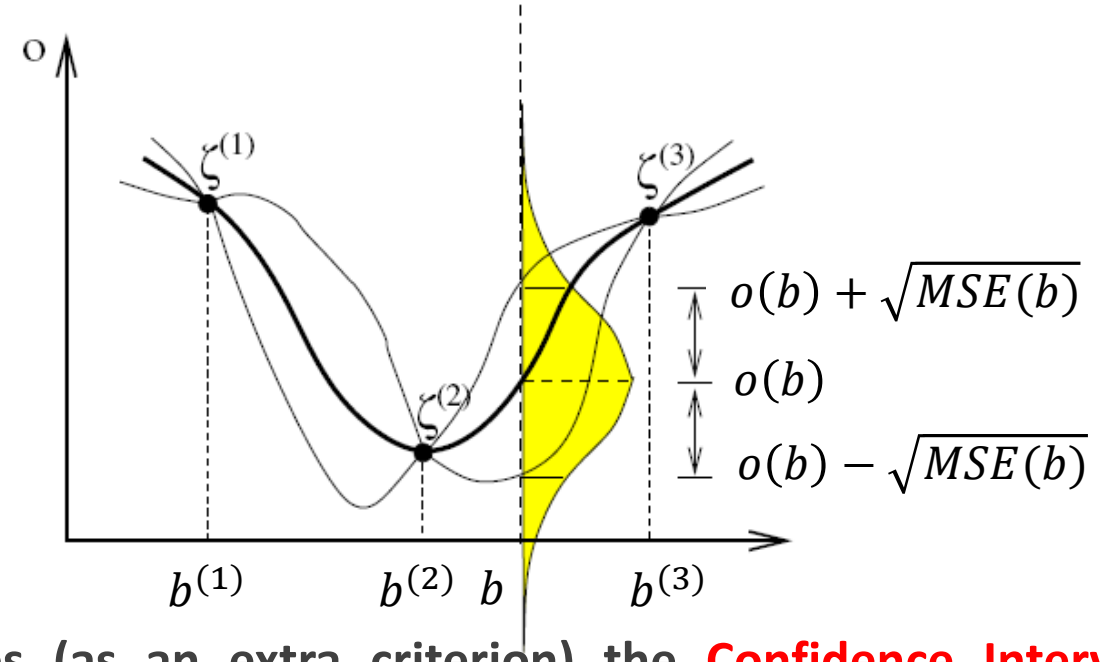


Attention should be paid to:

- **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.**

Kriging

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.

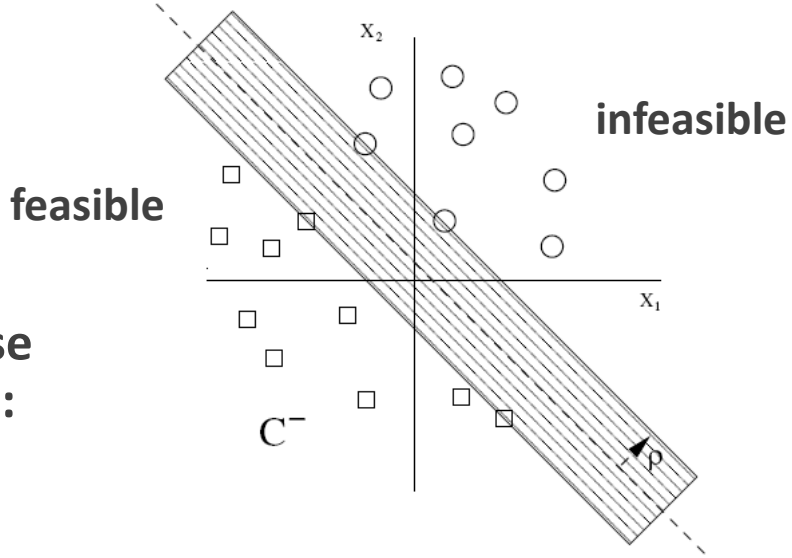


Kriging estimates and uses (as an extra criterion) the Confidence Interval, pertinent to the predicted value of the objective or constraint function.

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

Constrained Optimisation in MAEA – Support Vector Machines (SVM)

Infeasible population members, penalised with an (+/-) infinite F value, should not be used as training patterns for the metamodel, but, on the other hand, the MAEA should take them into account. A special treatment is needed.



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:

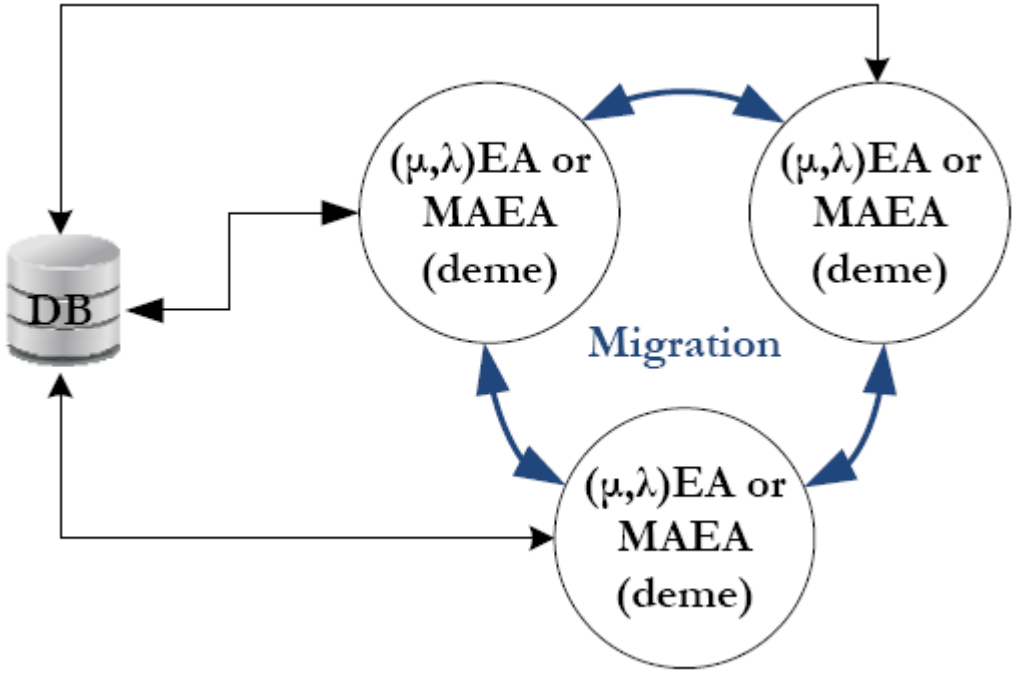
- **the RBFN is used only for individuals marked as "feasible".**
- **Individuals marked as "infeasible" are penalised.**

Distributed EAs and MAEAs

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!**

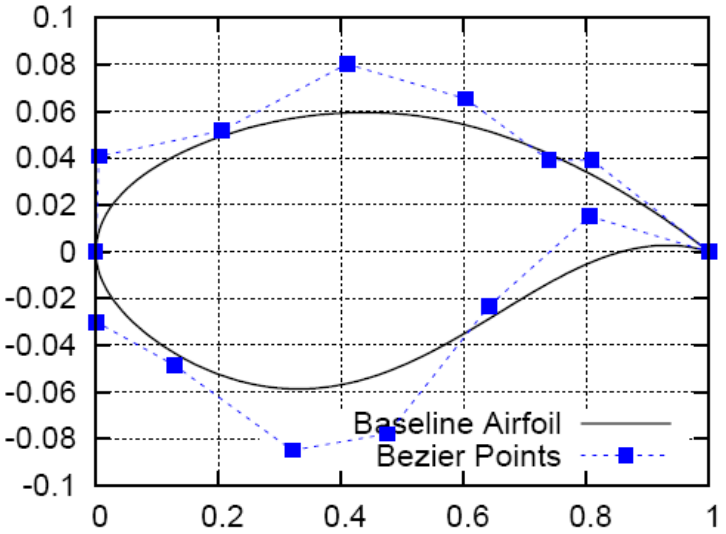


Common DB for all demes.

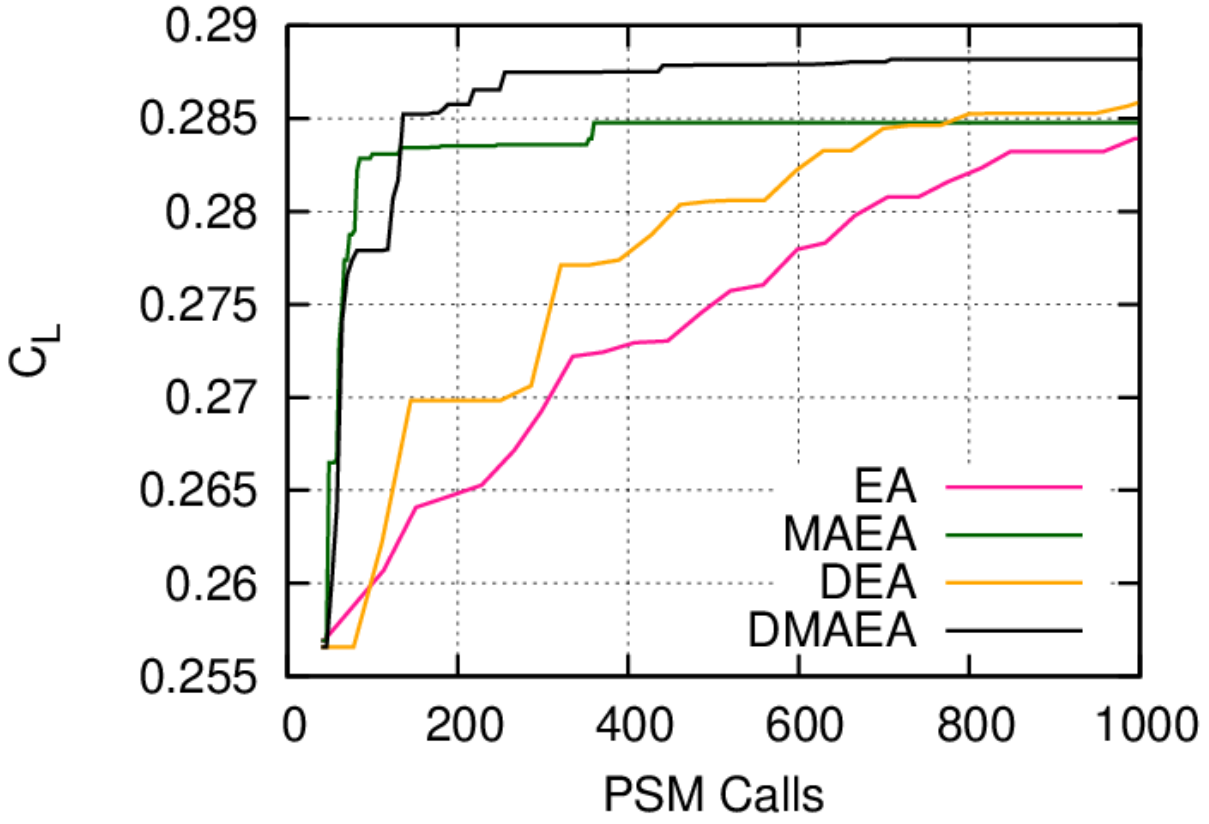
International Journal for Numerical Methods in Fluids, 53:455-469, 2007.

