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

Introduction to Optimisation Methods – Stochastic Optimisation Methods

Dr. Kyriakos C. Giannakoglou, Professor NTUA

Dr. Varvara Asouti, Adjunct Lecturer NTUA

Contributors:

Dr. A. Giotis

Dr. M. Karakasis

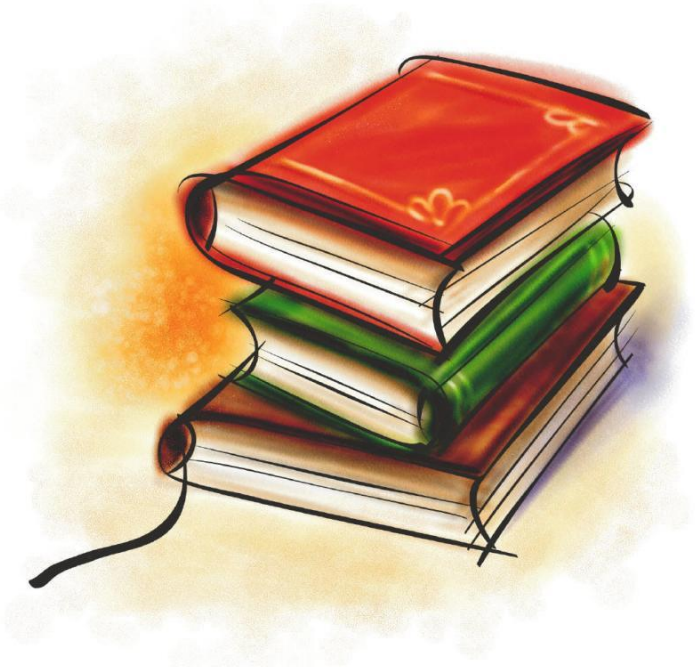
Dr. S. Kyriacou

Dr. D. Kapsoulis

Dr. E. Papoutsis-Kiachagias

Dr. X. Trompoukis

Dr. M. Kontou



Introduction

Preface – Terminology & Focus

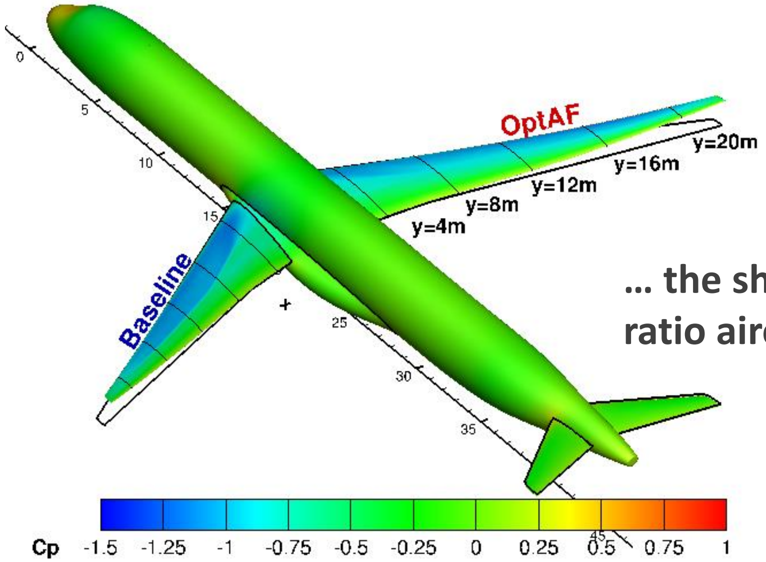
- Optimisation is inherently linked with all scientific/technological fields & takes place even if this is not explicitly stated. *The enemy of the good is the best!*
- This course is about optimisation methods which are, typically, classified as gradient-based (deterministic, GBM) or gradient-free (stochastic, GFM) ones.
- Optimisation under Uncertainties (OuU); Robust-Design Optimisation (RDO).
- Single-Objective Optimisation (SOO); Multi-Objective Optimisation (MOO).
- Unconstrained or Constrained optimisation.
- Optimisation problems for which analysis (primal problem) relies on the numerical solution of (systems of) PDEs.
- Looking for optimisation methods which are both effective and efficient; getting the optimal solution(s) at the minimal computational cost is of great importance!
- Select the most appropriate optimisation method based on the development cost (if code development is needed) as well as the run-time cost.
- Optimisation vs. Improvement!

Also:

- **This course is not necessarily for engineers, though most of the applications are in the field of (mechanical/aeronautical) engineering.**
- **Though many applications are related to cars, aircraft, etc, they can easily be understood by those studying Computational Sciences. Don’t worry: a simplified short introduction to all problems will be provided.**
- **Mathematical examples will often be used in this course.**
- **As optimisation scientists, we assume that the analysis method/tool/software is already in place.**
- **All presented results of industrial applications have already been published.**

Real-world Optimisation (1/2)

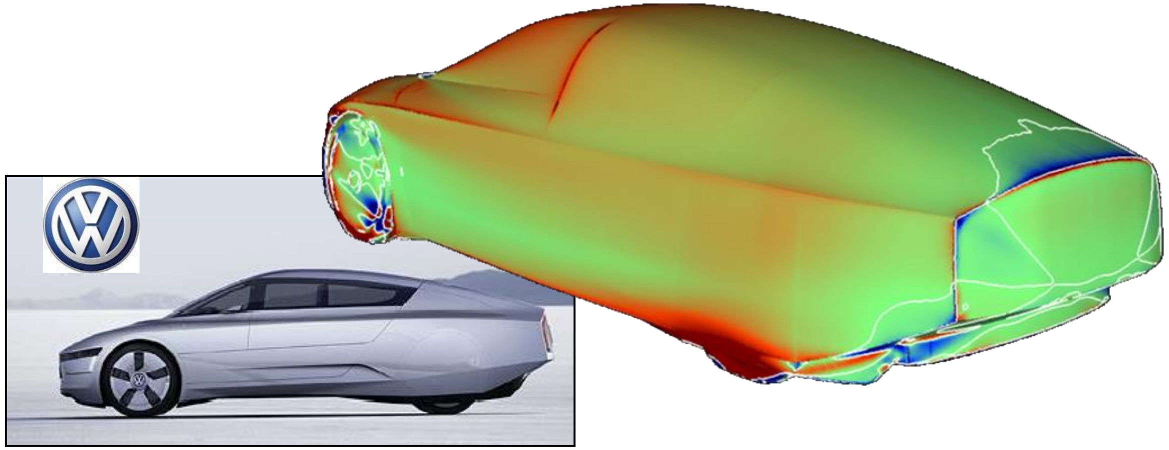
… the shape of an ultra-lightweight vehicle



… the shape of a high-aspect ratio aircraft wing (DLR-F25).

NEXTAIR

Funded by the European Union

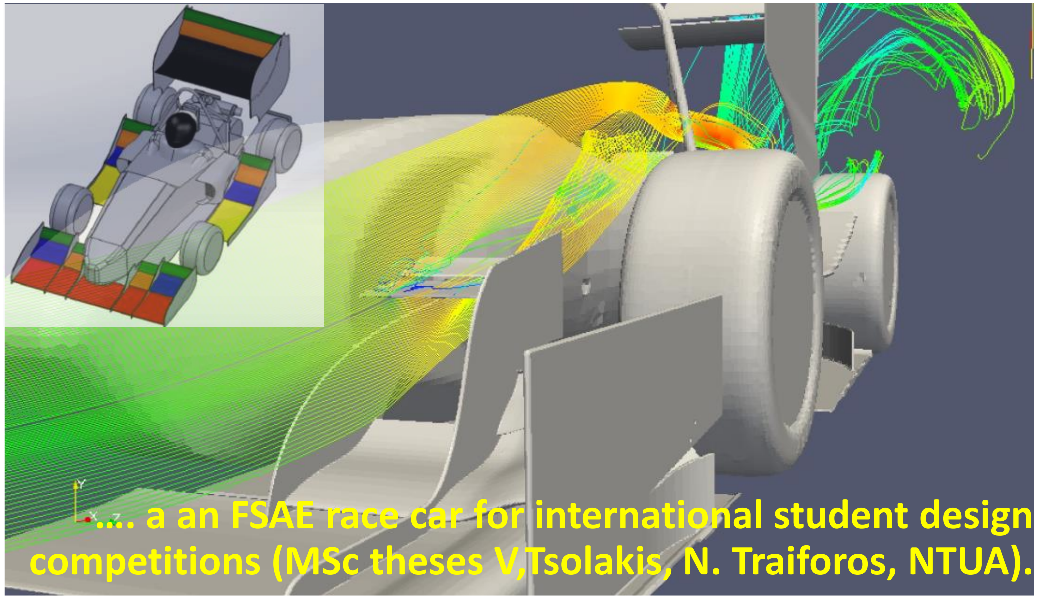


… the shape of a passenger car.

TOYOTA
STLSolidLabeling

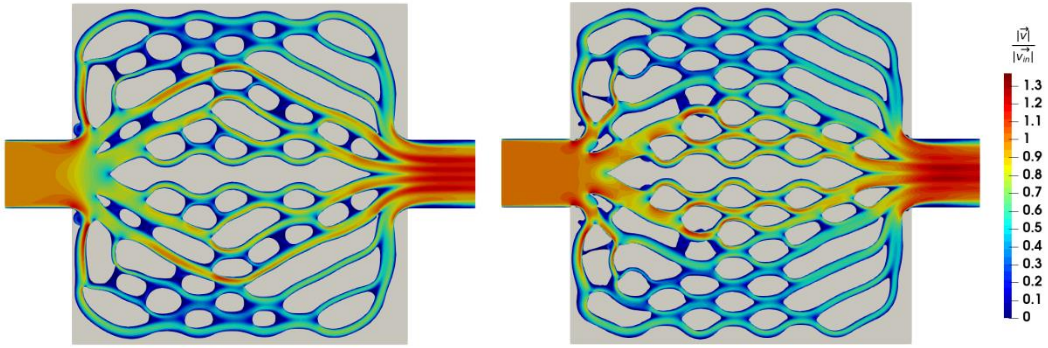
… the shape of the defroster nozzle of the HVAC unit of a passenger car.

Real-world Optimisation (2/2)

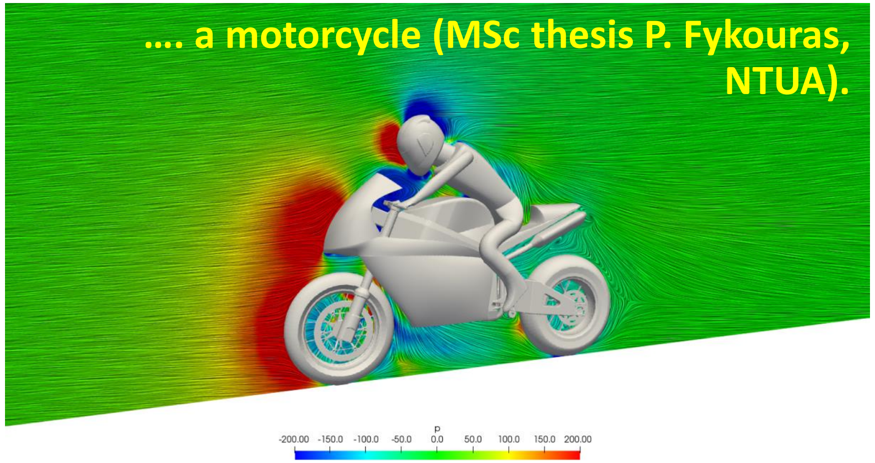


…. a an FSAE race car for international student design competitions (MSc theses V,Tsolakis, N. Traiforos, NTUA).

… a bicycle frame.



… the cooling of heat sinks (PhD N.Galanos, NTUA).

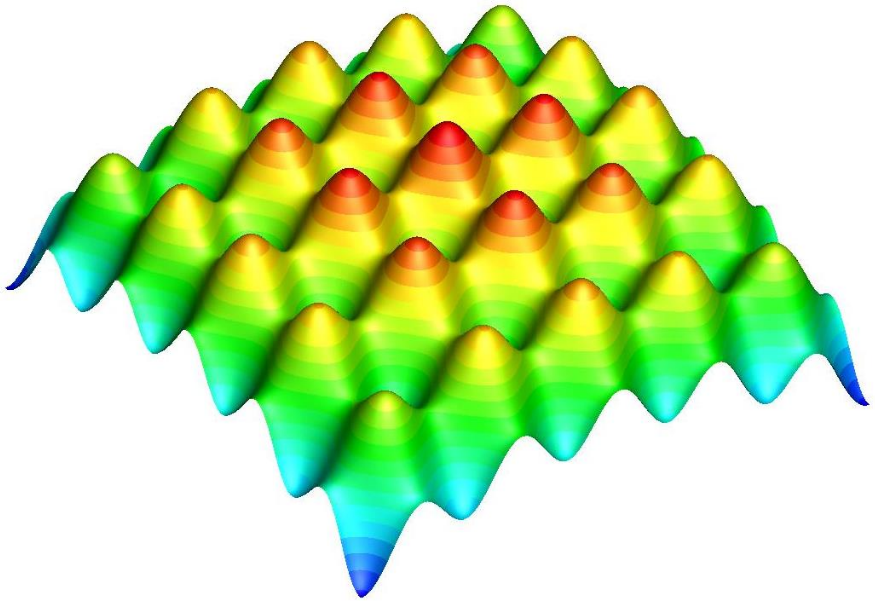


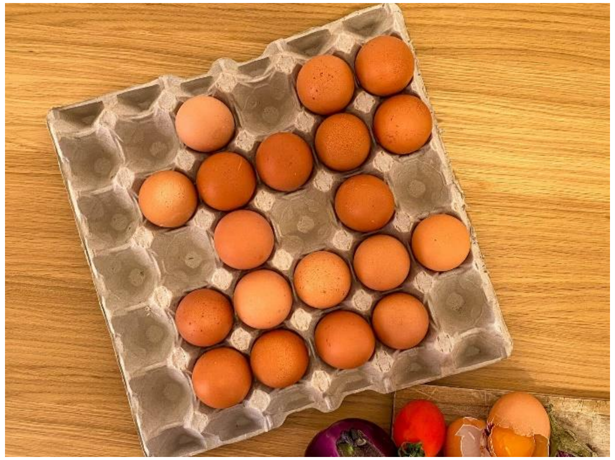
…. a motorcycle (MSc thesis P. Fykouras, NTUA).

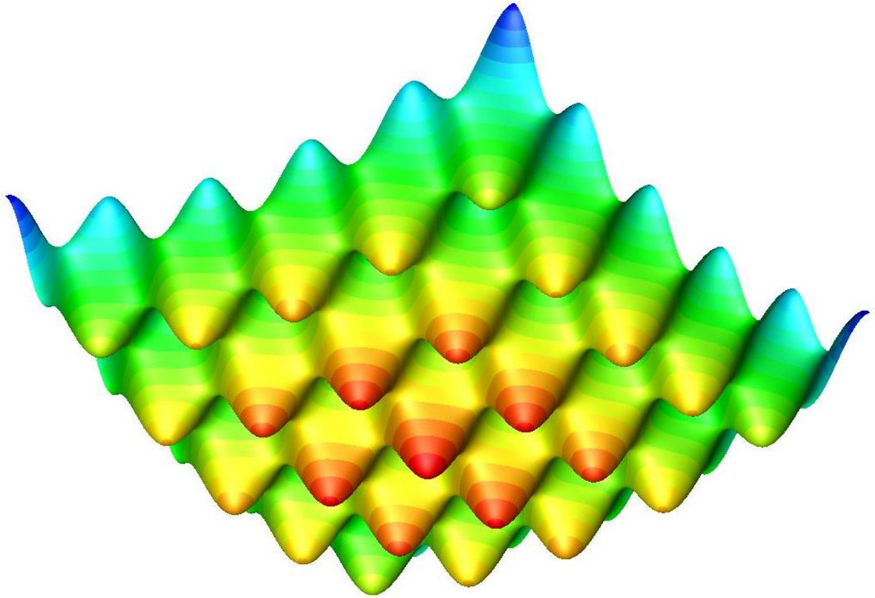
Mathematical (Multimodal) Optimisation Problems

- Local vs. Global optimum.
- Min. vs. max.

- For benchmarking new methods.
- Multimodal functions.

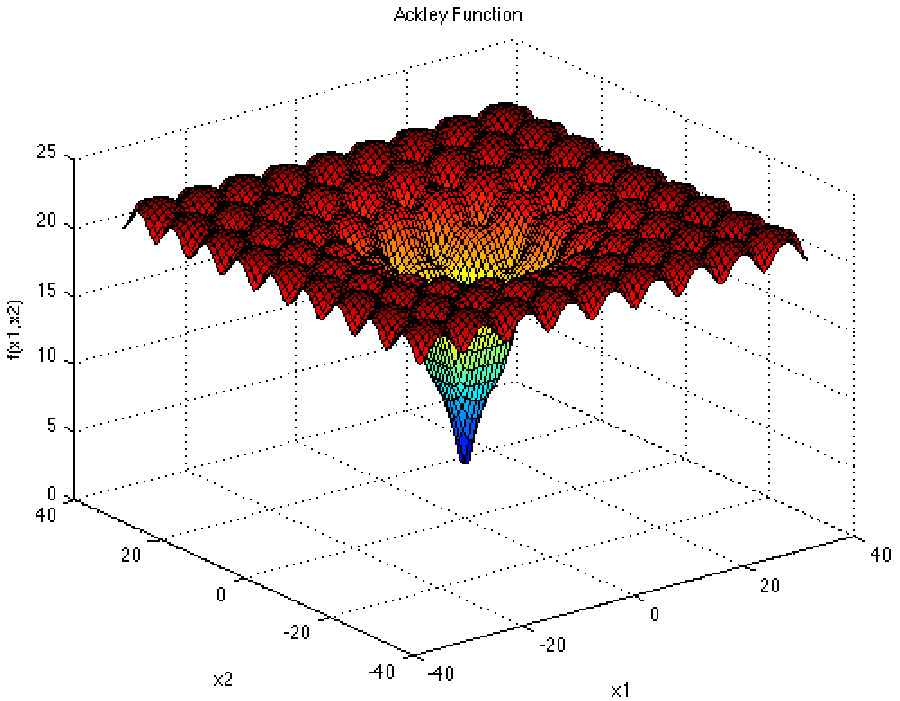


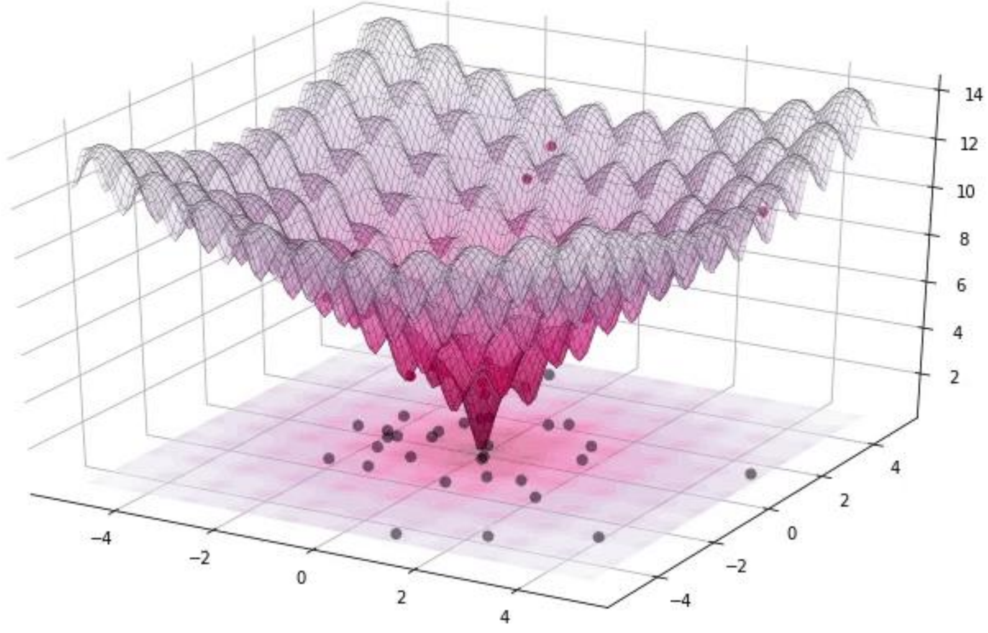




There are plenty of egg-holder-like functions. Typical max/minimisation problem, max. (or min.) F(b_1,b_2), N=2, with two design variables. Benchmarking new optimisation methods frequently rely on multidimensional extensions (N>>) of such functions.

Example: the Ackley Function





The non-convex Ackley function is widely used for testing optimisation algorithms. In its 2D form, as shown in the plot above, it is characterised by a nearly flat outer region, and a large hole at the center. N should take on any integer value.

$$f_{AC}\left(\vec{b}\right) = 20 + \exp(1) - 20exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} b_i^2}\right) - exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi b_i)\right) \qquad -32.768 \le b_i \le 32.768$$

From Analysis to Optimisation (a CFD example)

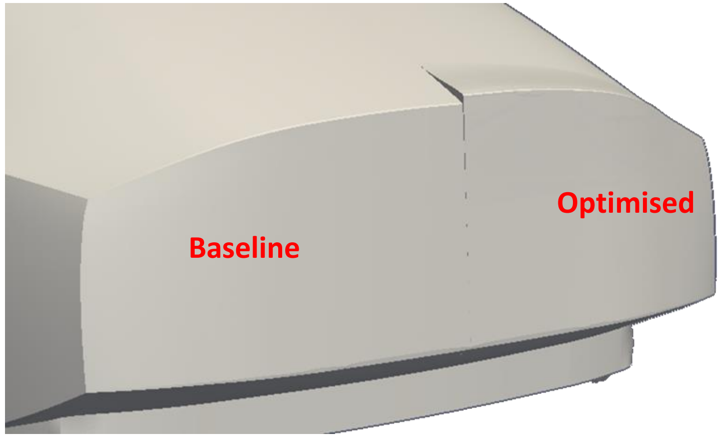
Working Assumption: the analysis method/tool/software is already in place.

Analysis:

Given the shape/geometry of a car or aircraft, etc., in a fixed environment (fixed boundary conditions), the analysis problem stands for the computation of the flow field around it (CFD analysis, followed by the integration of forces over its surface) in order to compute quantities (drag, lift, etc.) determining its performance.

Design-Optimisation:

Given the environment (environmental/flow conditions,…), compute the car or aircraft shape/geometry that gives best performance (min. drag, max. lift, a combination of both, etc).



VW XL1 drag minimisation, using methods & s/w developed by the PCOpt/NTUA. The optimised shape has >2% reduction in drag (for an aerodynamically nearly perfect car) and 30% increase in lift.

Optimisation under Uncertainties (OuU)

Analysis:

Given the shape/geometry of a car or aircraft, etc., in a fixed environment (fixed boundary conditions), analysis problem stands for the computation of the flow field around it (CFD analysis, followed by the integration of forces over its surface) in order to compute integral quantities (drag, lift, etc.) determining its performance.

Shape/geometry → Manufacturing Uncertainties!!!

Boundary (flow) conditions → Changing based on some probabilistic laws!!!

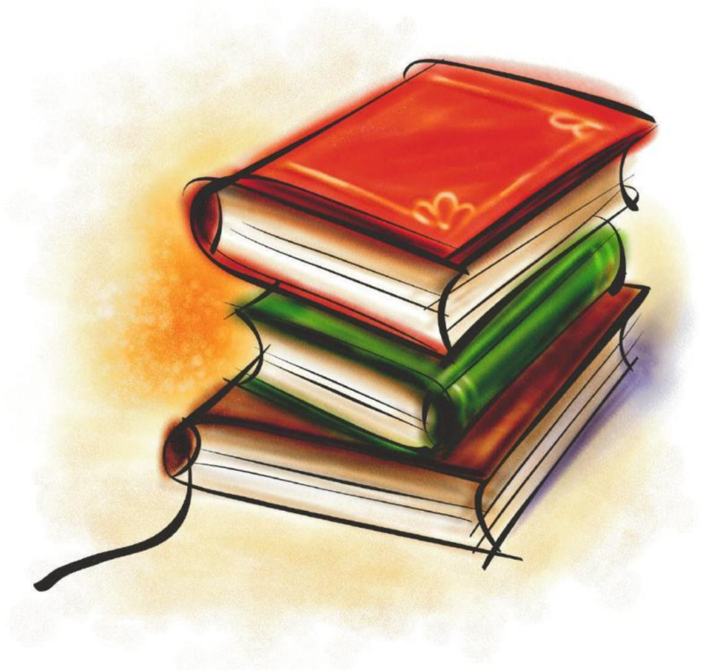
OuU – Robust Design Optimisation (RDO):

Given the uncertain environment or shape, compute the car or aircraft shape/geometry that gives statistically best performance. Uncertainty Quantification (UQ) is a prerequisite for this!

A Few Comments

- **Analysis s/w: Commercial or in-house CFD code.**
- **Having or not access to the analysis code (source code) seriously affects the selection of the optimisation method.**
- **The frequency of running an optimisation task affects this selection, since development and run-time costs are usually contradictory.**
- **S/w parallelisation on CPUs and/or GPUs. Parallelise the CFD analysis or the search process or both?**
- **Artificial Intelligence (AI) / Machine Learning (ML) inherently use optimisation methods and, in return, are extensively used in optimisation workflows to reduce the computational cost.**

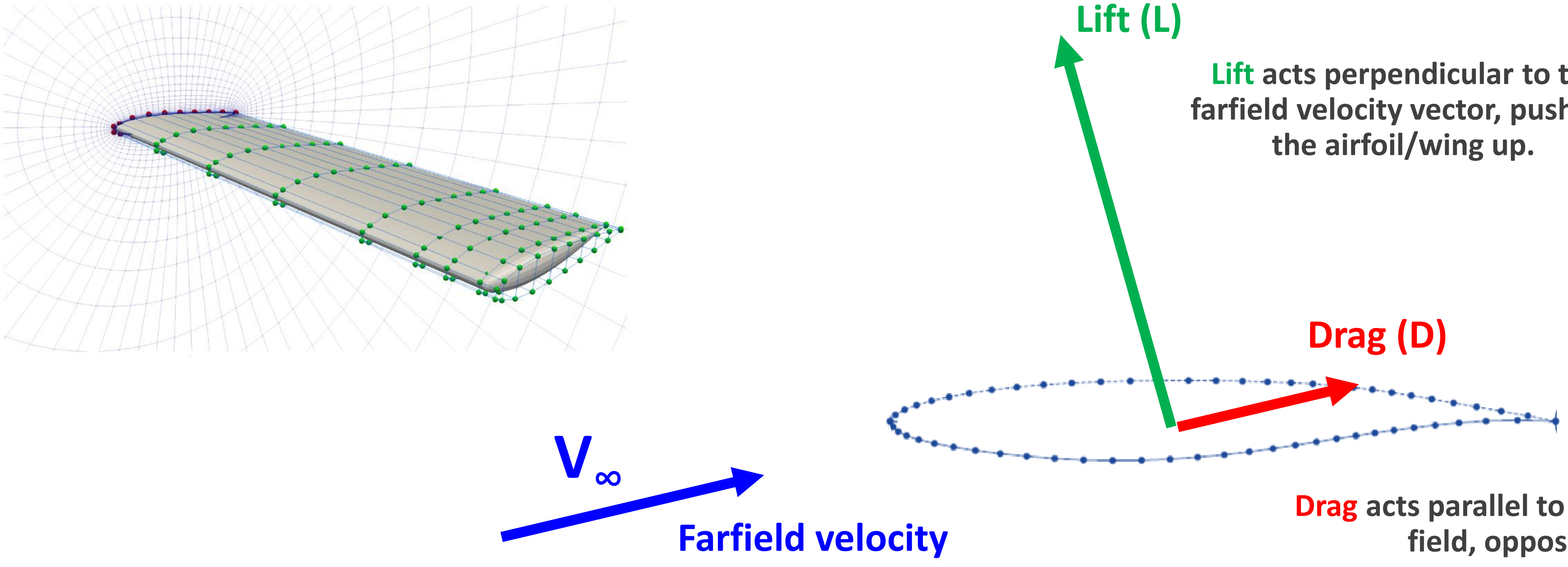
The above comments are valid for any other discipline such as Structural Mechanics, Electromagnetism, etc. or Multi-Disciplinary Optimisation (MDO) problems.



Preparing a CFD-based Shape Optimisation (ShpO)

**We are using CFD as an intuitive example.
This is not a course on CFD!**

Fundamentals of Aerodynamics / Fluid Mechanics (1/2)

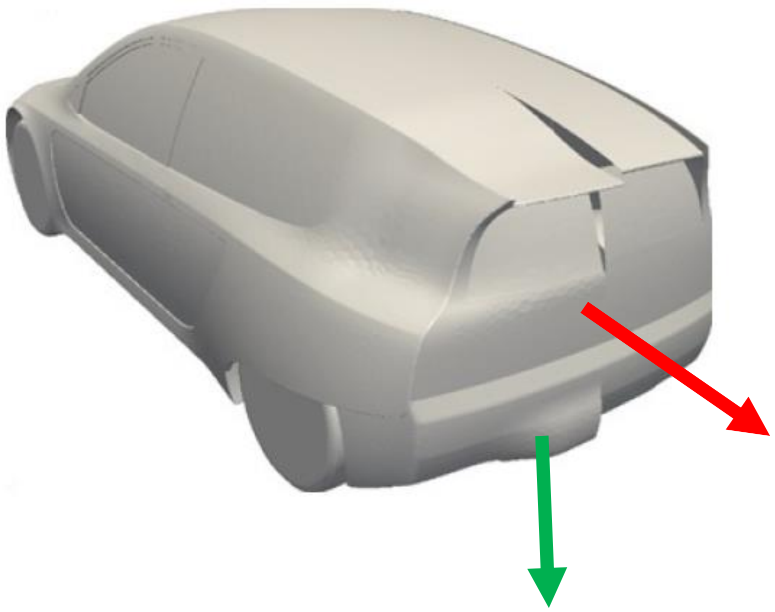


Lift acts perpendicular to the farfield velocity vector, pushing the airfoil/wing up.

Drag acts parallel to the farfield velocity field, opposing motion.

Lift (C_L) and Drag (C_D) coefficients are non-dimensional quantities frequently involved in optimisation runs, for the purpose of similarity (large- and small-scale).

Fundamentals of Aerodynamics / Fluid Mechanics (2/2)



Drag (D)

Downforce (L)

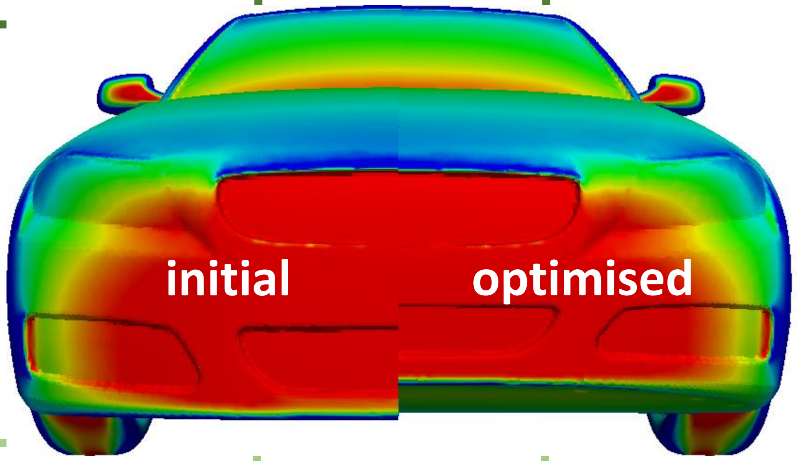
Prerequisites for running Shape Optimisation (ShpO) in CFD

PR1

Flow (CFD) Model (& s/w) incl. grid generation & grid displacement.

PR3

Objective & Constraint Function(s); gradient-based optimisation requires their derivatives w.r.t. b_n, n=1,…,N.



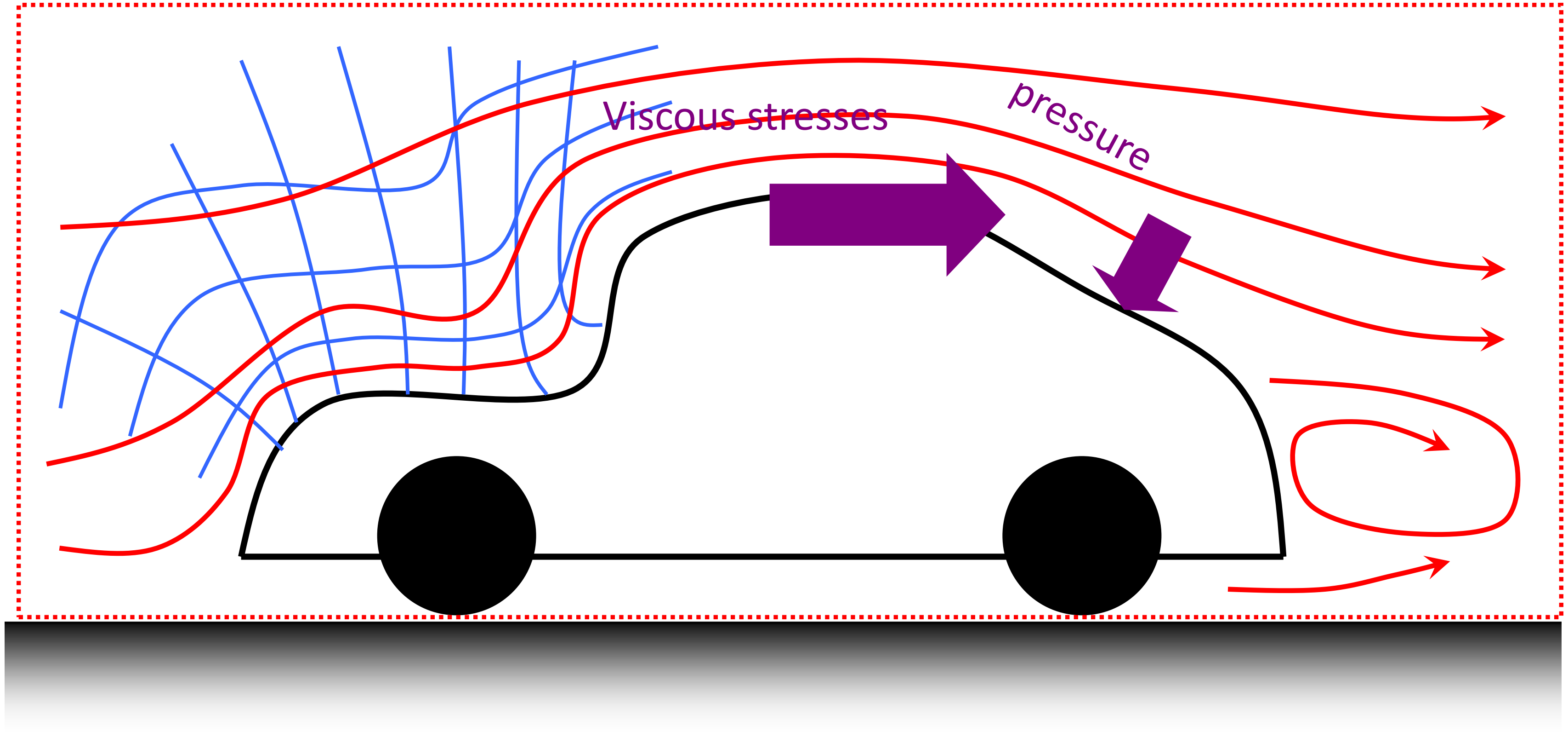
PR2

Case Parametrisation, defining the design variables (Degrees of Freedom, DoFs) b_n, n=1,…,N of the case.

Optimisation Method, incl. the computation of grad(…) if needed.

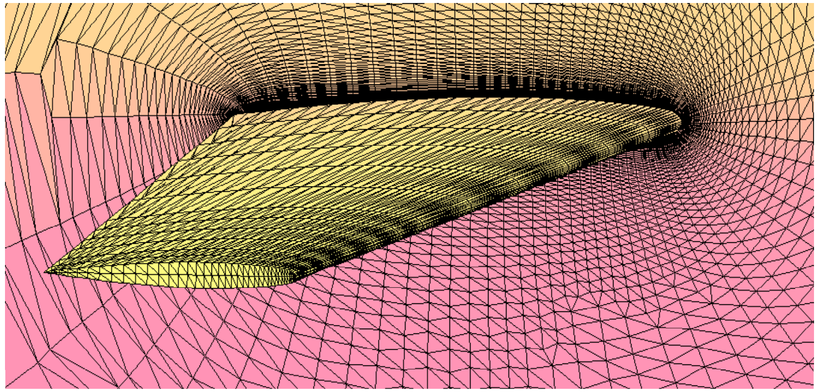
PR1 - Flow Model (& peripherals) (1/10)

Here, "Analysis" is based on a CFD s/w and some peripherals (primal code). The analysis cost is, practically, the time-unit (TU) to be used to compare optimisation methods.

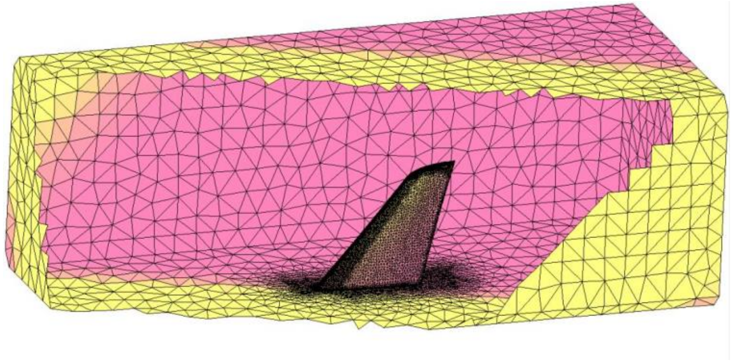


$$C_D = \oint_{carcontour} forces$$

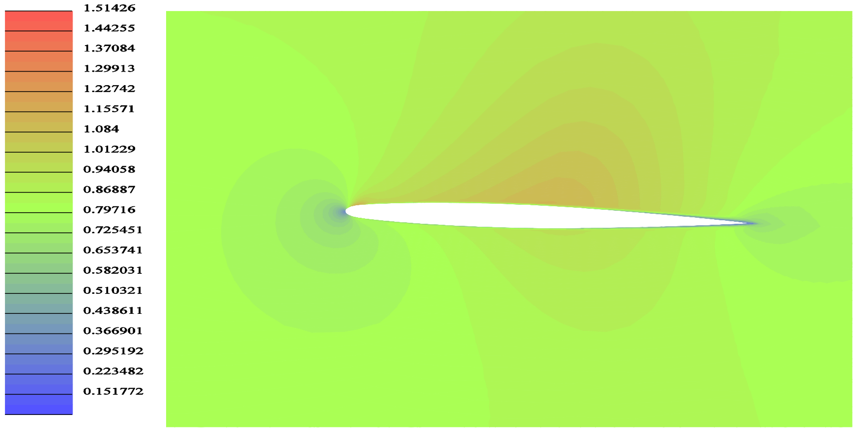
PR1 - Example: Aerodynamic Analysis of a Wing (2/10)



Wing geometry



Generate the computational grid

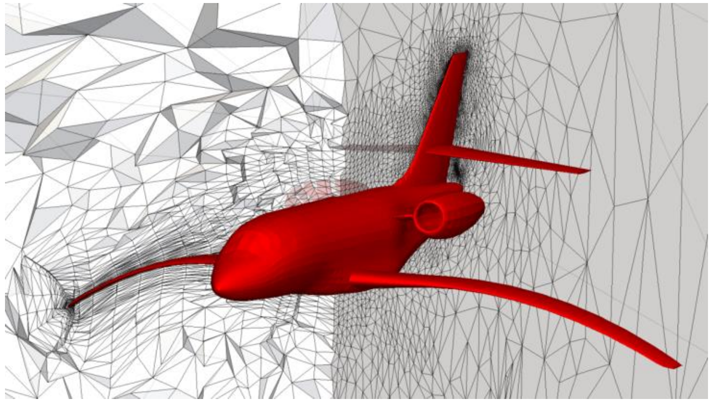


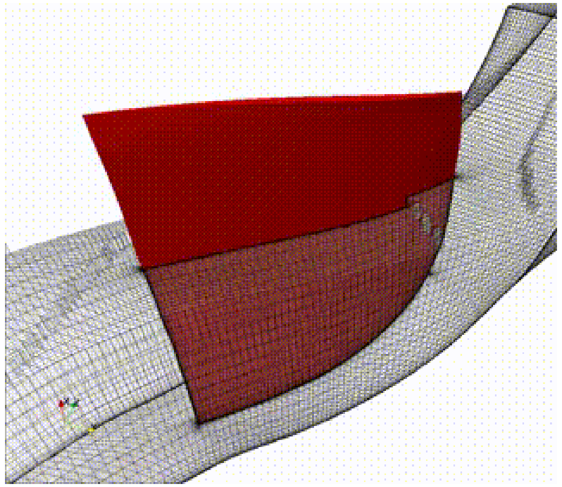
Numerically solve the Navier-Stokes equations

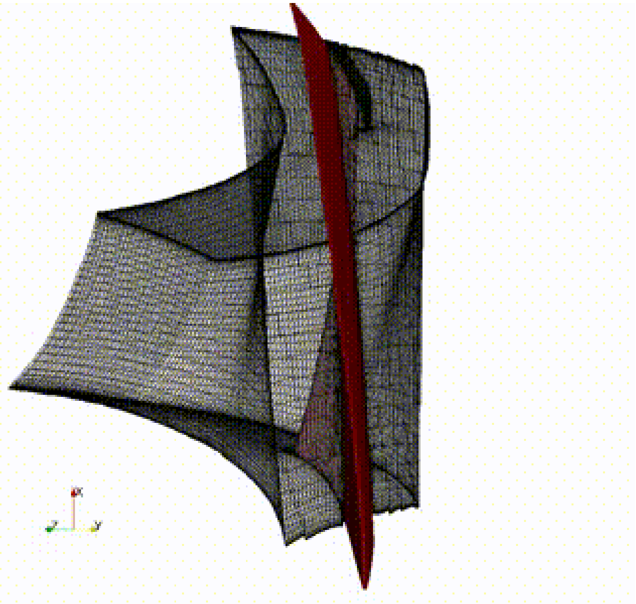
C_L=…
C_D=…

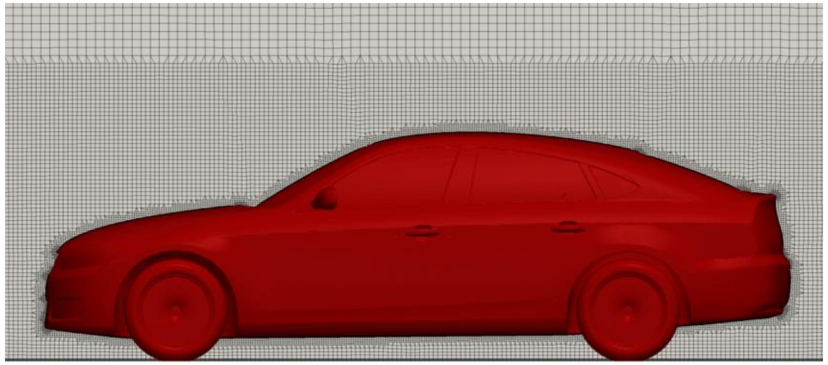
Post-process the results

PR1 - Grid Displacement Tools to Support ShpO (3/10)









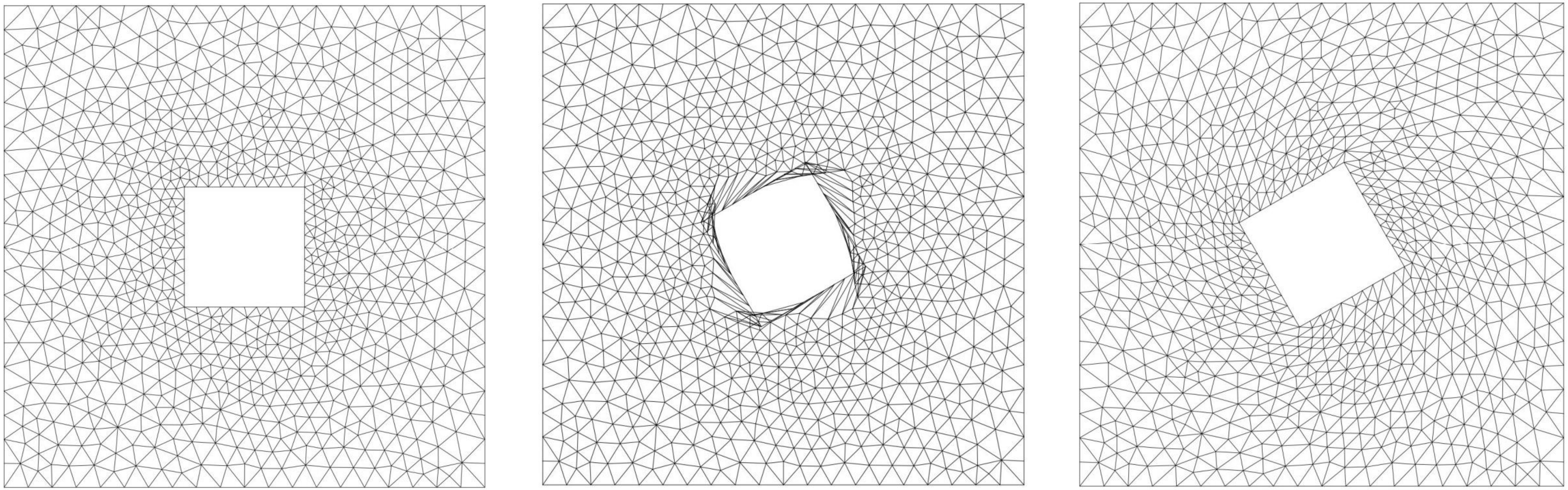
Remeshing or Grid Displacement for each new candidate solution?

Fast grid displacement tools to support ShpO. Internal grid is adapted to the displaced boundaries, in each optimisation cycle.

These are the same tools supporting CFD runs with moving walls (aeroelasticity, etc.).

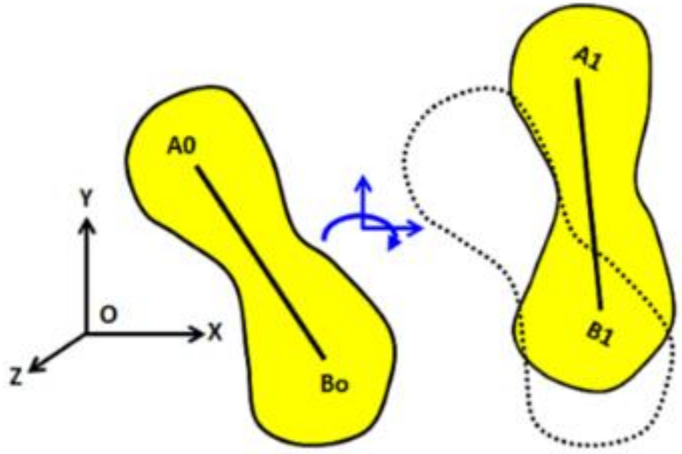
Some Grid Displacement methods: Algebraic models (Inverse Distance Weighted, IDW), Laplacian Models, Volumetric B-Splines, Linear/Torsional Spring Analogies, Harmonic Coordinates, Delaunay Graph, etc.

PR1 - The Importance of Having Access to a Robust Grid Displacement Tool (4/10)



PR1 - Rigid Body Motion for Grid Displacement (5/10)

This is a connectivity-agnostic method which applies the rules following the motion of a solid body (the distance between any two points on the body does not change!). Since this is not possible in case of varying domain boundaries, an optimisation problem should be solved.

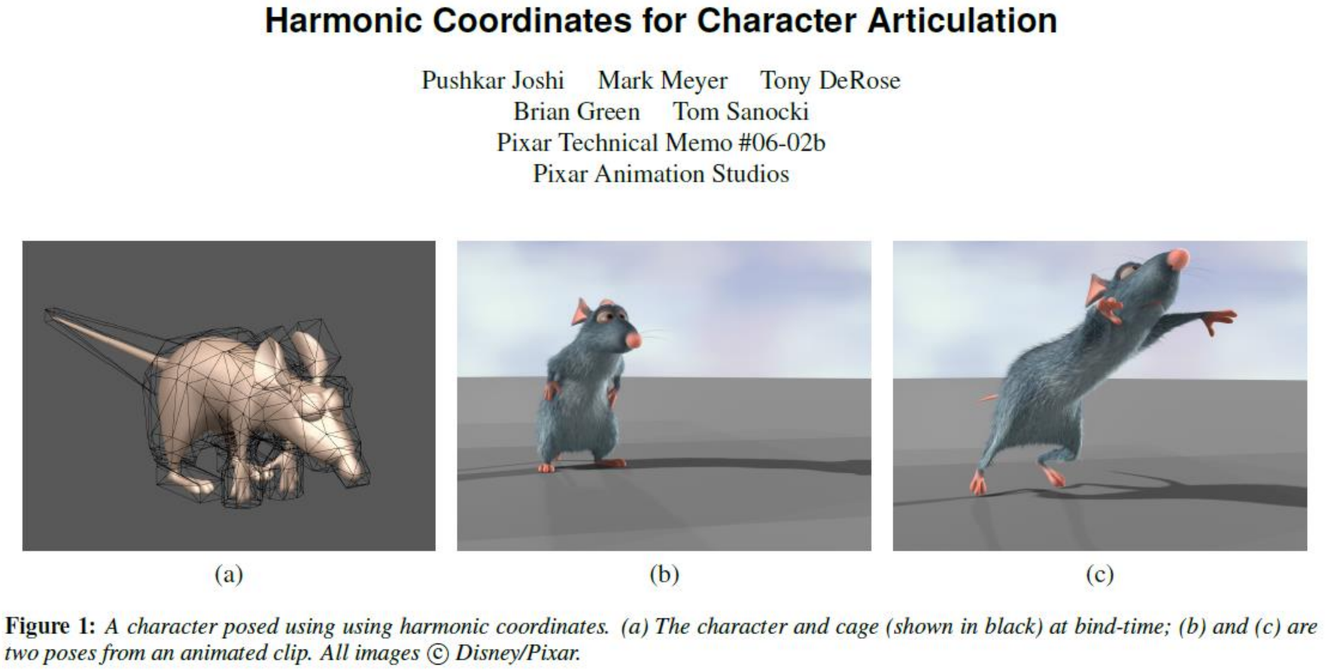


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Δx, Δy = translation; θ = rotation

$$F_i = \frac{1}{2} \sum_{j \in \mathcal{N}(i)} \left[(x_j^{ideal} - x_j^{new})^2 + (y_j^{ideal} - y_j^{new})^2 \right]$$

PR1 - Harmonic Coordinates (6/10)

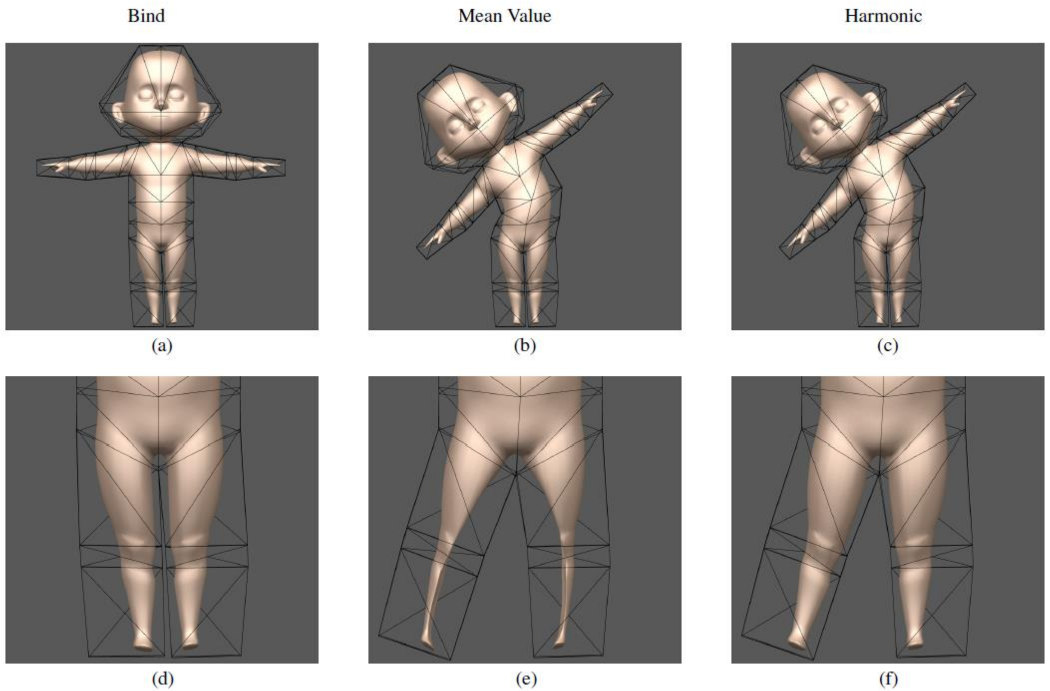


Harmonic Coordinates for Character Articulation

Pushkar Joshi Mark Meyer Tony DeRose
Brian Green Tom Sanocki
Pixar Technical Memo #06-02b
Pixar Animation Studios

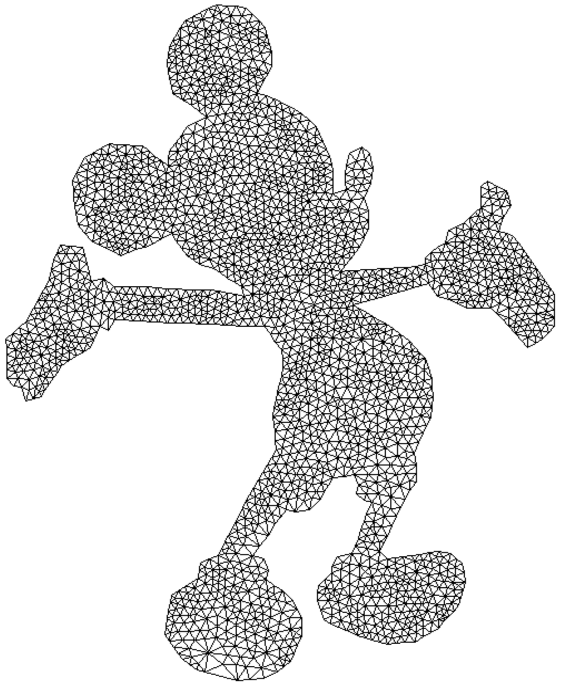
(a) (b) (c)

Figure 1: *A character posed using using harmonic coordinates. (a) The character and cage (shown in black) at bind-time; (b) and (c) are two poses from an animated clip. All images © Disney/Pixar.*

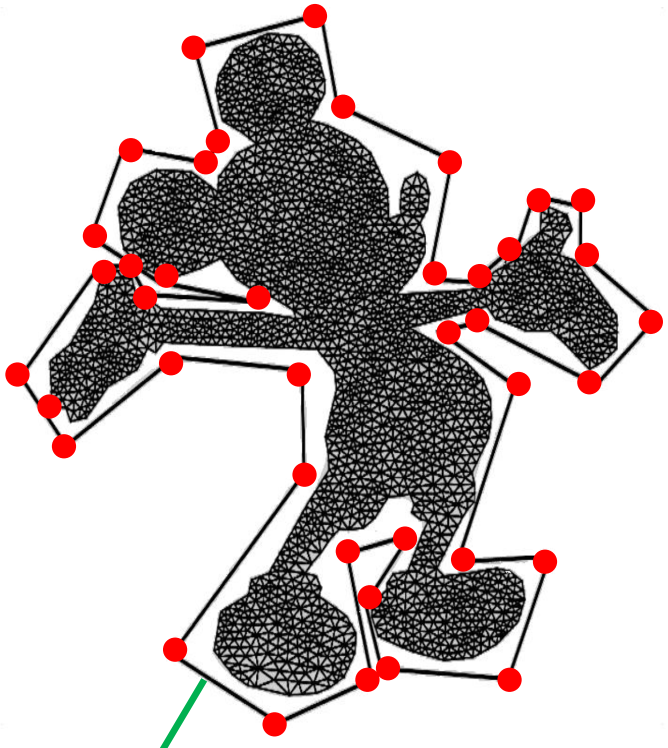


… the problem of creating and controlling volume deformations used to articulate characters for use in highend applications such as computer generated feature films. We introduce a method we call harmonic coordinates that significantly improves upon existing volume deformation techniques. Our deformations are controlled using a topologically flexible structure, called a cage, that consists of a closed 3D mesh. The cage can optionally be augmented with additional interior vertices, edges, and faces to more precisely control the interior behavior of the deformation. We show that harmonic coordinates are generalised barycentric coordinates that can be extended to any dimension.

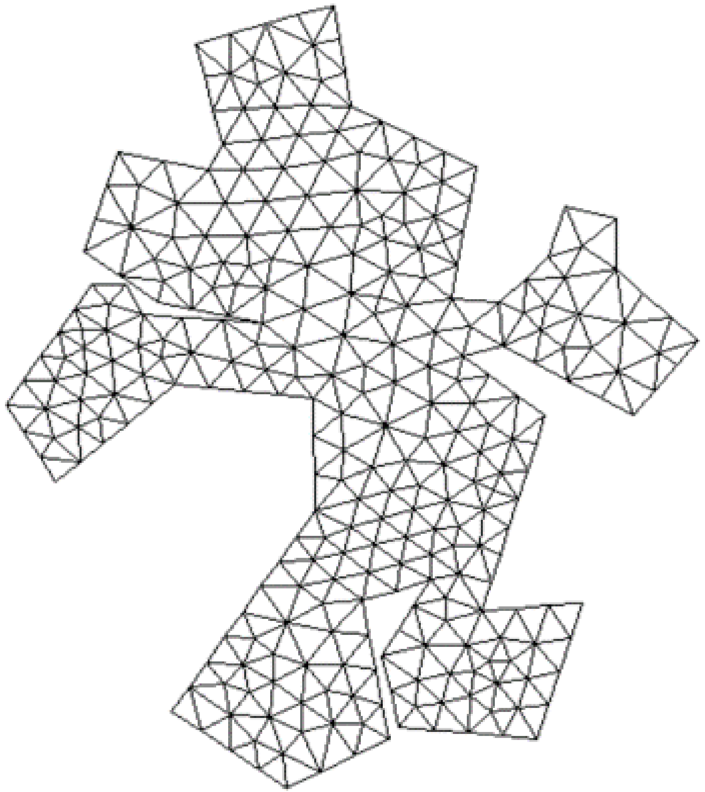
PR1 - Harmonic Coordinates (7/10)



Computational Grid

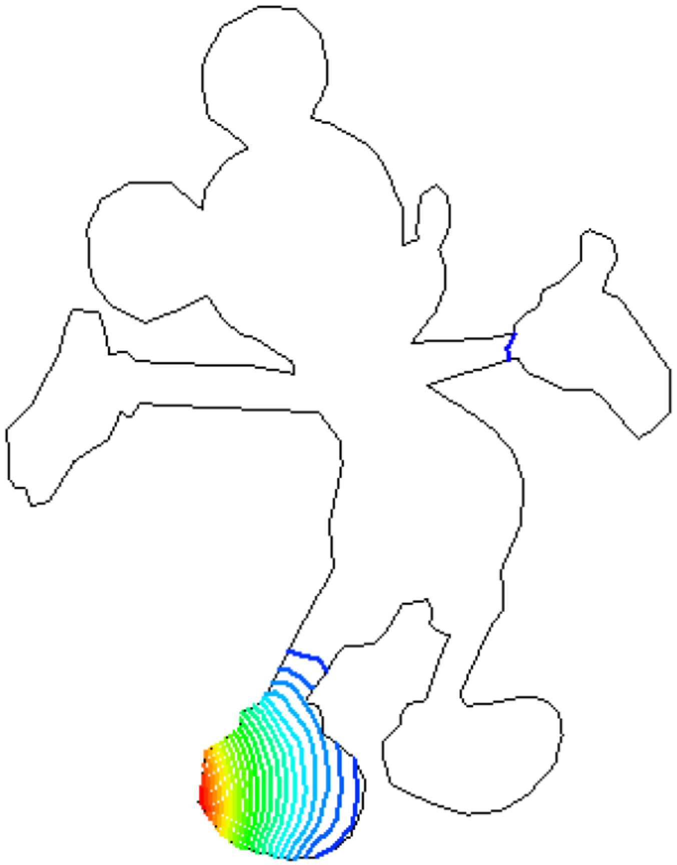


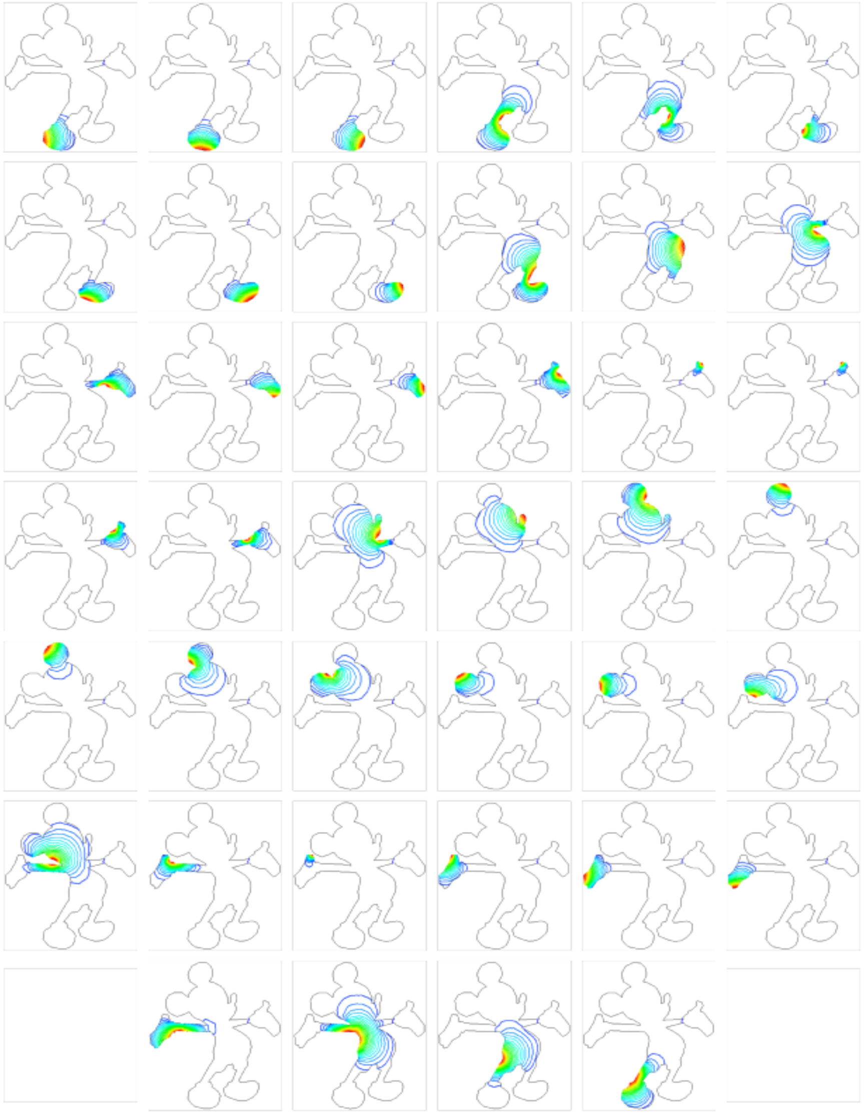
Cage (40 control-points)

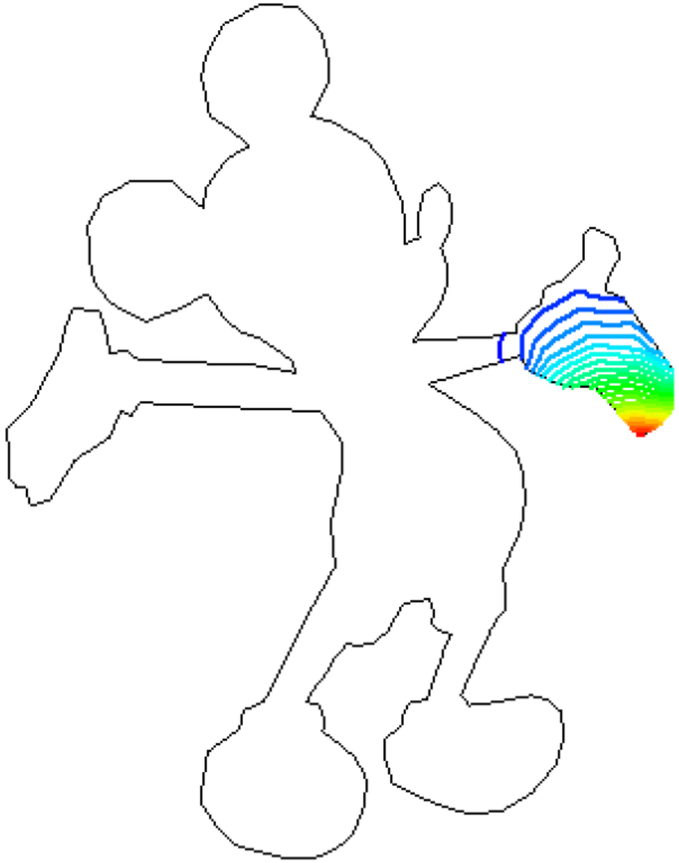


Control Grid

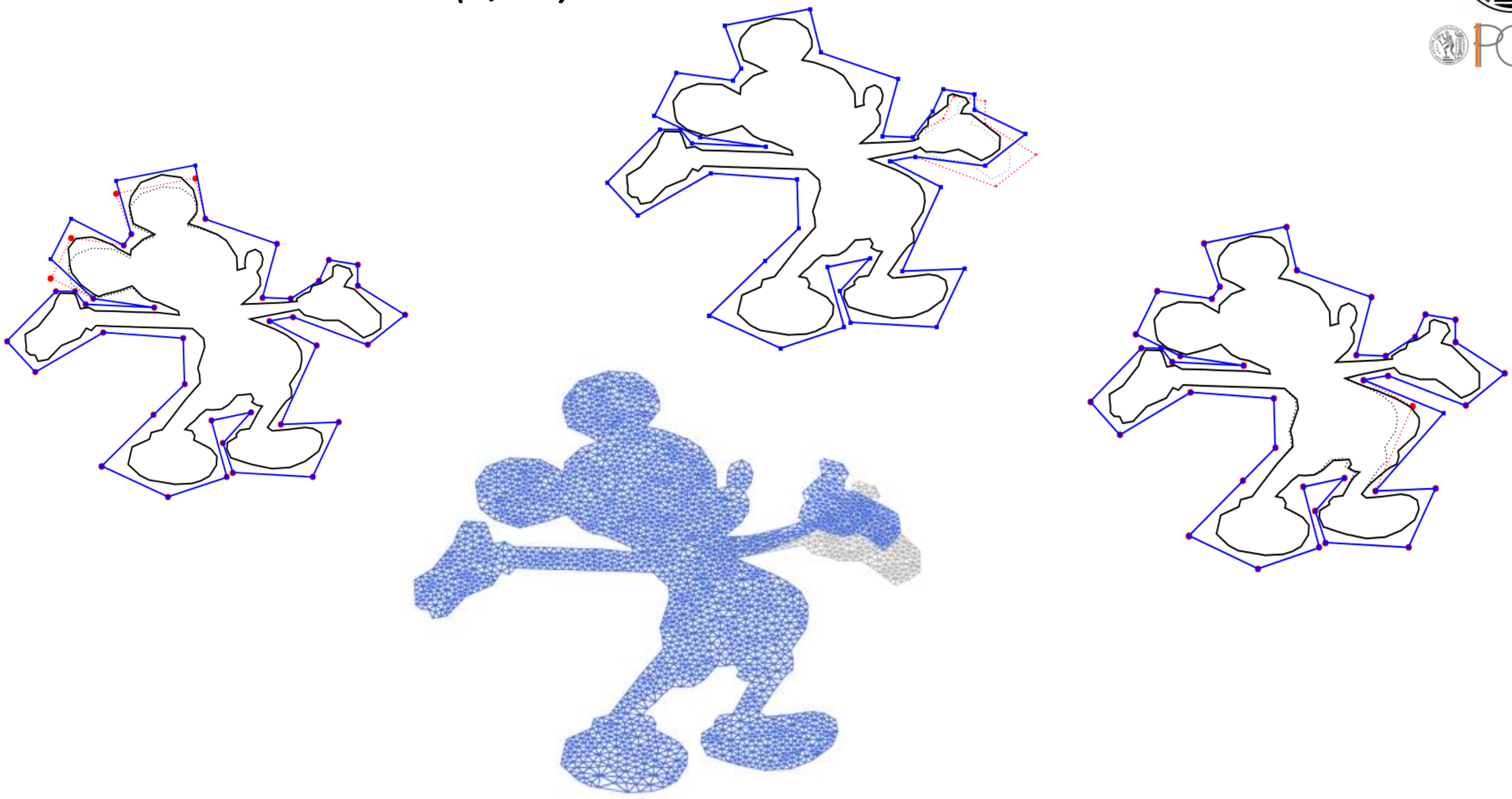
PR1 - Harmonic Coordinates (8/10)



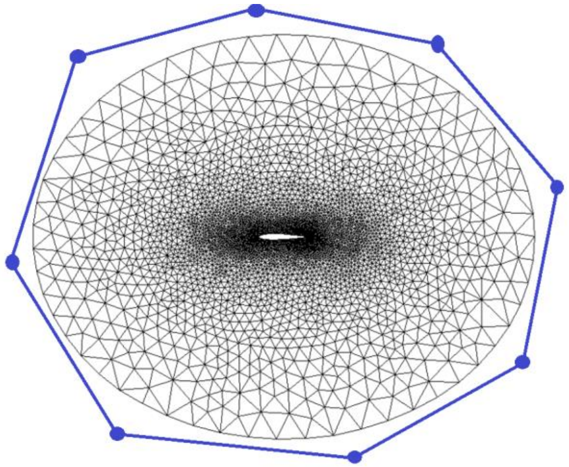


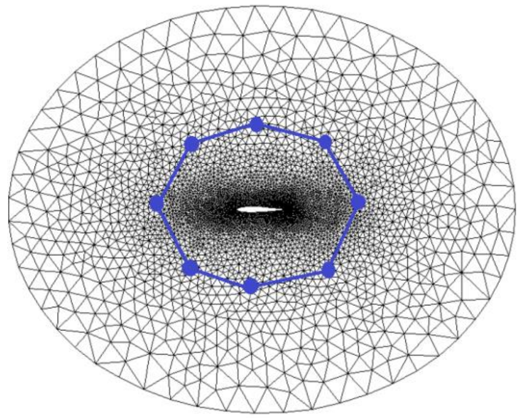


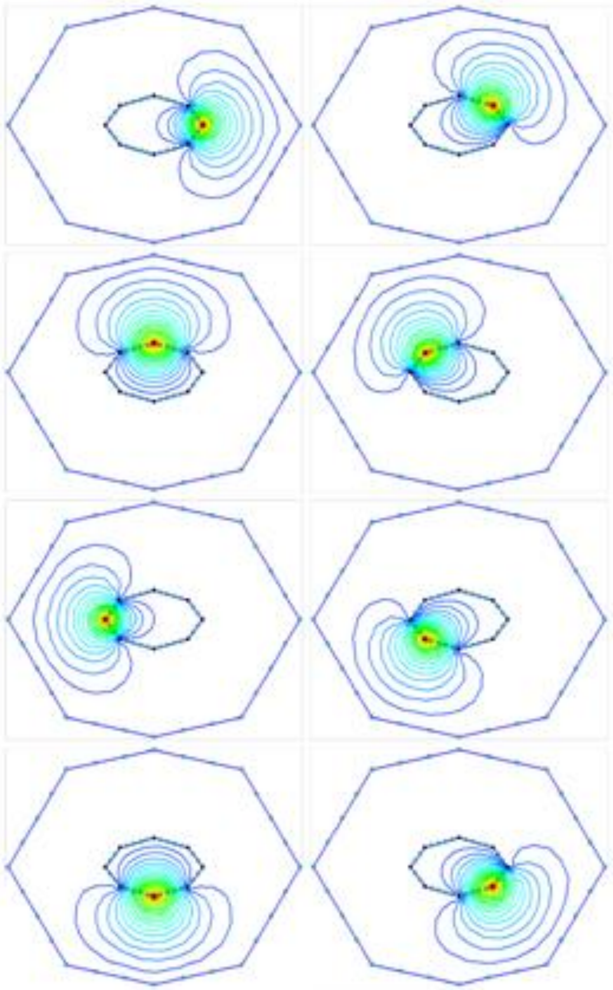
PR1 - Harmonic Coordinates (9/10)

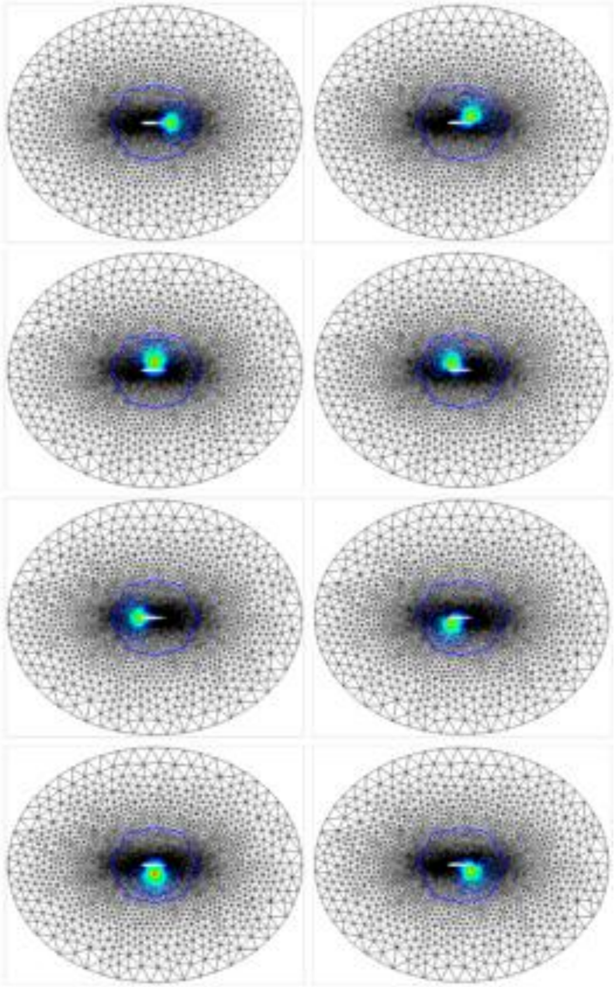


PR1 - Harmonic Coordinates in Aerodynamic ShpO (10/10)



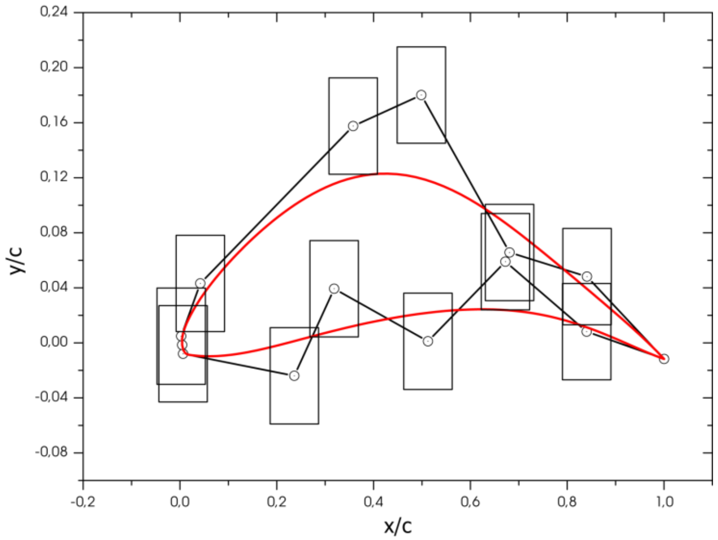


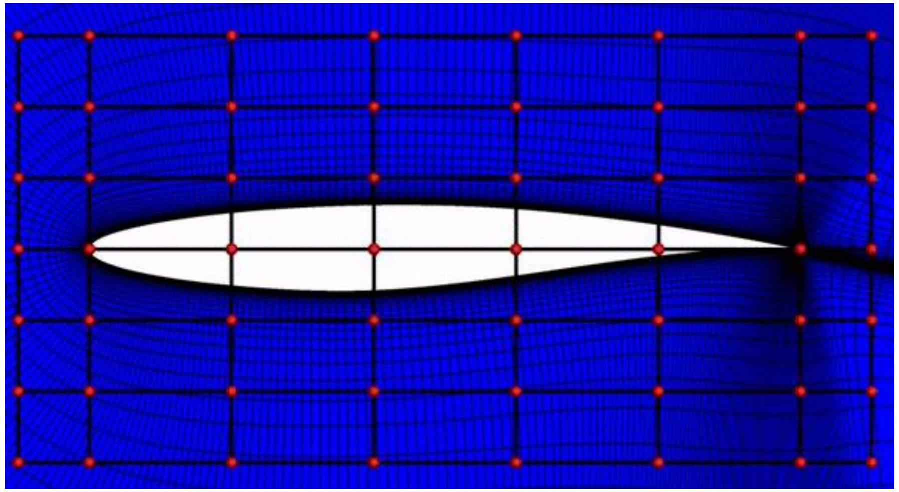




PR2 - Shape Parametrisation (1/2)

The selection of a CAD-based or CAD-free parametrisation affects the quality of shapes generated in the course of the optimisation & its convergence rate. Different parametrisations, with different sets of design variables (Degrees of Freedom, DoFs) b_n, n=1,…,N, may lead to different optimal solutions.

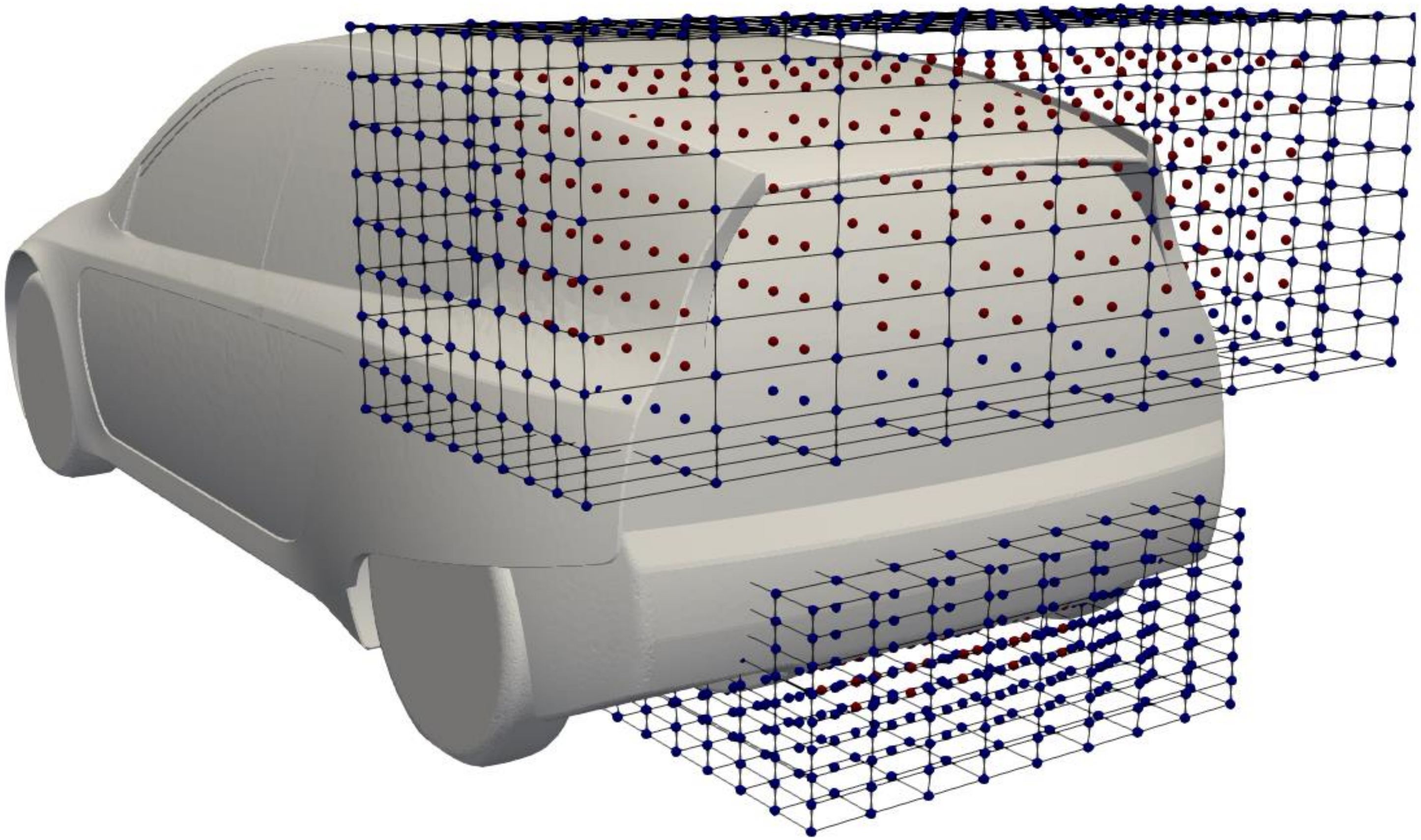




CAD-based Parametrisation: Using Bezier, NURBS, Splines etc patches and native CAD parameters. The coordinates of displaceable control points stand for the N design variables.

CAD-free Parametrisation: Using morphing techniques based, for instance, on volumetric B-Splines. Shape & CFD grid are simultaneously adapted. An alternative way is node-based parametrisation. Returning to CAD is difficult, possibly impairing the quality of the designed shapes.

PR2 - Shape Parametrisation (CAD-Free, for a Car) (2/2)



Symmetry?
Other constraints?

PR3 - Objective Function (1/3)

- Fastest
- Less expensive
- More comfortable
- Best regarding ecological driving
- Best regarding safety, etc.

- Objective Function (min. or max.)
- Cost Function (min.)
- Fitness Function (max.)

Is the selected objective function differentiable???

PR3 - About the Objective Function (2/3)

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

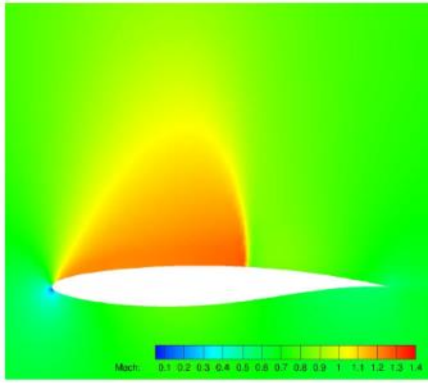
A two-objective problem to be solved using a MOO method:

max C_L
min C_D

min $-C_L$
min C_D

min $1/C_L$
min C_D

max C_L
max $-C_D$



Solve a MOO problem using a SOO method:

min C_D+ w/C_L

min C_D+ $1/C_L$

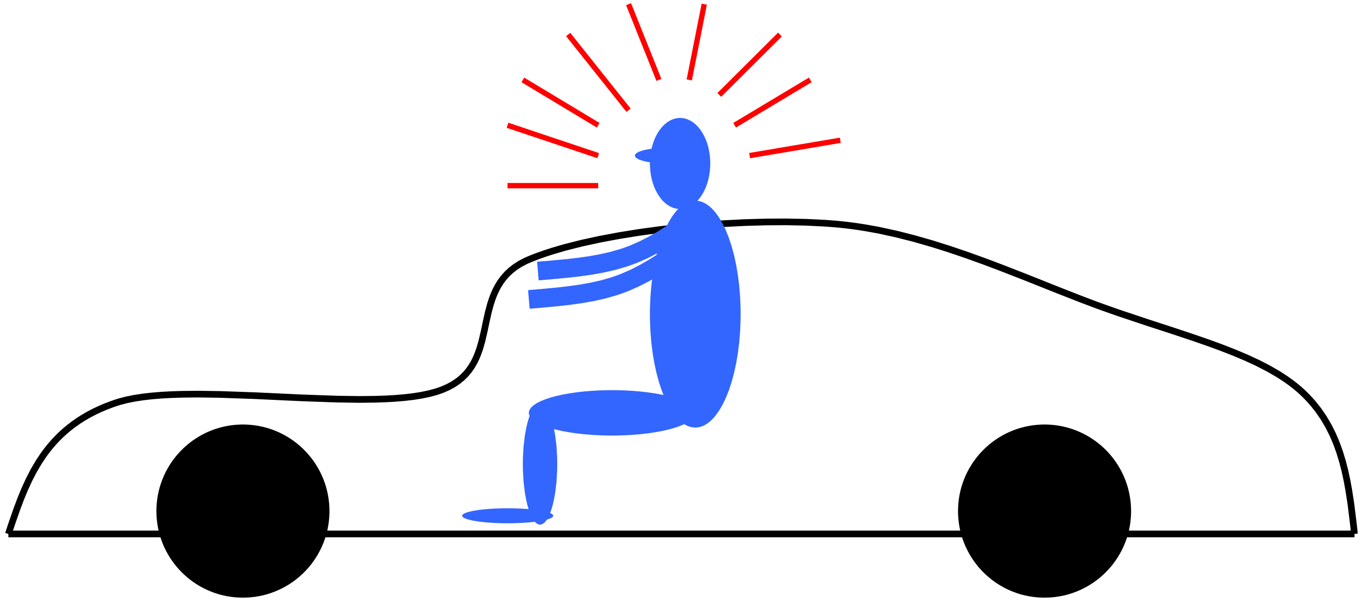
min C_D+ $10/C_L$

Reformulate it as a Constrained SOO problem:

min C_D
subject to: C_L=1.2

min C_D
subject to: C_L>1.2

PR3 – Constraints (3/3)



Infeasible solution

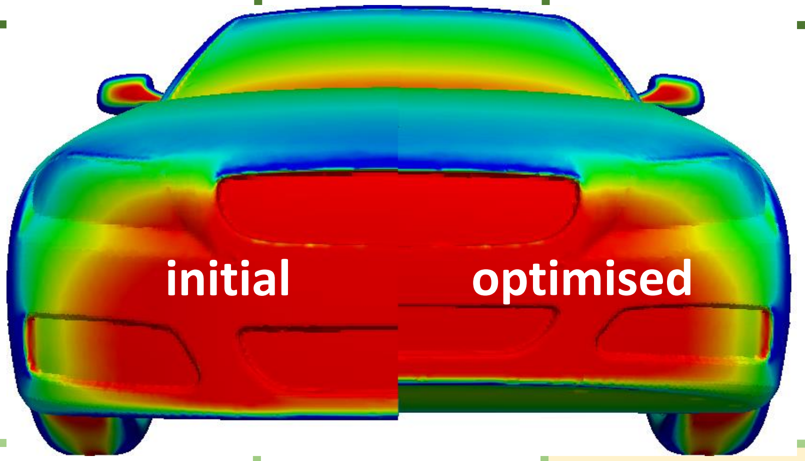
Prerequisites for running ShpO in CFD

PR1

Flow (CFD) Model (& s/w) incl. grid generation & grid displacement.