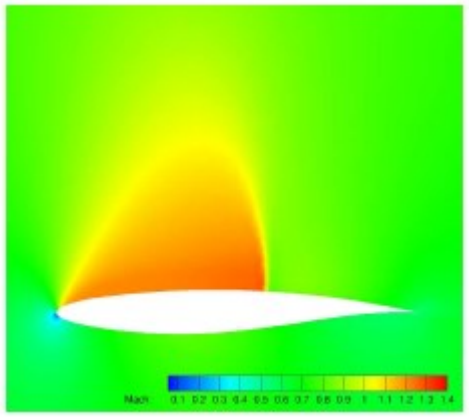
Demo Case A (SOO)

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.

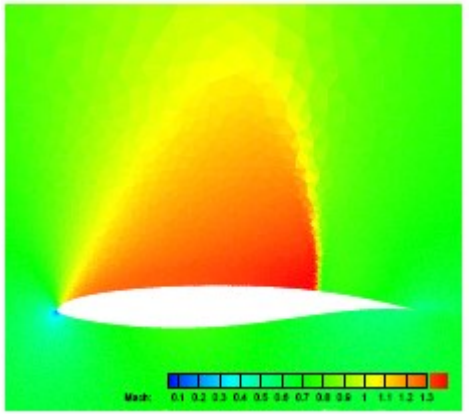


Demo Case A (SOO)





Starting



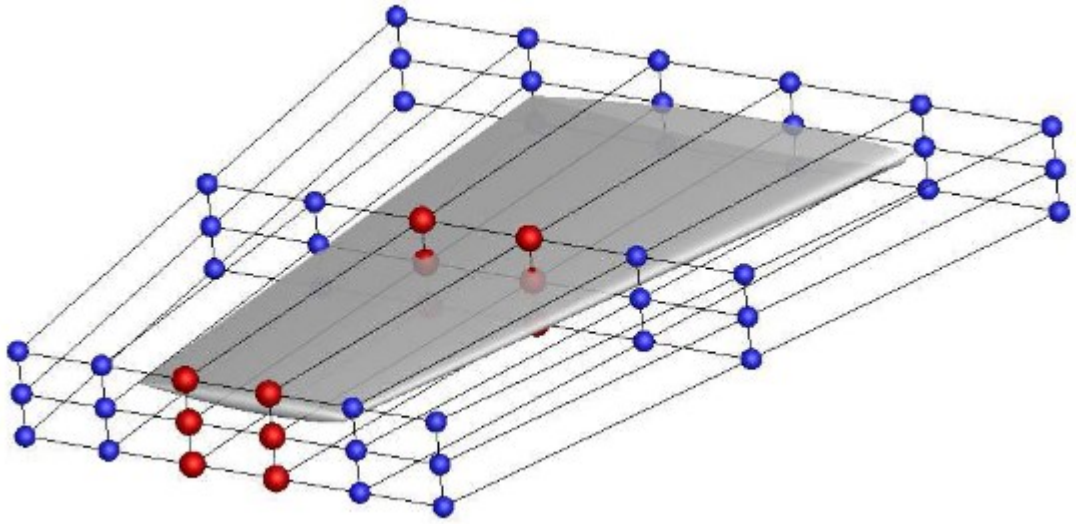
Optimal

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.

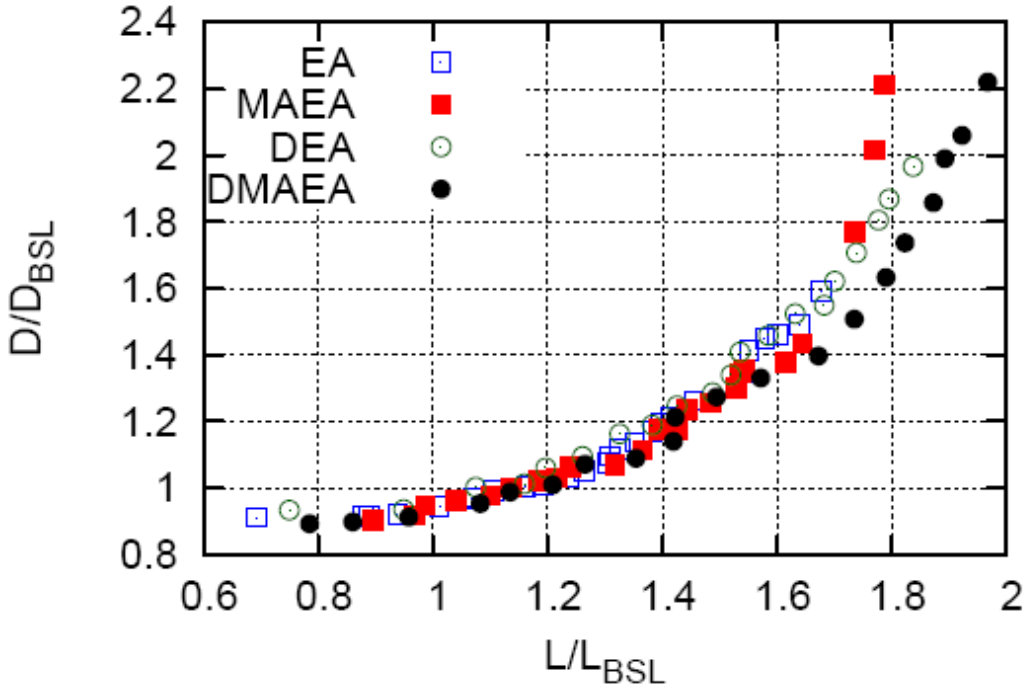
(20,40) EA
(20,40) MAEA, T^{MM}=40, λ_e=3
DEA or DMAEA: two demes (10,20)

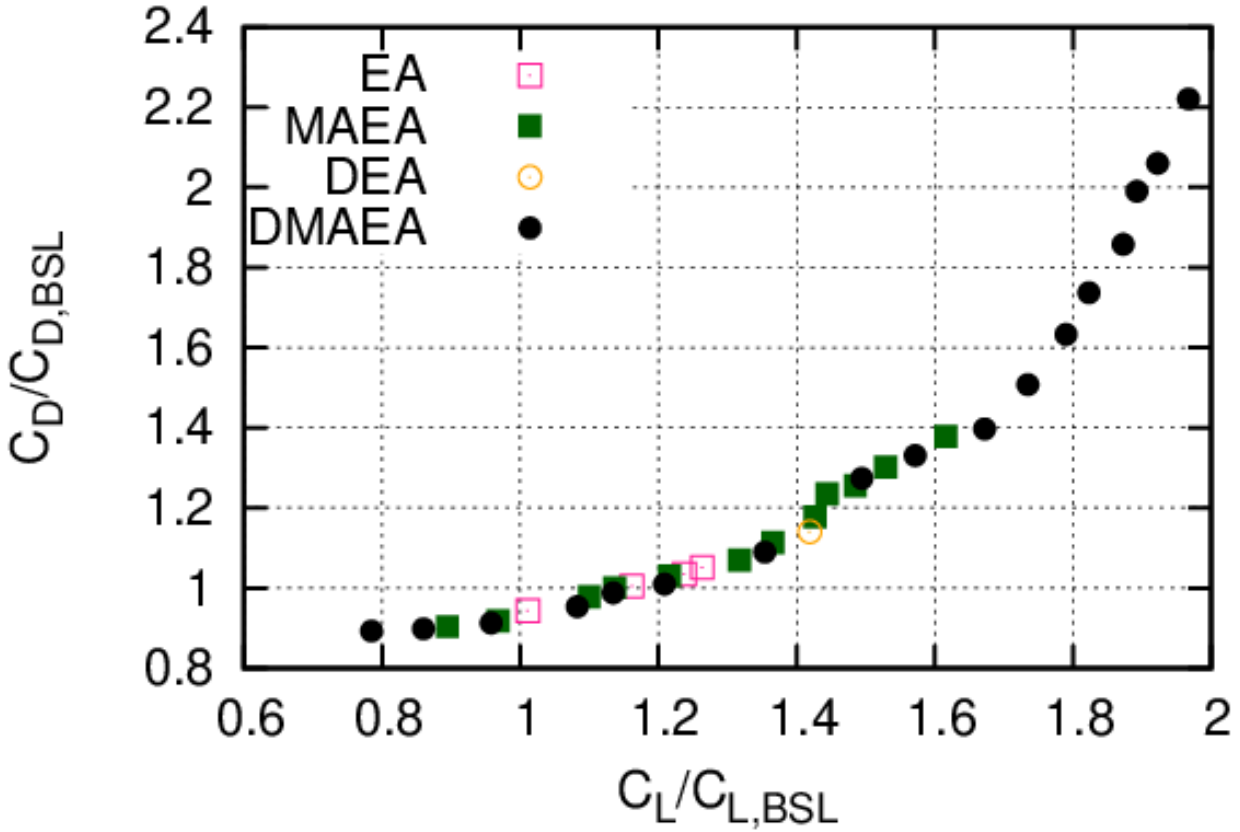
Demo Case B (MOO)

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^o, $a_{yaw,inf}$=0^o.
Parametrisation: 6x3x3 Volumetric NURBS Control Grid.
N=24 DoFs (internal control points moving normal to the planform).
Basis of Comparison: a (10,20)EA.