PR3

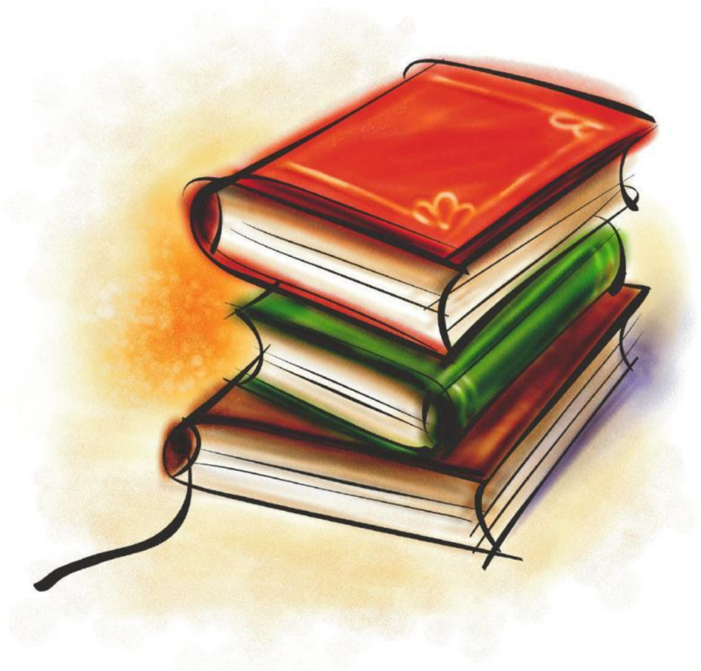
Objective & Constraint Function(s); gradient-based optimisation requires their derivatives w.r.t. b_n, n=1,…,N.



PR2

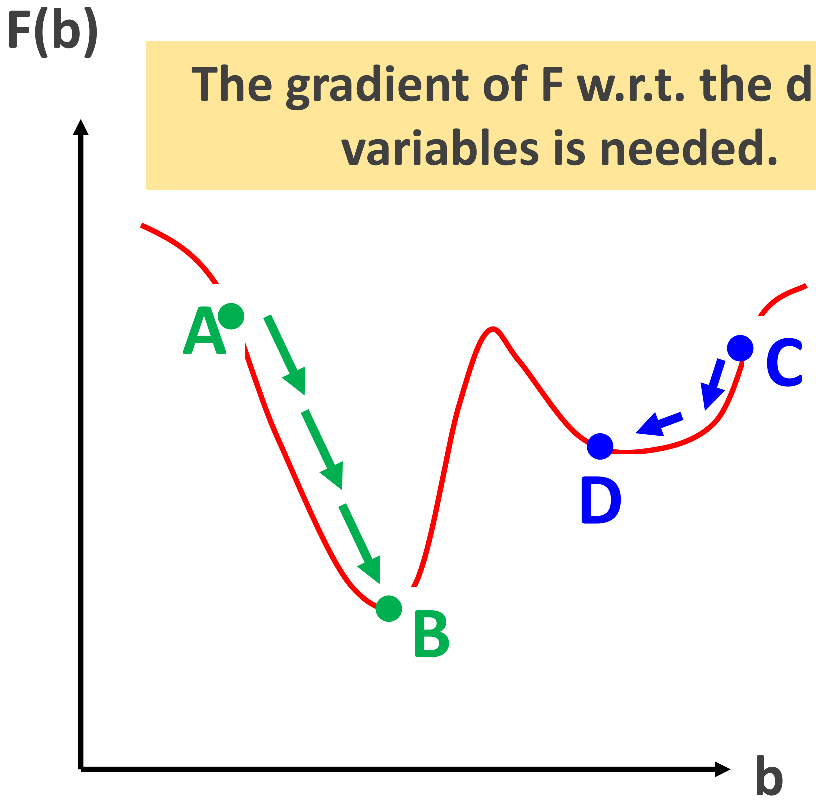
Case Parametrisation, defining the design variables (Degrees of Freedom, DoFs) b_n, n=1,…,N of the case.

Optimisation Method, incl. the computation of grad(…) if needed.

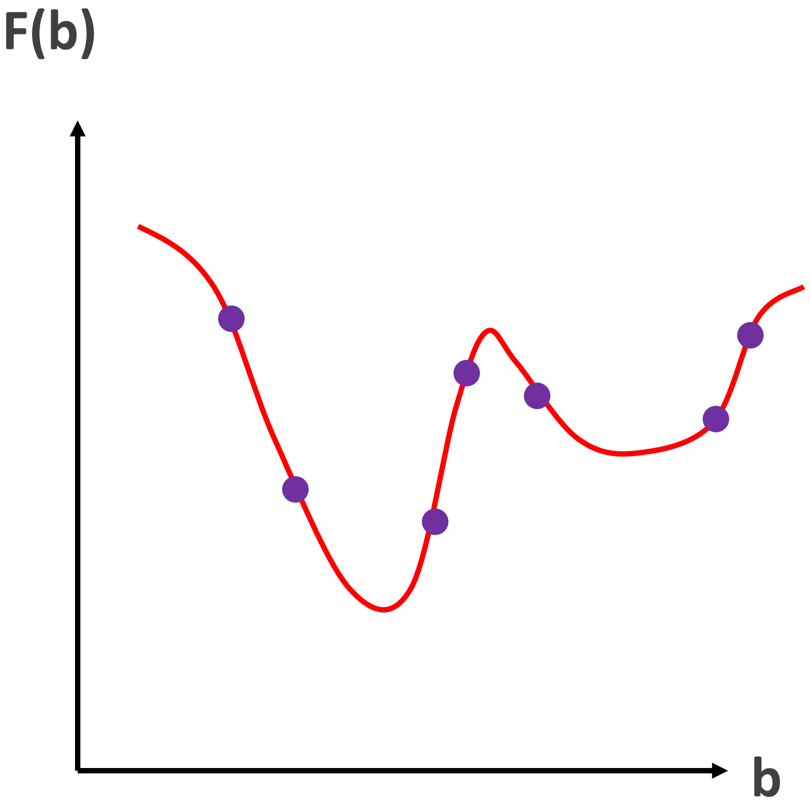


Gradient-Free (GFM) & Gradient-Based (GBM) Optimisation Methods

Classification of Optimisation Methods (1/3)



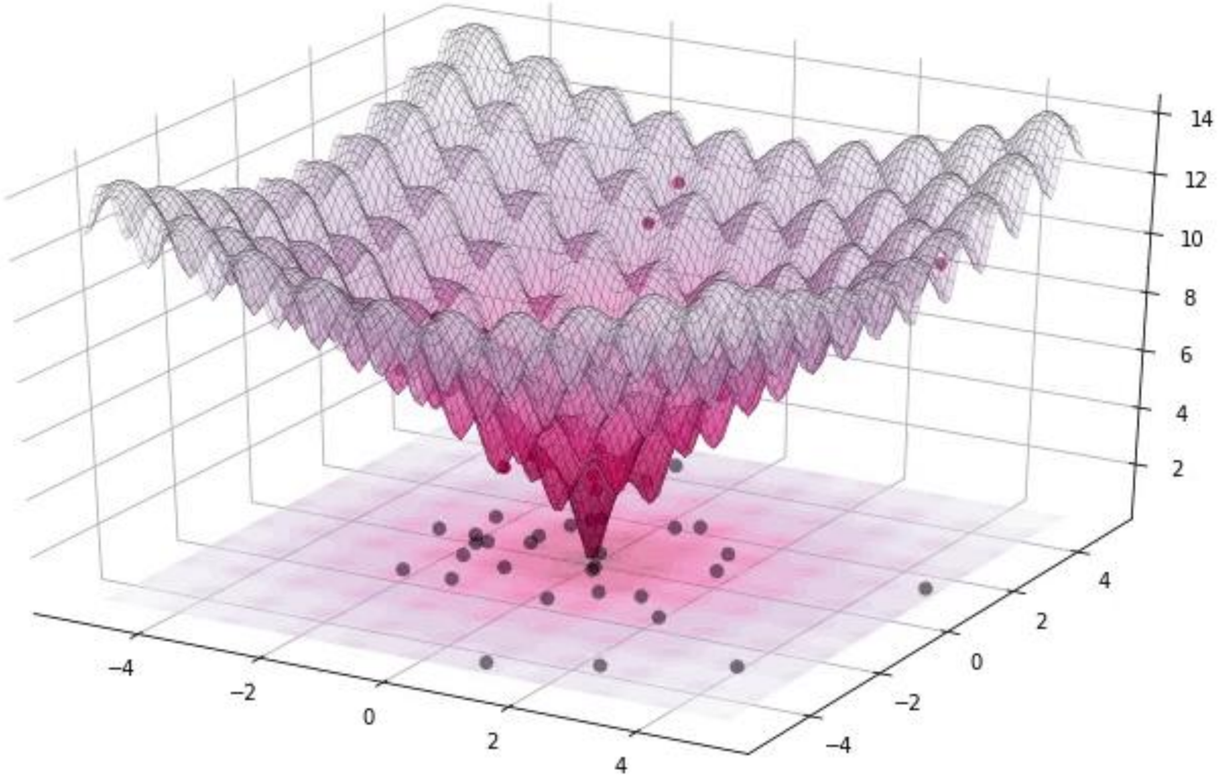
The gradient of F w.r.t. the design variables is needed.



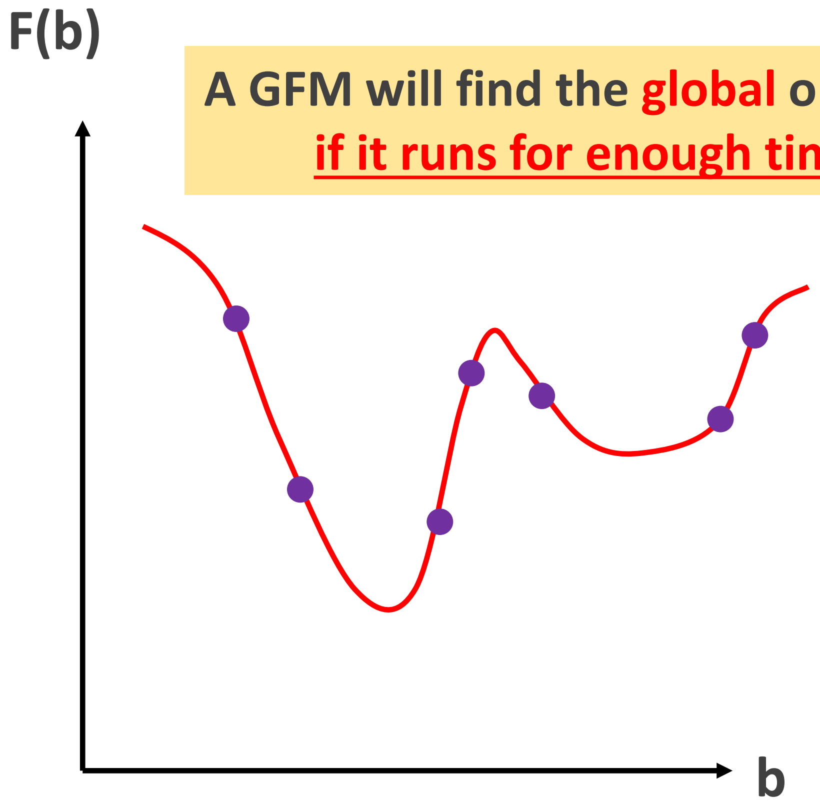
Gradient-Based Methods (GBM)

Stochastic (Gradient-Free, GFM) Methods

Classification of Optimisation Methods (2/3)



A GFM will find the global optimum if it runs for enough time!



Stochastic (Gradient-Free, GFM) Methods

Classification of Optimisation Methods (3/3)

Gradient-Based Methods (GBM)

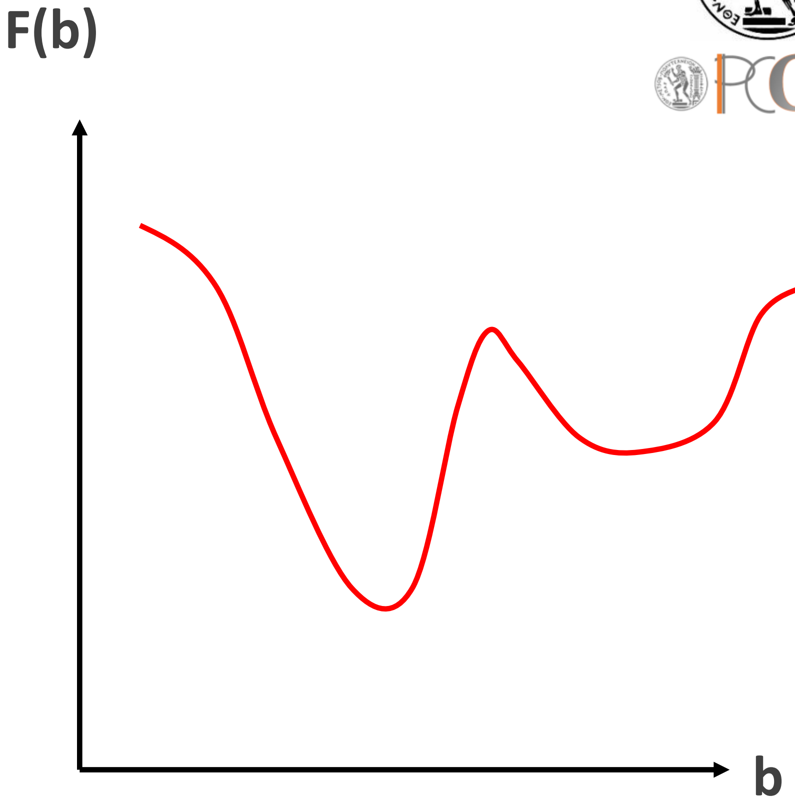
vs.

Stochastic (Gradient-Free, GFM) Methods

Individual-based Methods

vs.

Population-based Methods



GBM or GFM?

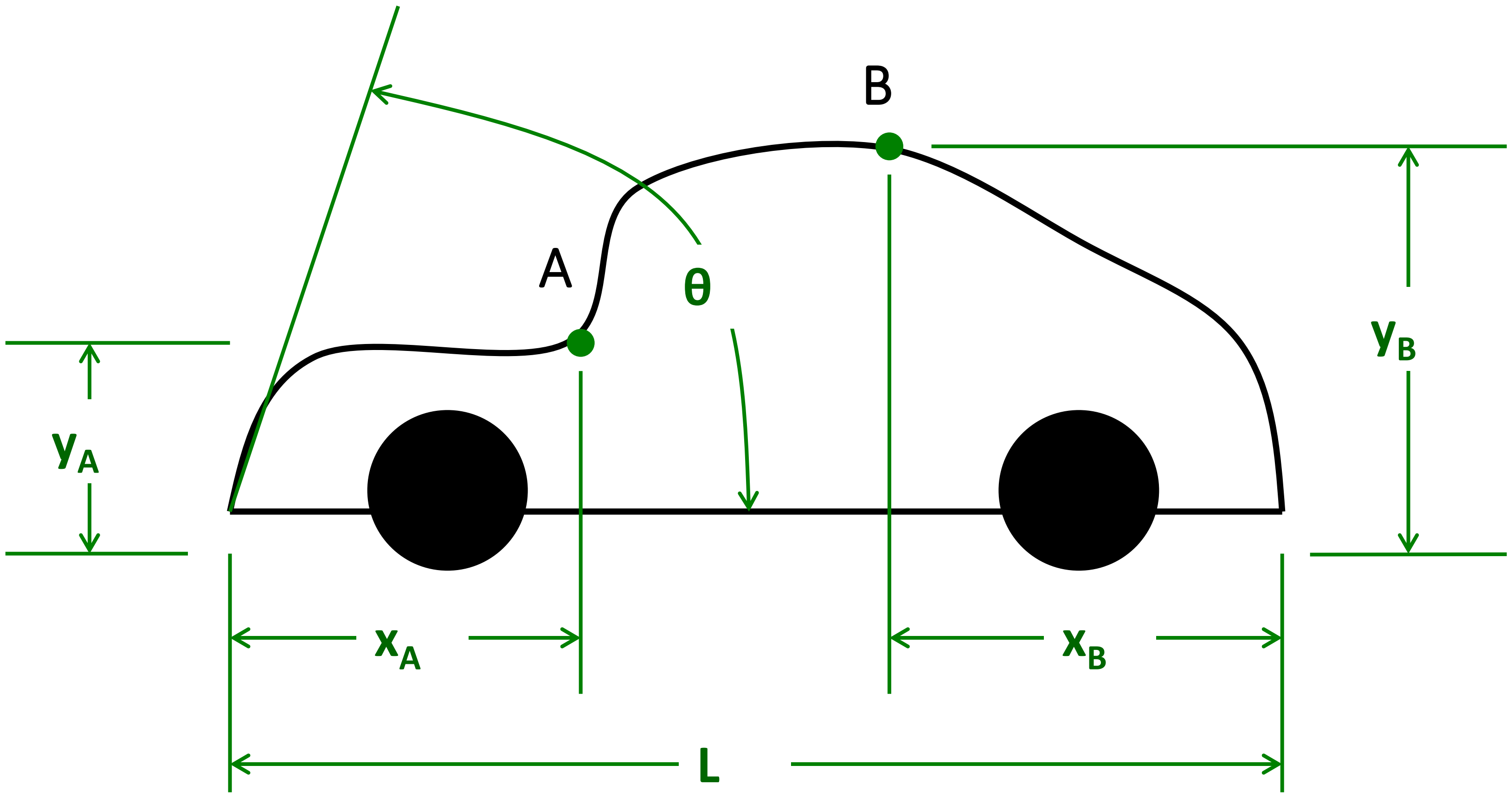
Criteria for deciding whether a gradient-based (GBM) or gradient-free (GFM) optimisation method must be selected, for a particular application, might be:

- A GFM can always be used! The computational cost, however, might become prohibitively high!
- To develop a GBM, the analysis/primal code must be available and the developer should be familiar with its mathematical background, discretization and numerical schemes, etc.
- For an optimisation that needs to be performed only once, use a GFM!
- If the optimisation method is used routinely, the GBM is by far the most efficient approach, provided that the development time/effort can be afforded.

Number of objectives (M_o) & constraints (M_C).

Number N of design or optimisation variables (degrees of freedom).

A Very Simple Working Example



N=6 degrees of freedom (DoFs), design or optimisation variables.

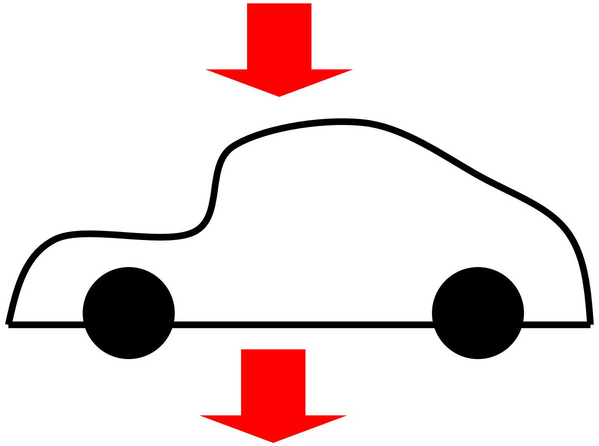
Objective (cost) function: min. drag (M_o=1). Let M_c=0.

Evaluation s/w: a CFD solver

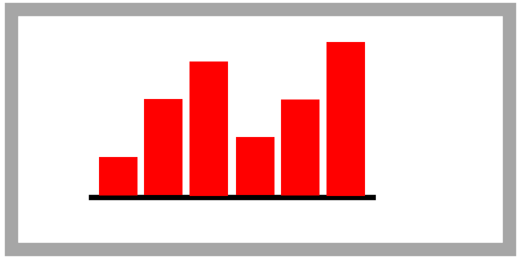
CFD evaluation

$\vec{b}$ =

L	x_A	y_A	x_B	y_B	θ
b_1=…	b_2=…	b_3=…	b_4=…	b_5=…	b_6=…



$F=C_D=…$



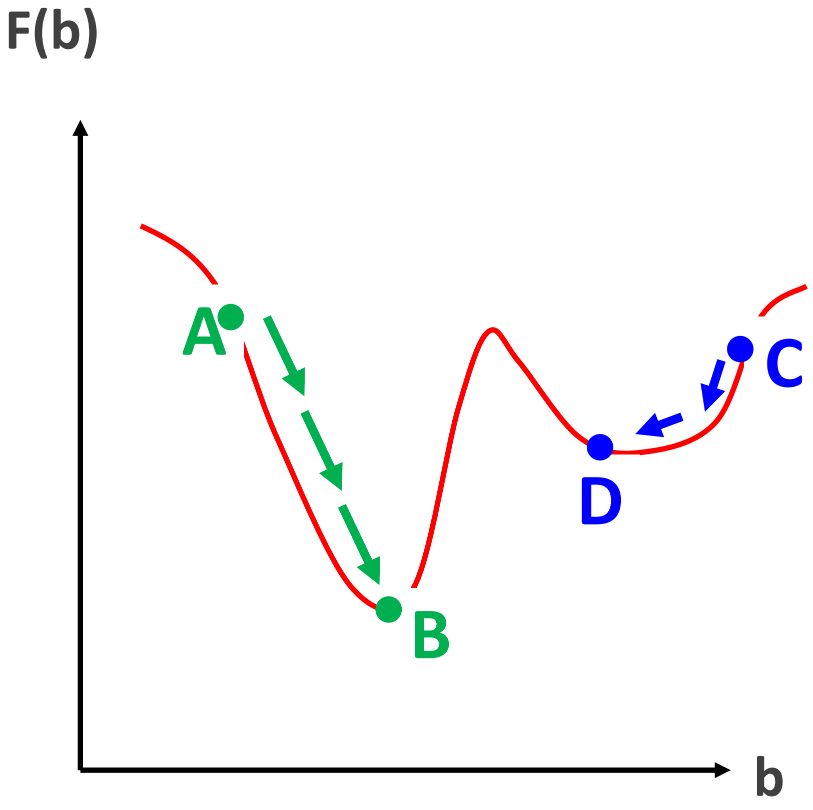
A typical way to measure the computational cost of the optimisation run is in terms of the number of calls to the eval.exe – one time unit=1TU).

Gradient-based (Deterministic) Optimisation Methods (GBMs) (1/4)

For a SOO problem (or a MOO one cast in the form of a SOO) (min. F) with N design variable.:

Having computed the gradient of F:

$$\nabla F = \left(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \ldots \frac{\partial F}{\partial b_N}\right)$$



Gradient-based (Deterministic) Optimisation Methods (GBMs)

Steepest Descent

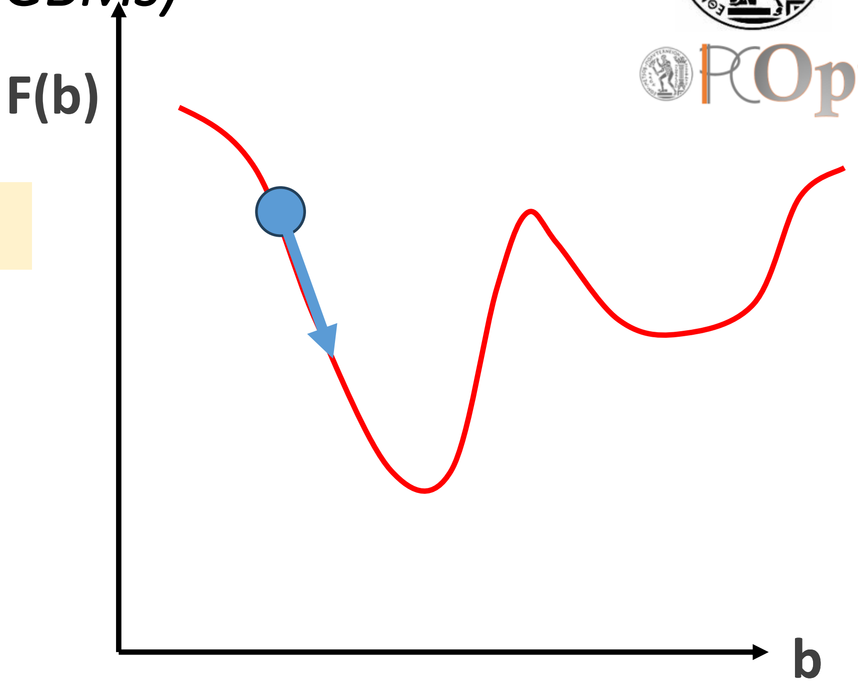
$$\vec{b}^{\,new} = \vec{b}^{\,old} - \eta \nabla F(\vec{b}^{\,old})^T$$

about η

$$b_n^{new} = b_n^{old} - \eta \frac{\delta F}{\delta b_n}^{old}, \qquad n = 1, \cdots N$$

Newton's Methods

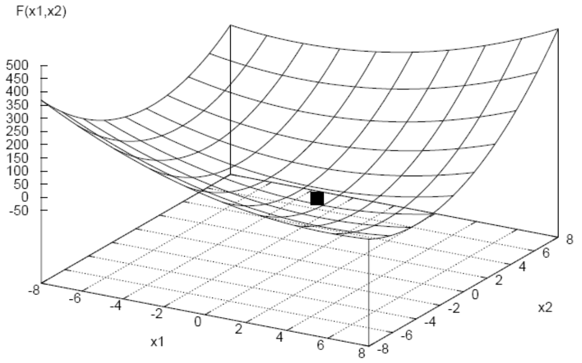
$$\vec{b}^{\,new} = \vec{b}^{\,old} - \nabla^2 F(\vec{b}^{\,old})^{-1} \nabla F(\vec{b}^{\,old})^T$$

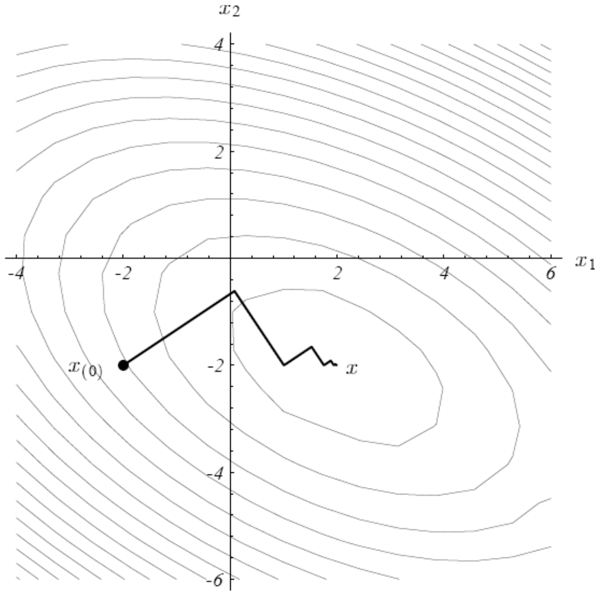


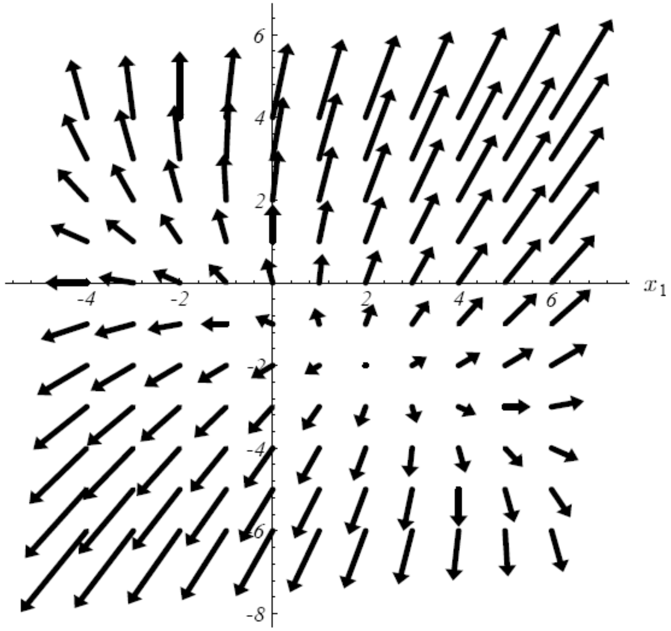
Compute (the exact) or approximate grad(F)? Accuracy vs. Computational cost.

Computing the exact Hessian! Quasi-Newton methods!

Gradient-based (Deterministic) Optimisation Methods (GBMs) (3/4)







- Risk to result to a local, rather than the global, optimum.
- Optimal solution heavily depending upon the initialisation. Many restarts?
- Expected to be fast, if the computation of Grad(F) isn’t time-consuming.

Possible Ways to Compute Grad(F) (4/4)

In a problem governed by PDEs (or linear systems of equations):

- **Finite Differences (FD)**
- **Complex Variable (CV) Method**
- **Direct Differentiation (DD)**
- **Automatic or Algorithmic Differentiation (AD)**
- **Adjoint Method (AM)**

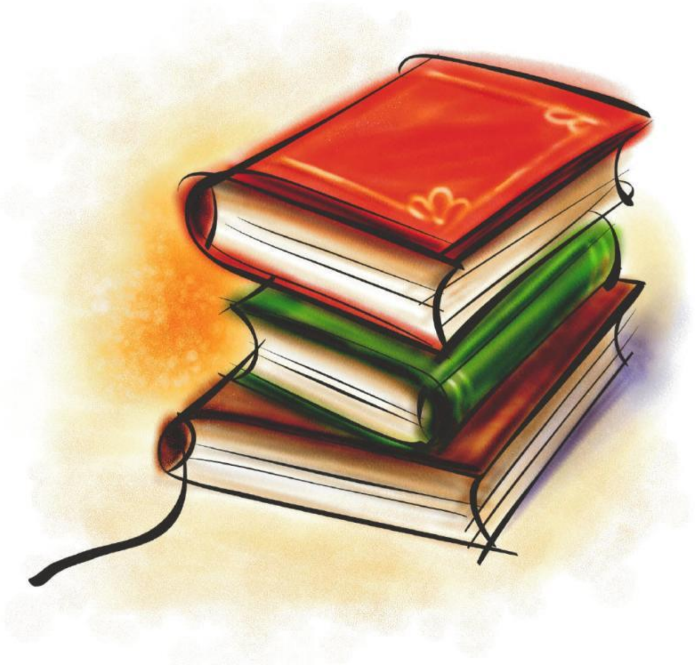
The computation of the gradient of the objective and/or constraint functions is the most important and costly part of a GBM. Its cost should be measured in terms of the cost of solving the analysis problem (one time unit, 1 TU or 1 EPS= Equivalent Problem Solution).

When GFMs should be used?

- If the evaluation software (for instance, the CFD code) is not available as open-source.
- If there is no available method/software to compute gradients.
- If there is no way to develop such a software.
- If computational cost can be afforded (for instance, solving the optimisation problem just once!)
- If the optimisation problem does not have a great number of unknowns!

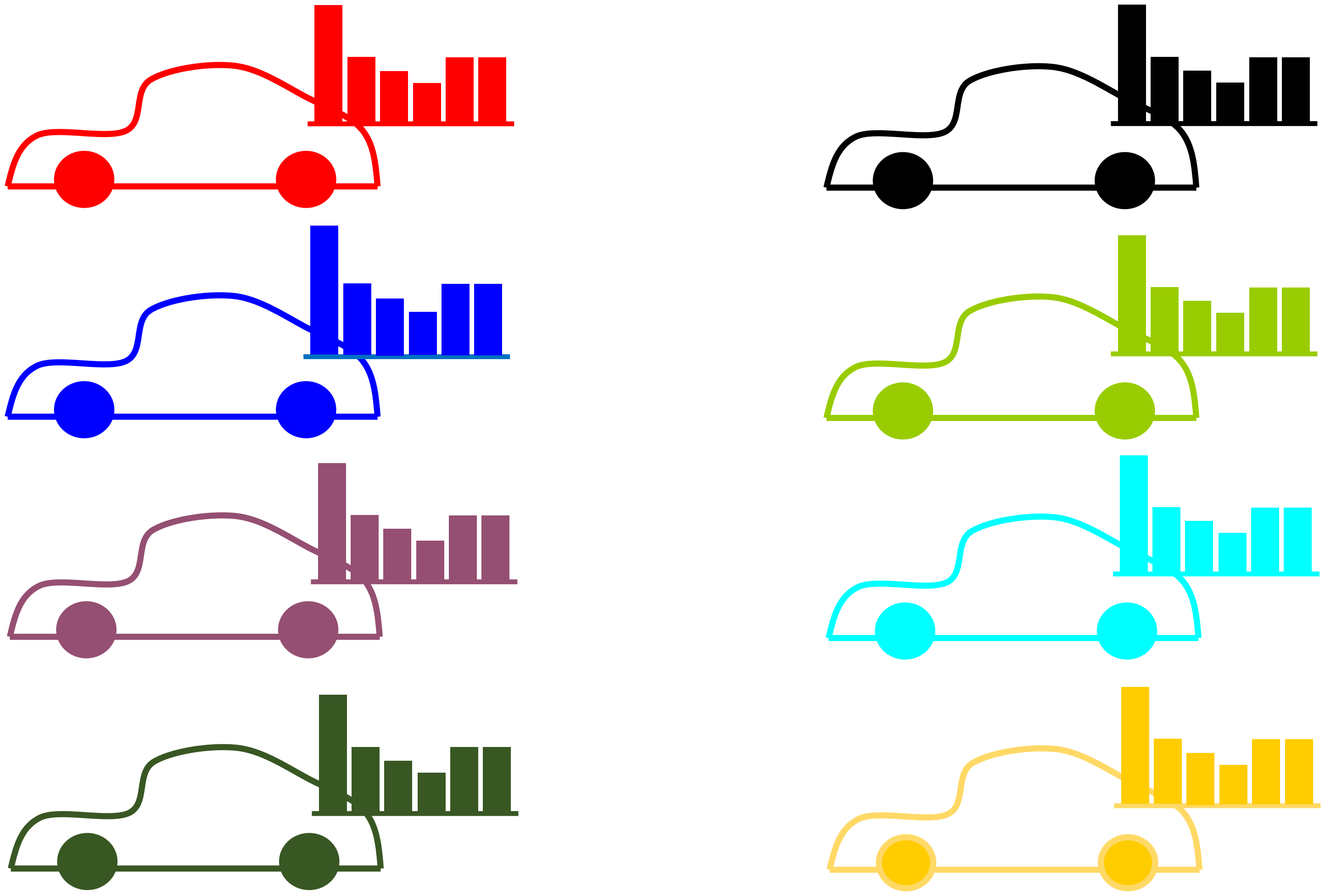
Should this be the case, continue with: **Gradient-Free Methods (GFMs)**, such as Evolutionary Algorithms (EAs), Particle Swarm Optimisation (PSO), Differential Evolution (DE), Covariance Matrix Adaptation (CMA) & other bio-inspired optimisation methods.

In any opposite case, try with **Gradient-Based Methods (GBMs)**, that is expected to have a long development period (if the gradient computing tool is not available to you) but, once this is done, the optimisation itself is expected to be much faster!



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

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



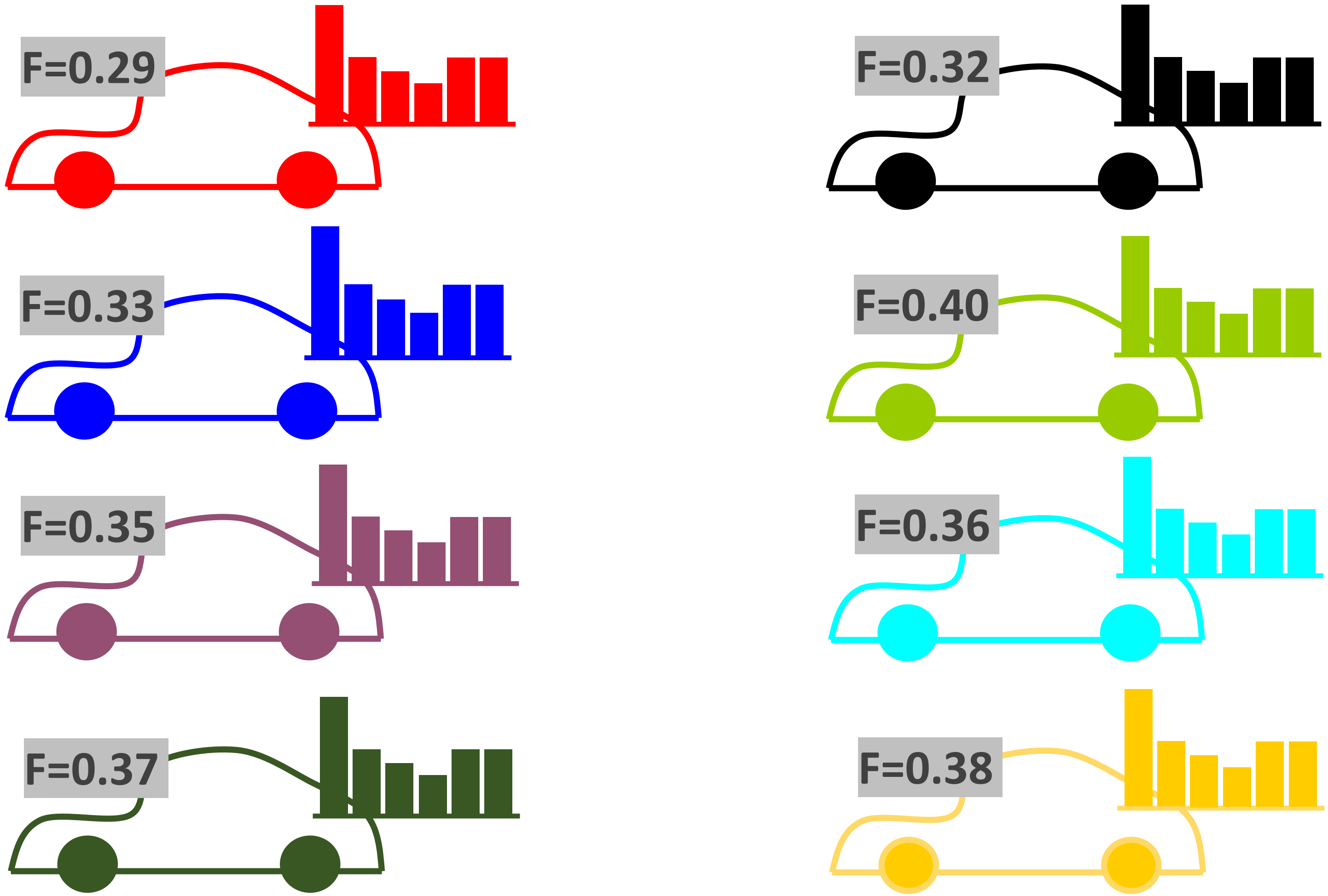
(μ,λ) Evolutionary Algorithm in a minimisation problem (min. F)

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 simplification!)

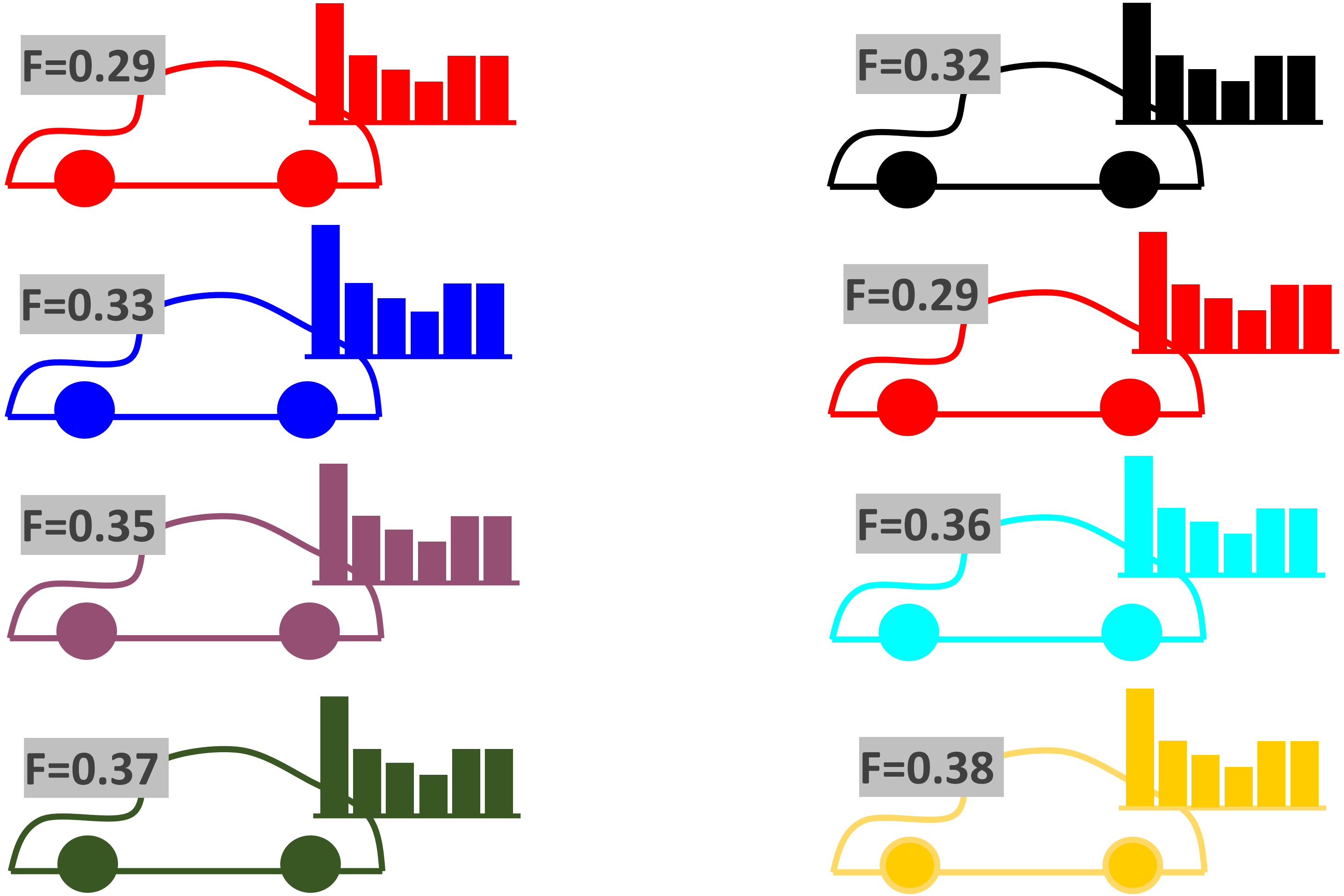
A simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (2/8)



Evaluate the offspring population.
Cost: λ (=8) calls to eval.exe.
min. F=C_D

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

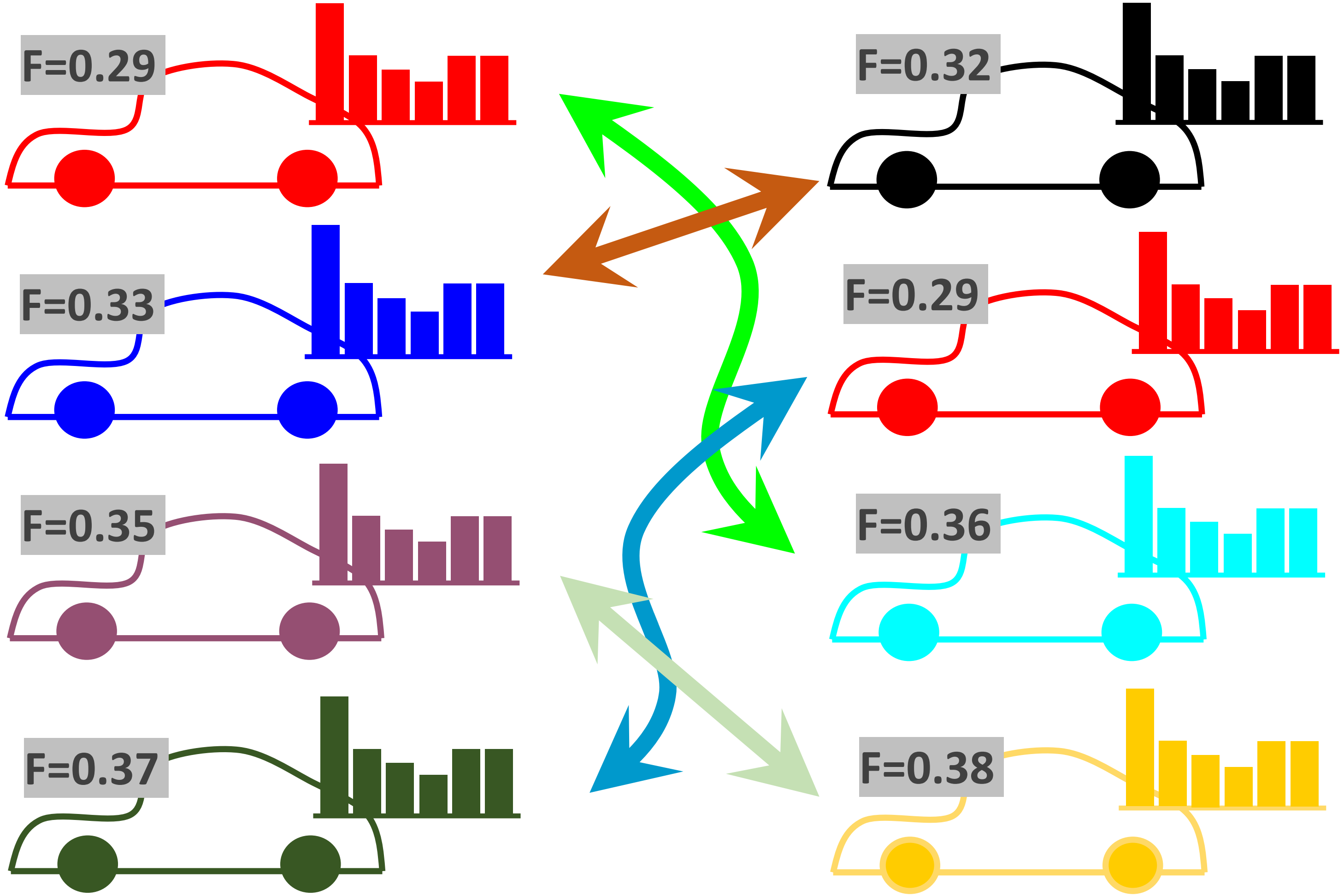
A simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (3/8)



Select parents: Replace the worst performing individual with the best.

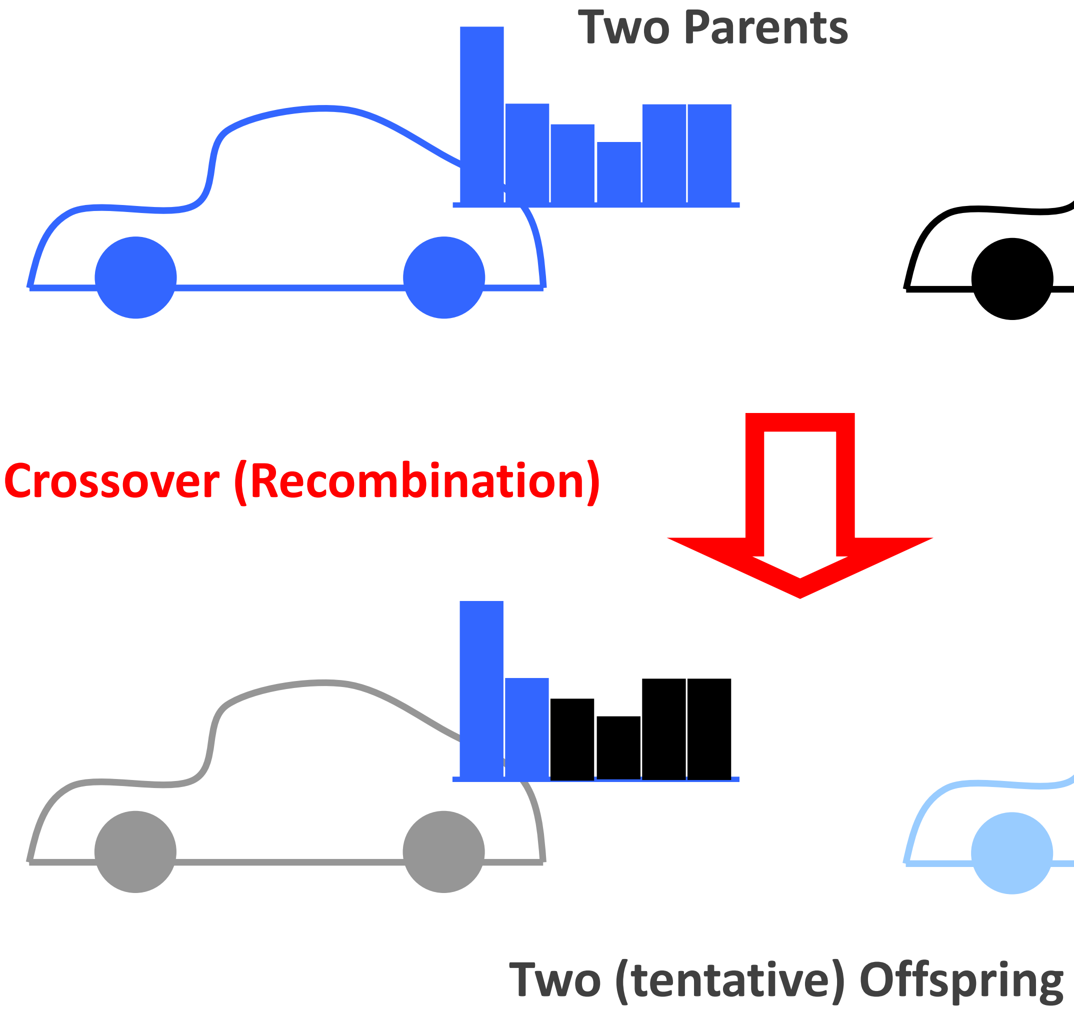
The population of μ parents will be used to create new λ offspring.

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



Randomly **Mate** them.

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

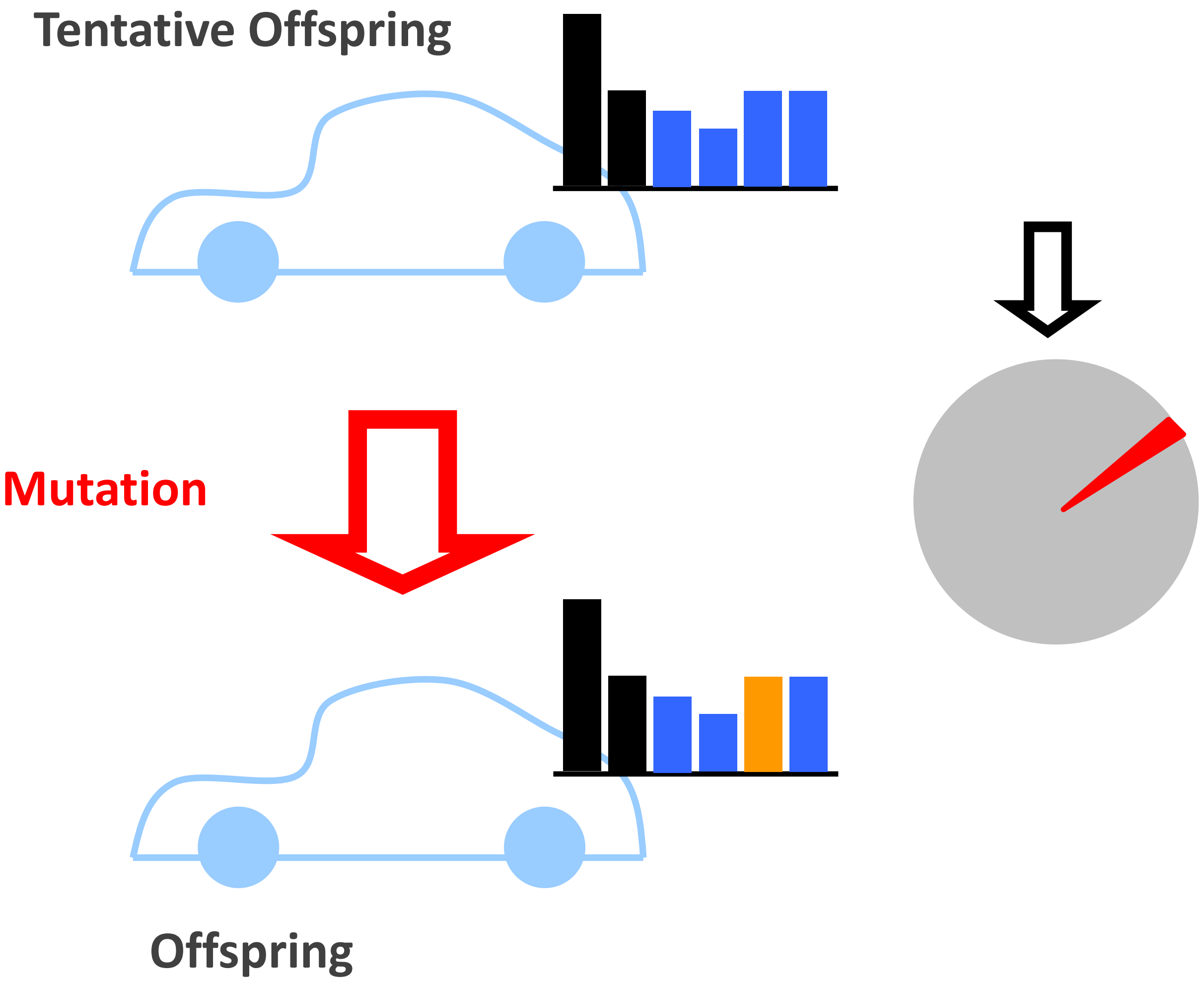


Apply Crossover (evolution operator) on the selected pairs of parent individuals.

(Based on a Random Number Generator or RNG)

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

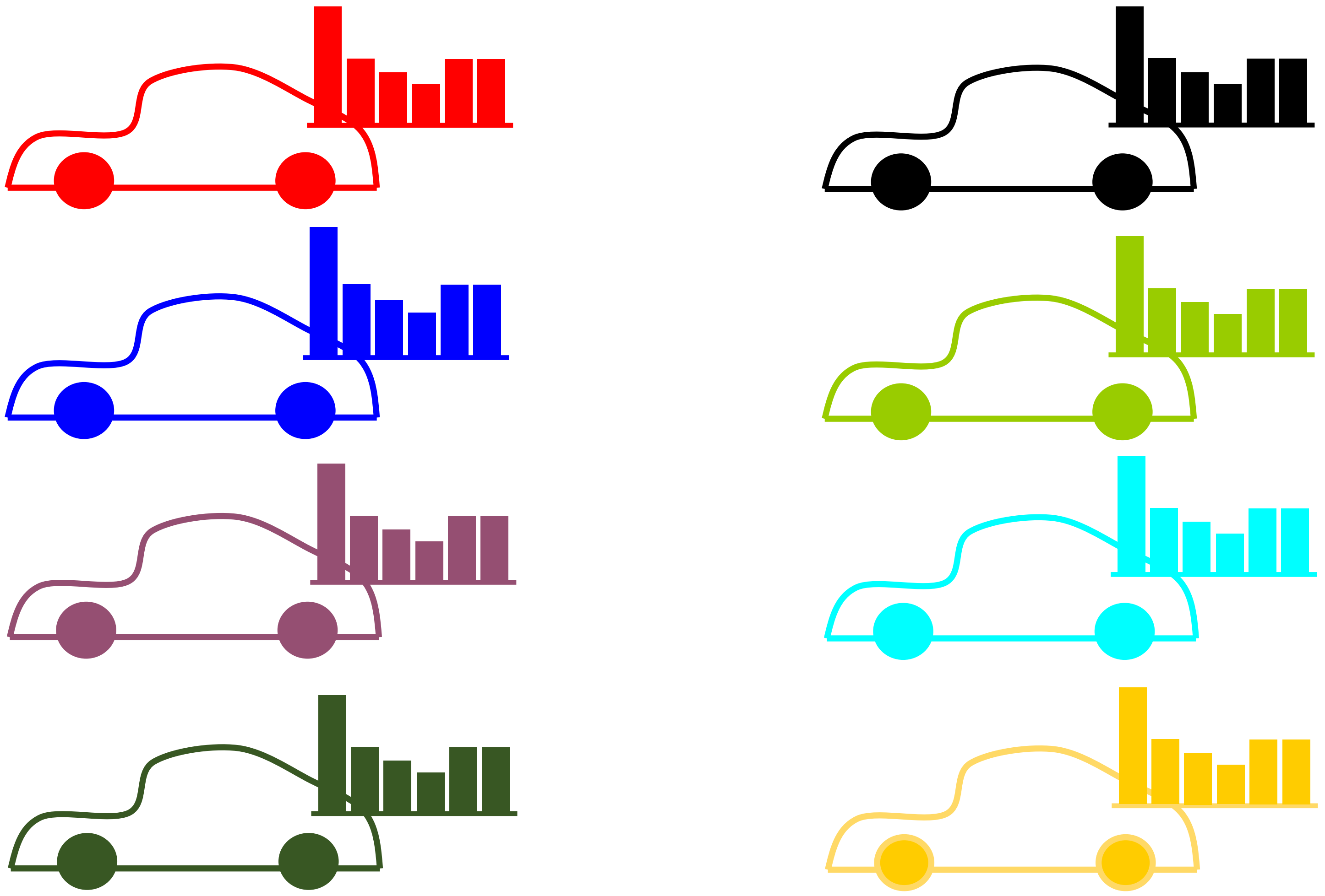
A simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (6/8)



Apply **mutation** (evolution operator).

+ Elitism!

A 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) should keep all evaluated individuals paired with their objective function values, to avoid repeating the same evaluations in future generations (and, thus, save computational cost).

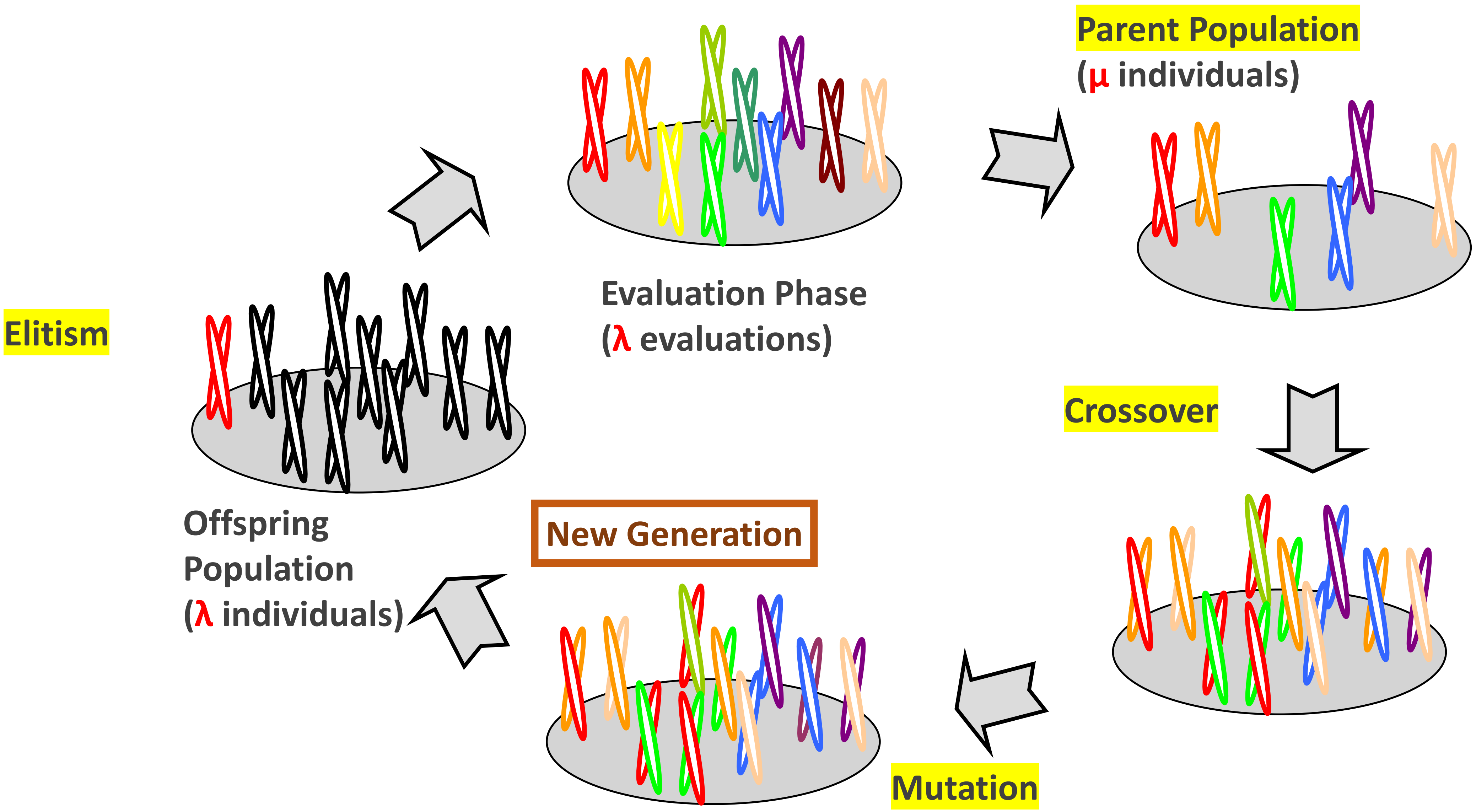
A simple example of a (μ,λ)EA with μ=8 parents & λ=8 offspring (SOO) (8/8)

Termination Criteria

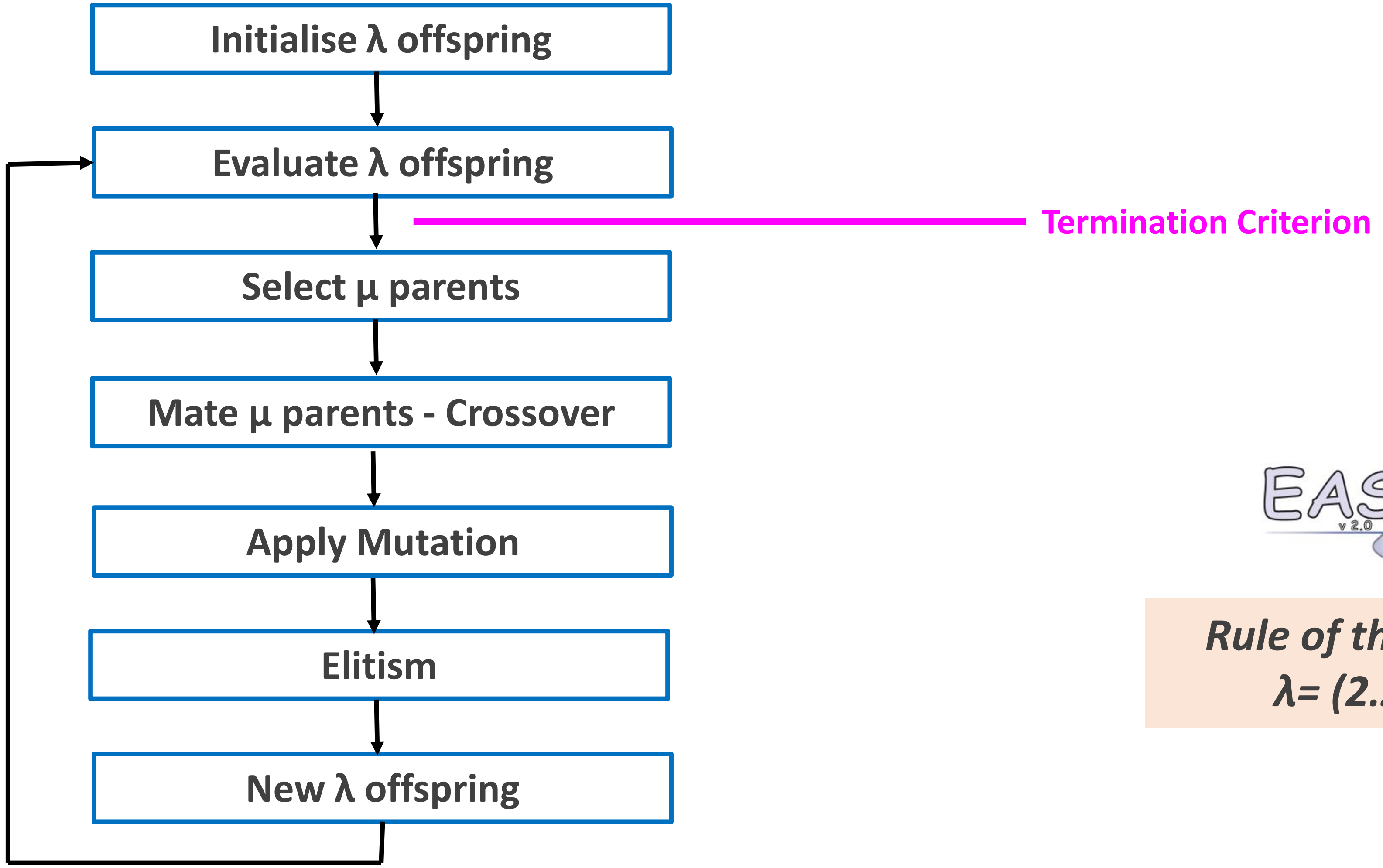
Terminate the evolution when:

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

The (μ,λ) EA


Parent Population
(μ individuals)
Evaluation Phase
(λ evaluations)
Elitism
Crossover
Offspring
Population
(λ individuals)
New Generation
Mutation

The (μ,λ) EA for SOO



Initialise λ offspring

Evaluate λ offspring

Select μ parents

Mate μ parents - Crossover

Apply Mutation

Elitism

New λ offspring

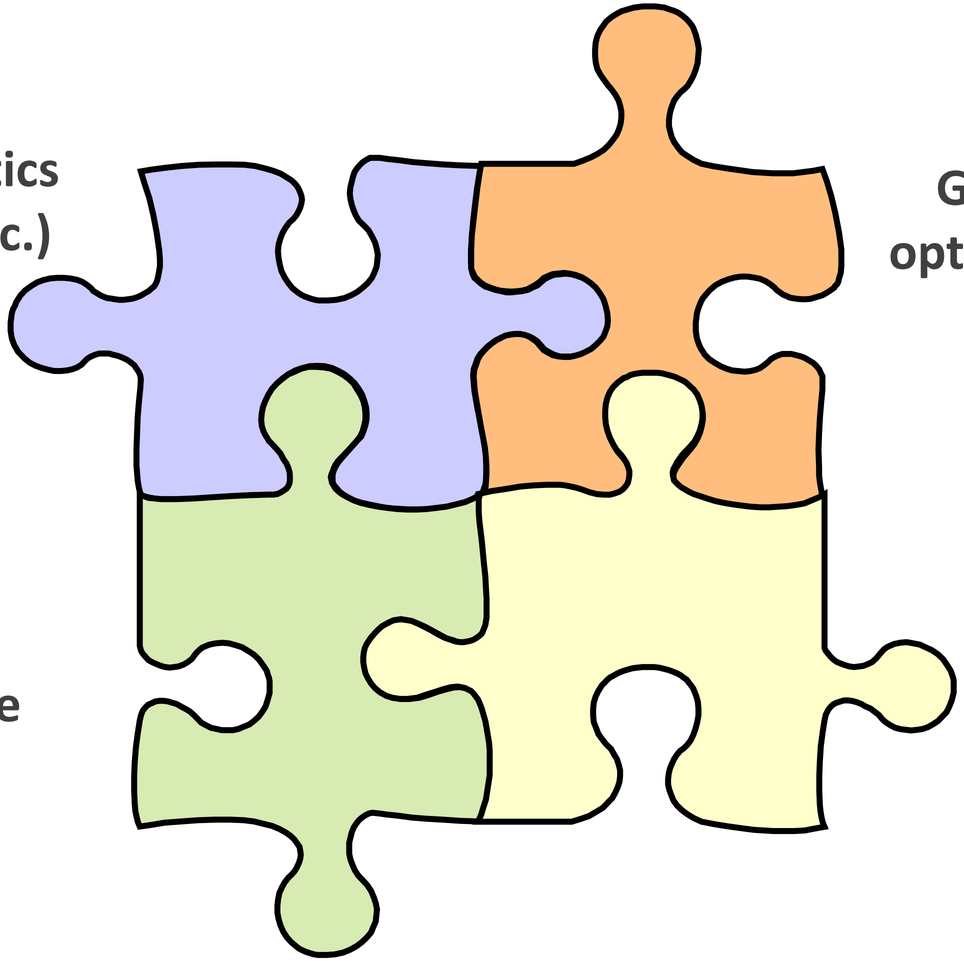
Termination Criterion

Rule of the Thumb:
λ= (2…4) · μ

Hybrid Optimisation Methods

Meta-Heuristics (GFM, EAs etc.)

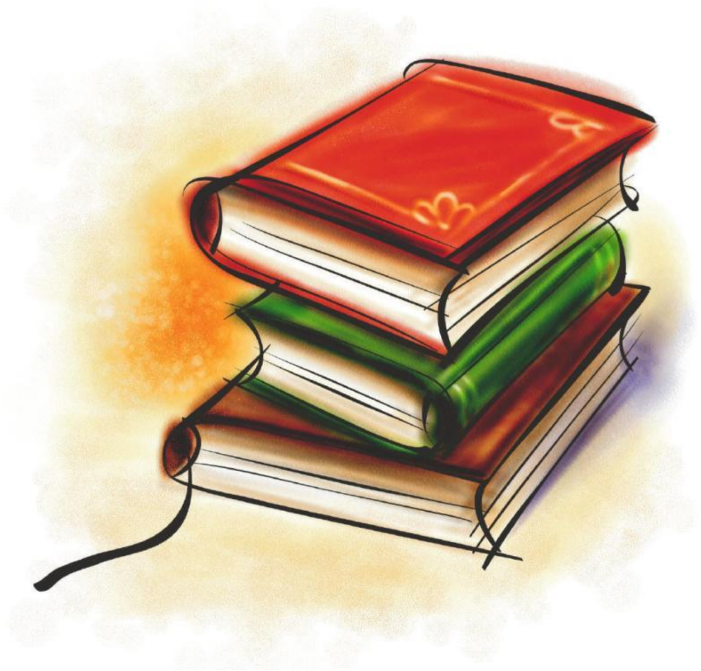
Gradient-based optimisation (GBM)



Artificial Intelligence

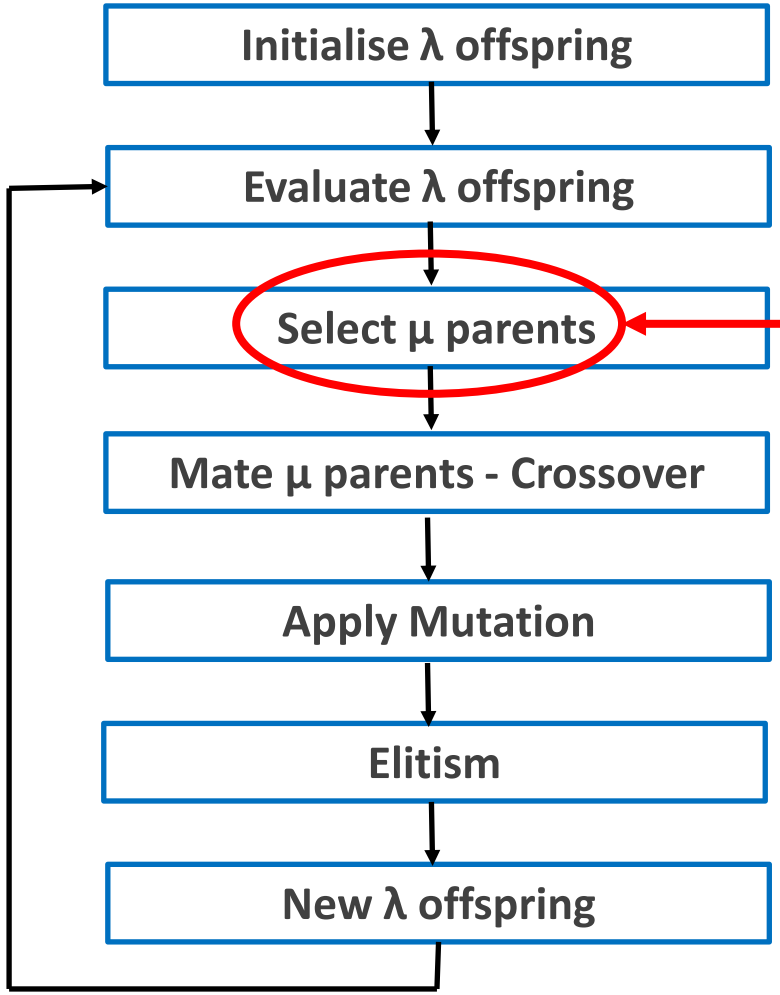
Parallel Computing

Parallel Hybrid Optimisation Methods



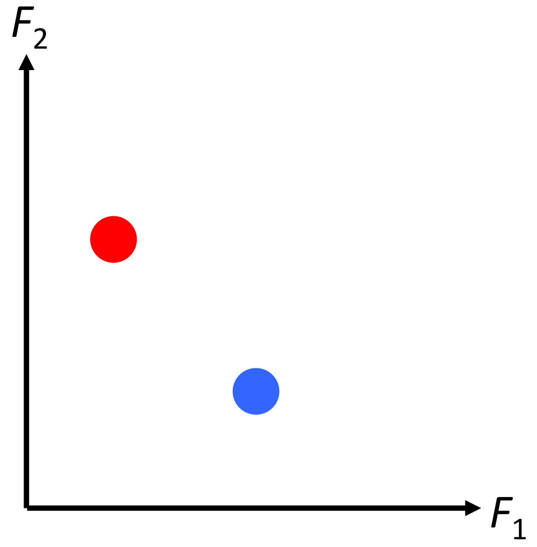
The (μ,λ) EA for Multi-Objective Optimisation (MOO)

The (μ,λ) EA for MOO

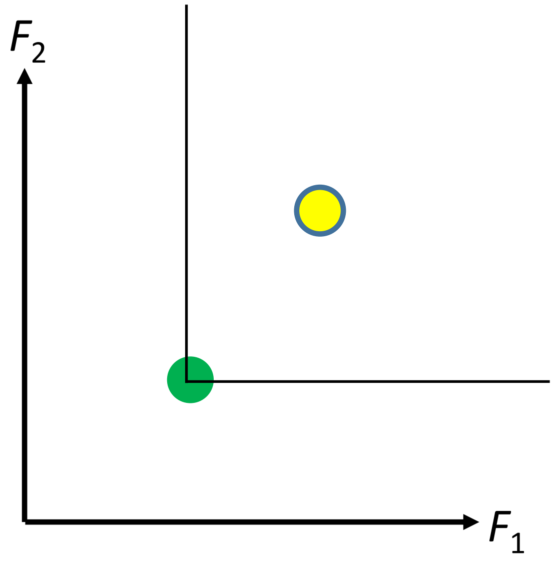


Transform vector (F_1,…, F_{Mo}) to a **scalar utility function Φ** (dominance driven) before selecting μ parents based on **Φ** values

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



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

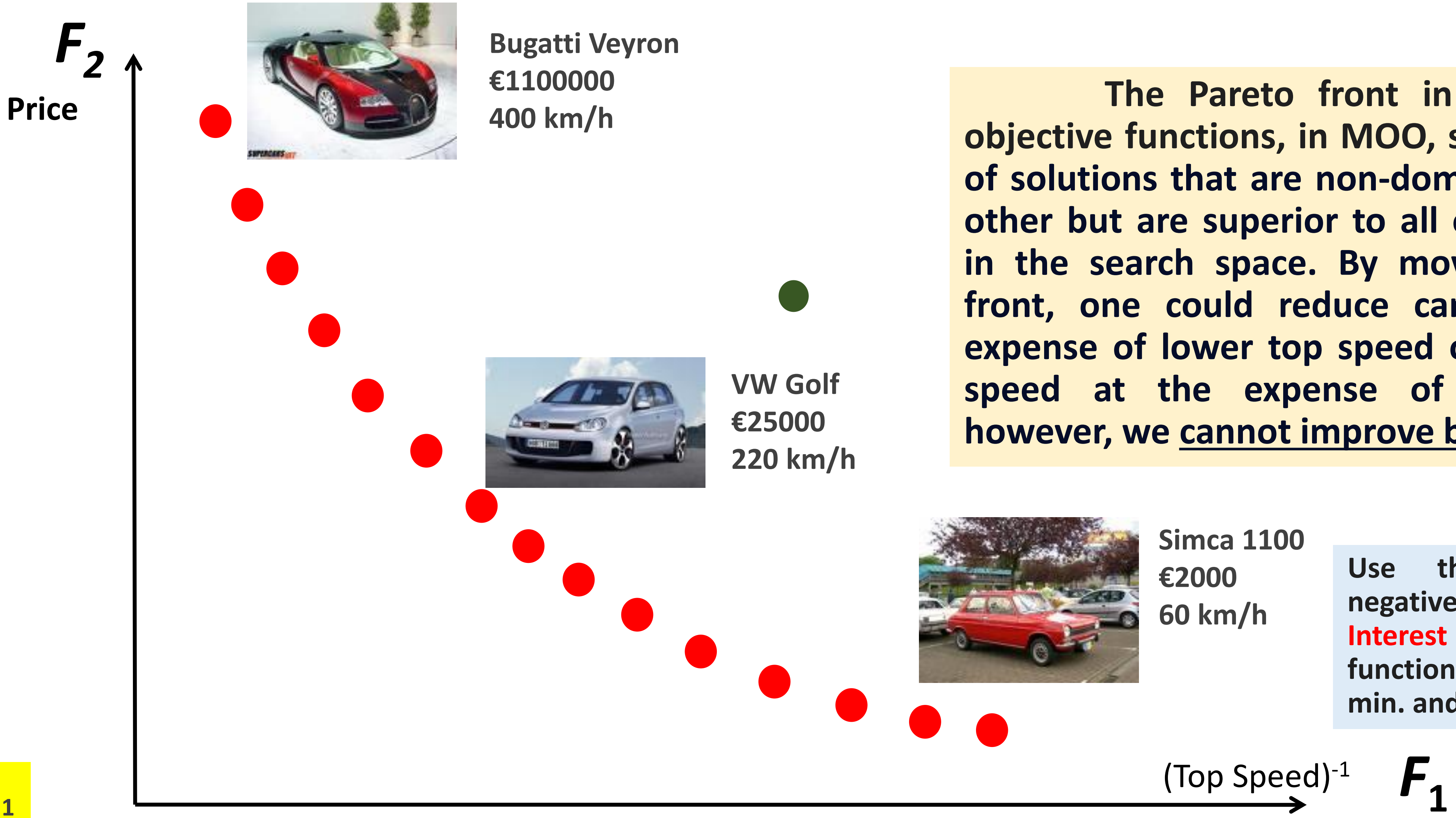


The yellow solution is dominated by the green one. Or, a point A dominates B, if A is better or equal in all criteria and strictly better in at least one.

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

Min. F_1
Min. F_2

MOO – The Pareto front (1/2)



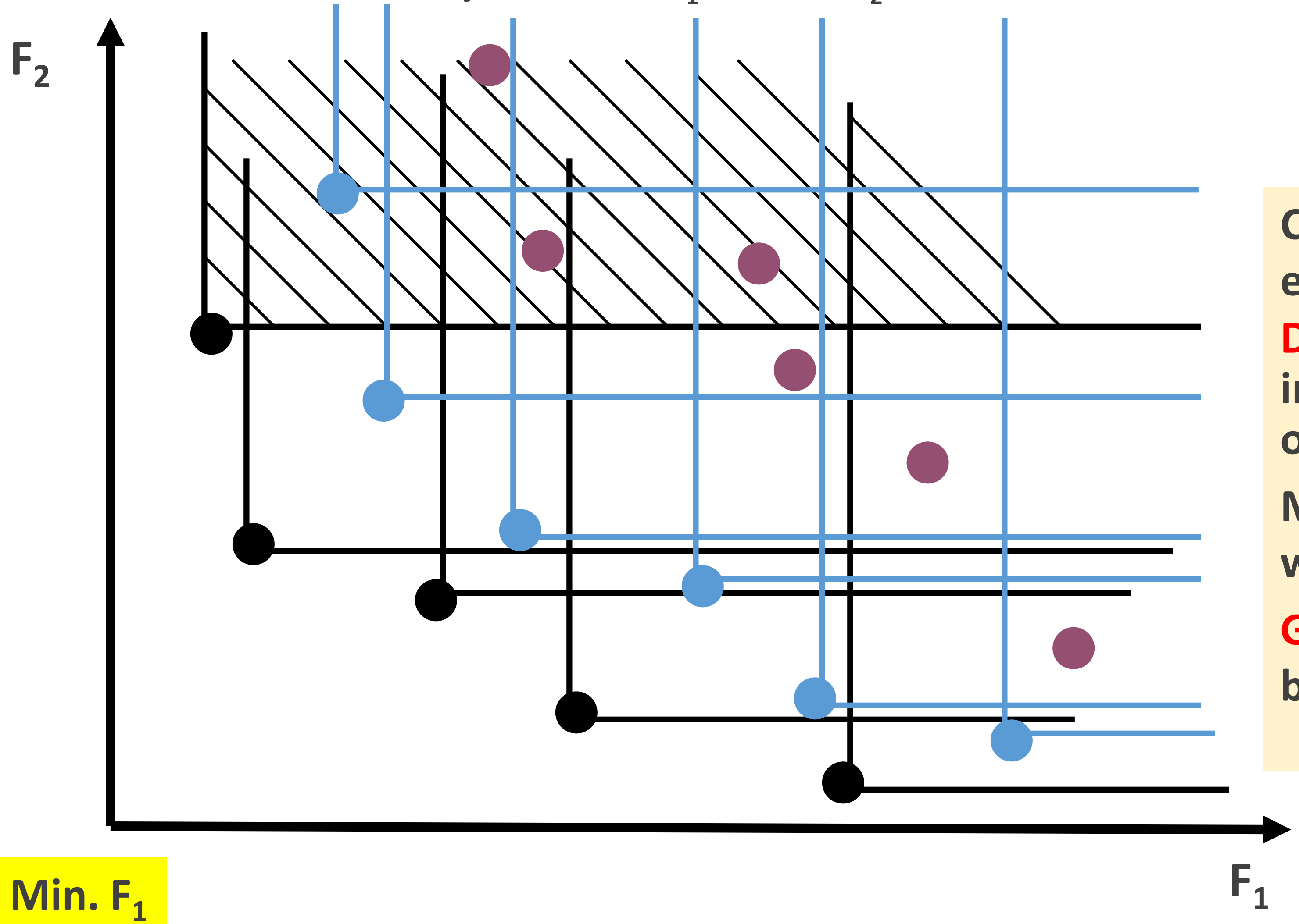
The Pareto front in the space of objective functions, in MOO, stands for a set of solutions that are non-dominated to each other but are superior to all other solutions in the search space. By moving along the front, one could reduce car price at the expense of lower top speed or increase top speed at the expense of higher price; however, we 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.

Min. F_1
Min. F_2

MOO – The Pareto front (2/2)

Objectives: min F_1 and min. F_2.



● Members of the Pareto front or front of non-dominated solutions.

Computing the (whole?) Pareto front is more expensive than a single optimal solution.

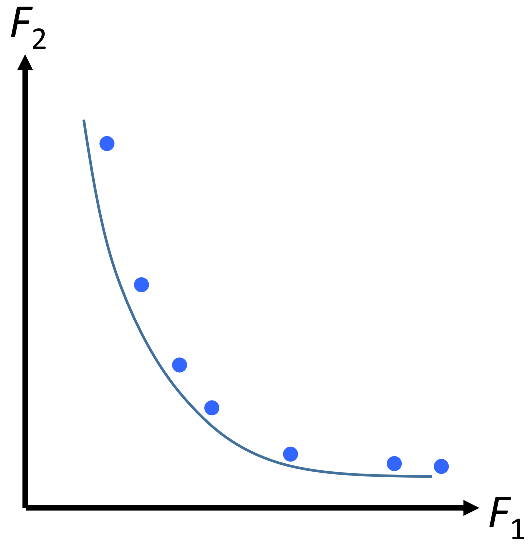
Decision making on the Pareto front, by involving more criteria than those used in the optimization, may follow.

Most GFMs (such as an EA) may compute the whole front in one shot. Cost?

GBMs computing Pareto fronts have recently been developed.