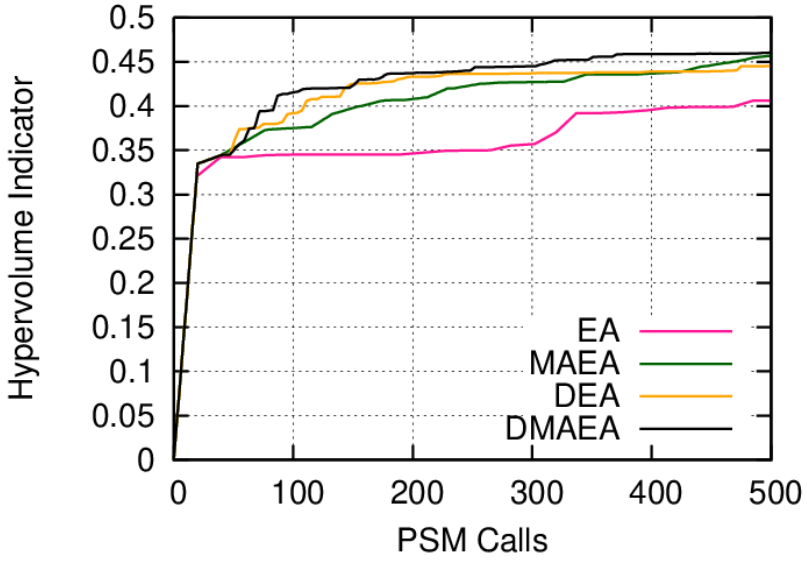
Demo Case B (MOO)



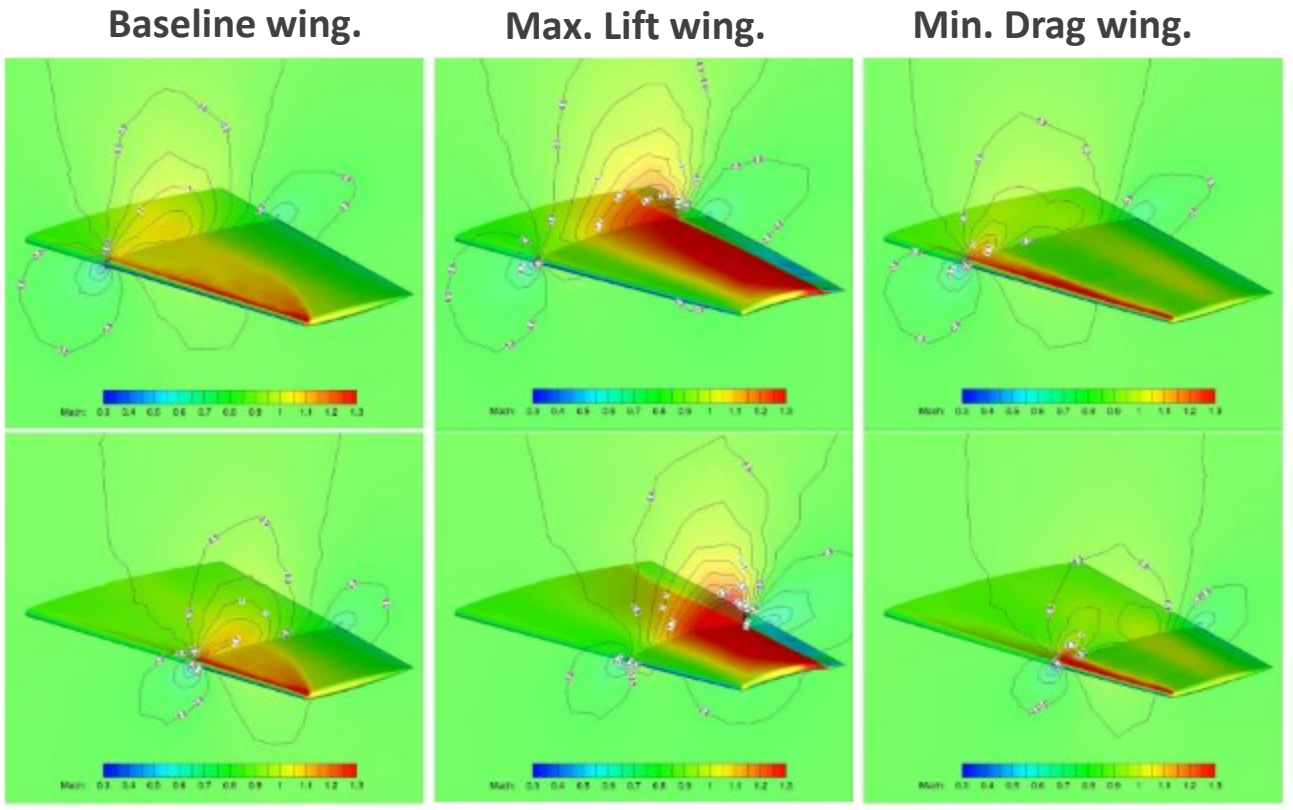


MOO problems may also benefit from the use of metamodels and distributed search.

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.

(MA)EAs & the Curse of Dimensionality

Separable Problems:

Function $G(x_1, x_2, \ldots, x_N)$ is separable if it can be written in the form:

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

Then, each $g_k(x_k)$ can be minimised in isolation.

Separable:

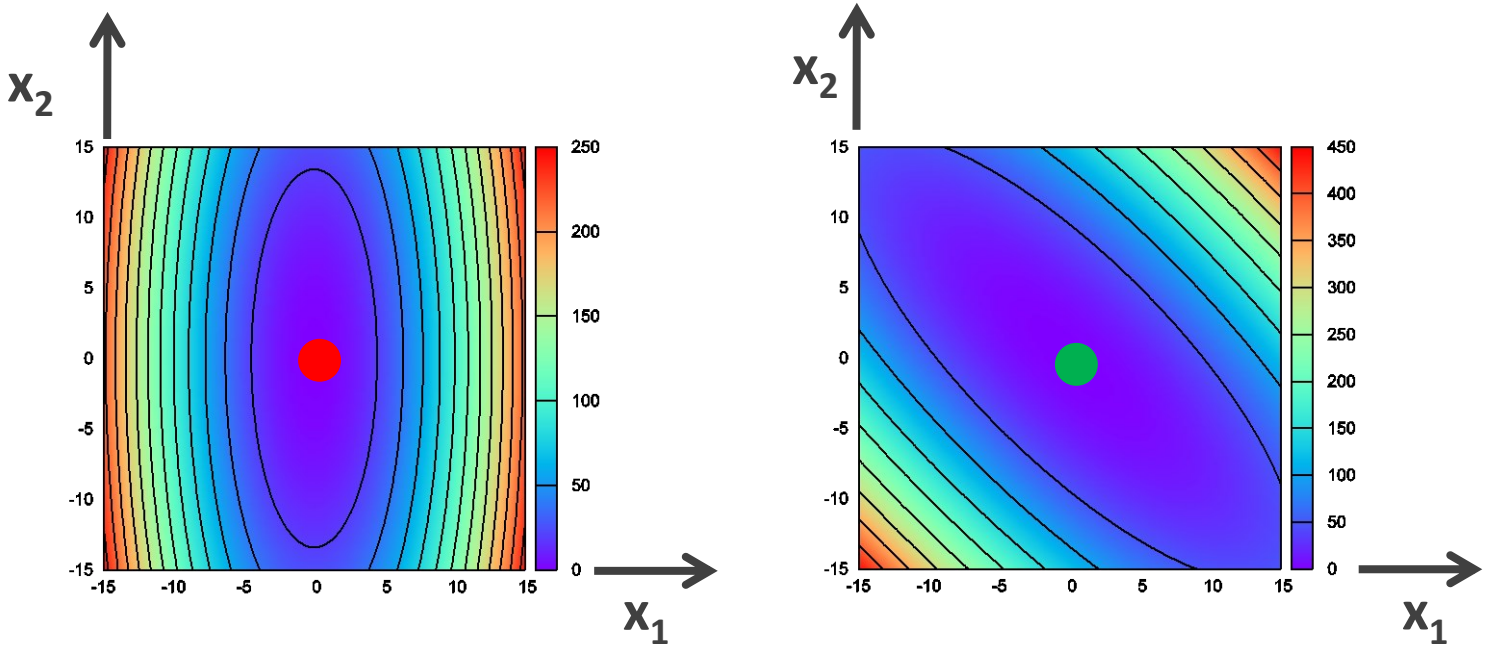
$$F(\vec{x}) = x_1^2 + \frac{1}{9}x_2^2$$

vs. non-separable function:

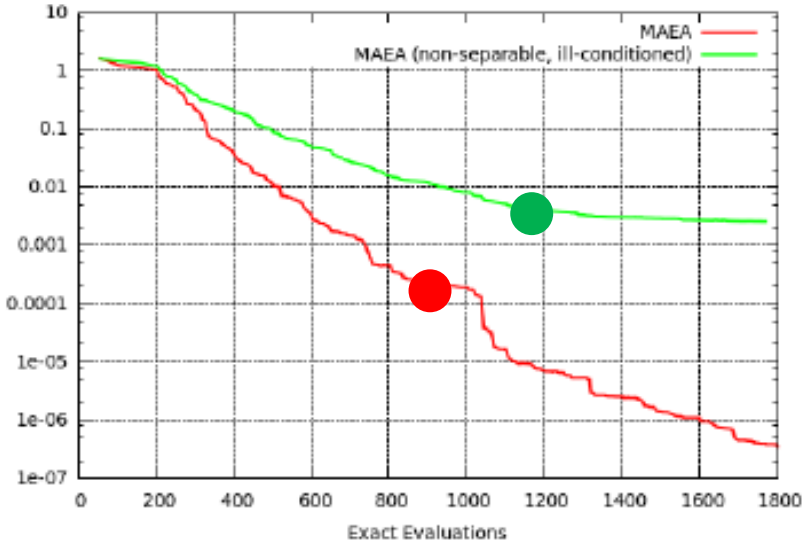
$$F(\vec{x}) = \left[x_1 cos(-\frac{\pi}{4}) - x_2 sin(-\frac{\pi}{4})\right]^2 + \frac{1}{9}\left[x_1 sin(-\frac{\pi}{4}) + x_2 cos(-\frac{\pi}{4})\right]^2$$

Optimisation in Separable Problems

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



Example with 10 DoFs

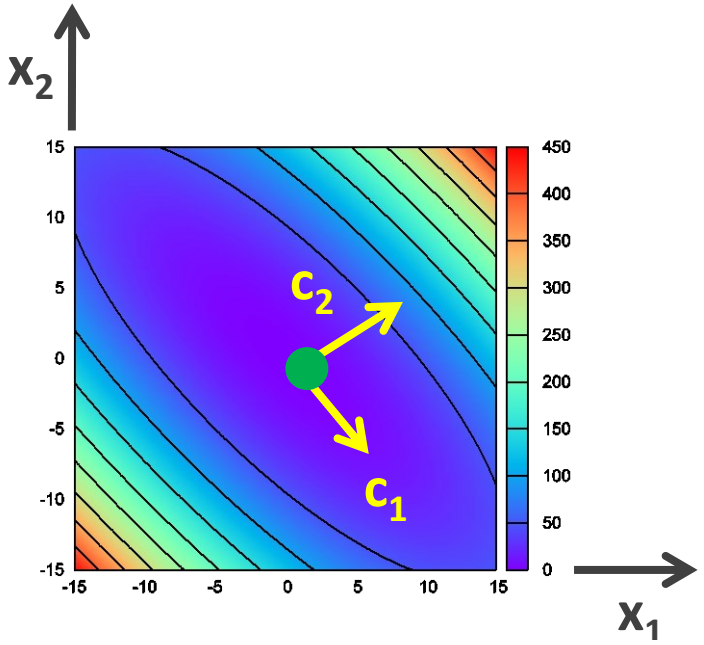


● $F(\vec{x}) = x_1^2 + \frac{1}{9}x_2^2$

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

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:

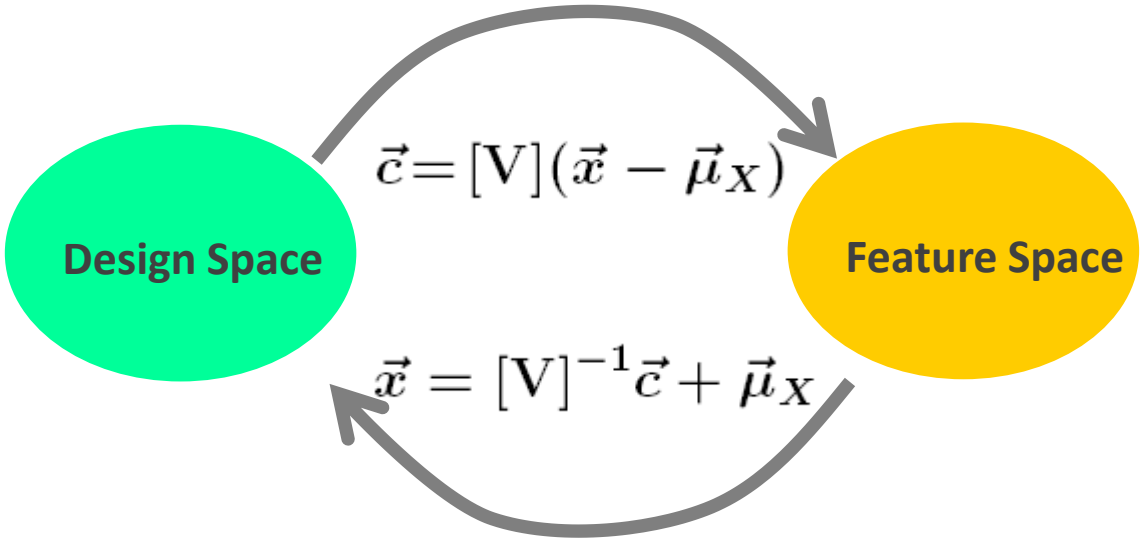
$$E[[\mathrm{X}]] = 0 \qquad E[[\mathrm{X}]^2] = 1$$

- Compute the covariance matrix:

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

- Eigen-Decomposition:

$$[\mathrm{P}] = [\mathrm{V}][\Lambda][\mathrm{V}]^T$$

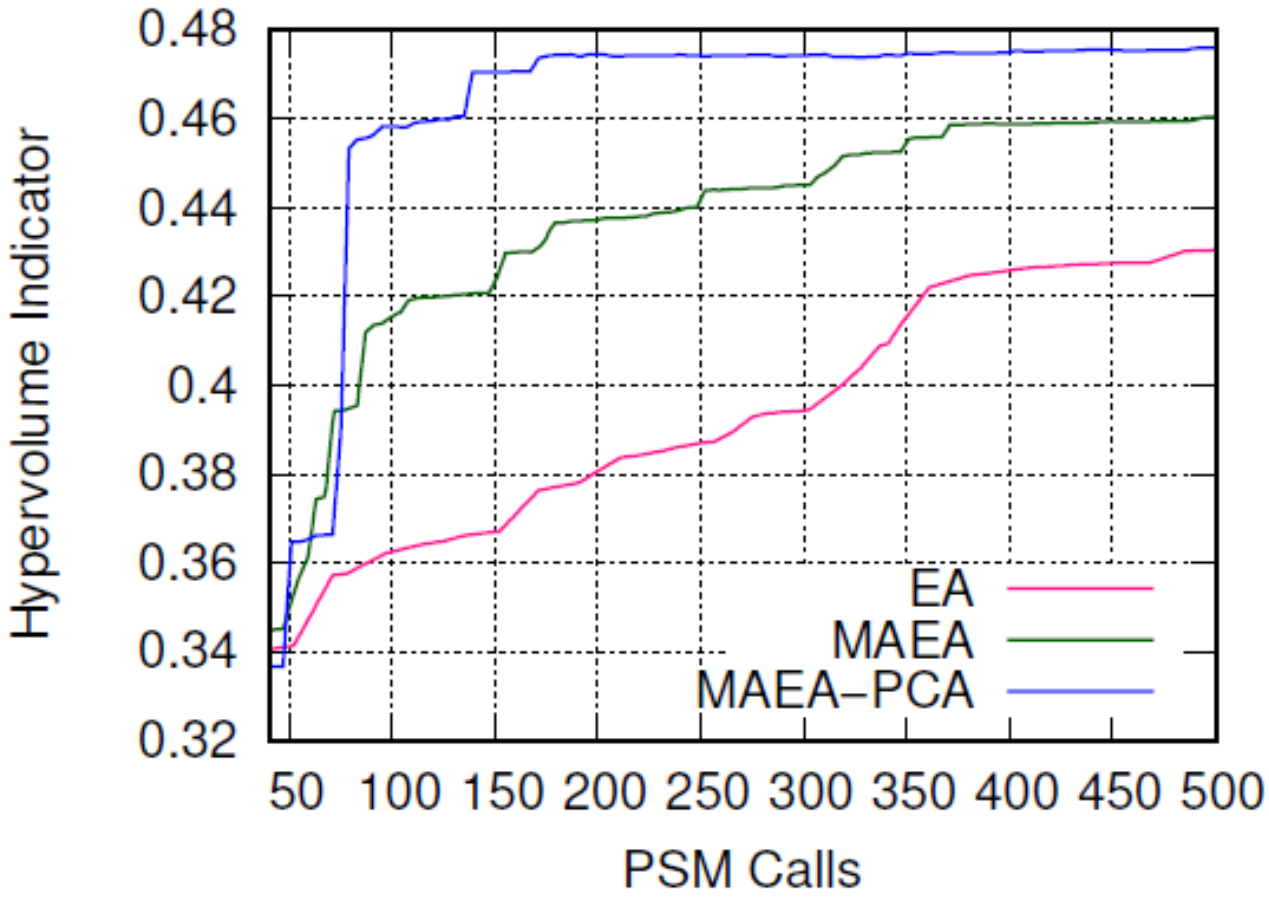


where the eigenvectors are the principal components defining the feature space and the eigenvalues are their variances.

Engineering Optimization, 46:895-911, 2014.

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

ShpO of an isolated wing
Target: max. Lift (L) and min. Drag (D)
New 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.

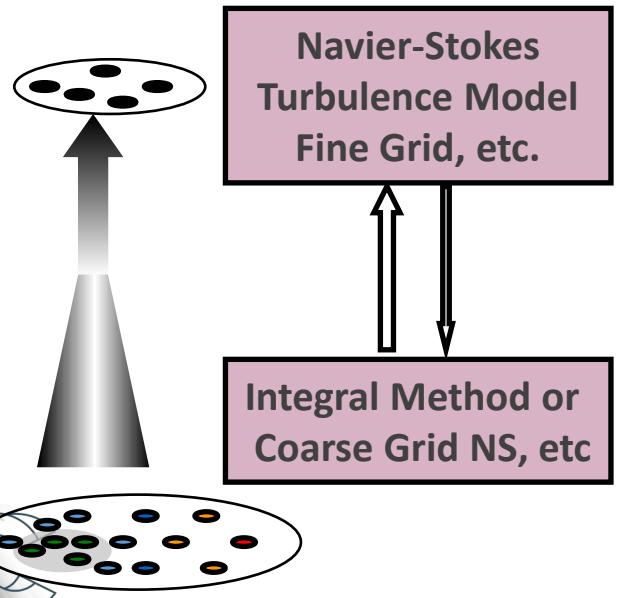
Hierarchical/Multilevel Schemes

Hierarchical/Multilevel Schemes

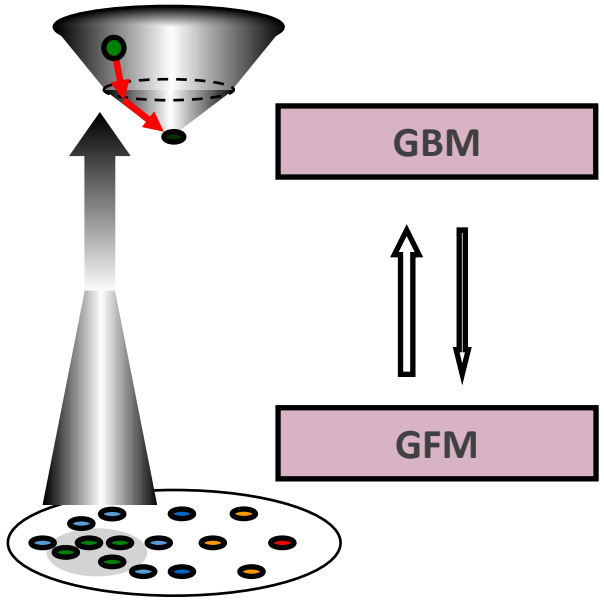
Splitting the optimisation at different (usually two!) levels, allows the combined use of evaluation models of different accuracy and computational cost, different search methods (EAs, MAEAs, DMAEAs, etc. or gradient-based ones) and/or different sets/subsets of design variables.

Exploitation vs. Exploration-based levels.

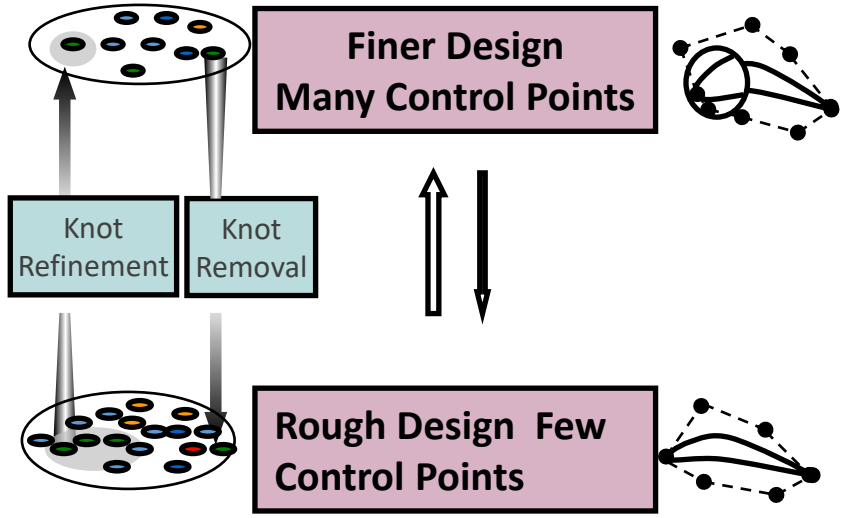
Hierarchical Evaluation



Hierarchical Search



Hierarchical Parametrisation



Dr. Varvara G. Asouti, vasouti@mail.ntua.gr

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.