Min. F_1
Min. F_2

Comparing Fronts of Non-Dominated Solutions (1/2)

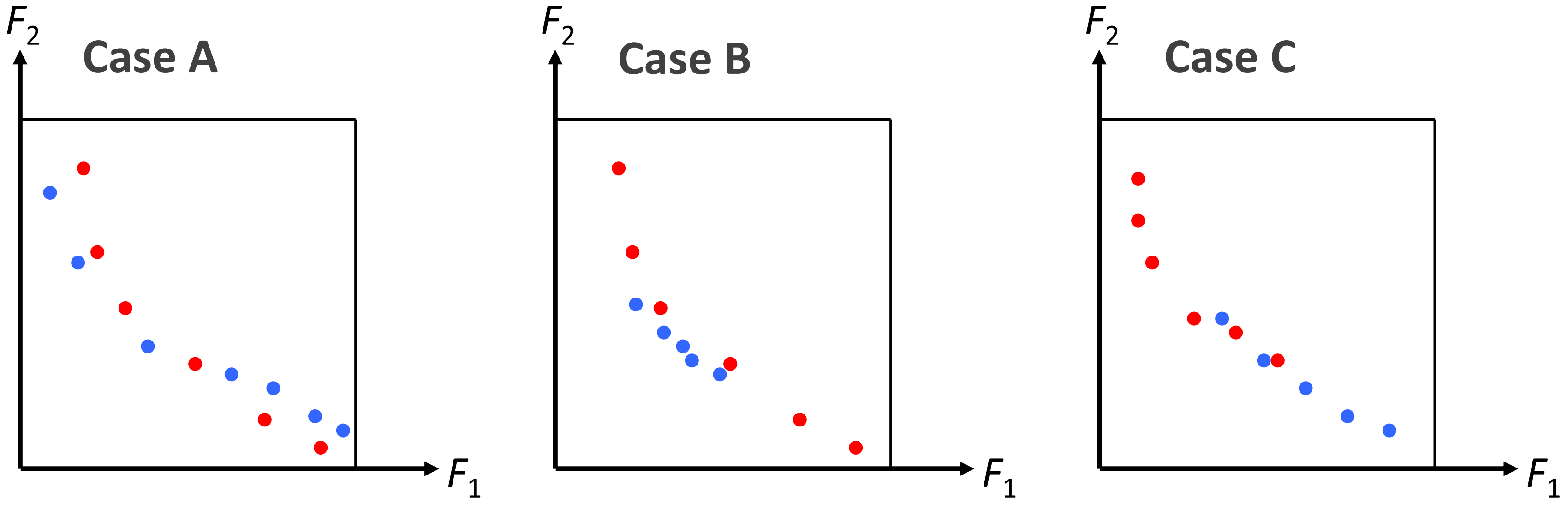


A MOO method seeks a discrete approximation to the Pareto front.

Pareto front vs. Front of Non-Dominated Solutions.

Min. F_1
Min. F_2

Comparing Fronts of Non-Dominated Solutions (2/2)



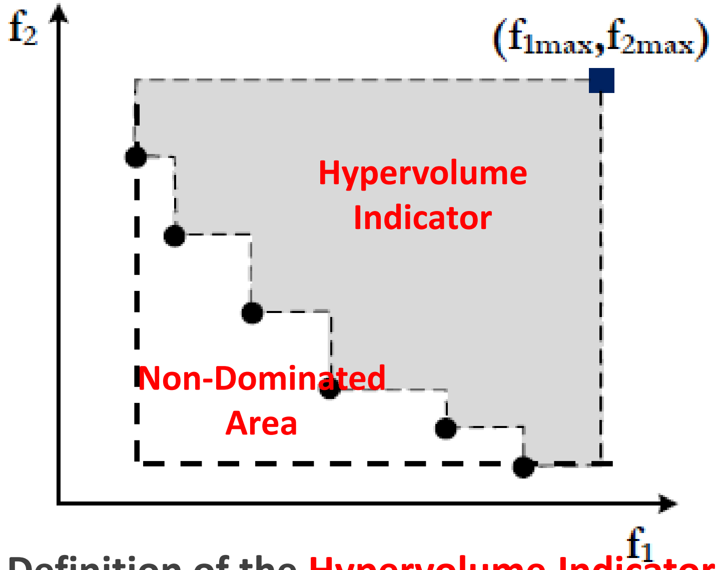
Comparison of computed fronts of non-dominated solutions

Visualisation/interpretation issues if M_o>3.

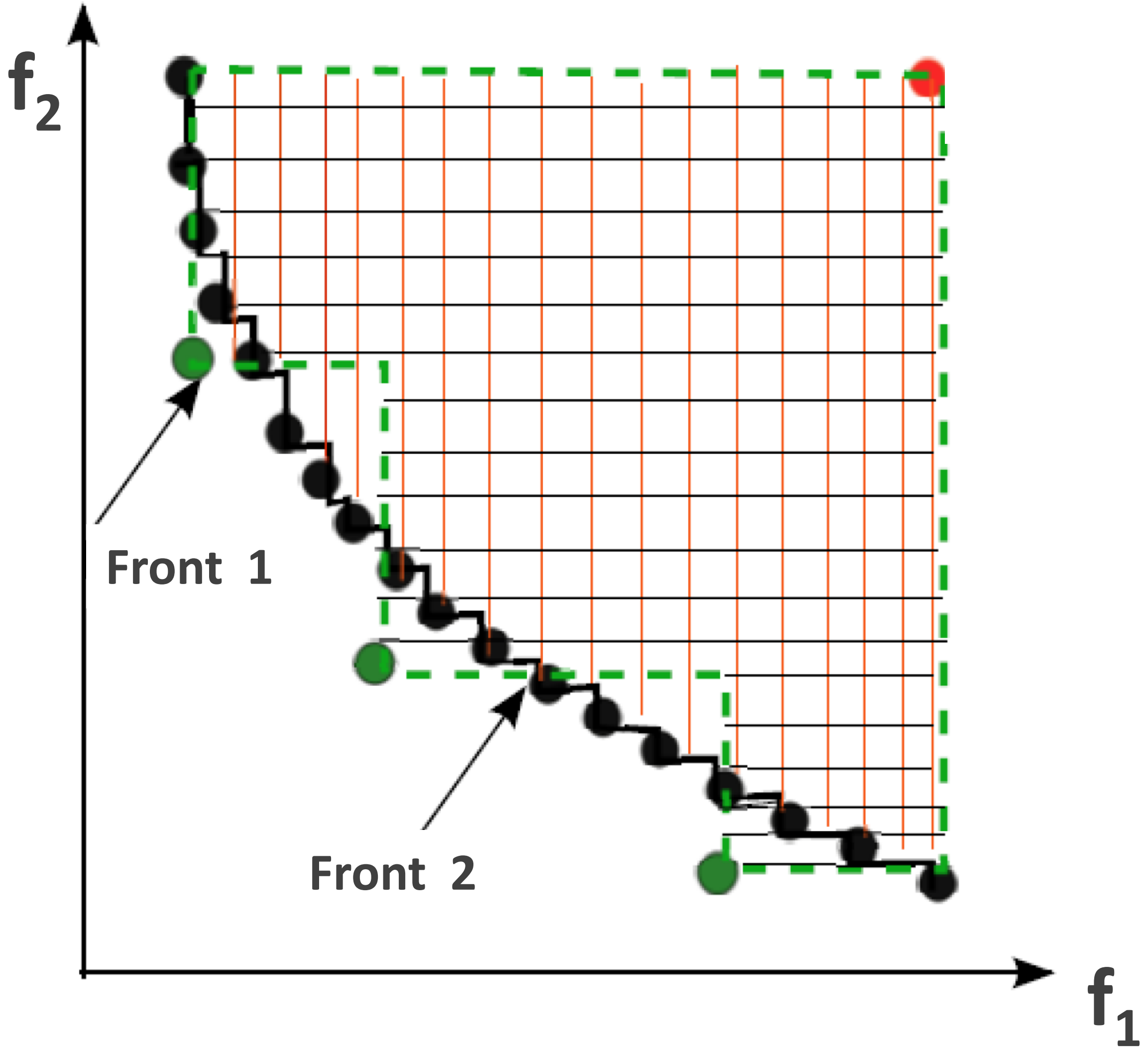
Decision Making!

Min. F_1
Min. F_2

The Hypervolume Indicator (HI)



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



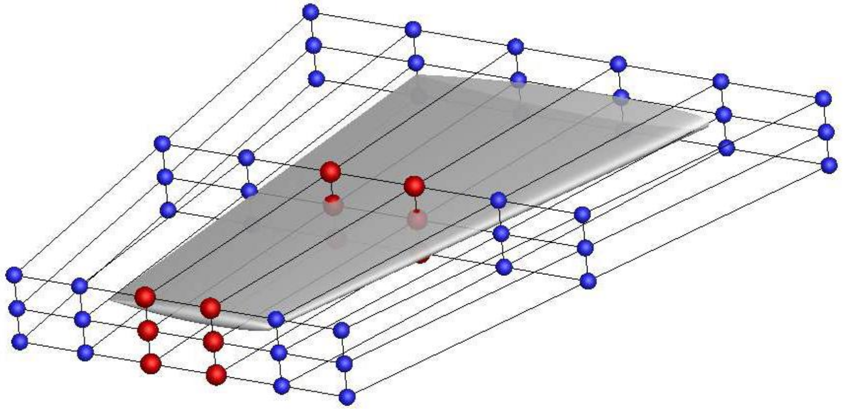
Min. F_1
Min. F_2

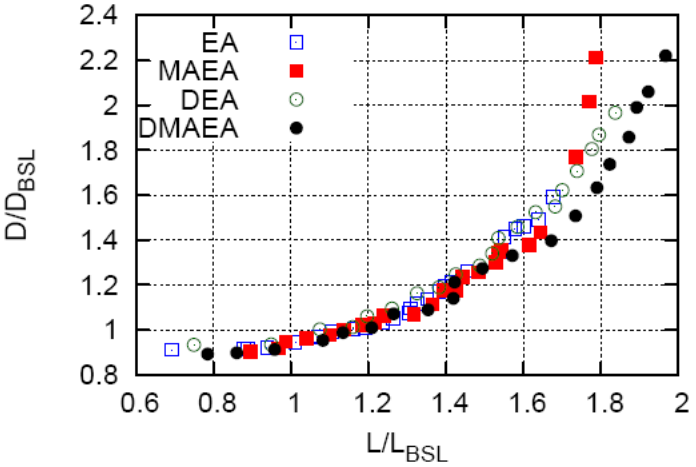
Comparing Fronts of Non-Dominated Solutions – An Example (1/2)

ShpO of an isolated wing for max. Lift (L) and min. Drag (D).

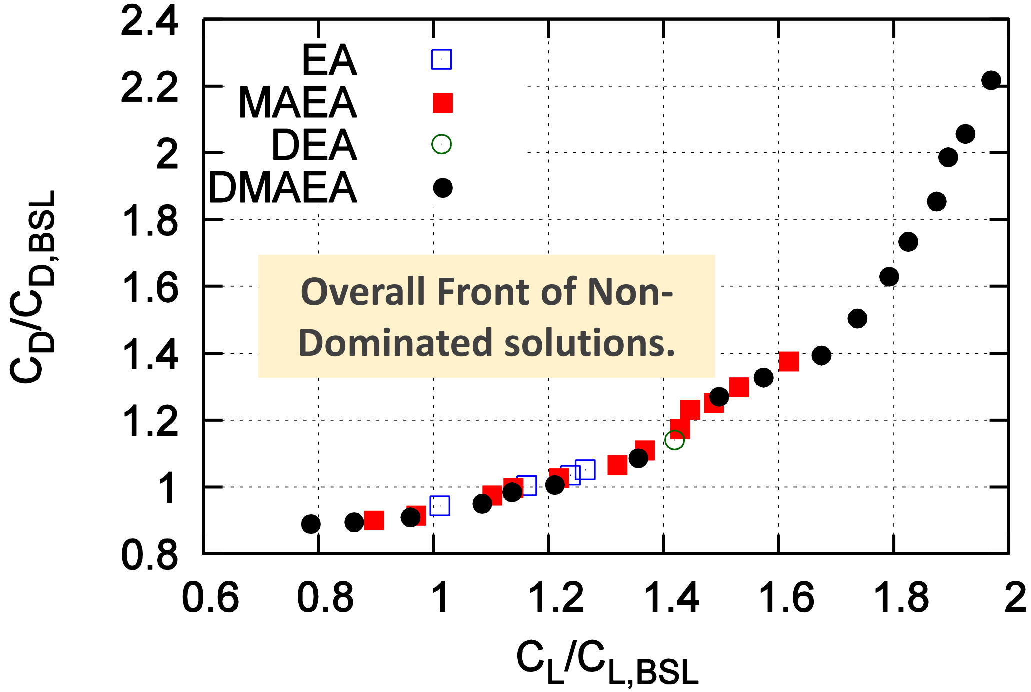
Inviscid flow, M_{inf}=0.40, $a_{pitch,inf}$=3.06°, $a_{yaw,inf}$=0°.

N=24 DoFs (internal control points moving normal to the planform).





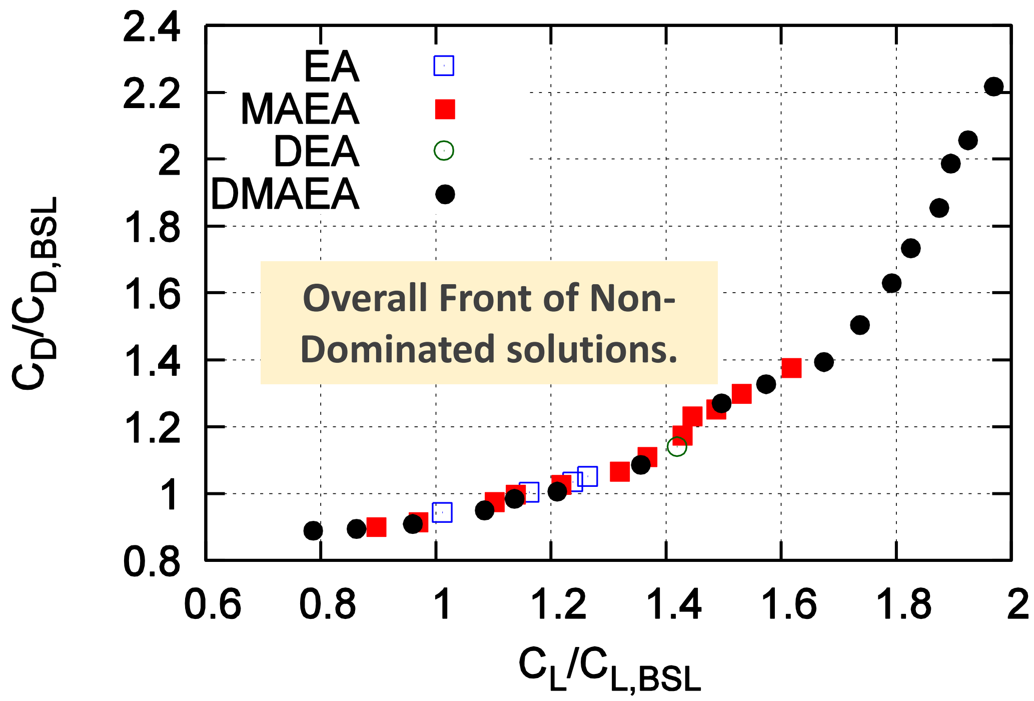
Four methods (to be defined later) computed different fronts. Which of them is the Winner?



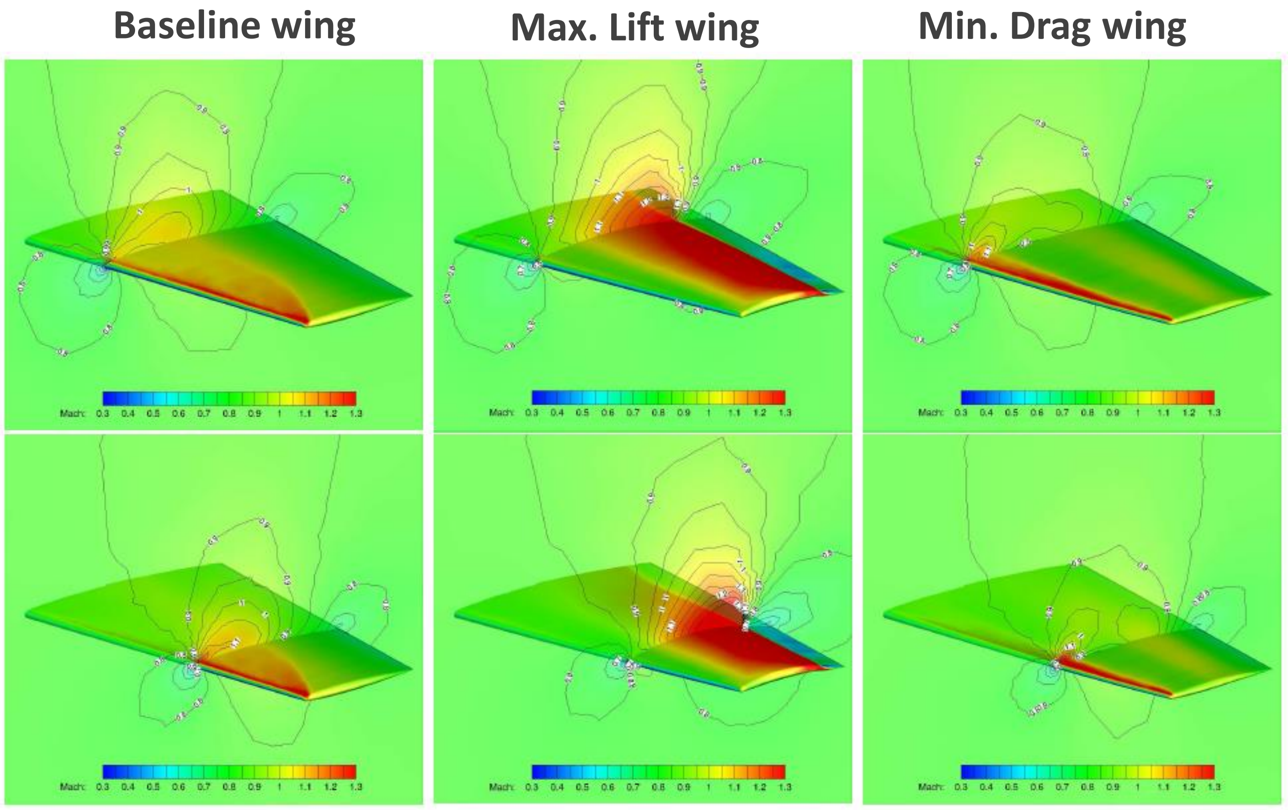
Overall Front of Non-Dominated solutions.

Max. F_1
Min. F_2

Comparing Fronts of Non-Dominated Solutions – An Example (2/2)



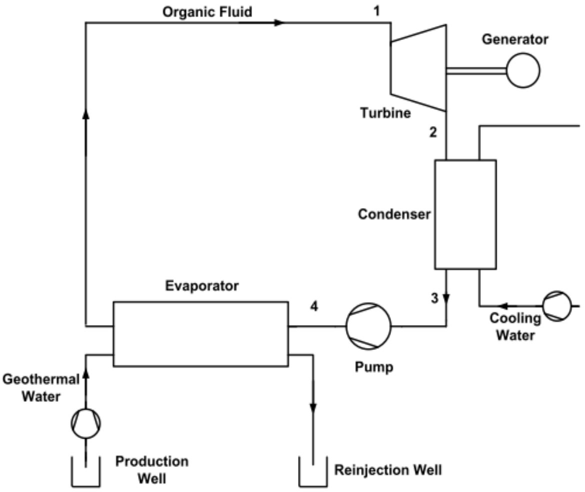
Overall Front of Non-Dominated solutions.



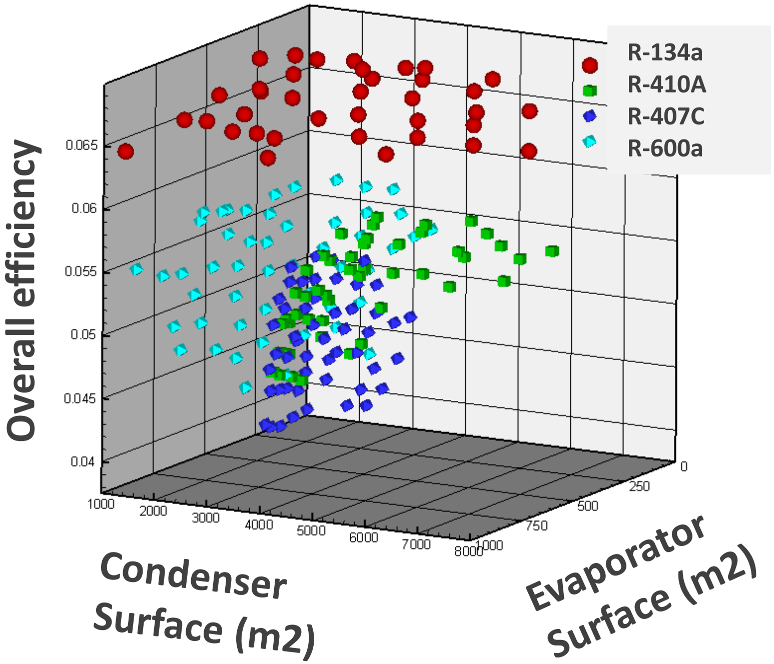
Mach number fields on & around three wing geometries

Max. F_1
Min. F_2

Optimisation of a Geothermal ORC System

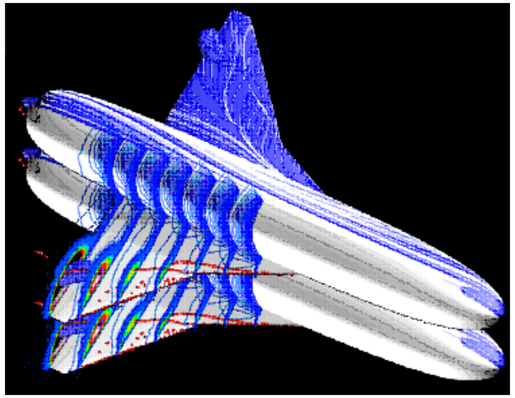


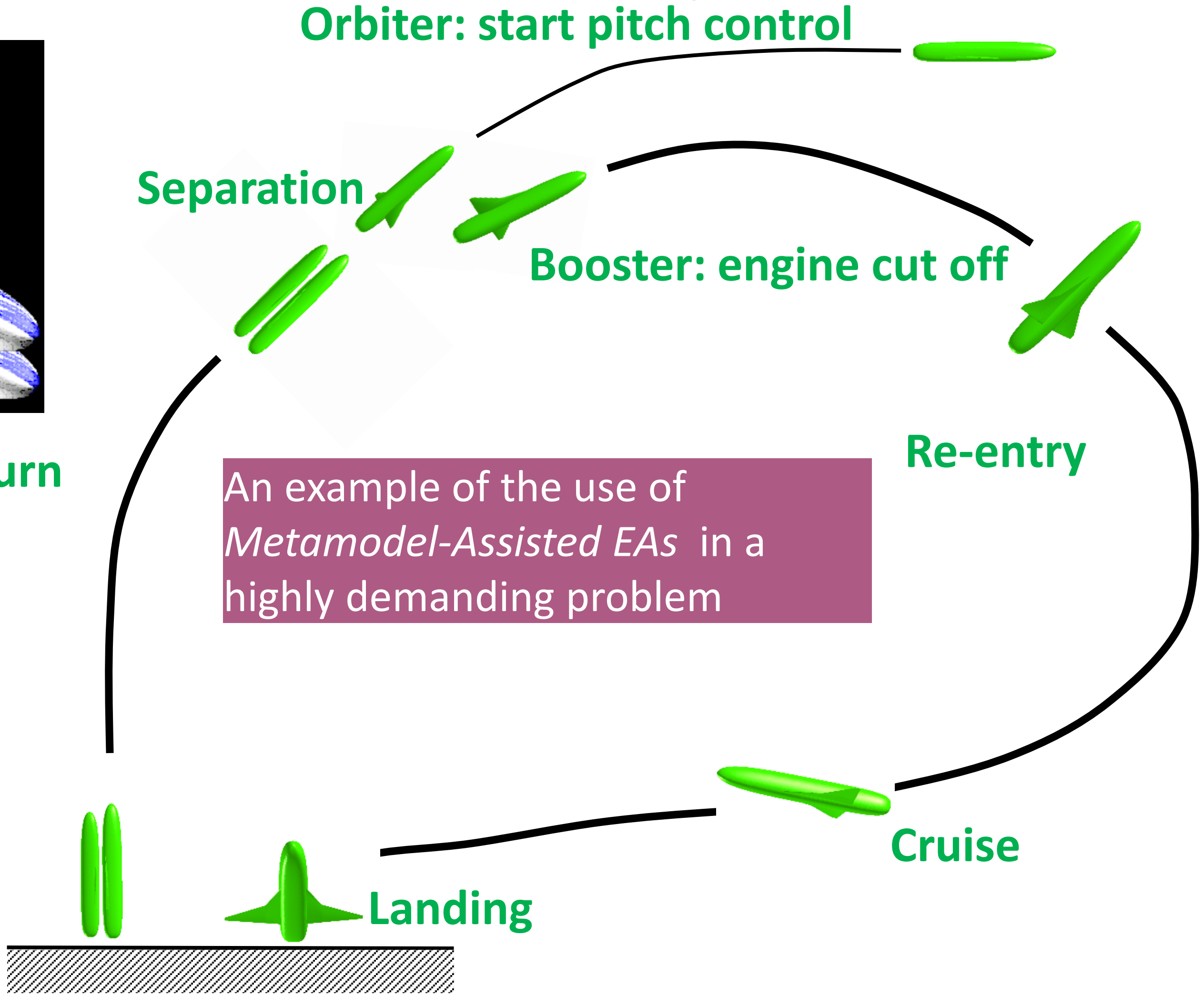
P_{elec} = 200 kW , Simbach, Germany



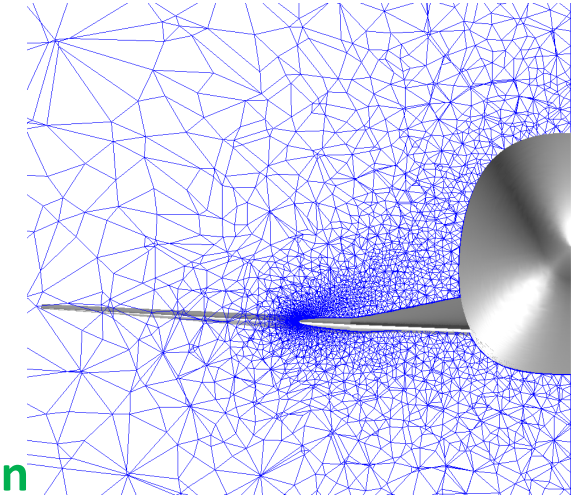
Design- Optimisation of an innovative Organic Rankine Cycle (ORC) System for electricity production using low-enthalpy geothermal energy, with three objectives (M_o=3). Design carried out at the Center for Renewable Energy Sources (CRES) using EASY, for the EU-funded project LOW-BIN. Installed & measured at Simbach, Germany.

Multi-Dimensional Pareto Fronts – Challenges (1/2)





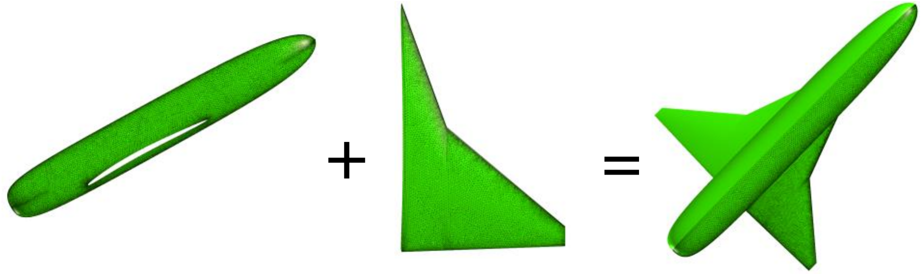
An example of the use of *Metamodel-Assisted EAs* in a highly demanding problem



Optimisation of a Reusable Launch Vehicle (RLV), $M_o=4$

In collaboration with TOHOKU UNIVERSITY

Multi-Dimensional Pareto Fronts – Challenges (2/2)



Think of possible ways to visualise a 4D Pareto front!!!!

Objective 1: Min. shifting of the aerodynamic center for min. control mechanisms.

$$F_1 = \left| C_{Mp}^{\text{supersonic}} - C_{Mp}^{\text{transonic}} \right|$$

Objective 2: Min. pitching moment at transonic flight, for stability purposes.

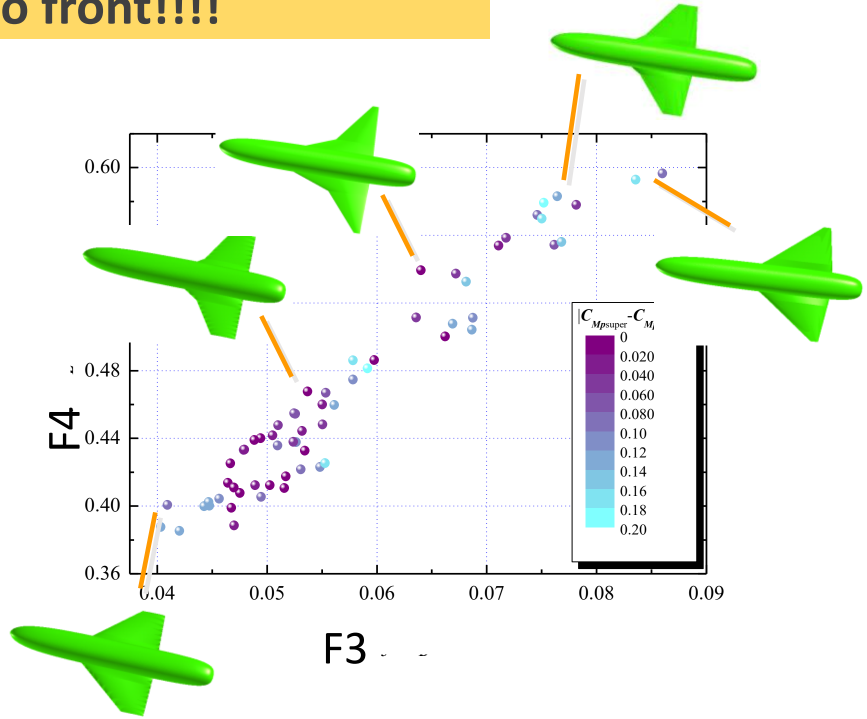
$$F_2 = \left| C_{Mp}^{\text{transonic}} \right|$$

Objective 3: Min. drag at transonic flight (major part of the range) for max. range.

$$F_3 = C_D^{\text{transonic}}$$

Objective 4: Max. lift at subsonic flight for min. runaway distance.

$$F_4 = C_L^{\text{subsonic}}$$



K. Chiba, S. Obayashi, K. Nakahashi, A. Giotis and K.C. Giannakoglou: 'Design Optimization of the Wing Shape of the RLV Booster Stage using Evolutionary Algorithms and Navier-Stokes Computations on Unstructured Grids', EUROGEN 2003, Barcelona, Spain, Sept. 15-17, 2003.

In collaboration with

TOHOKU UNIVERSITY

Optimisation of a Reusable Launch Vehicle (RLV), $M_o=4$

Example: CCGTPP with Supplementary Firing (1/3)

<u>Associated with the Lavrion Units (by the Greek Power Corporation)</u>
(a) 173 MW net ISO capacity, two GTs, two HRSGs and one ST,
(b) 550.2 MW, three GTs, three HRSGs and one ST.
All natural-gas fired.

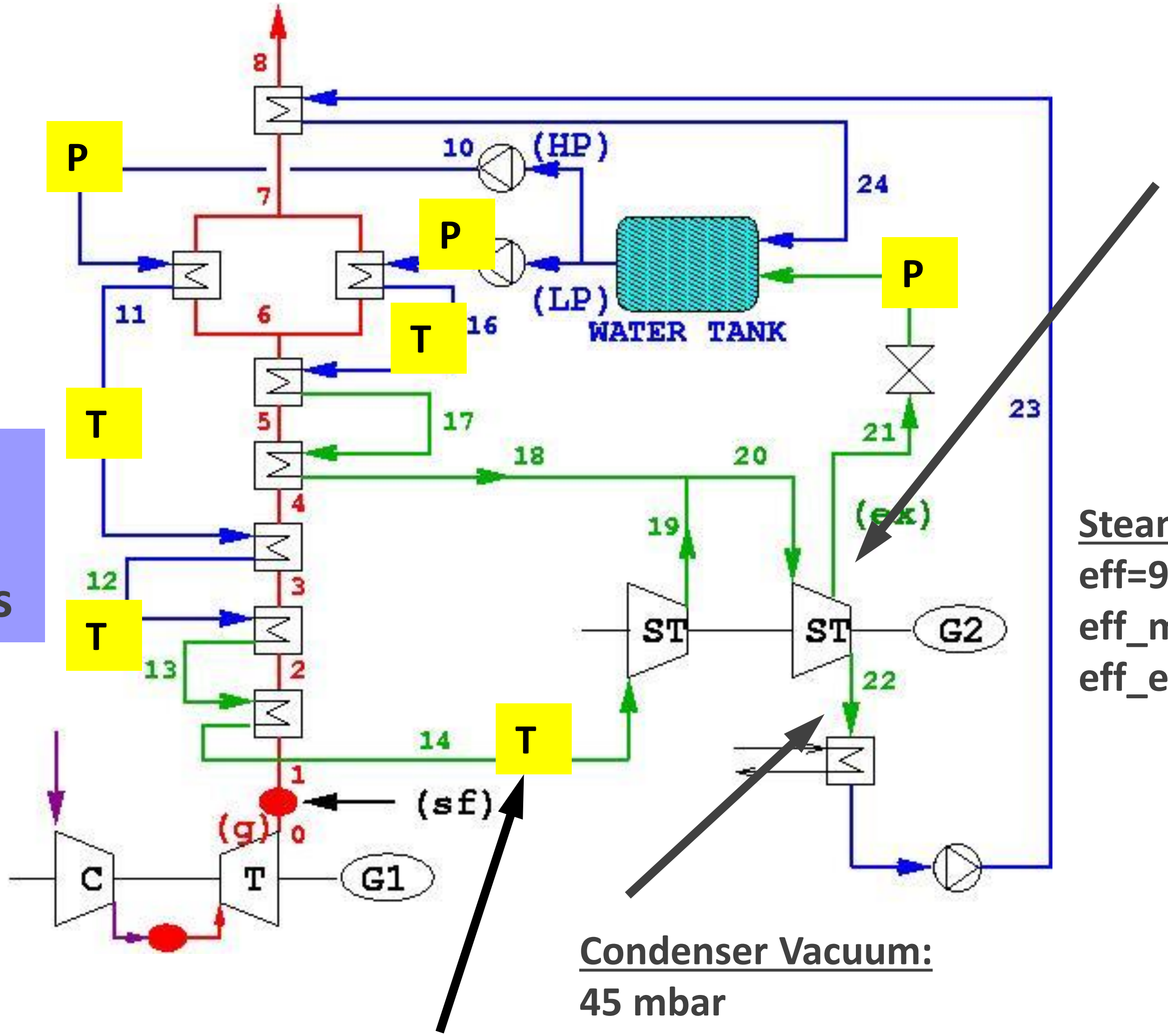
<u>Targets:</u>
(a) max. efficiency,
(b) max. power output,
(c) min. investement cost.

<u>Tools:</u>
(a) for the thermodynamic analysis of the water-steam cycle,
(b) for the CCGT capital cost estimation

Example: CCGTPP with Supplementary Firing (2/3)

Fixed parameters, optimisation variables and 12 constraints

Feasible Heat Exchangers



Extraction: 2.5 bar

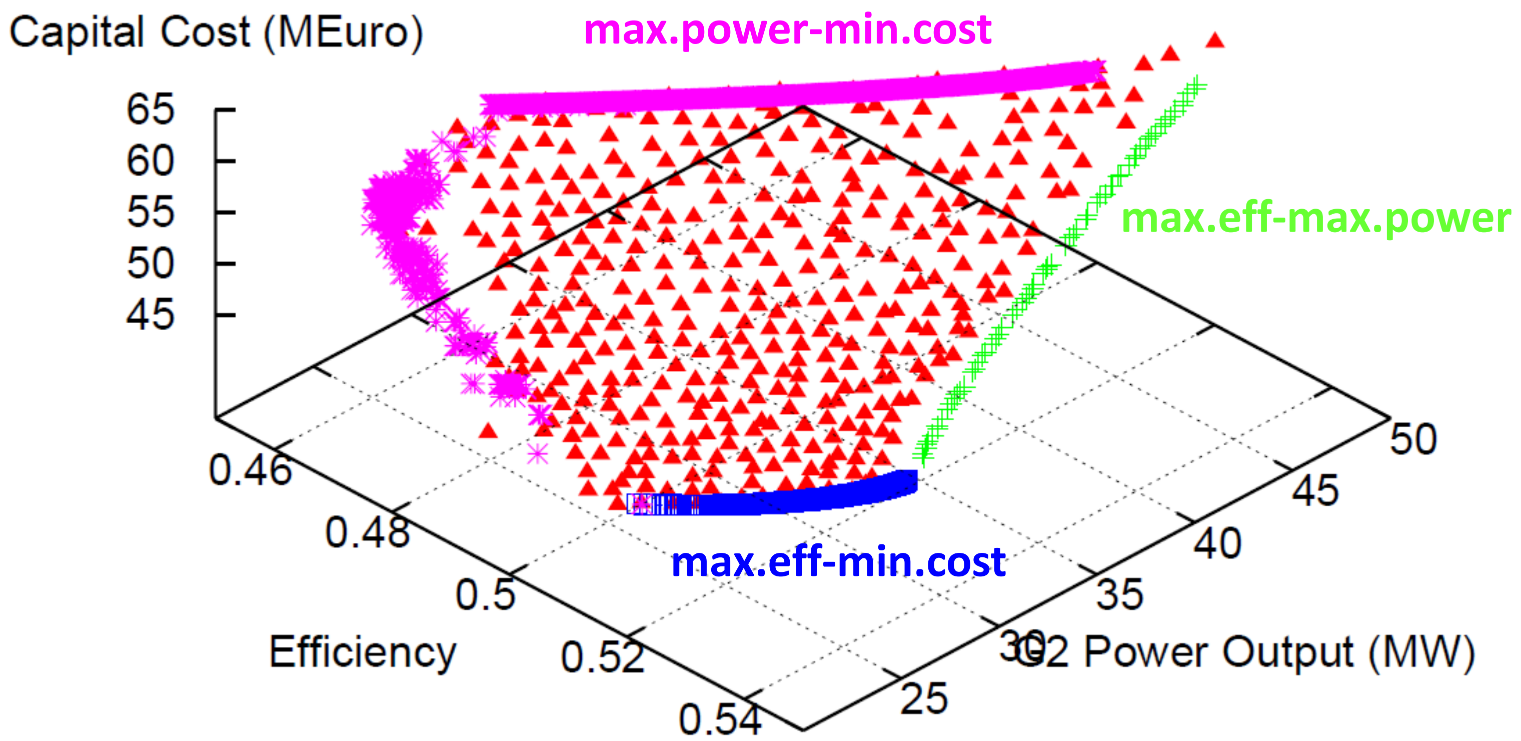
Steam Turbine:
eff=90%
eff_m=90%
eff_el=100%

Two Gas Turbines:
70 MW
eff=40%
m=265 kg/sec
Tfg=440C
(Aeroderivatives)

Condenser Vacuum:
45 mbar

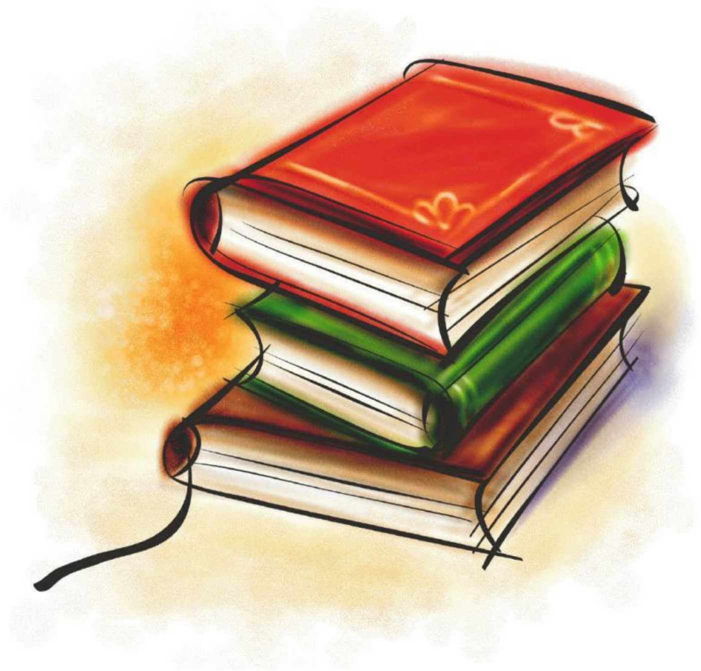
max. T = 565 C

Example: CCGTPP with Supplementary Firing (3/3)



Pareto fronts resulting from a three-objective EA and three two-objective EAs.

E.T. Bonataki, L.S. Georgoulis, C. Georgopoulou and K.C. Giannakoglou: 'Optimal Design of Combined Cycle Power Plants based on Gas Turbine Performance Data', ERCOFTAC Design Optimization: Methods & Applications, Athens, Greece, March 31-April 2, 2004.



MOO: Scalar Φ Computation Techniques

MOO: Scalar Φ Computation Techniques

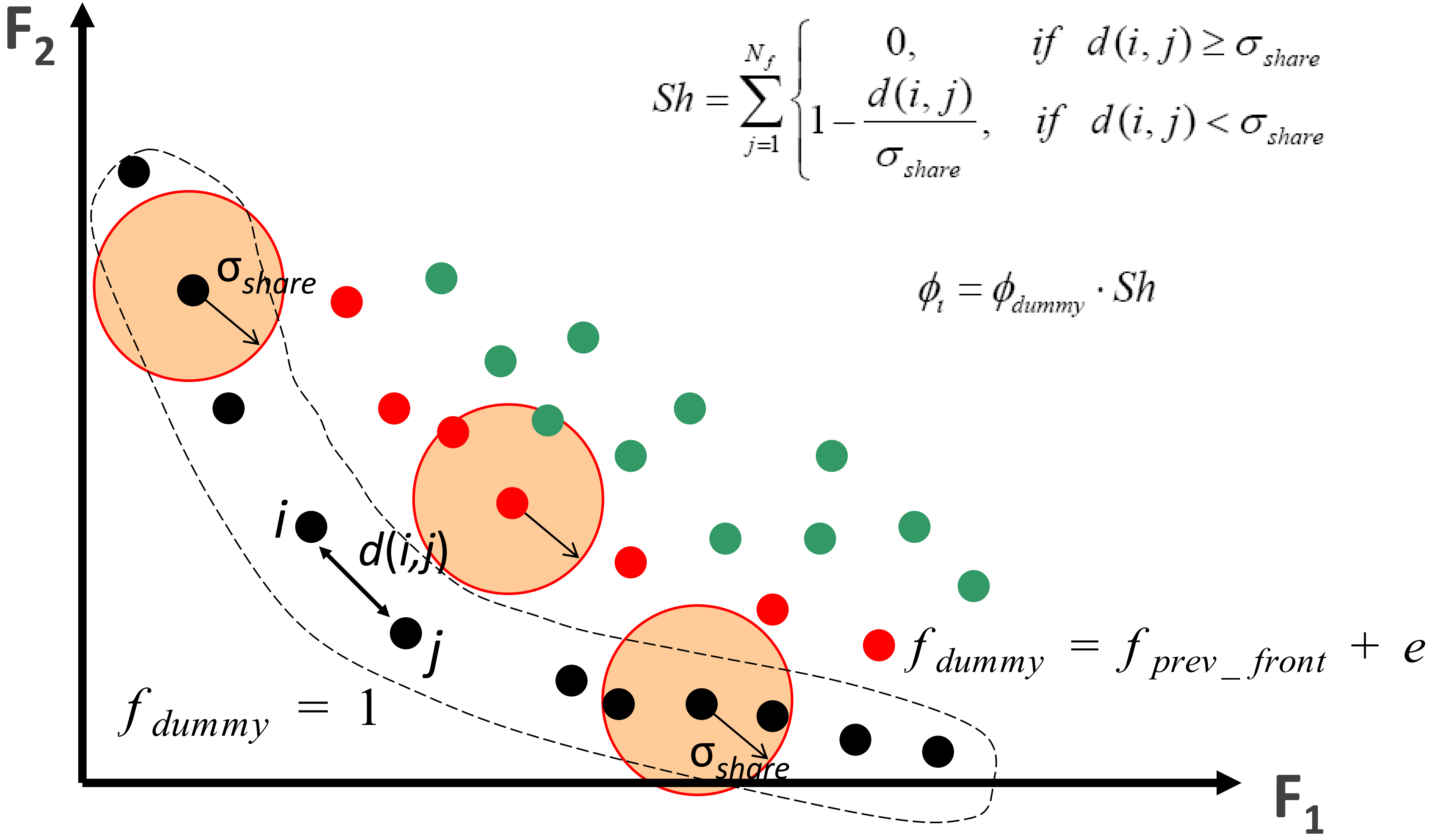
- **Front Ranking [Goldberg, 1989]**
- **Niched Pareto GA (NPGA) [Horn, Nafpliotis, 1993]**
- **Nondominated Sorting GA (NSGA) [Srinivas, Deb, 1994]**
- **Strength Pareto EA (SPEA) [Zitzler, Thiele, 1998]**
- **Pareto Envelope-based Selection Algorithm (PESA) [Corne, Knowles, Oates, 2000]**
- **NSGA-II [Deb, Agrawal, Pratap, Meyarivan, 2000]**
- **Strength Pareto EA II (SPEA II) [Zitzler, Laumanns, Thiele, 2001]**
- **etc**

K. Deb, Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design, in Evolutionary Algorithms in Engineering and Computer Science, Wiley, 1999.
E. Zitzler, L. Thiele, An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach, Technical Report No 43, Computer Engineering and Communication Networks Lab (TIK), Switzerland, May 1998.