Industrial Applications

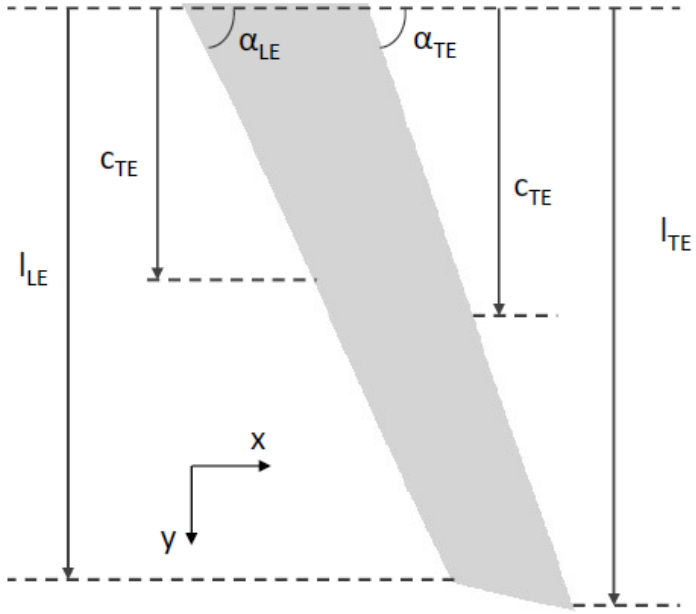
Optimisation of an Aircraft Wing-Body Configuration (1/3)

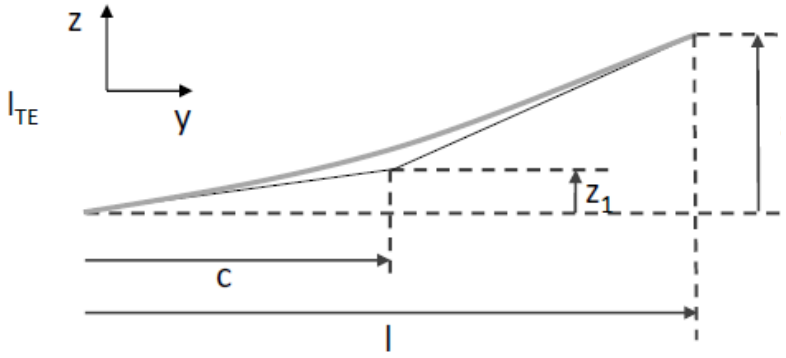
Re-design of an aircraft wing-body configuration, by changing only the wing shape, for max. C_L & min. C_D.

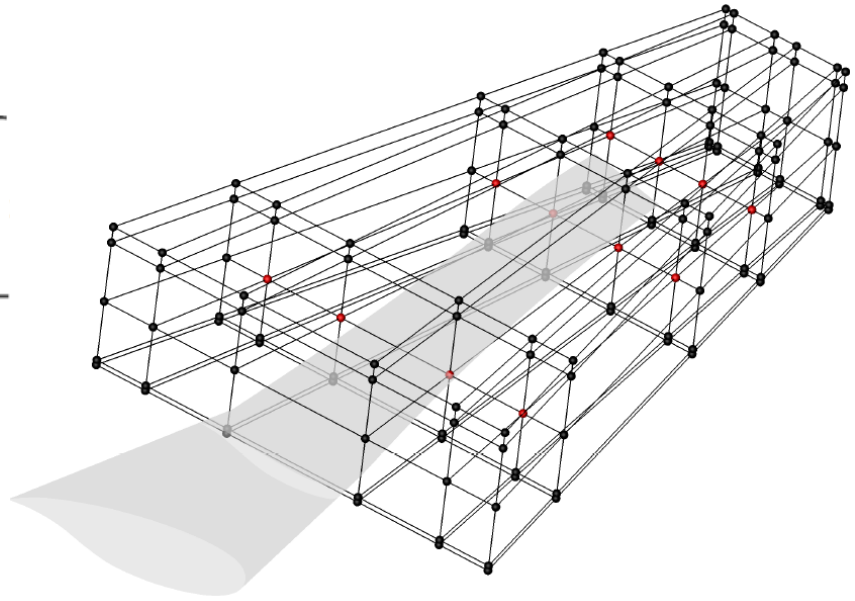
Flow conditions: $Re_c = 10^6$, $M_\infty = 0.75$, $a_\infty = 0^o$.

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

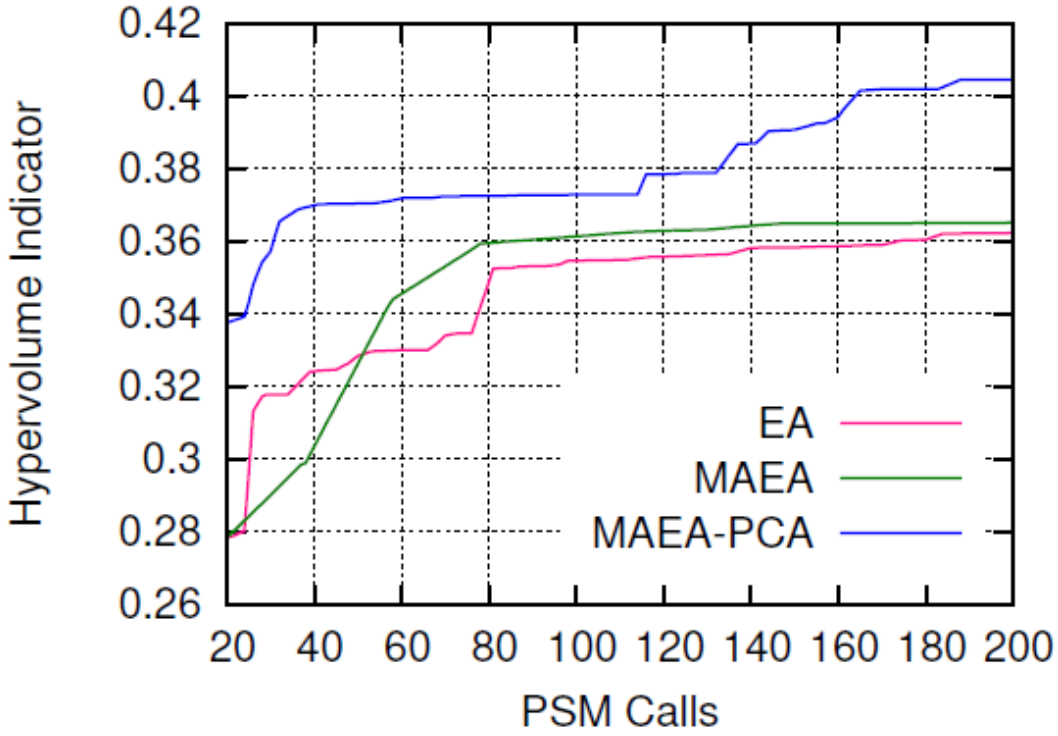
Customised parametrisation (sweep angles, LE/TE curvatures, wing bending & twist) with 8 DoFs, affecting the coordinates of the 3×5×4 NURBS morphing box.

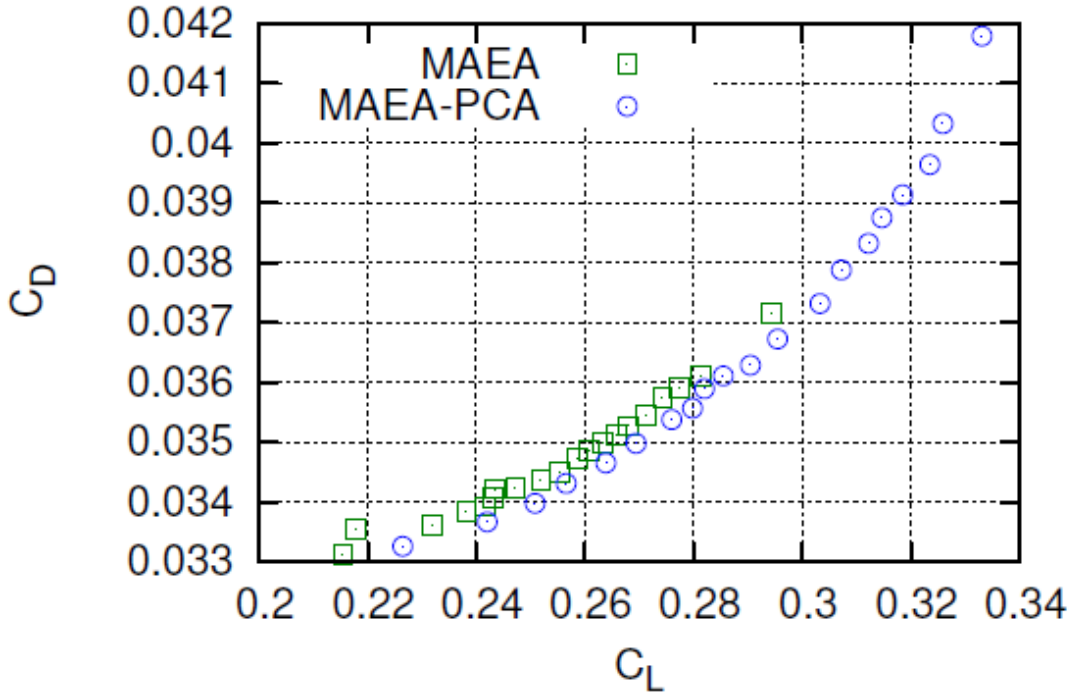






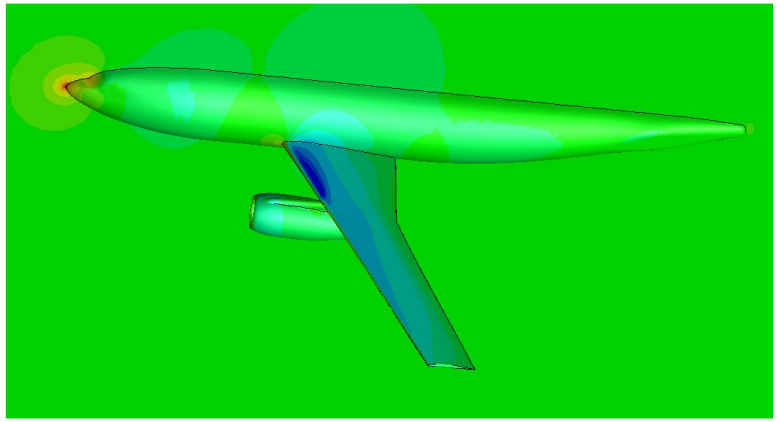
Optimisation of an Aircraft Wing-Body Configuration (2/3)



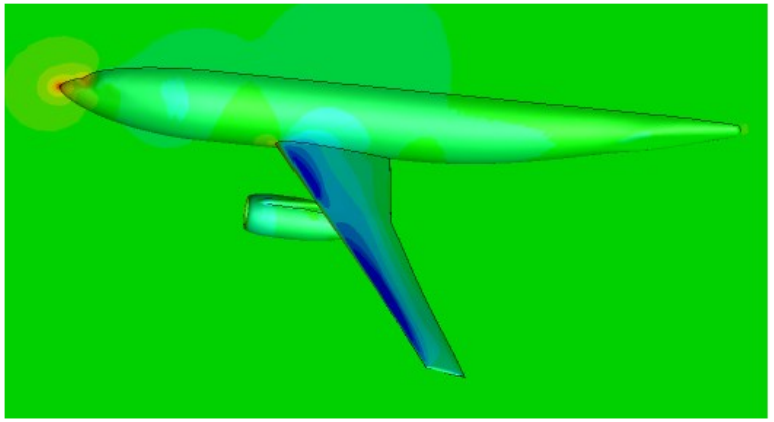


Comparison of (5, 10)EA, MAEA and MAEA-PCA. The LCPE phase starts after T^{MM} = 20 calls to the PSM; λ_e =4. Stopping criterion = 200 calls to the PSM.

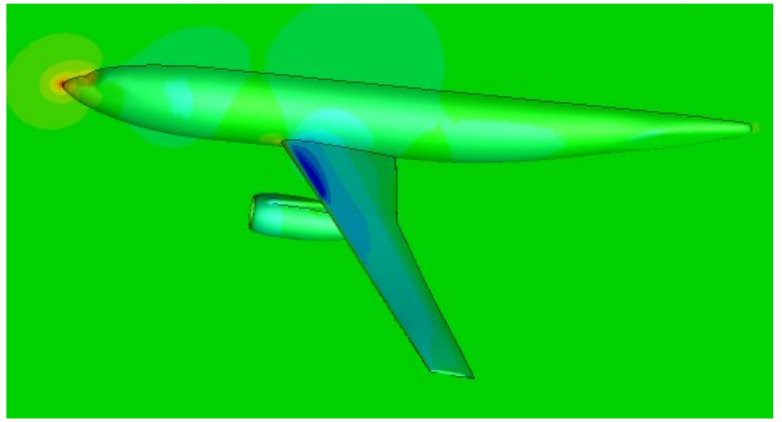
Optimisation of an Aircraft Wing-Body Configuration



Reference shape

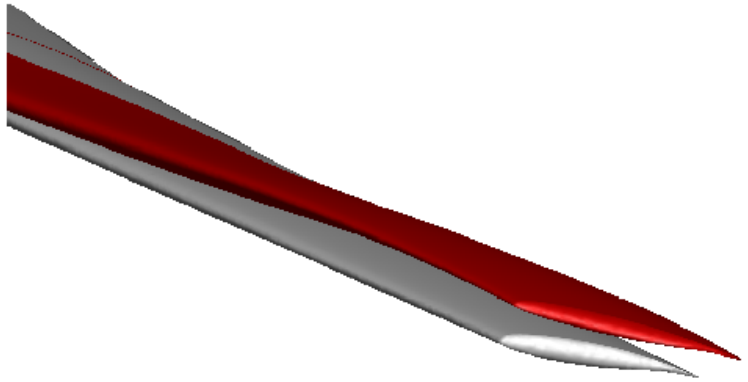


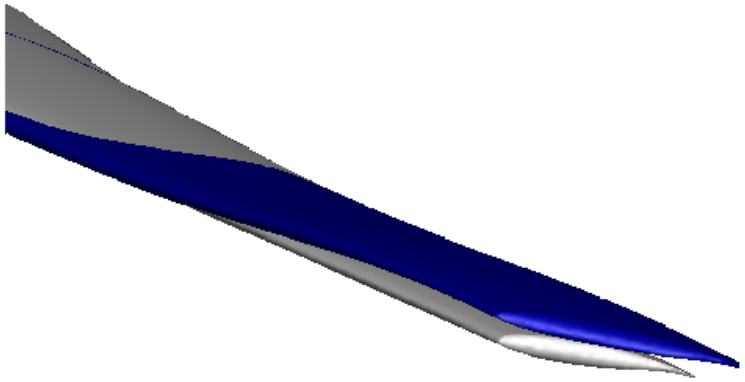
Optimised for max. C_L



Optimised for min. C_D

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





Dr. Varvara G. Asouti, vasouti@mail.ntua.gr

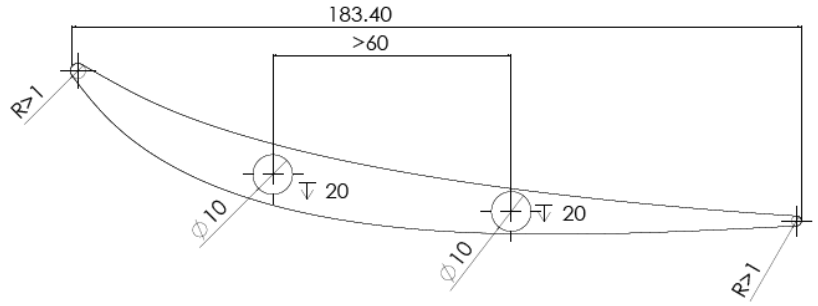
Shape Optimisation of of theTU Berlin Turbolab Stator (1/2)

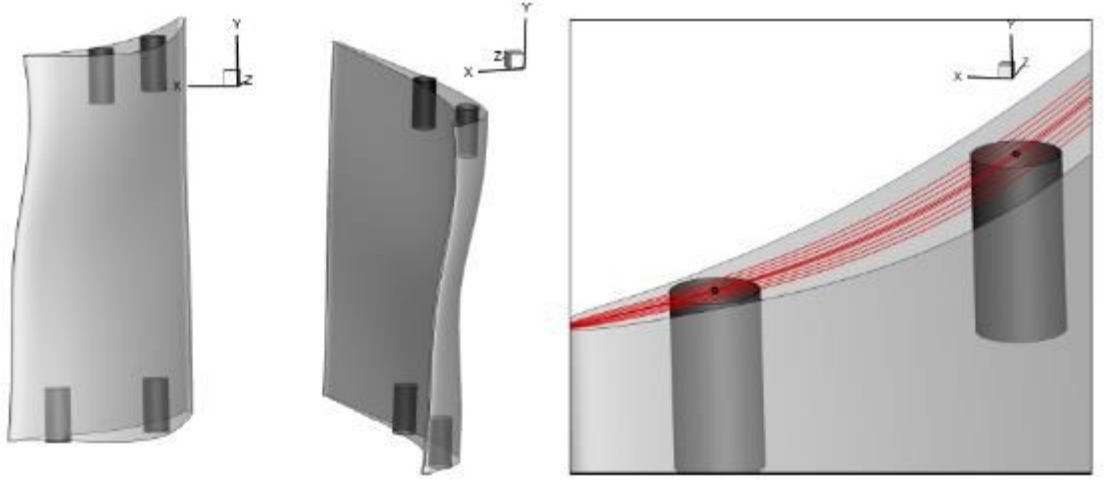
ShpO of the TU Berlin Turbolab stator with two objectives: (a) min. Δp_t (total pressure losses) and (b) min. $\Delta\alpha$ (deviation of the exit angle from the axial direction)

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

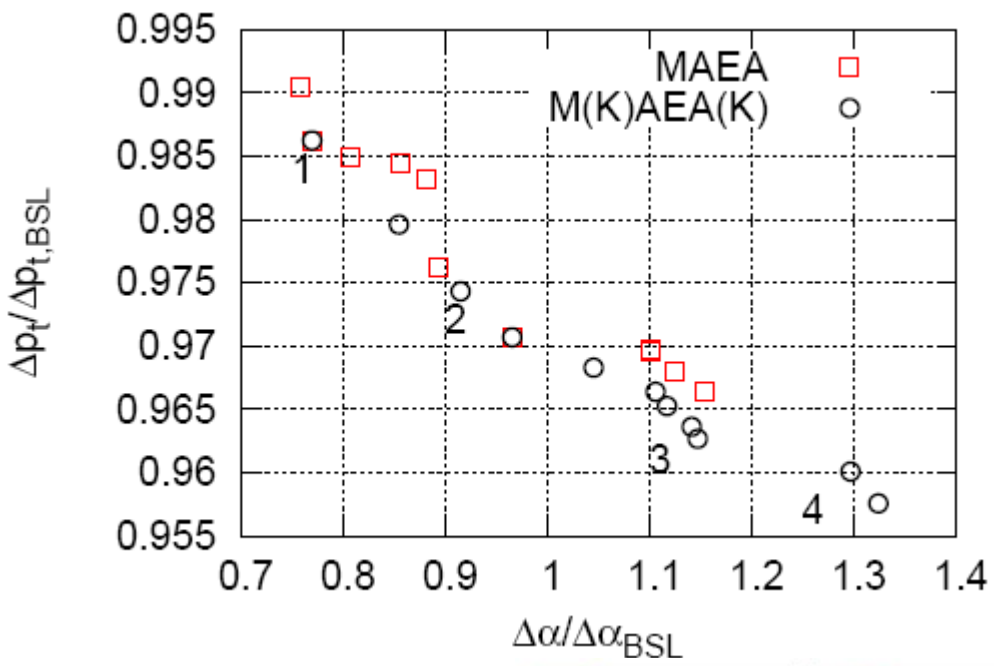
Constraints:

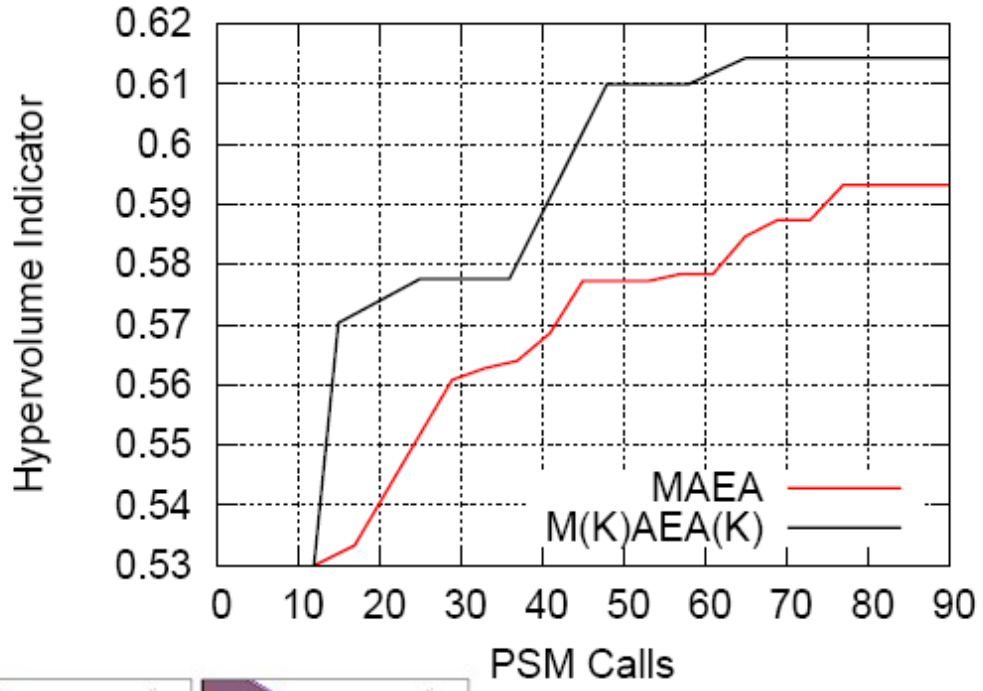
- **Ensure blade mounting in the casing (hub and shroud).**
- **Blade thickness must accommodate cylinders of material of 5mm radius and 20mm depth.**
- **Min. distance of 60 mm between fixture holes.**
- **The first cylinder is fitted by marching along the camber line.**
- **Fitting of the second cylinder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