MOO: Scalar Φ Computation Techniques – NSGA

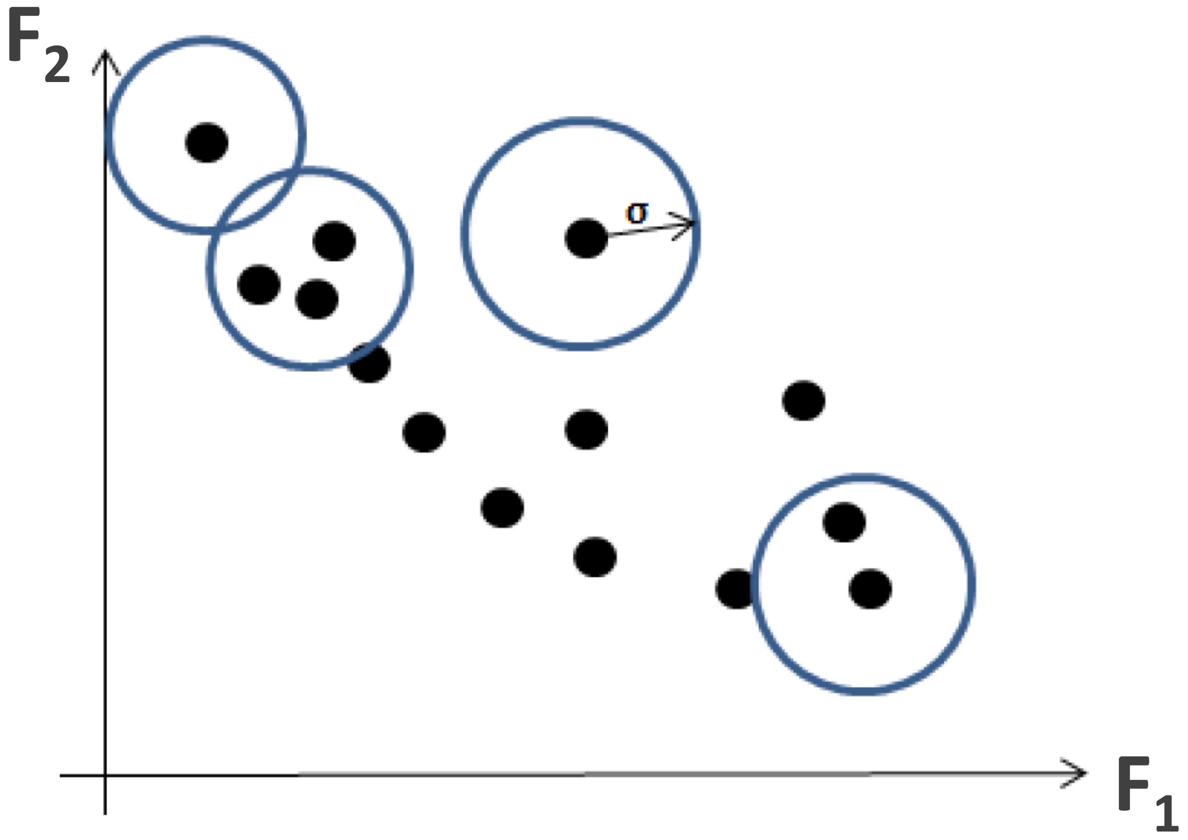
$$Sh=\sum_{j=1}^{N_f}\begin{cases}0, & if \;\; d(i,j)\geq\sigma_{share}\\ 1-\dfrac{d(i,j)}{\sigma_{share}}, & if \;\; d(i,j)<\sigma_{share}\end{cases}$$

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



Min. F_1
Min. F_2

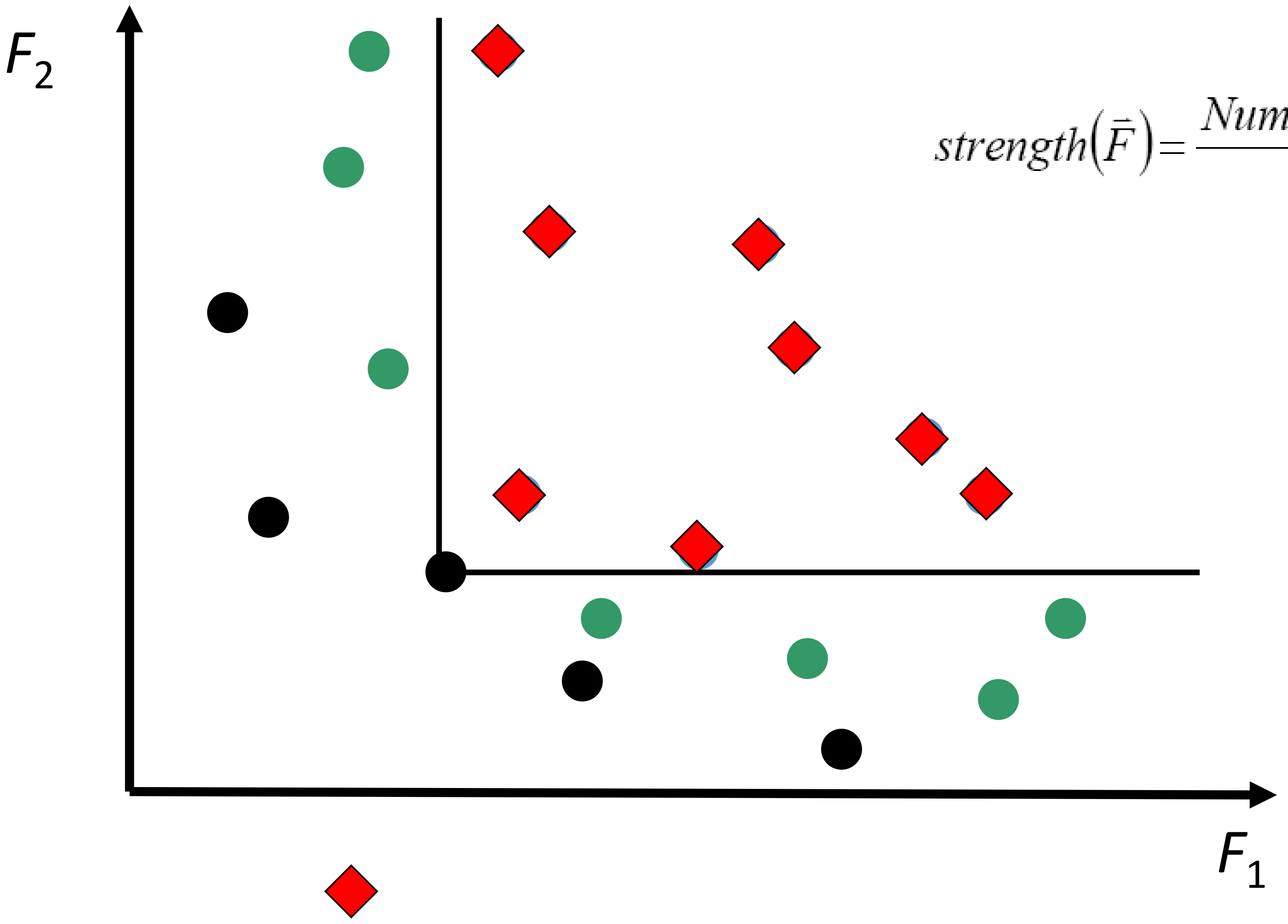
MOO: Distances in the Objective Space



An appropriate regularization of all objectives might be needed!

Min. F_1
Min. F_2

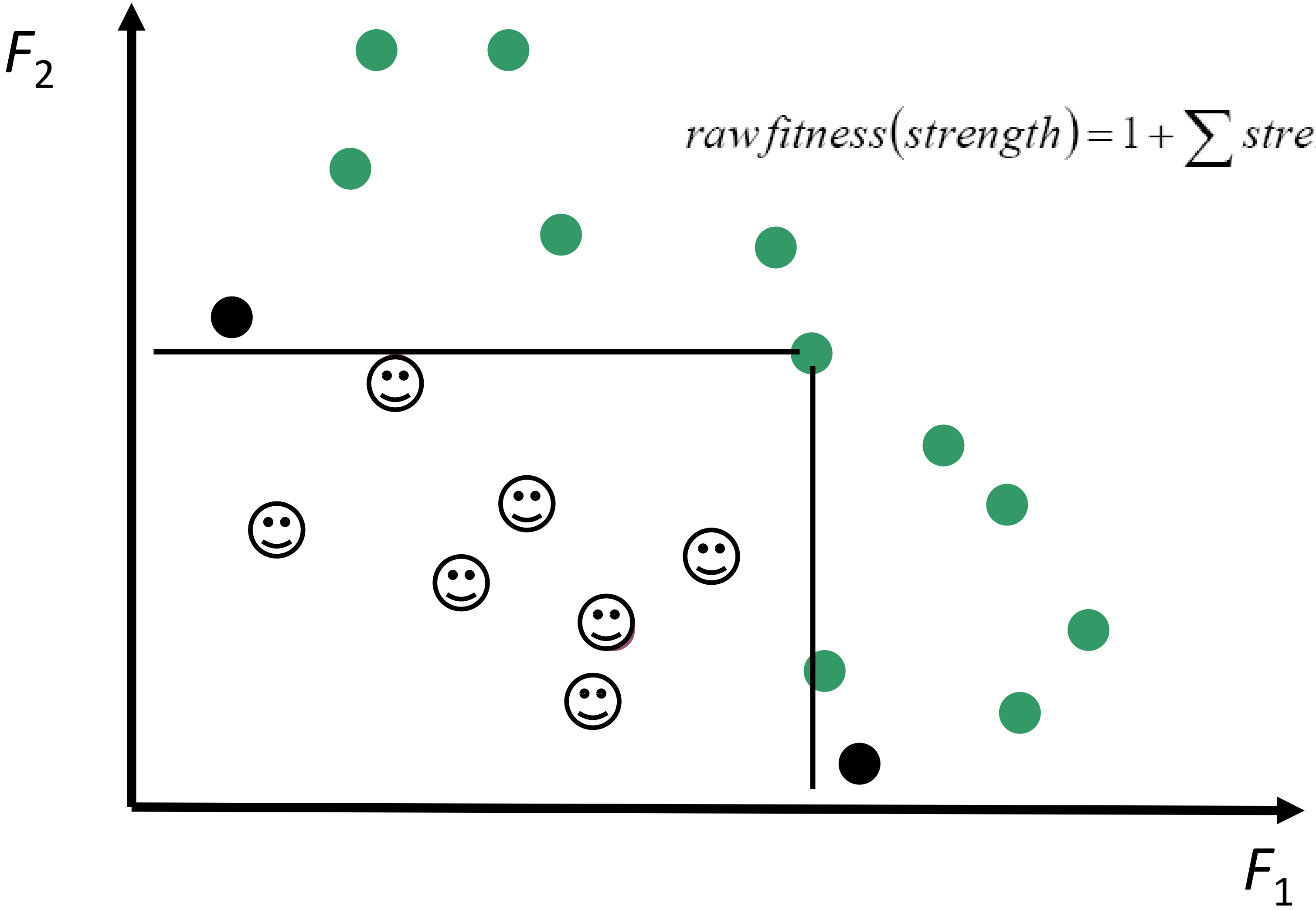
MOO: Scalar Φ Computation Techniques - SPEA



$$strength(\bar{F}) = \frac{Number\ of\ Do\min ated}{\mu+\lambda+1} \in [0,1)$$

Min. F_1
Min. F_2

MOO: Scalar Φ Computation Techniques - SPEA



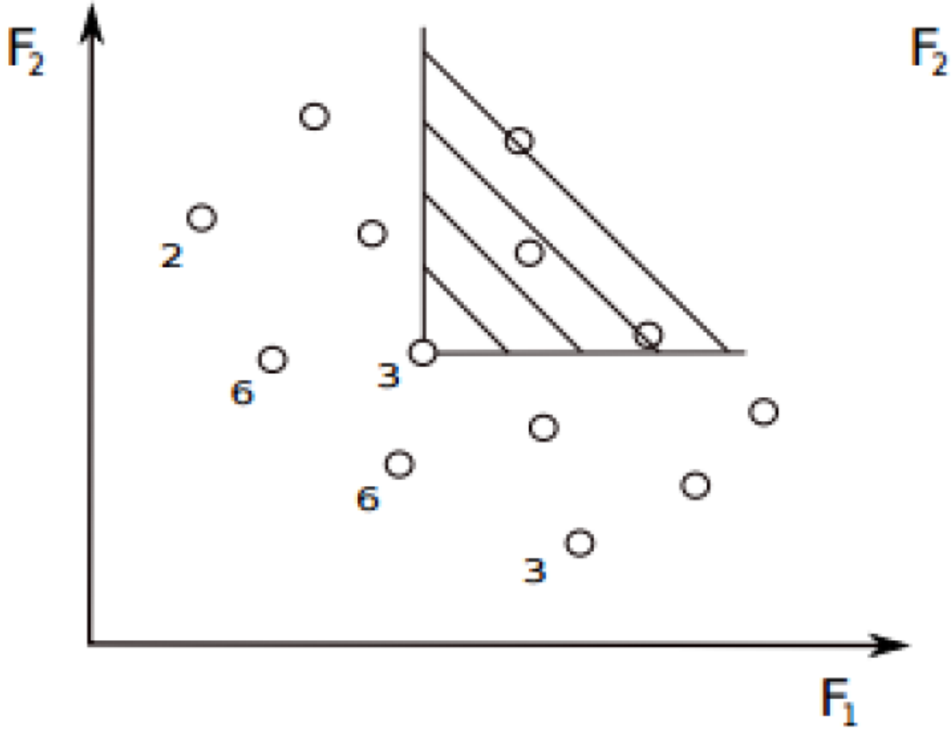
☺

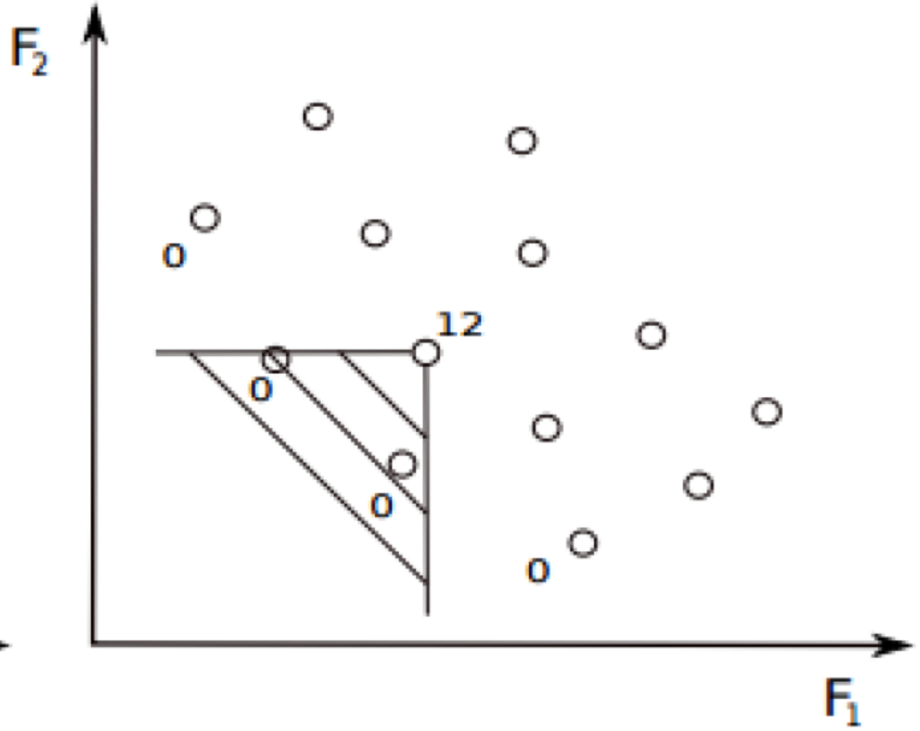
$$raw\,fitness(strength) = 1 + \sum strength(do\min ating) > 1$$

Min. F_1
Min. F_2

MOO: Scalar Φ Computation Techniques - SPEA

Summarizing:





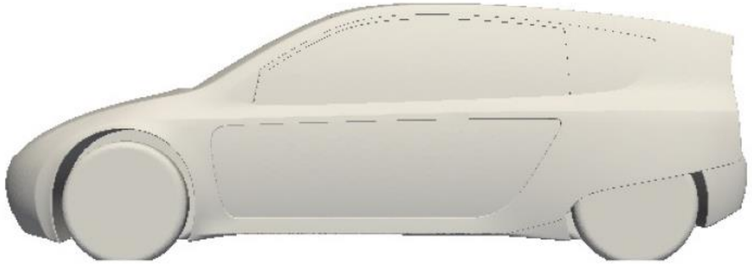
Min. F_1
Min. F_2

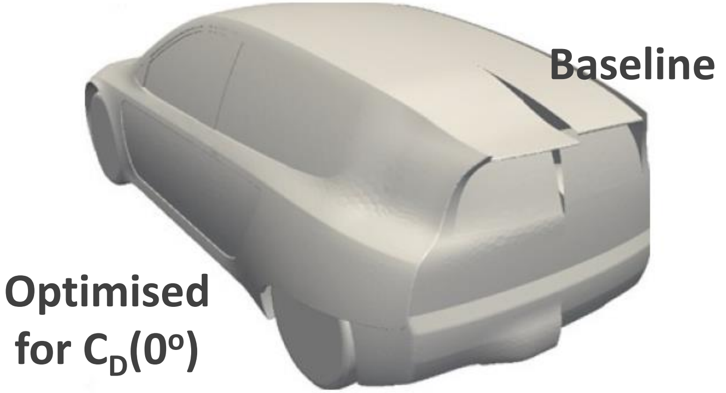
A Naïve Way to Compute Pareto Fronts, using a GBM

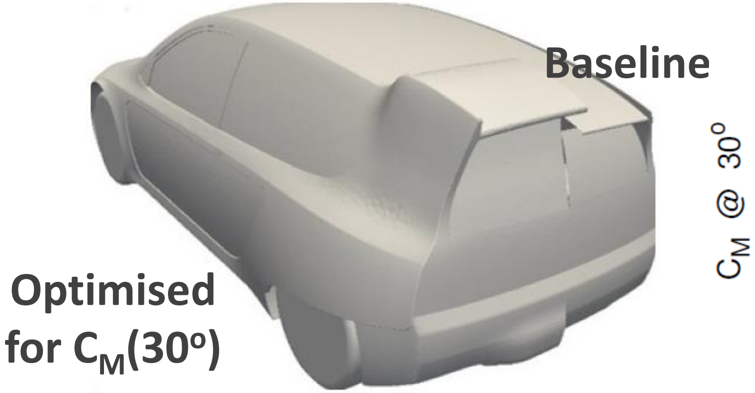
- ShpO of an ultra-lightweight vehicle, designed by the Toyota Aerodynamic Dept., for making it less sensitive to side-wind (30° from the port side; min. yaw moment), while maintaining a very low drag at 0° (longitudinal wind).
- Objective function:

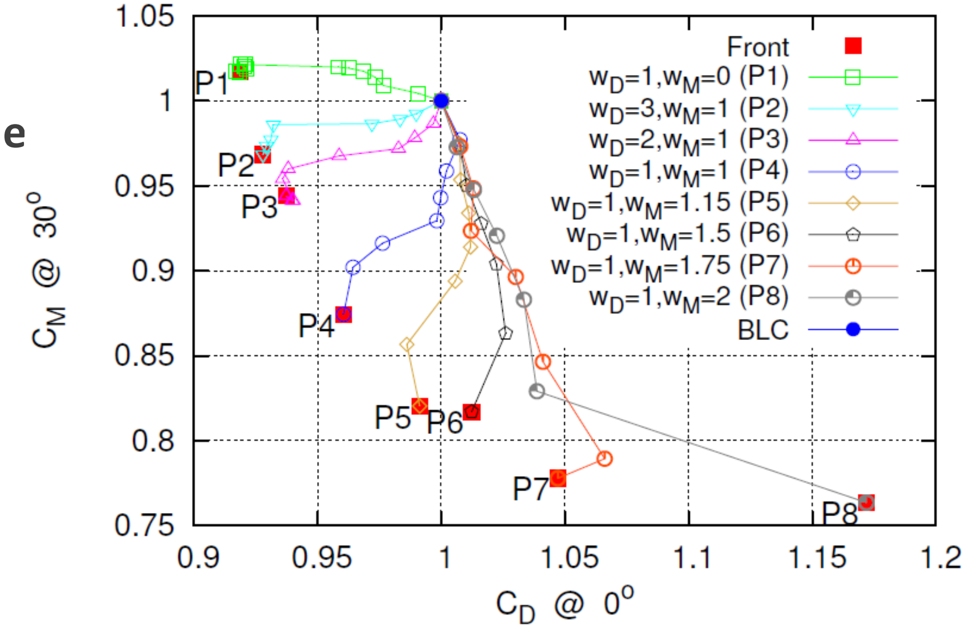
$$\boxed{J = \omega_D C_D^{0^o} + \omega_M C_M^{30^o}}$$

- Pareto front computed by optimising for different values of (ω_D,ω_M).



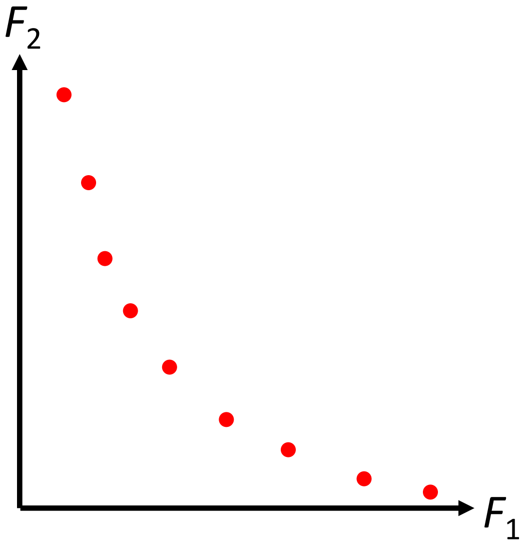


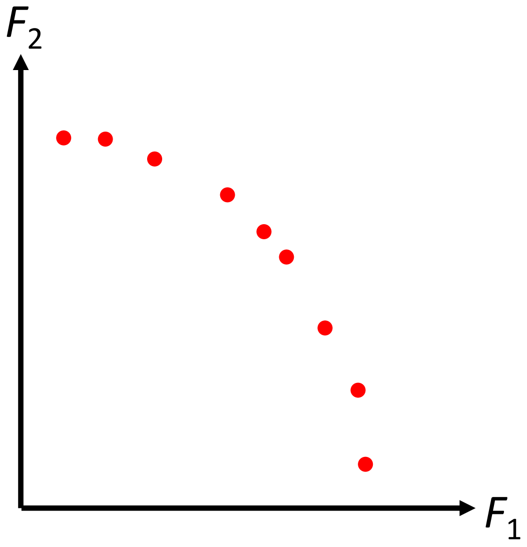




Research funded by TOYOTA

Solving a MOO problem with a SOO method (1/4)





Min. F_1
Min. F_2

$$F = w_1 F_1 + w_2 F_2$$

Working with weights (i.e. by transforming the MOO problem to a SOO problem, with arbitrary weights) is not the best way to solve a MOO problem.
Convex- and Non-Convex Pareto Fronts.

Solving a MOO problem with a SOO method (2/4)

Convex Functions in R^N:

F is a **convex** function if, for any two points ($\overrightarrow{x}$ and $\overrightarrow{y}$) in R^N, if:

$$F(\lambda\overrightarrow{x} + (1-\lambda)\overrightarrow{y}) \;\leq\; \lambda F(\overrightarrow{x}) + (1-\lambda)F(\overrightarrow{y})$$

for any $\lambda \in (0,1]$.

Practically, any line segment connecting any two points on the graph lies above or on the graph itself.

Solving a MOO Problem with a SOO Method (3/4)

Quiz: **What an SOO method will compute if used to solve the two-objective minimisation problem using weighted sum of the two objectives:**

$$\min F_1(\omega) = \sin\omega, \qquad F_2(\omega) = \cos\omega, \qquad \omega \in [0, \frac{\pi}{2}]$$

Objective Function (SOO) ; *k*=weight:

$$F = k\,\sin\omega + (1-k)\,\cos\omega$$

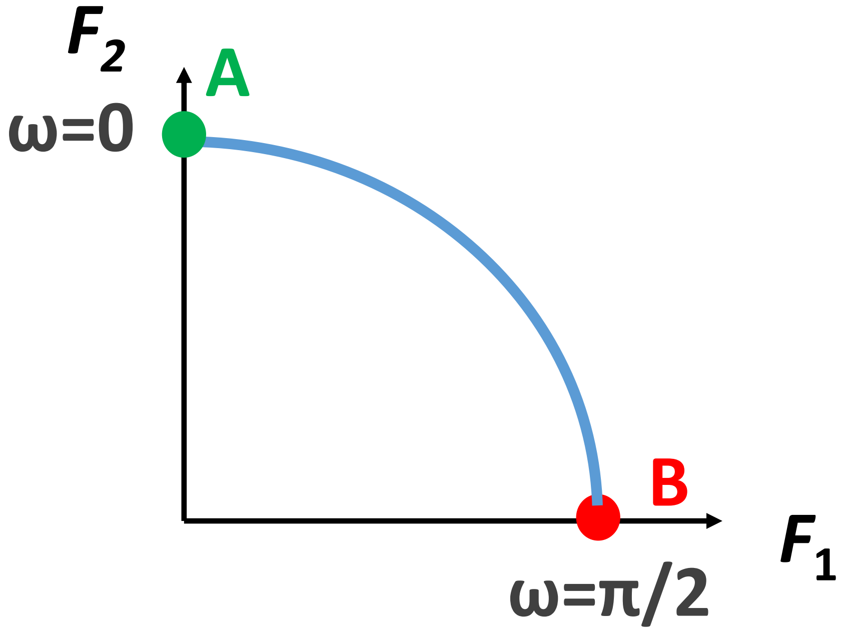
B

Solving a MOO Problem with a SOO Method (4/4)

$$F=k\sin\omega+(1-k)\cos\omega$$

$k<1/2$ → B

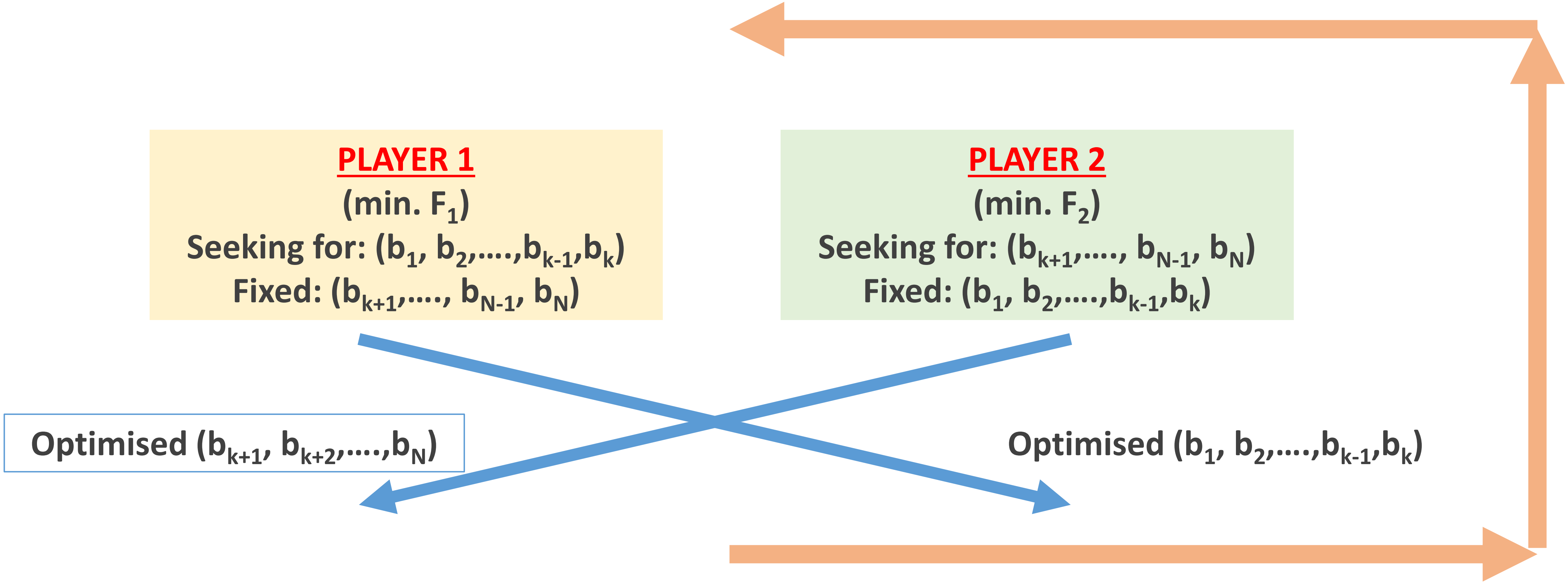
$k>1/2$ → A



B

From a Cooperative (Pareto) to a Competitive Game (Nash) – Nash Game (1/3)

(Min. F_1, Min. F_2). Optimisation variables ($b_1, b_2, \ldots, b_{k-1}, b_k, b_{k+1}, \ldots, b_{N-1}, b_N$)



PLAYER 1

(min. F_1)

Seeking for: ($b_1, b_2, \ldots, b_{k-1}, b_k$)

Fixed: ($b_{k+1}, \ldots, b_{N-1}, b_N$)

PLAYER 2

(min. F_2)

Seeking for: ($b_{k+1}, \ldots, b_{N-1}, b_N$)

Fixed: ($b_1, b_2, \ldots, b_{k-1}, b_k$)

Optimised ($b_{k+1}, b_{k+2}, \ldots, b_N$)

Optimised ($b_1, b_2, \ldots, b_{k-1}, b_k$)

From a Cooperative (Pareto) to a Competitive Game (Nash) – Nash Game (2/3)

$\min F_1(b_1,b_2)=(b_1-5)^2+(b_2-b_1+3)^2$ & $\min F_2(b_1,b_2)=(b_1+b_2+2)^2+(b_1-b_2)^2$

Game 1:

PLAYER 1
min. F_1; controlling b_1

PLAYER 2
min. F_2; controlling b_2

Game 2:

PLAYER 1
min. F_1; controlling b_2

PLAYER 2
min. F_2; controlling b_1

$\partial F_1/\partial b_1= 2(b_1-5)-2(b_2-b_1+3)=0 \rightarrow 2b_1-b_2=8$

$\partial F_1/\partial b_2= 2(b_2-b_1+3)=0 \rightarrow b_2-b_1=-3$

$\partial F_2 /\partial b_1=(b_1,b_2)=2(b_1+b_2+2) +2(b_1-b_2)=0 \rightarrow b_1=-1$

$\partial F_2 /\partial b_2=2(b_1+b_2+2)-2(b_1-b_2)=0 \rightarrow b_2=-1$

From a Cooperative (Pareto) to a Competitive Game (Nash) – Nash Game (3/3)

$$\min F_1(b_1,b_2)=(b_1-5)^2+(b_2-b_1+3)^2 \quad \& \quad \min F_2(b_1,b_2)=(b_1+b_2+2)^2+(b_1-b_2)^2$$

Game 1:

PLAYER 1
min. F_1; controlling b_1

$\partial F_1/\partial b_1 \rightarrow 2b_1-b_2=8$

PLAYER 2
min. F_2; controlling b_2

$\partial F_2/\partial b_2 \rightarrow b_2=-1$

$$\boxed{b_1=7/2 \quad ; \quad b_2=-1}$$

Game 2:

PLAYER 1
min. F_1; controlling b_2

$\partial F_1/\partial b_2 \rightarrow b_2-b_1=-3$

PLAYER 2
min. F_2; controlling b_1

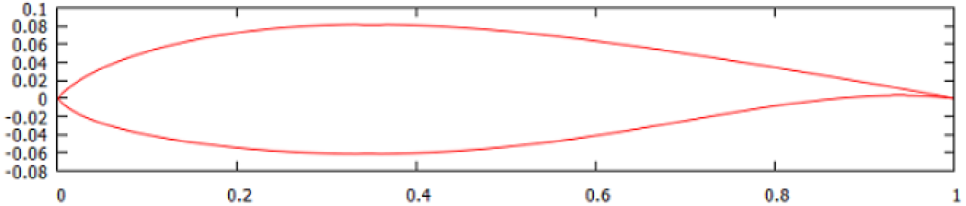
$\partial F_2/\partial b_1 \rightarrow b_1=-1$

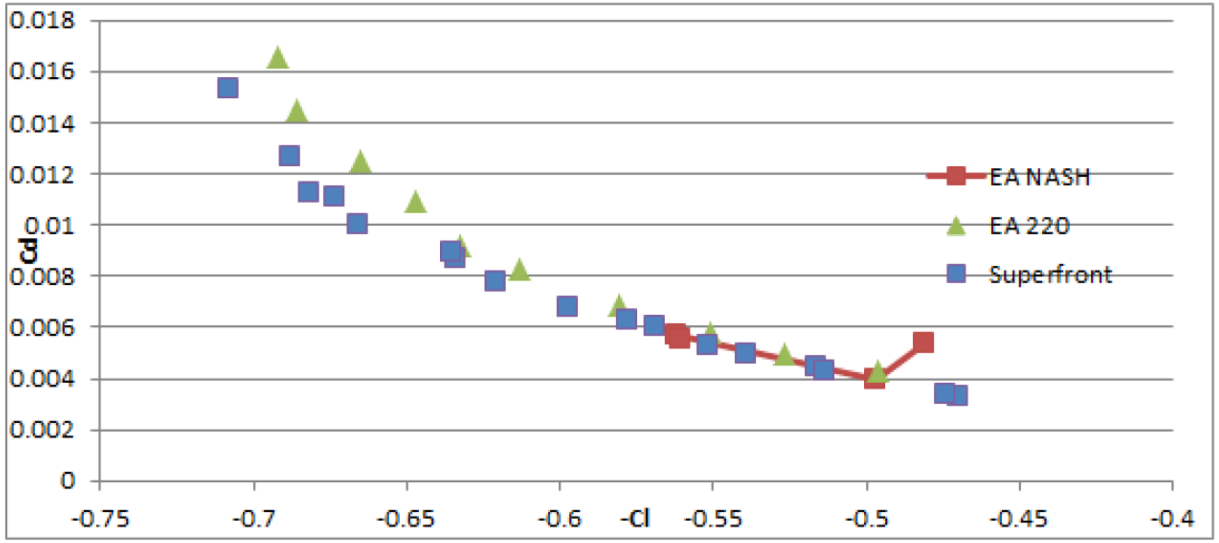
$$\boxed{b_1=-1 \quad ; \quad b_2=-4}$$

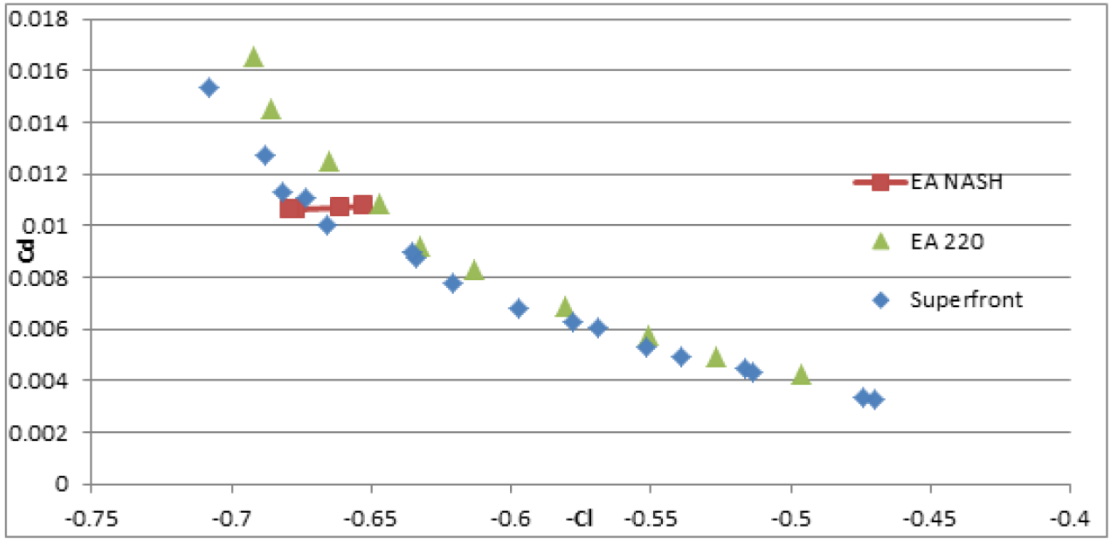
From a Cooperative Game (Pareto) to a Competitive one (Nash)

Example:

Optimisation of an Isolated Airfoil (inviscid flow) for Min C_D, min. $-C_L$)
Cost of a Nash run: 90 TU
Vs. 220 TU of an incomplete EA

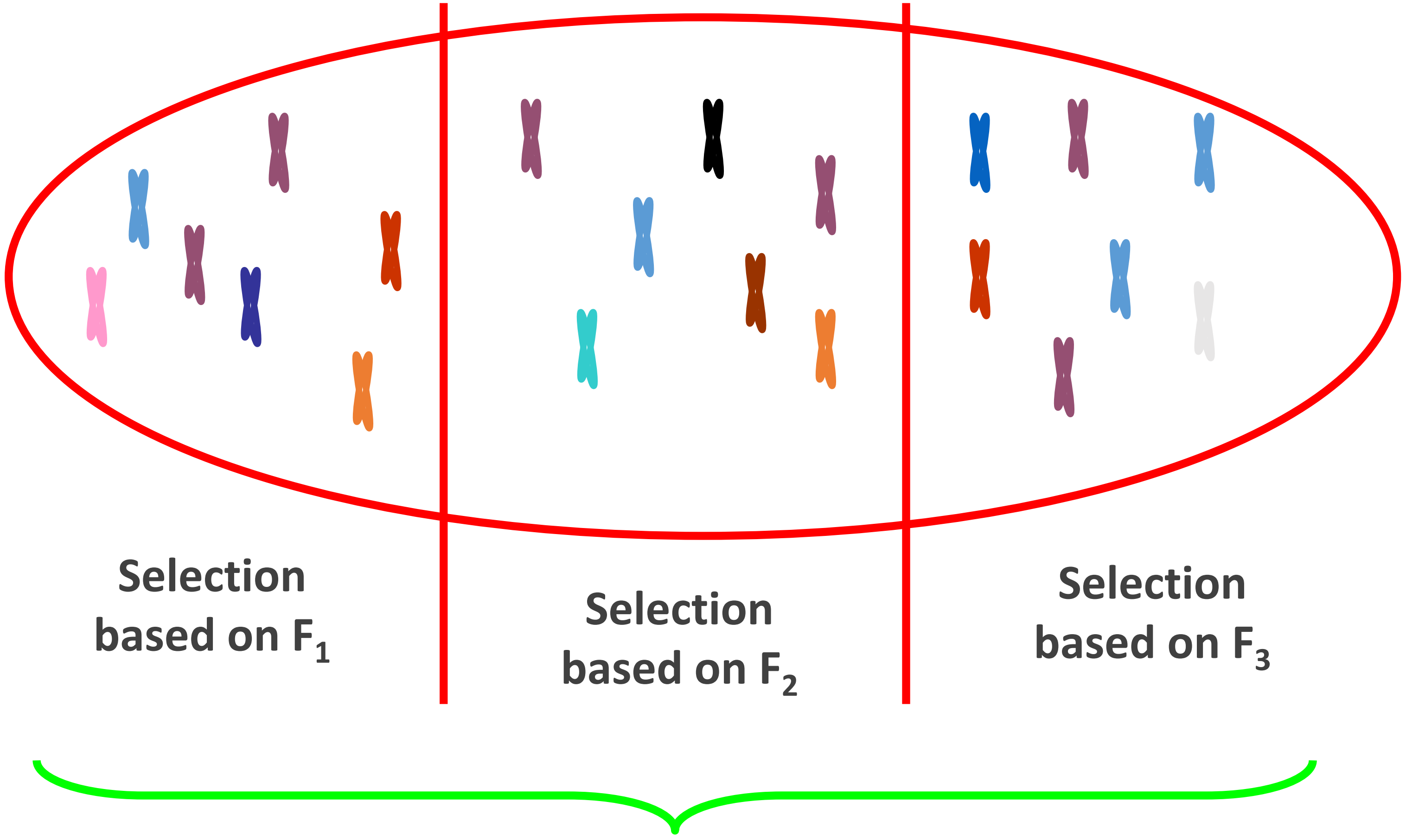






Vector-Evaluated Genetic Algorithm (VEGA) (1/2)