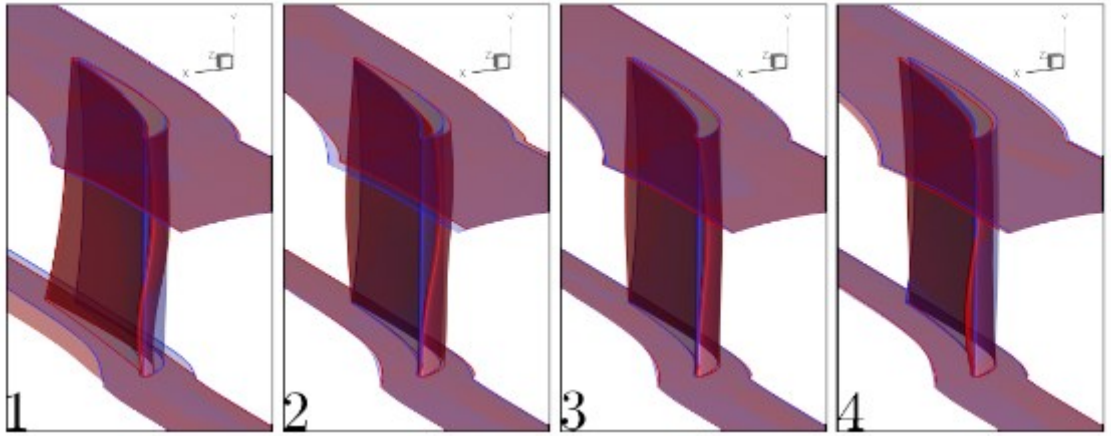




Shape Optimisation of of theTU Berlin Turbolab Stator (2/2)

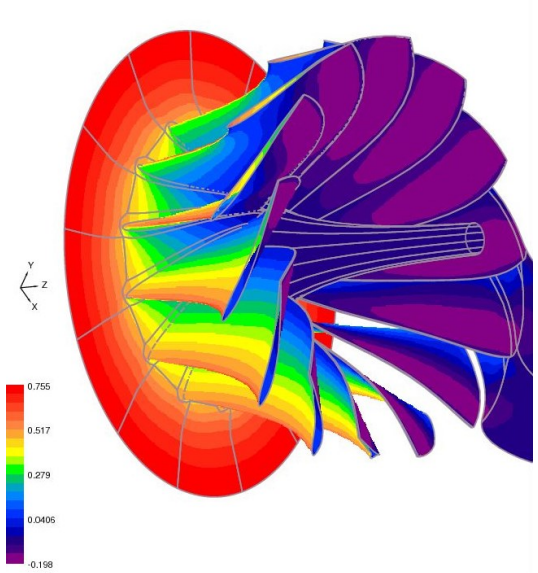


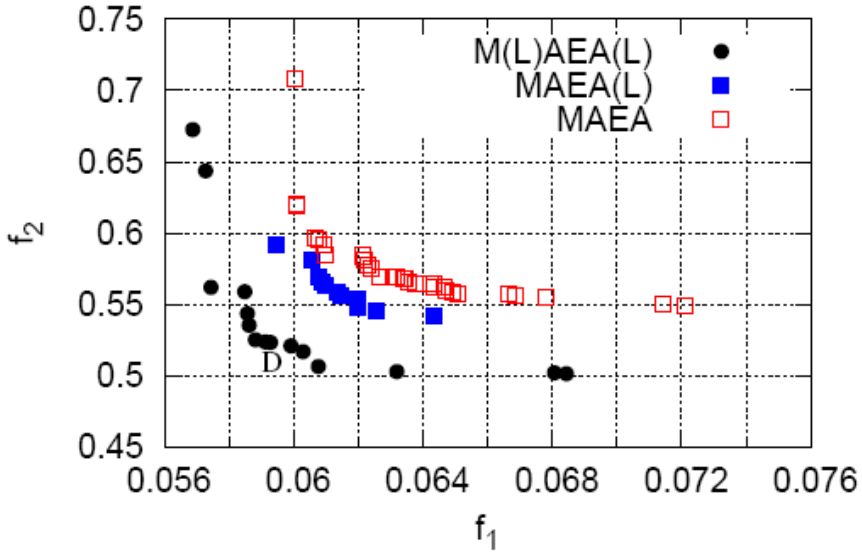


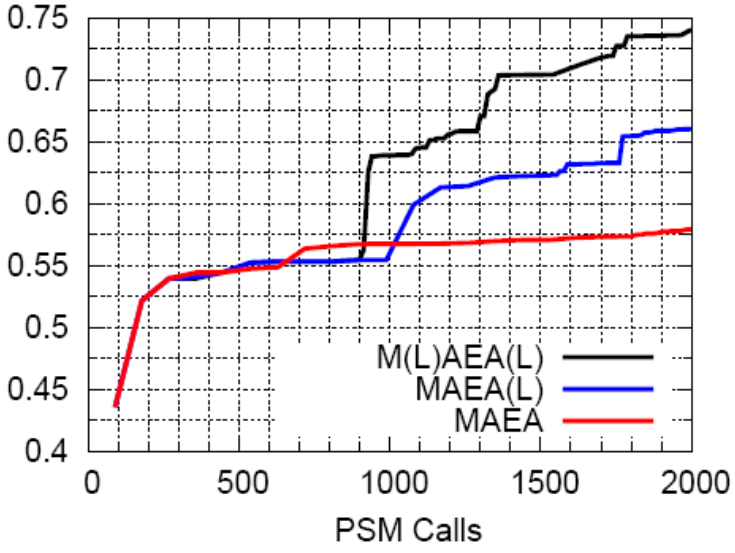


Four **optimised solutions** selected from the computed Pareto front, compared with the **reference blade**.

Shape Optimisation 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!**

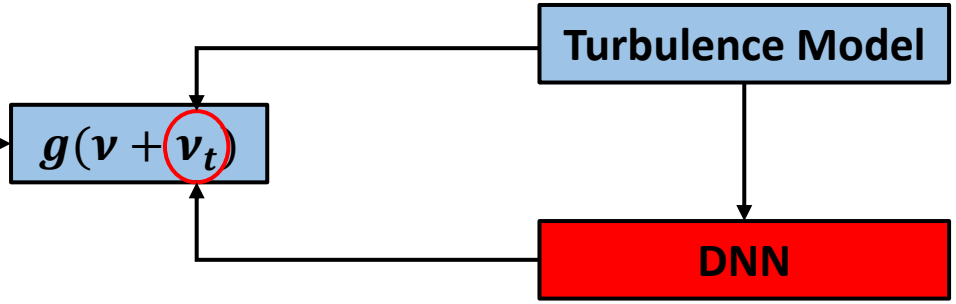
In collaboration with ANDRITZ Hydro

Deep Neural Network Surrogates for CFD

DNN Surrogates for RANS – Implementation in EAs/MAEAs

- **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.**

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



Automated Process for finding the Optimal DNN Configuration

Which is the DNN architecture?

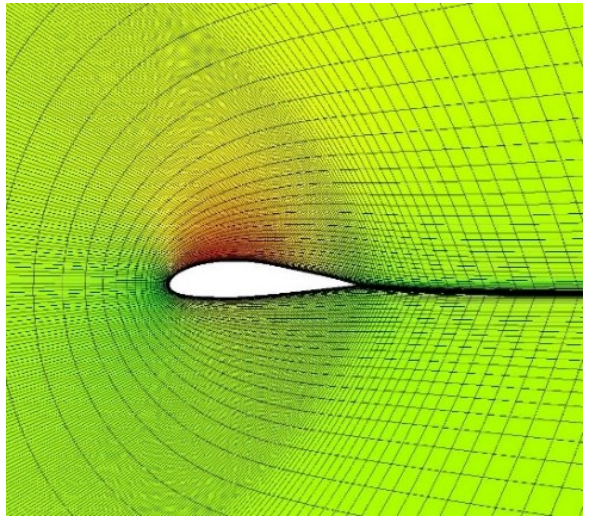
#layers, #neurons, activation functions etc

Which are the DNNs inputs?