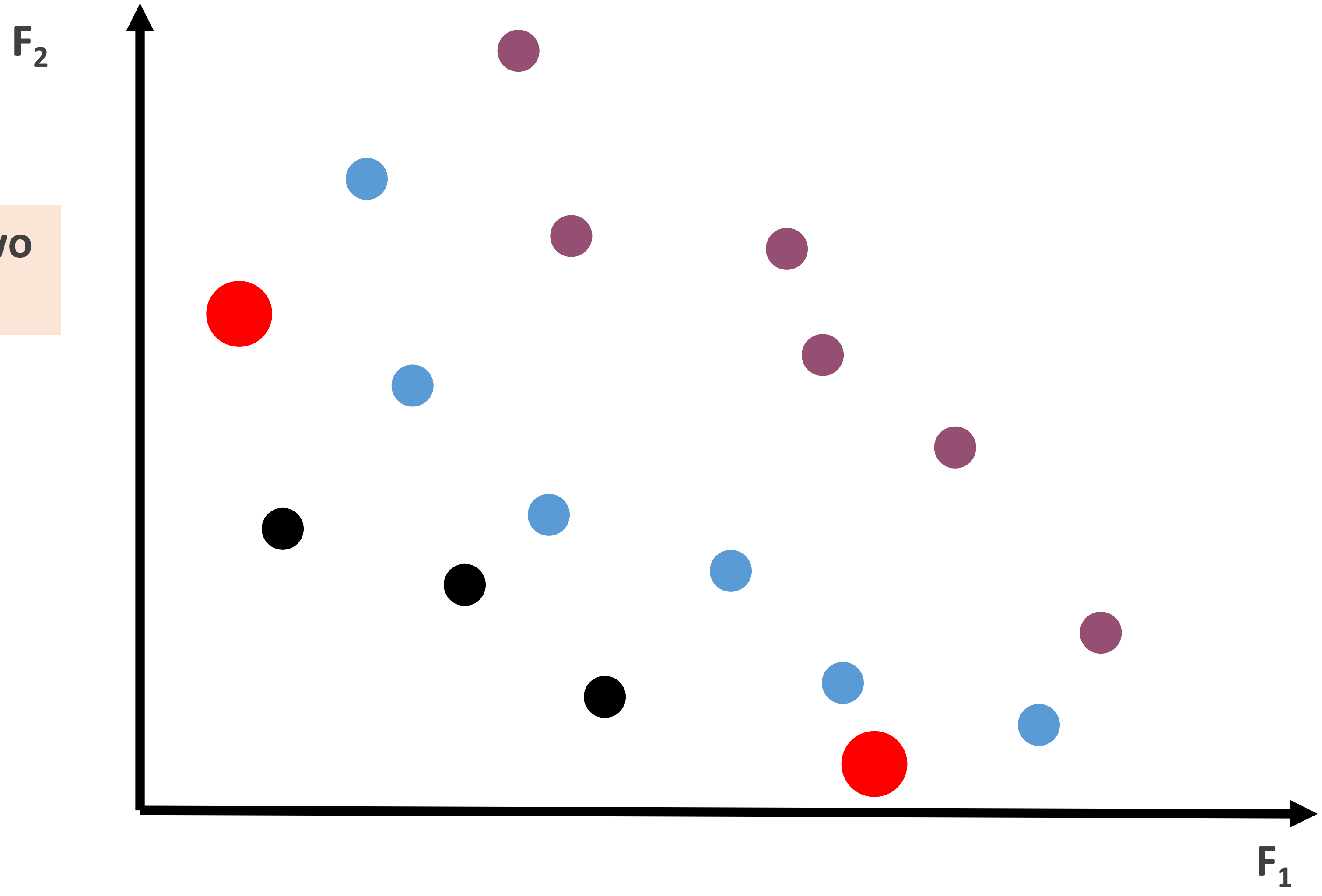
Example in case of three objectives (M_o=3)

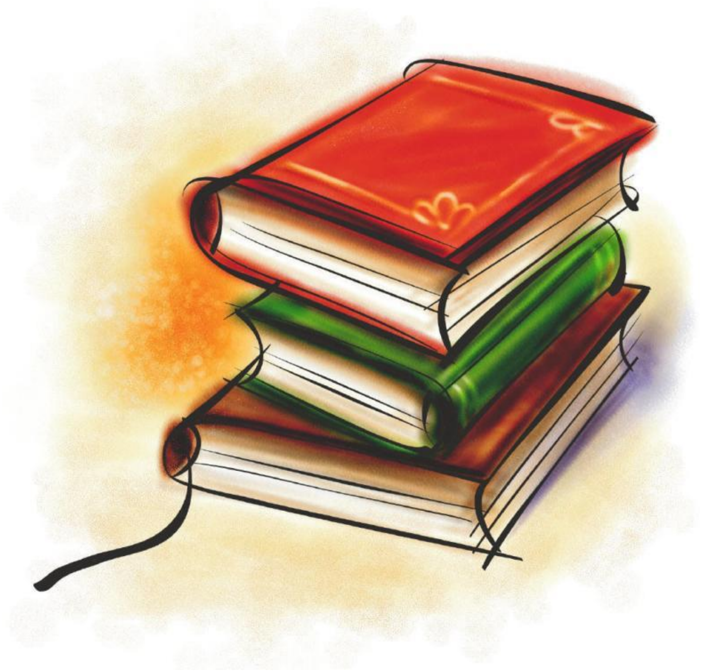


Apply Evolution Operators: Recombination, Mutation, Elitism,…

VEGA – Guess the final solutions (2/2)

Example in case of two objectives (M_o=2)





Constraint Handling in EAs

Constraint Handling in EAs – Escalated Penalties

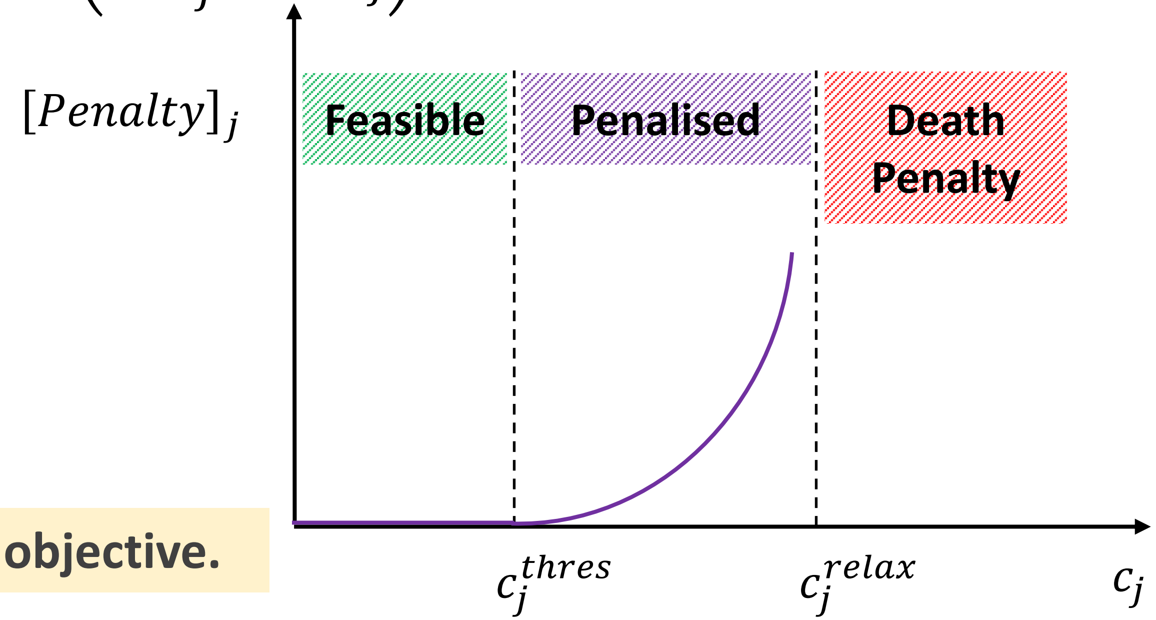
Exponential penalty if

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

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

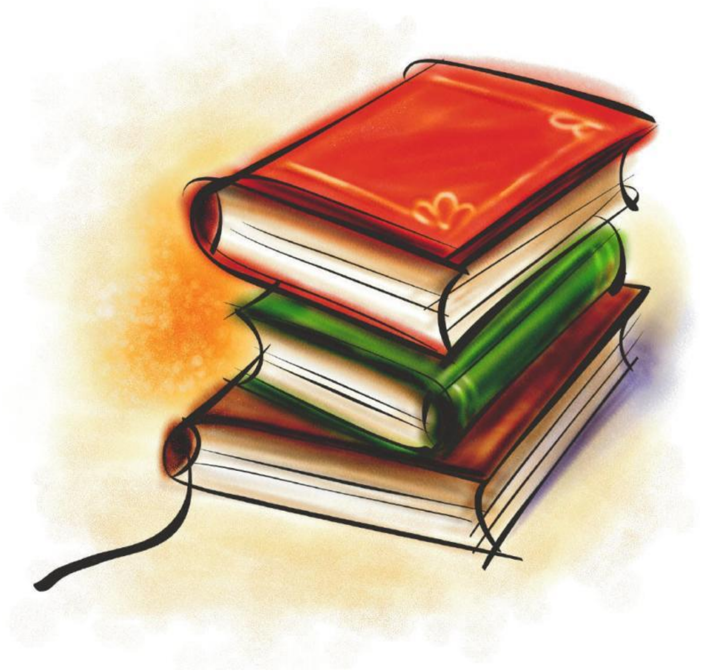
Death penalty if

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



(* minimisation problem)

In MOO, this technique applies separately to each objective.



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

Stochastic, Population-Based Optimisation Methods. Pros & Cons

Pros (GFM):

- **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 (GFM):

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

Our EA: The EASY Platform

All theory and computations presented below, in the field of GFM) are based on the EASY s/w 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

Industrial Use of EAs

<u>Requirements for the penetration of EAs in the industry:</u>

◆ **Find the optimal solution(s) with less (CFD-CAA S/W based) evaluations:**

- **Better evolution operators or carefully tuned parameters**
- **Distributed search**
- **Replace calls to the <u>costly/exact</u> evaluation tool (PSM, for instance CFD or CAA) with calls to <u>cheaper/less-accurate</u> surrogate evaluation models (metamodels)**
- **Dimensionality reduction (curse of dimensionality)**
- **Hierarchical/multilevel search.**

◆ **Reduce the wall-clock time of the optimisation:**

- **Parallelisation.**
- **Overcoming the generation synchronisation barrier (asynchronous search).**

◆ **Combine some (all?) of the above.**

EAs: Exploration vs. Exploitation

In an effective & efficient EA, Exploration and Exploitation should be balanced.

Exploration means that the algorithm can search for better solutions in new regions.

Exploitation means to profit of existing solutions and refine them to increase their fitness.

(μ,λ) EA: Notations – Key Terms

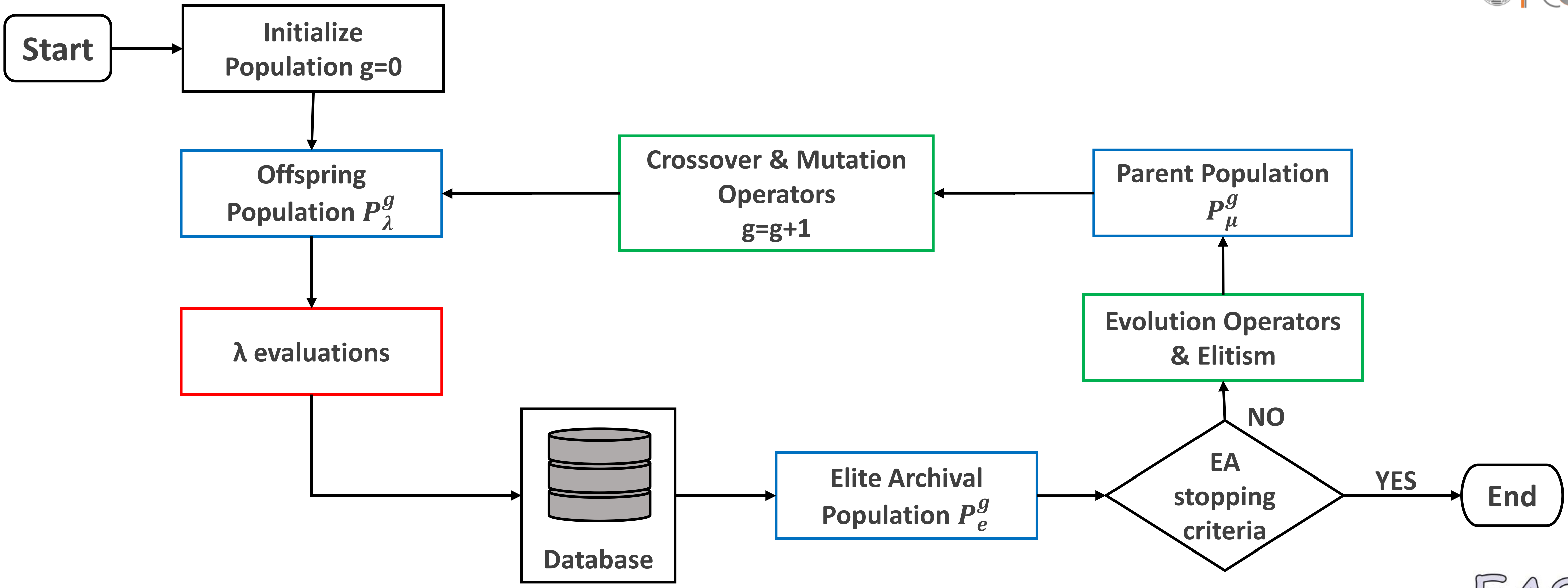
Notations:

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

μ = parent population size
λ = offspring population size
e = elite population size
g = generation index
ρ = number of parents to form an offspring

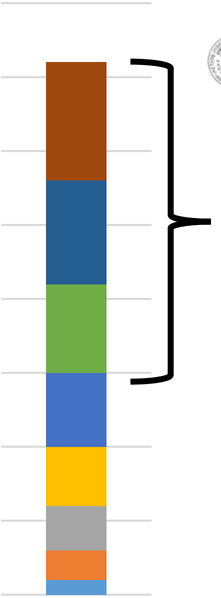
Key Terms:
Individual: Candidate solution
Genes: parameters (design variables) characterizing an individual
Chromosome (genotype): the set of genes of an individual

Flowchart of a Generalised (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
Database
Elite Archival Population P_e^g
NO
EA stopping criteria
YES
End

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.**

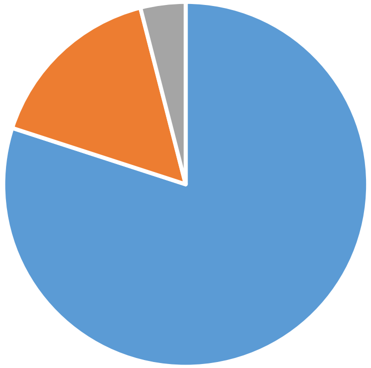
Proportional Selection: $P_\mu^{g-1} \cup P_\lambda^g$ **individuals are associated with a probability based on their fitness value (smaller values correspond to higher selection probability). A roulette wheel is considered, in which each individual is associated with a slot with angular width proportional 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 rank-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 (based on a roulette wheel). 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. [0110100111010100110]

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

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 Coding

For all genes in the phenotype (all optimisation variables), the user selects the corresponding number of binary digits, which determines its resolution. The corresponding/overall binary string becomes:

$$\underbrace{0101}_{b_1}\underbrace{101}_{b_2}.....\underbrace{10111}_{b_m}$$

For the b_m optimisation variable (L_m and U_m are its lower and upper bounds) , with *bit*$_m$ binary digits assigned to it, its (real) value is given by:

$$b_m = L_m + \frac{U_m - L_m}{2^{bit_m} - 1} \sum_{i=1}^{bit_m} 2^{i-1} d_{m,i}$$

Binary vs. Real Coding – Pros & Cons

Binary Coding:

- **Creates lengthy binary strings if high accuracy is required.**
- **Offers the maximum number of schemata per bit of information, compared to any other coding.**
- **Facilitates theoretical analysis and the development of new genetic/evolutionary operators.**
- **Smallest alphabet that allows a natural expression of the problem (Goldberg, 1989).**

Real Coding:

- **Problem-tailored genetic operators can readily be devised.**
- **High-cardinality alphabets contain more schemata (Antonisse, 1989).**

Binary-Gray Transformation

	Binary	Gray
3	0011	0010
4	0100	0110
…		
7	0111	0100
8	1000	1100

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

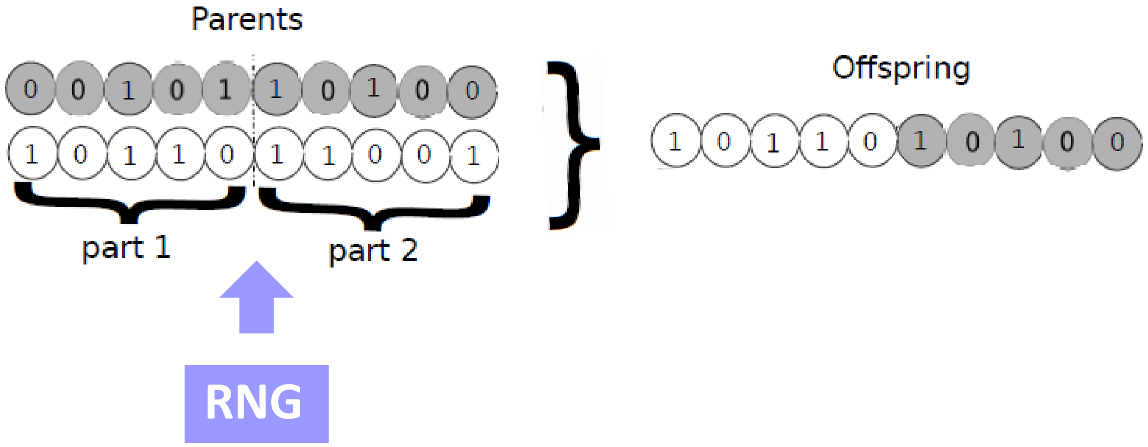
$$p_1 = [0 \quad 1 \quad 1 \quad 1] = 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 7$$
$$p_2 = [1 \quad 0 \quad 0 \quad 0] = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 8$$

$$G \cdot p_1^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = [0 \quad 1 \quad 2 \quad 2] = [0 \quad 1 \quad 0 \quad 0] = p_1^G$$
$$G \cdot p_2^T = \quad \ldots \quad = \quad \ldots \quad = [1 \quad 1 \quad 0 \quad 0] = p_2^G$$

Think about the inverse transformation!!!

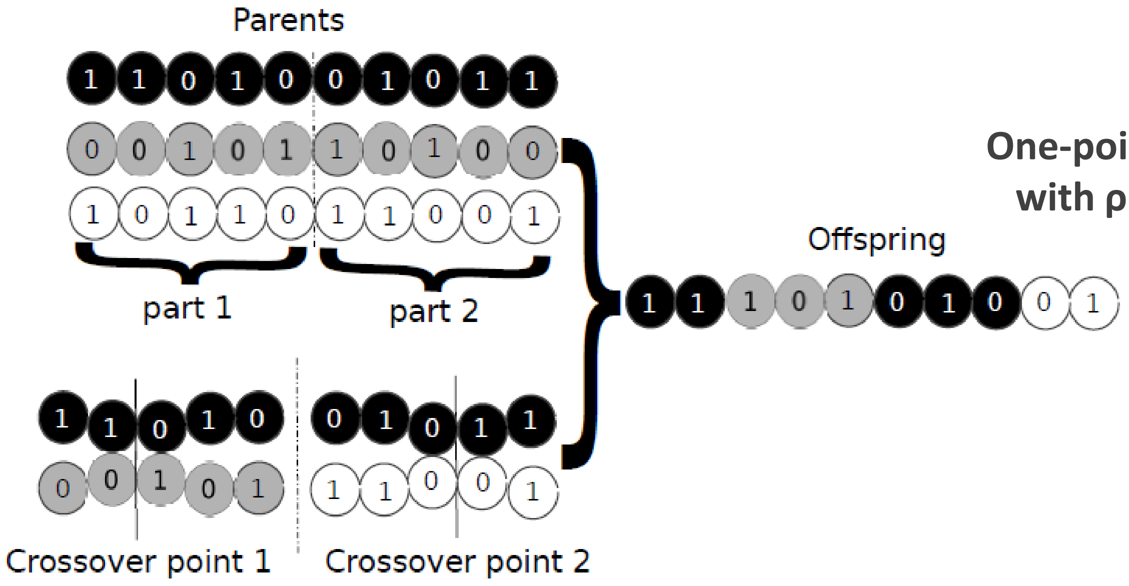
Binary Coding: Crossover

- **One-point crossover**

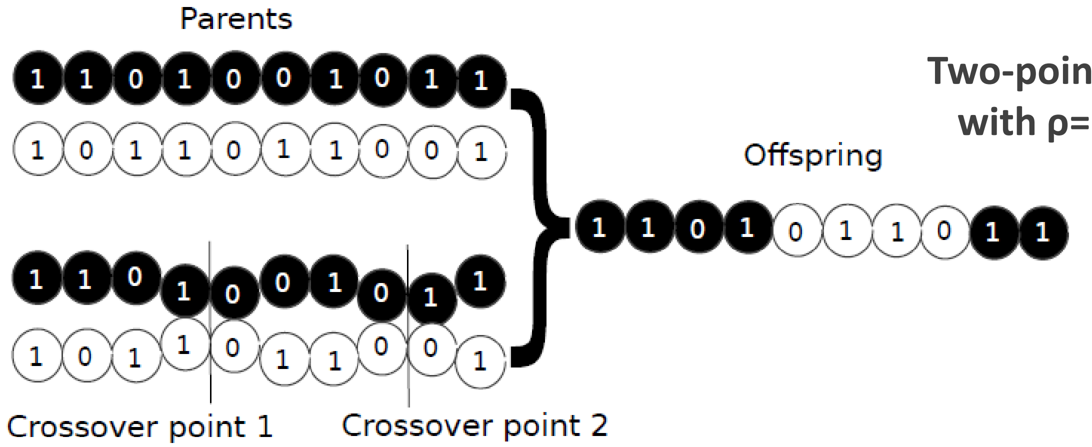


Binary Coding: Crossover

- One-point crossover
- Two-point crossover
- One- or two-point crossover per (design) variable
- etc.



One-point crossover with ρ=3 parents



Two-point crossover with ρ=2 parents

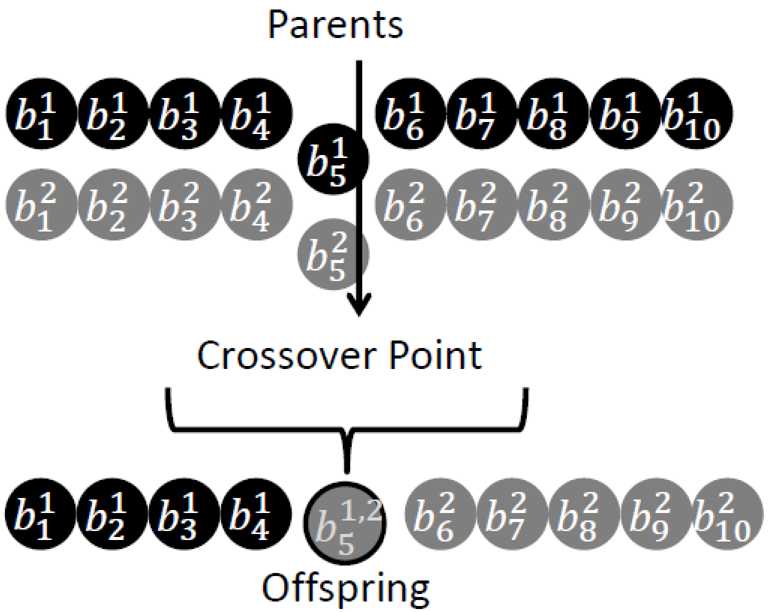
Real Coding: Crossover

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

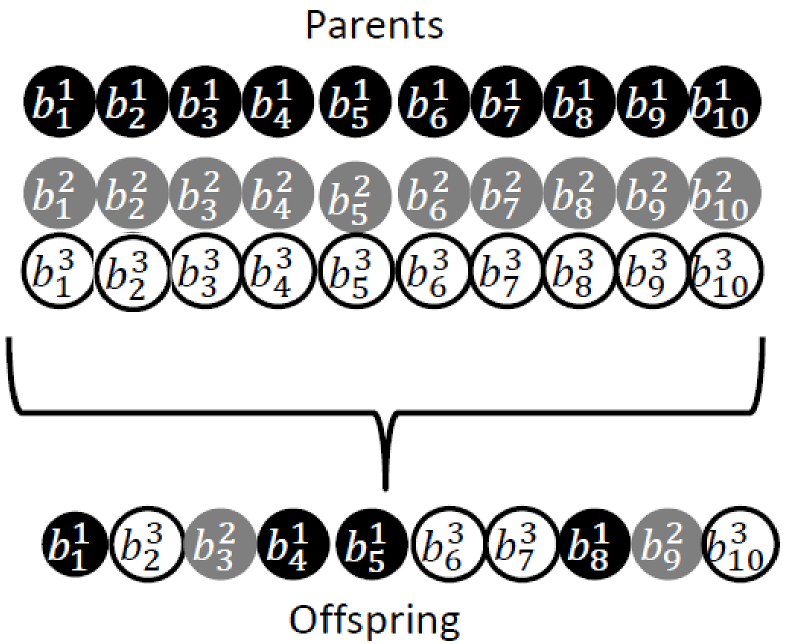
$$\vec{b} = \left[(1+\beta)\vec{b}^{P1} + (1-\beta)\,\vec{b}^{P2}\right] \quad \boldsymbol{\beta} = \begin{cases} (\mathbf{2}\boldsymbol{u})^{\mathbf{1}/(\boldsymbol{n}+\mathbf{1})} & ,\boldsymbol{u} < \mathbf{0.5} \\ \left(\dfrac{\mathbf{1}}{\mathbf{2}(\mathbf{1}-\boldsymbol{u})}\right)^{\mathbf{1}/(\boldsymbol{n}+\mathbf{1})}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

Simulated Binary Crossover (SBX)

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


Parents
b_1^1 b_2^1 b_3^1 b_4^1 b_5^1 b_6^1 b_7^1 b_8^1 b_9^1 b_{10}^1
b_1^2 b_2^2 b_3^2 b_4^2 b_5^2 b_6^2 b_7^2 b_8^2 b_9^2 b_{10}^2
Crossover Point
b_1^1 b_2^1 b_3^1 b_4^1 $b_5^{1,2}$ b_6^2 b_7^2 b_8^2 b_9^2 b_{10}^2
Offspring

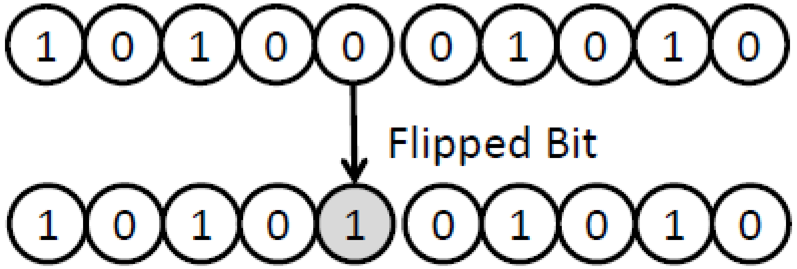
One-point crossover with ρ=2 parents


Parents
b_1^1 b_2^1 b_3^1 b_4^1 b_5^1 b_6^1 b_7^1 b_8^1 b_9^1 b_{10}^1
b_1^2 b_2^2 b_3^2 b_4^2 b_5^2 b_6^2 b_7^2 b_8^2 b_9^2 b_{10}^2
b_1^3 b_2^3 b_3^3 b_4^3 b_5^3 b_6^3 b_7^3 b_8^3 b_9^3 b_{10}^3
b_1^1 b_2^3 b_3^2 b_4^1 b_5^1 b_6^3 b_7^3 b_8^1 b_9^2 b_{10}^3
Offspring

Discrete crossover with ρ=3 parents

Mutation

Binary/Gray Coding:



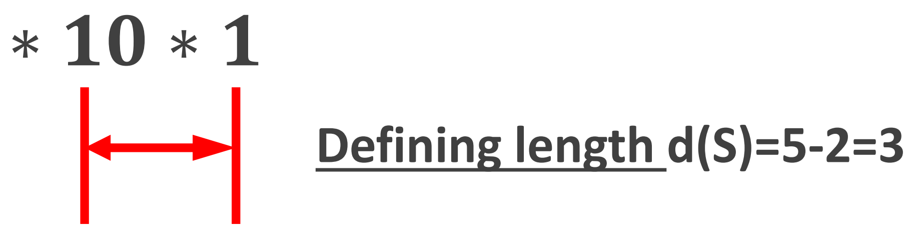
Real Coding:

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

D: depends on the current generation and the maximum generations

Schemata in Binary Strings

A **schema** is a similarity template describing a subset of strings with similarities at certain string positions (Holland, 1968)



$$* \, 1 0 * 1$$

Defining length d(S)=5-2=3

Order of schema o(S)=3 (fixed digits)

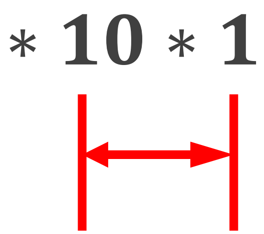
01001	01011	11001	11011

m=length of string(=5)
r=number of *'s (=2)

2^m possible schemata

2^r strings matched by this schema

Why dealing with Schemata? – Schema Theorem



$$*\,\mathbf{10}\,*\,\mathbf{1}$$

<u>Defining length </u>d(S)=5-2=3

<u>Order of schema </u>o(S)=3 (fixed digits)

The order of a schema affects its survival probabilities during mutation.

The defining length of a schema affects its survival probabilities during crossover

Schema Theorem (1/4) – Effect of Parent Selection

(m_g) examples of a particular schema (S), in the current generation (g)

F(S) = average fitness of the strings matching schema (S)

$$m_{g+1} = m_g \frac{F(S)}{F_{mean,g}}$$

Reproductive Schema Growth Equation

$m_{g+1}>m_g$	if $F(S)>F_{mean,g}$
mg+1<mg	if $F(S)<F_{mean,g}$

$$m_{g+1} = m_g(1+k)$$

$$m_{g+1} = m_0(1+k)^{g+1}$$

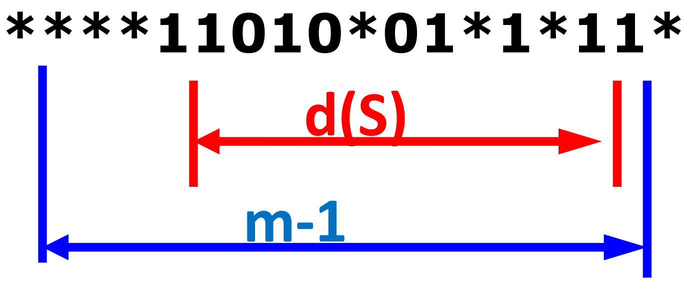
Long term effect of selection (k=constant)

Min. F

Schema Theorem (2/4) – Effect of Crossover

Possibility of destructing the schema S during crossover (depends on its defining length):

$$p_{des} = \frac{d(S)}{m-1}$$



****11010*01*1*11*

Possibility of maintaining the schema S:

$$p_{main} = 1 - p_{Xover}\frac{d(S)}{m-1}$$

Schema Growth Equation (after selection & crossover):

$$m_{g+1}(S) \geq m_g(S)\frac{F(S)}{F_{mean,g}(S)}(1 - p_{Xover}\frac{d(S)}{m-1})$$

Schema Theorem (3/4) – Effect of Mutation

Possibility of maintaining the schema S after mutation:

$$\boldsymbol{p_{surv} = (1 - p_{mut})^{o(S)} \approx 1 - p_{mut}\,o(S)}$$

****11010*01*1*11*

Final Schema Growth Equation:

$$m_{g+1}(S) \ge m_g(S)\frac{F(S)}{F_{mean,g}(S)}\left(1 - p_{Xover}\frac{d(S)}{m-1} - o(S)p_{mut}\right)$$

Short, low-order, above-average schemata should receive an (exponentially) increasing number of strings in the next generations

Schema Theorem (3/4) – Effect of Mutation

Possibility of maintaining the schema S after mutation:

$$p_{surv} = (1 - p_{mut})^{o(S)} \approx 1 - p_{mut}o(S)$$

******11010*01*1*11***

Final Schema Growth Equation:

$$m_{g+1}(S) \geq m_g(S)\frac{F(S)}{F_{mean,g}(S)}\left(1 - p_{Xover}\frac{d(S)}{m-1} - o(S)p_{mut}\right)$$

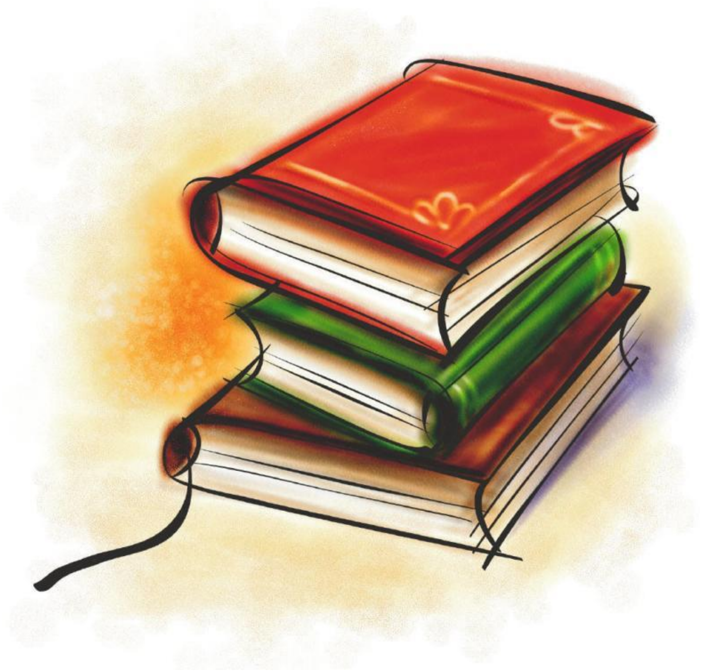
Short, low-order, above-average schemata should receive an (exponentially) increasing number of strings in the next generations

Schema Theorem (4/4) – Lessons Learned

- Short, low-order, above-average schemata should receive an (exponentially) increasing number of strings in the next generations (Schema Theorem).
- A GA (EA with binary coding) explores the search space by short, low-order schemata.
- GAs (EAs with binary coding) seek near-optimal performance through the juxtaposition of short, low-order, high-performance schemata (the so-called building blocks, Building Block Hypothesis).

- Exploration: seeking the global optimum in new and unknown areas in the search space.
- Exploitation: making use the knowledge gained from the previously examined points to guide the search towards new better points in the search space. [Holland 1975: for GAs]

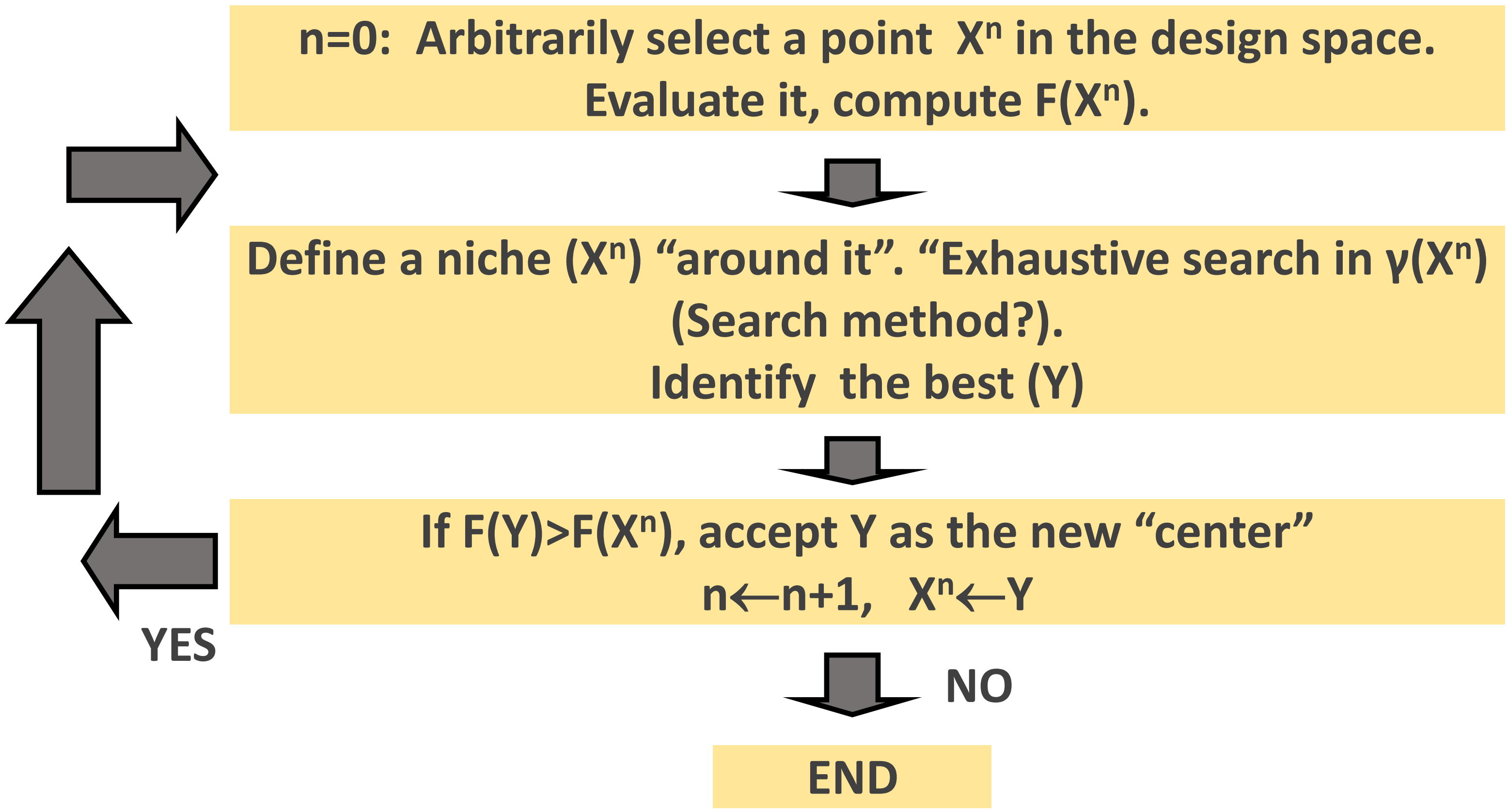
Under the assumptions of (a) infinite population size, (b) fitness function value accurately reflects the utility of a solution, and (c) genes in a chromosome do not interact significantly.



Other Stochastic Optimisation Methods (GFMs)

Other Stochastic Optimisation Methods (GFMs)

Target: max F



n=0: Arbitrarily select a point X^n in the design space.
Evaluate it, compute $F(X^n)$.

Define a niche (X^n) “around it”. “Exhaustive search in $\gamma(X^n)$
(Search method?).
Identify the best (Y)

If $F(Y)>F(X^n)$, accept Y as the new “center”
$n \leftarrow n+1, \quad X^n \leftarrow Y$

YES

NO

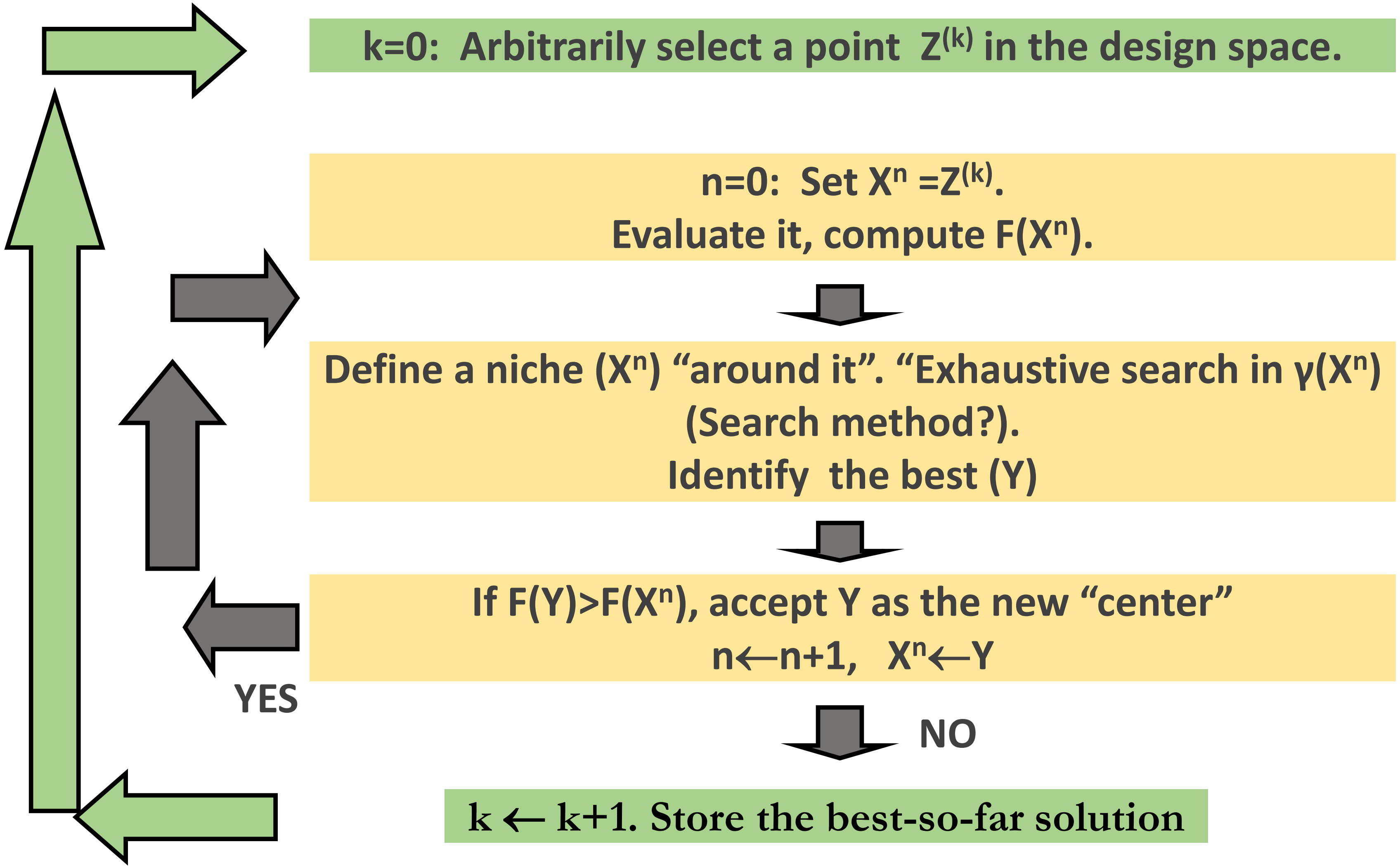
END

Other Stochastic Optimisation Methods (GFMs)

- **May end up at a local optimal solution.**
- **The user does not control the quality of the computed solution.**
- **The optimal solution to be computed depends on the initialisation.**
- **Difficult to control the computational cost.**

Iterated Hill Climber

Target: max F



k=0: Arbitrarily select a point $Z^{(k)}$ in the design space.

n=0: Set $X^n = Z^{(k)}$.
Evaluate it, compute $F(X^n)$.