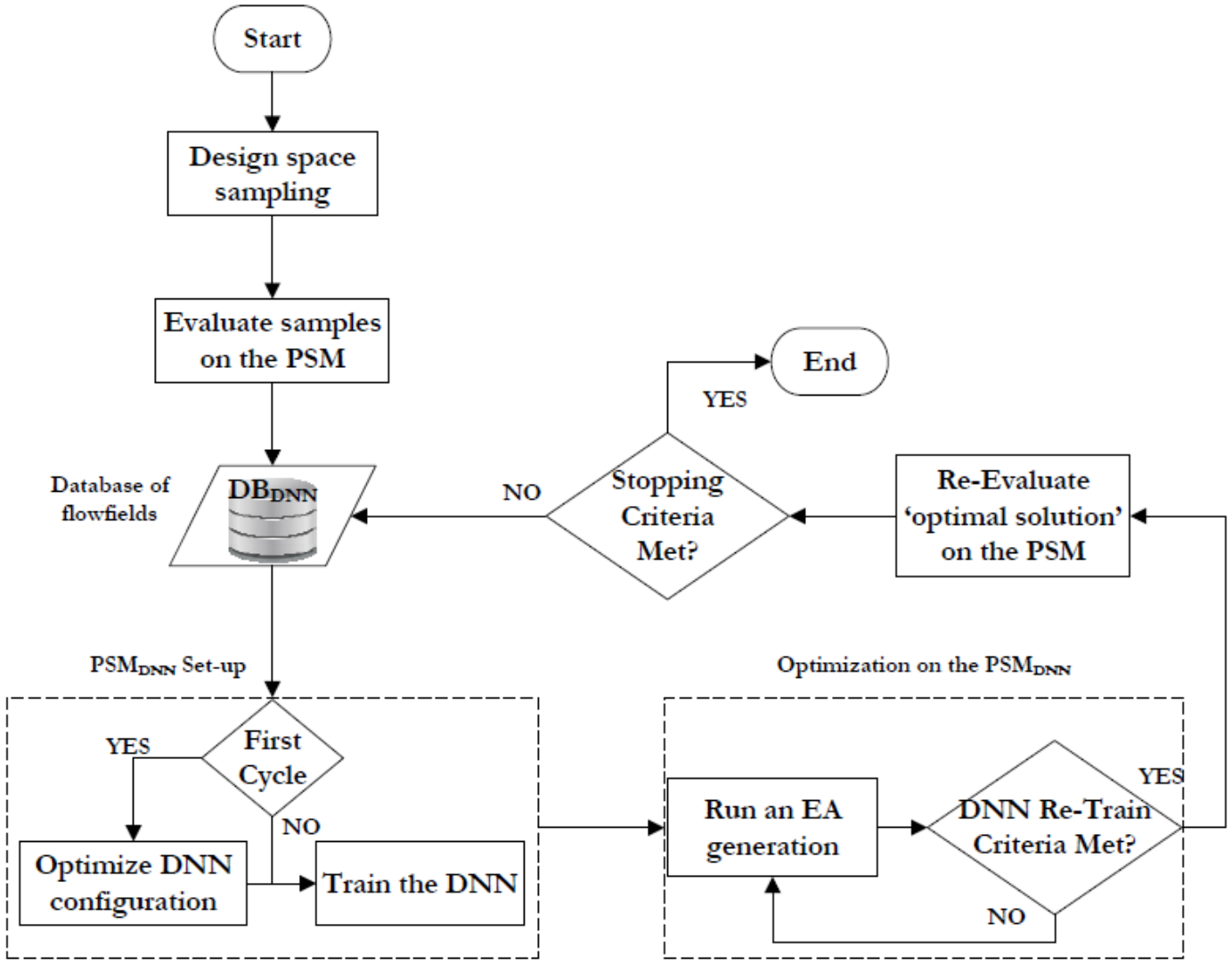
Available input data: $\vec{x}, \overrightarrow{U}, P, \Omega, S, d$ etc

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 EAs/MAEAs

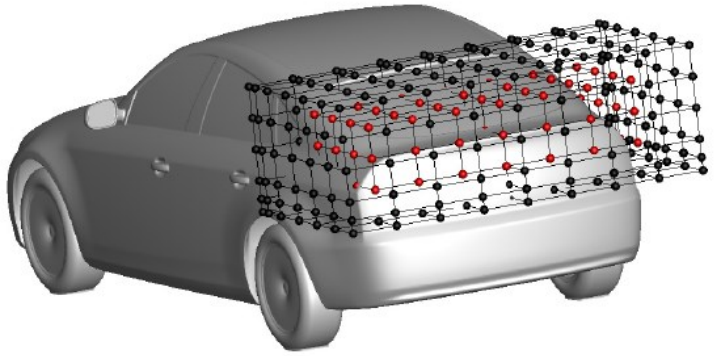


Shape Optimisation of the Drivaer Fast-back Car Model (1/2)

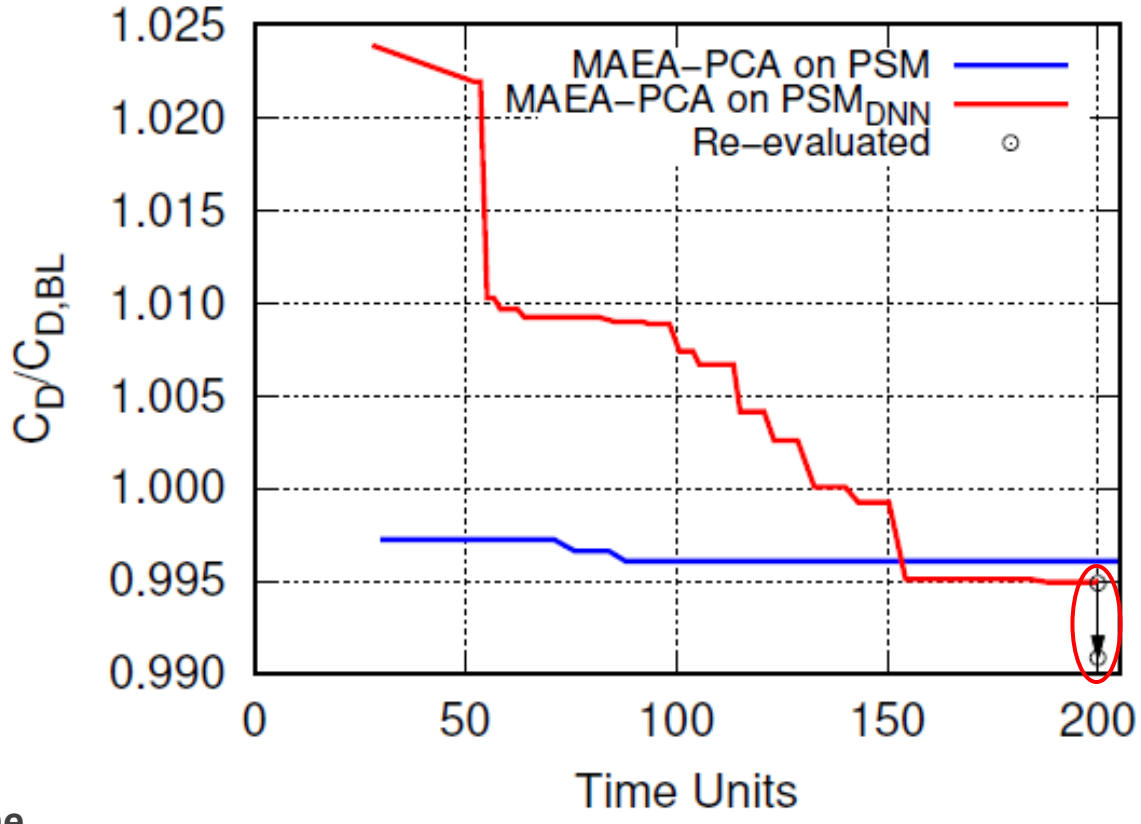
PSM_{DNN}: The PUMA code coupled with DNN to replicate the Spalart-Allmaras model. Cost=0.80PSM.

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

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

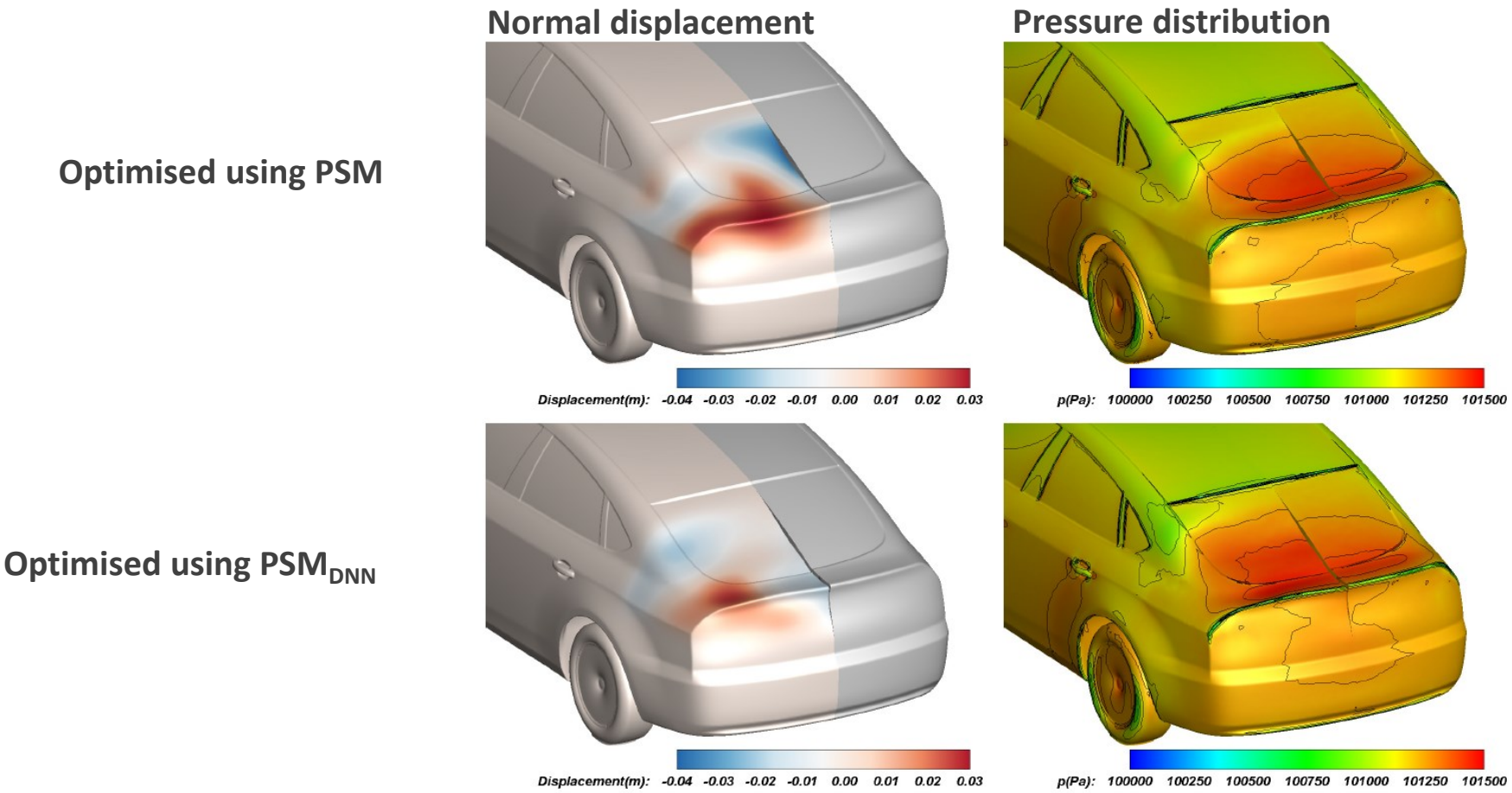


A 7×5×6 volumetric NURBS box is used to parameterise the top of the rear part of the car; control points in red are allowed to move by ±25% of their reference position in the longitudinal and ±60% in the normal-to-the ground direction. 96 DoFs. Cost per CFD evaluation ~ 1h on a single NVIDIA A100 GPU.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

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



Comparison with the baseline geometry (starboard side).

Conclusions – Lessons Learned

- **CFD-based constrained optimisation of real-world industrial problems is computationally demanding by itself.**
- **Assisting EAs 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).**
- **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).**
- ✓ **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.**