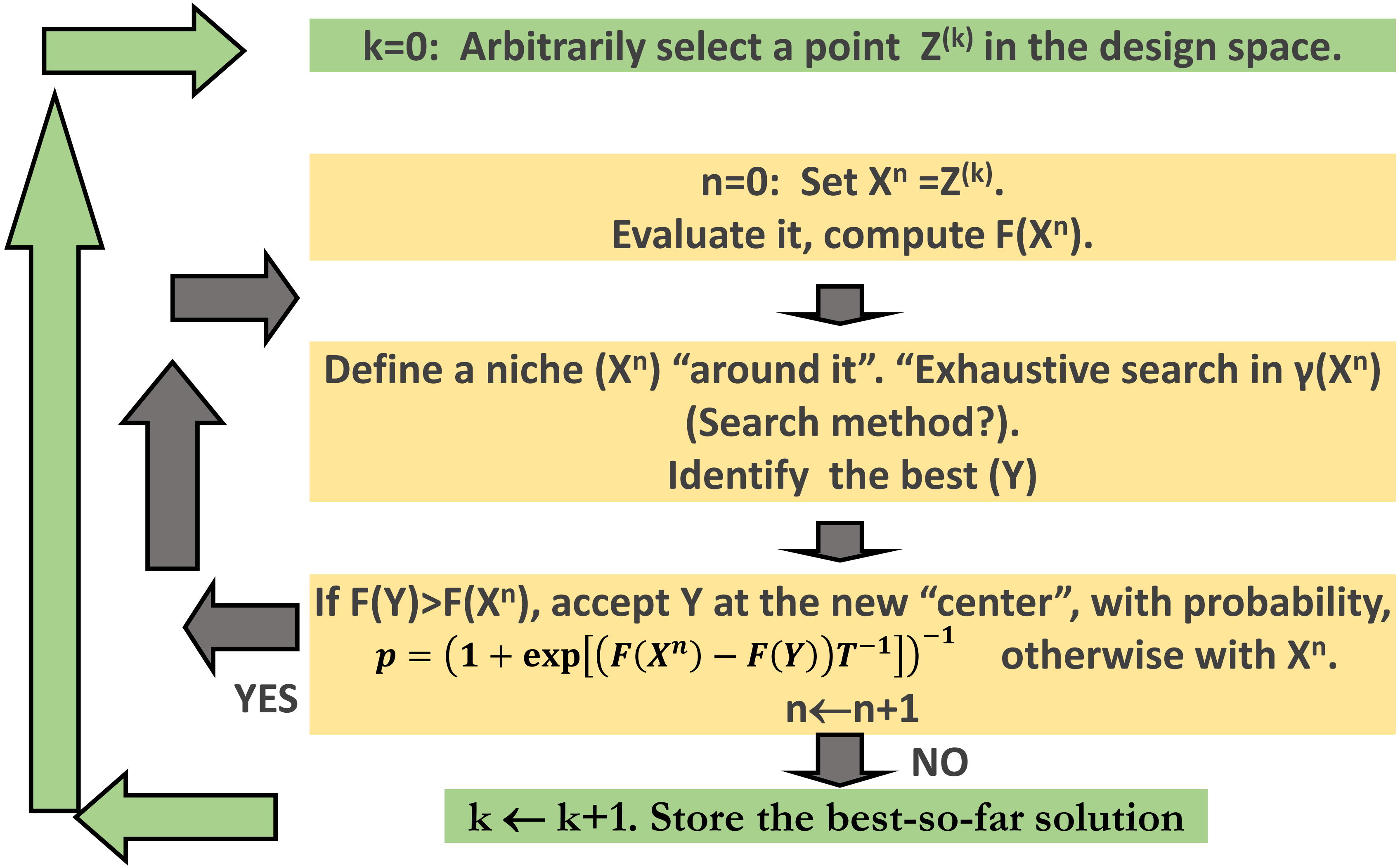
Define a niche (X^n) “around it”. “Exhaustive search in $\gamma(X^n)$
(Search method?).
Identify the best (Y)

If $F(Y)>F(X^n)$, accept Y as the new “center”
$n \leftarrow n+1, \quad X^n \leftarrow Y$

$k \leftarrow k+1$. Store the best-so-far solution

Stochastic Iterated Hill Climber

Target: max F



k=0: Arbitrarily select a point $Z^{(k)}$ in the design space.

n=0: Set $X^n = Z^{(k)}$.
Evaluate it, compute $F(X^n)$.

Define a niche (X^n) "around it". "Exhaustive search in $\gamma(X^n)$
(Search method?).
Identify the best (Y)

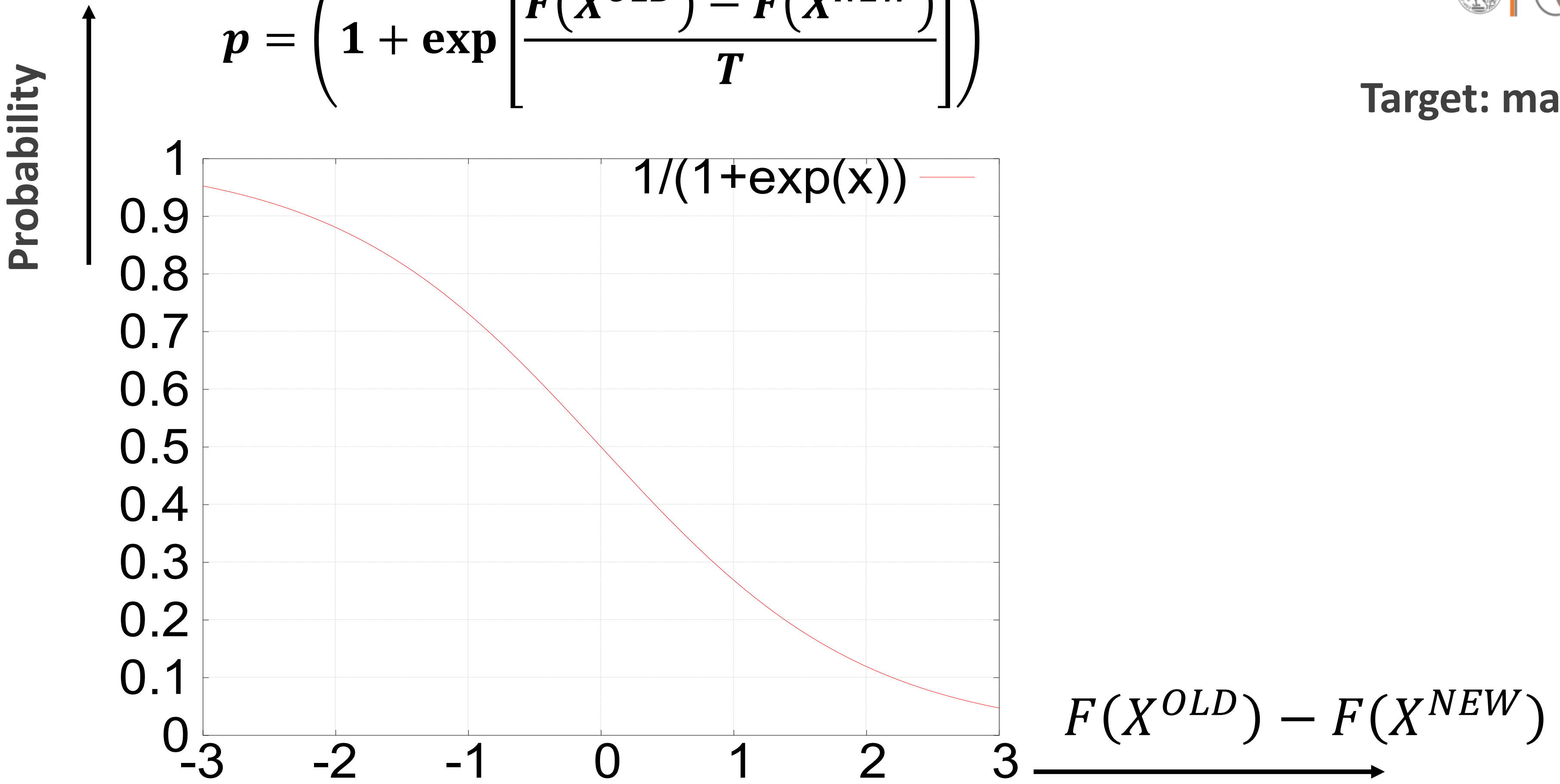
If $F(Y)>F(X^n)$, accept Y at the new "center", with probability,
$p = \left(1+\exp\left[\left(F(X^n)-F(Y)\right)T^{-1}\right]\right)^{-1}$ otherwise with X^n.
n←n+1

k ← k+1. Store the best-so-far solution

Stochastic Iterated Hill Climber

$$p = \left(1 + \exp\left[\frac{F(X^{OLD}) - F(X^{NEW})}{T}\right]\right)^{-1}$$

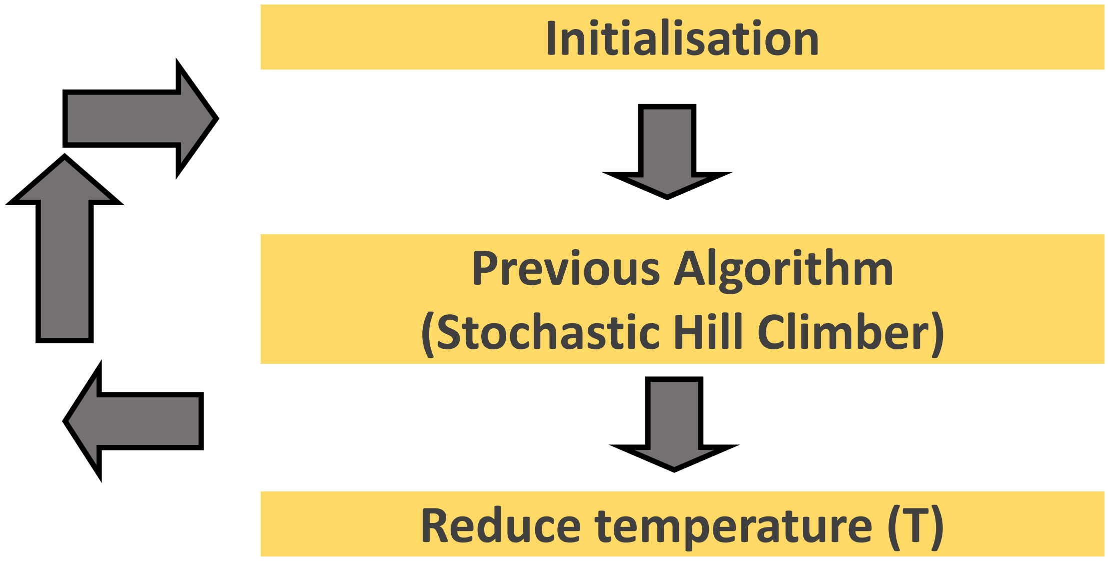
Target: max F



$F(X^{NEW})>F(X^{OLD})$ $F(X^{OLD})>F(X^{NEW})$

Simulated Annealing (SA)

Target: max F



Initialisation

Previous Algorithm (Stochastic Hill Climber)

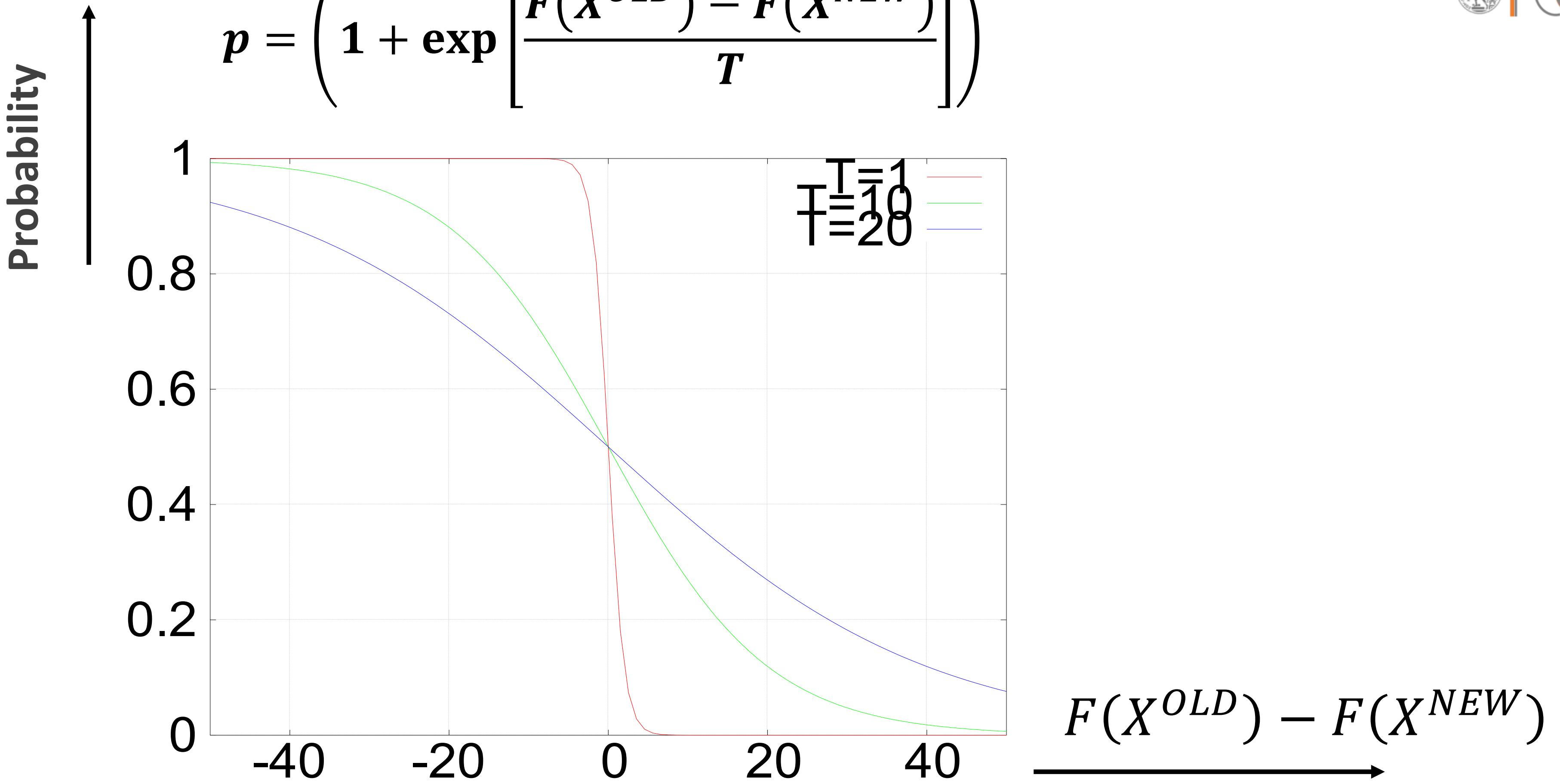
Reduce temperature (T)

$$p = \left(1 + \exp\left[\frac{F\left(X^{OLD}\right) - F\left(X^{NEW}\right)}{T}\right]\right)^{-1}$$

Annealing is a heat treatment process for materials like metals, glass, and polymers, involving heating to a specific temperature, holding, and controlled cooling to alter properties, primarily to increase ductility, reduce hardness, relieve internal stress, and improve workability for further shaping, essentially restoring a more flexible, original microstructure.

Stochastic Iterated Hill Climber – The Role of T

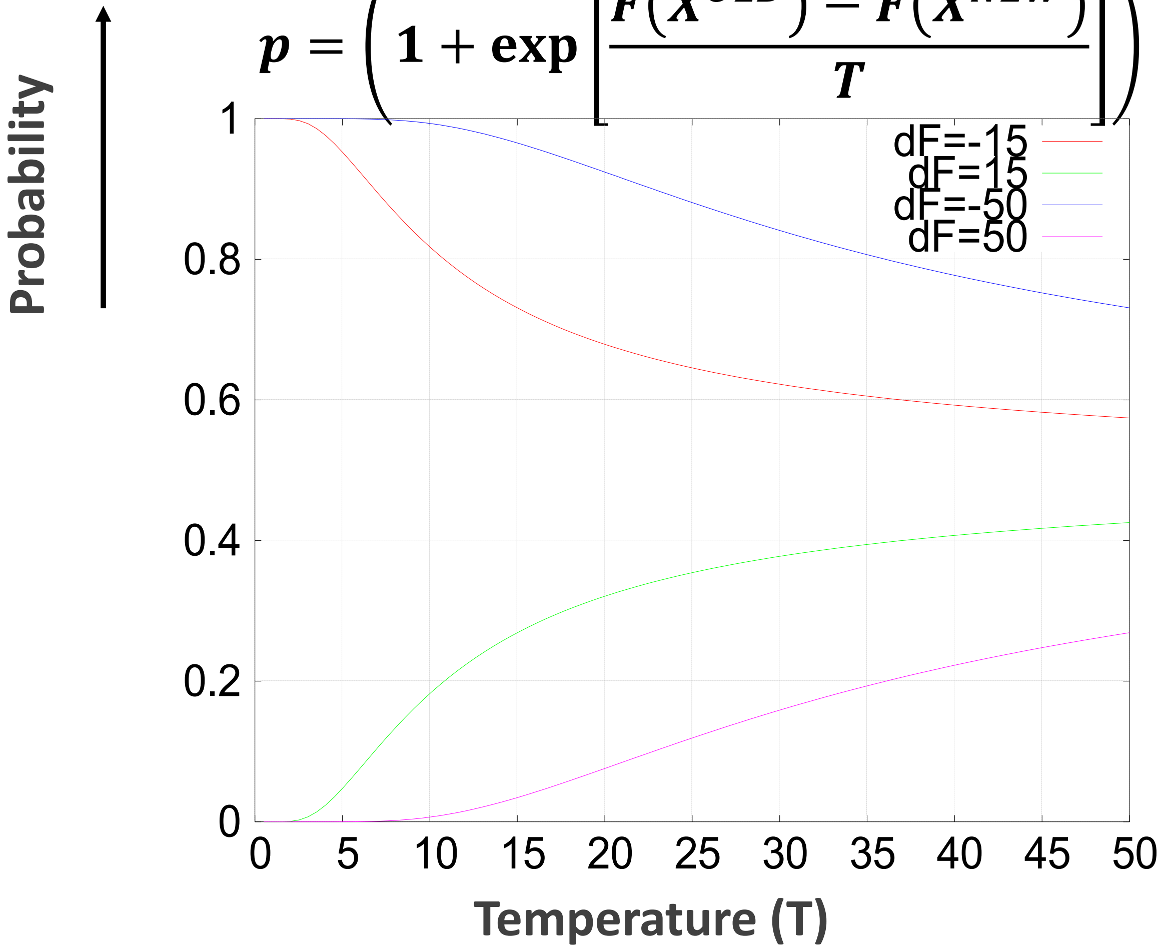
$$p=\left(1+\exp\left[\frac{F\left(X^{OLD}\right)-F\left(X^{NEW}\right)}{T}\right]\right)^{-1}$$



$F(X^{NEW})>F(X^{OLD})$ $F(X^{OLD})>F(X^{NEW})$

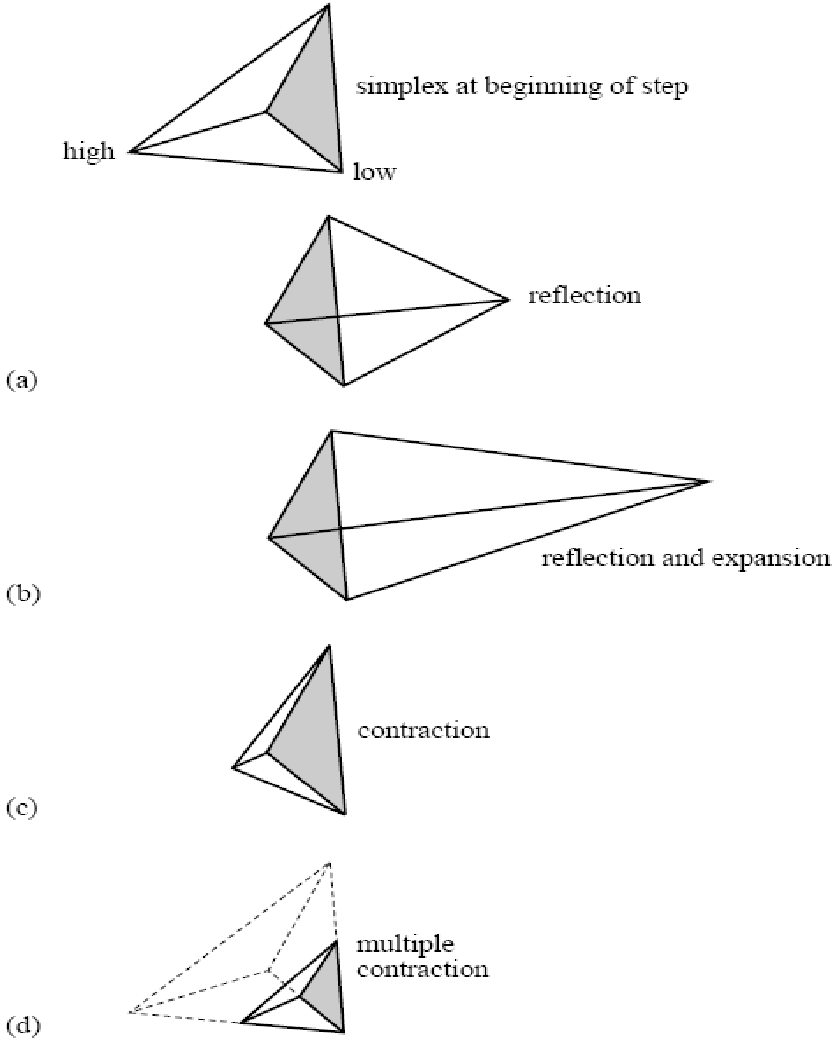
Stochastic Iterated Hill Climber

$$p = \left(1 + \exp\left[\frac{F\left(X^{OLD}\right) - F\left(X^{NEW}\right)}{T}\right]\right)^{-1}$$



$$dF = F\left(X^{OLD}\right) - F\left(X^{NEW}\right)$$

Nelder-Mead Method (Simplex or NL)



NL evolves a non-degenerate simplex with N+1 vertices in R^N.

In every NL iteration/cycle, the NP algorithm replaces the worst vertex of the current simplex with a hopefully better one, based on operations such as reflection, expansion, contraction, etc, trying to minimise the objective function.

(-) Near local minima, the NL algorithm may enter in oscillation, not converging to a single value.

Other Stochastic Population-based Optimisation Methods

Stochastic Population-based Optimisation/Search Methods:

- Genetic Algorithms (GA) & Evolution Strategies (ES)
- Particle Swarm Optimisation (PSO)
- Ant-Colonies Optimisation (ACO)
- Pity Beetle Algorithm
- Harmony Search
- …

Swarm Intelligence

Original definition: *“Systems of non-intelligent robots exhibiting collectively intelligent behavior evident in the ability to produce unpredictably specific (=not in a statistical sense) ordered patterns of matter in the external environment.” G. Beni and J. Wang, 1993.*

Explanation: Collective behaviour (learning and decision-making) of decentralised self-organised systems

Examples from nature:

- **Bird flocks**
- **Fish shoaling and schooling**
- **Ant colonies**
- **Bee swarms**
- **etc**

Particle Swarm Optimisation (PSO)

Modelled after the flocking behavior of birds and the shoaling and schooling of fishes.

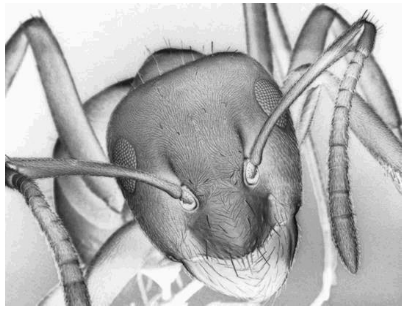
<u>Basic features:</u>

- **Particles (candidate solutions) move through the search (design) space, guided by their own experience and that of their neighbors.**
- **Each particle is described by its position vector and a velocity vector; assume timestep equal to unit. Velocity represents the distance that the particle will travel to reach its next position.**
- **Particles adjust their position based on a combination of personal best (cognitive influence) and group best (social influence) solutions; the momentum of the particle in motion is also considered.**

<u>Equivalence with EAs:</u>

- **Crossover & mutation → Velocity/momentum update formula**
- **Selection → Social influence**
- **Elitism → Cognitive influence**

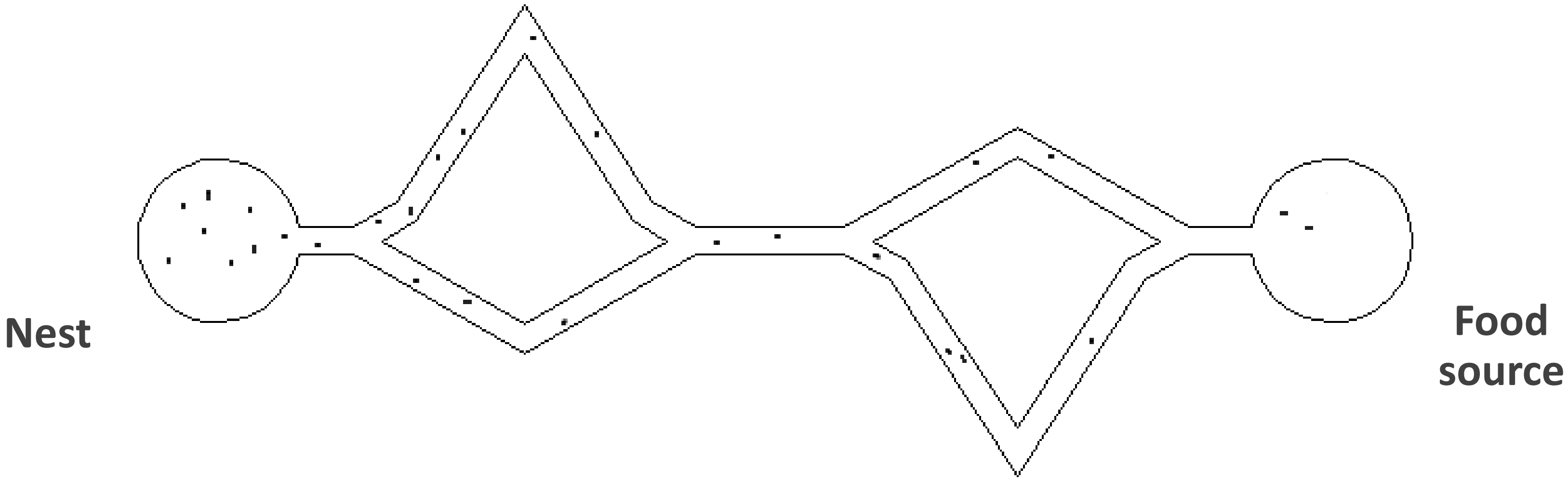
Ant Colonies Optimisation (ACO) (1/3)



Inspired by the foraging behavior of ants.

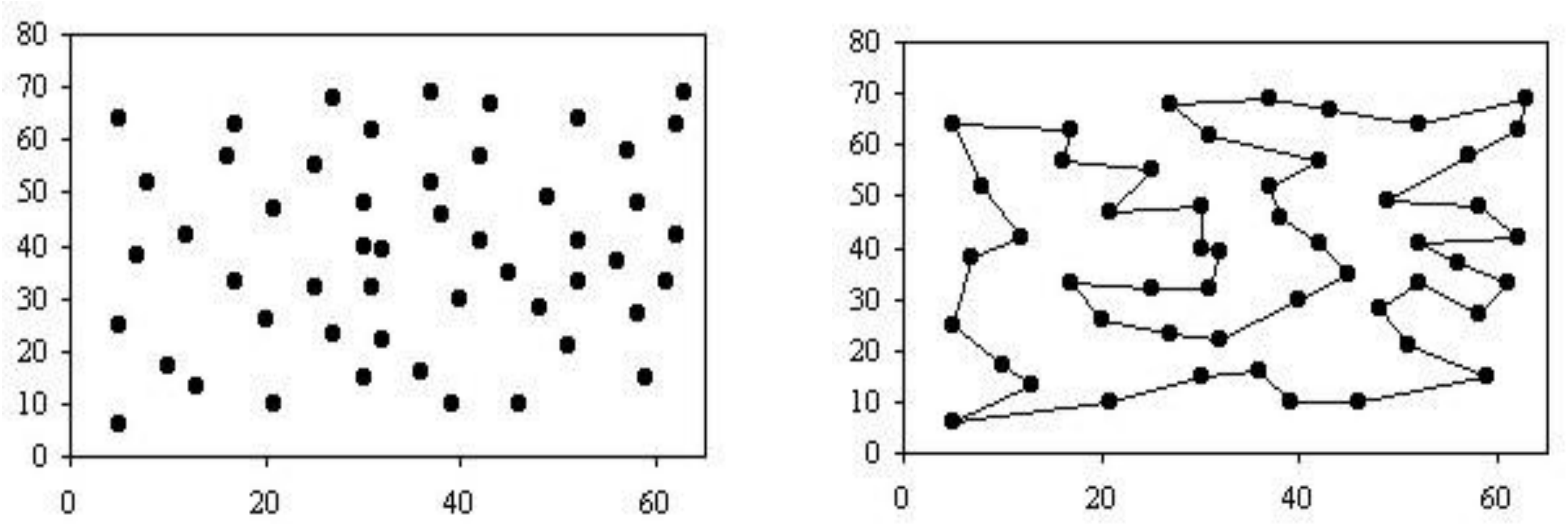
Basic features:

- **Ants (candidate solutions) move through the search (design) space, laying down pheromone guiding other ants to promising solutions (“food”).**
- **Over time, the pheromone trail strengthens for optimal paths and evaporates for less efficient ones.**



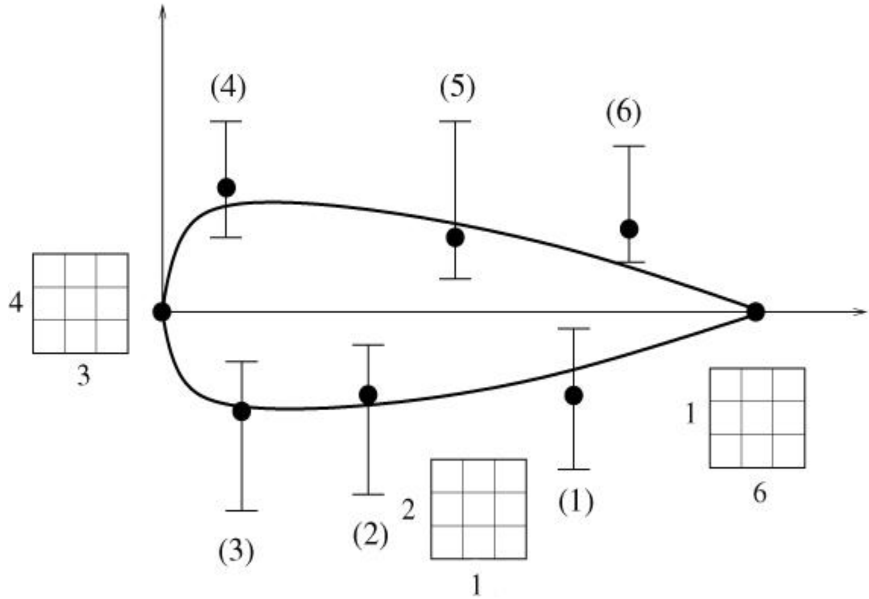
Ant Colonies Optimisation (ACO) (2/3)

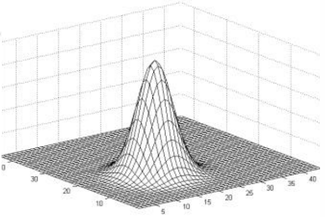
From the solution of the Traveling Salesman Problem to other applications, such its use in Aerodynamic ShpO.

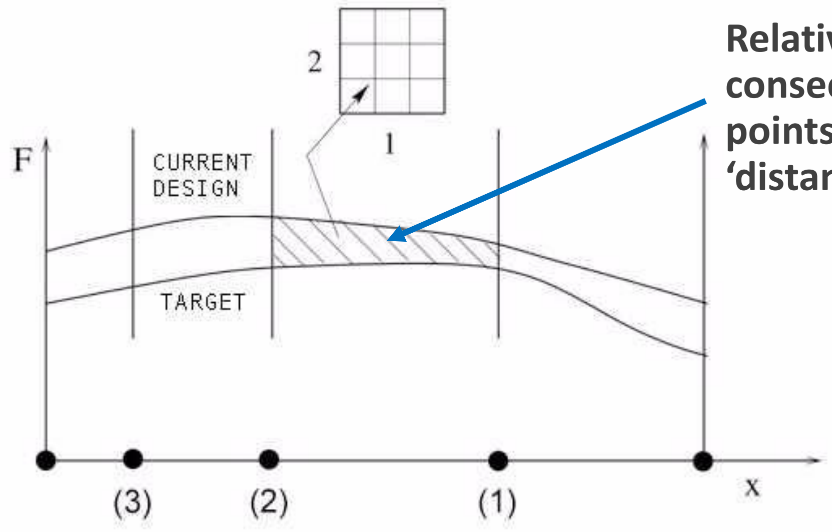


Ant Colonies Optimisation (ACO) (3/3)

From the Traveling Salesman Problem to other applications, such its use in Aerodynamic ShpO.







Relative position of two consecutive control points for defining the 'distance' parameter.

Ants should visit all territories associated with the DoFs (vertical lines). The sequence of territories is known and the tour is closed. The ACO algorithm aims at minimising the "length" of the path (objective function).

The destination of each ant (next territory) depends on two parameters, namely the distance (local) and by the pheromone trail (global).

Chicken Swarm Optimisation (CSO) (1/6)

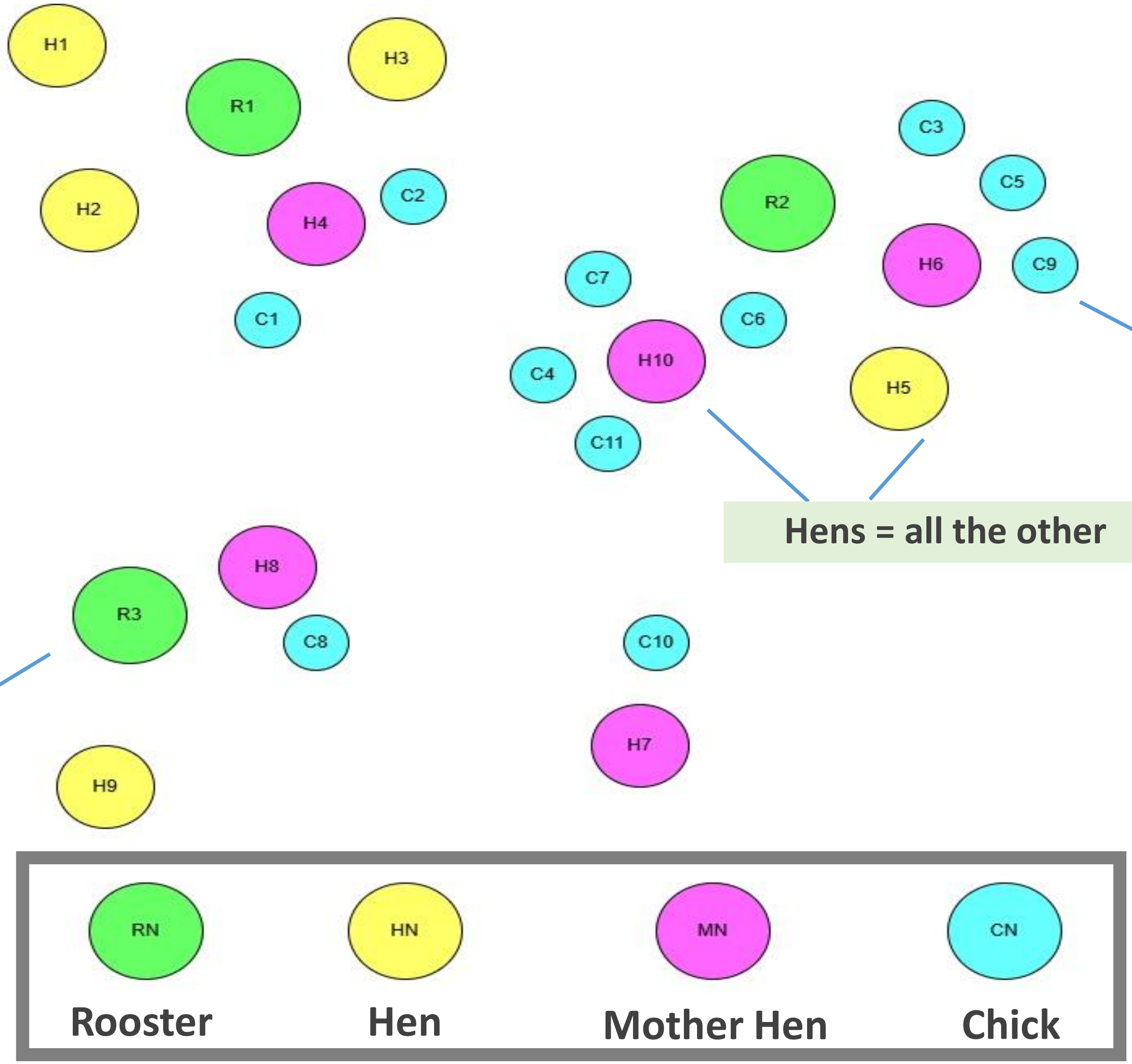
Inspired by the foraging behavior of chickens, CSO mimics the organic behavior of their flocks. Chicken coordinate their food-finding efforts in groups.

Basic features:

- **The flock contains three distinct populations, namely the roosters, hens and chicks.**
- **Chickens easily cooperate with others; taken as a swarm, they coordinate themselves as a team to search for food under specific hierarchal order.**
- **The population is divided into groups. Each group contains one rooster, a few hens and several chicks.**
- **Every hen in a flock has a matching rooster; every chick has a corresponding mother hen.**
- **For each group, the leading rooster actively seeks out food and defends group's area. Hens follow roosters while foraging and chicks typically feed near their mothers.**

👓 Binhe Chen, et al., "A comprehensive survey on the chicken swarm optimization algorithm and its applications: state-of-the-art and research challenges", Artificial Intelligence Review 57:170, 2024.

Chicken Swarm Optimisation (CSO) (2/6)



Chicks = the chickens with worst fitness values

Hens = all the other

Roosters = the chickens with best fitness values; roosters lead a group

Chicken Swarm Optimisation (CSO) (3/6)

Roosters' Motion (in a minimisation problem; SOO):

New location of the i-th rooster, affected by another (arbitrarily selected) rooster k:

$$x_{i,j}^{t+1} = x_{i,j}^{t} \cdot (1 + \mathcal{N}(0, \sigma^2))$$

$$\sigma^2 = \begin{cases} 1, & \text{if } f_i \leq f_k, \\ \exp\left(\frac{f_k - f_i}{|f_i| + \epsilon}\right), & \text{otherwise} \end{cases}$$

Roosters with the lower F value can search for food in a wider range than those with a higher value (in a minimisation problem).

Chicken Swarm Optimisation (CSO) (4/6)

Hens' Motion:

Each hen follows its group-mate rooster (x_{r1}) but is also affected by another randomly selected rooster (x_{r2}).

$$x_{i,j}^{t+1} = x_{i,j}^{t} + S_1 \cdot \mathrm{Rand}(0,1) \cdot \left(x_{r_1,j}^{t} - x_{i,j}^{t}\right) + S_2 \cdot \mathrm{Rand}(0,1) \cdot \left(x_{r_2,j}^{t} - x_{i,j}^{t}\right)$$

$$S_1 = \exp\left(\frac{f_i - f_{r_1}}{|f_i| + \epsilon}\right) \qquad S_2 = \exp\left(\frac{f_{r_2} - f_i}{|f_{r_2}| + \epsilon}\right)$$

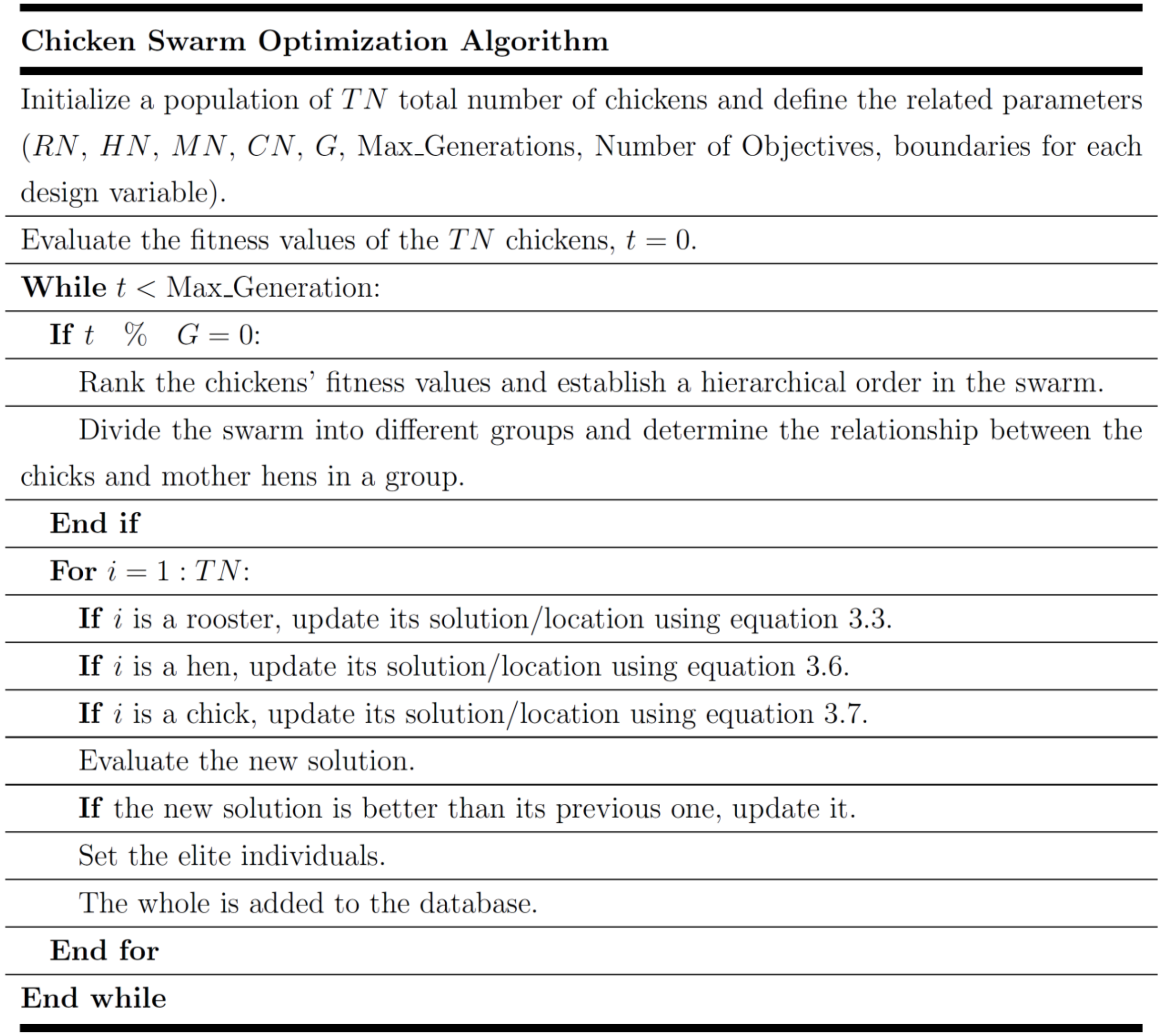
Chicken Swarm Optimisation (CSO) (5/6)

Chics' Motion:

Chics move around their mother (x_m) for food.

$$x_{i,j}^{t+1} = x_{i,j}^{t} + FL(0,2) \cdot (x_{m,j}^{t} - x_{i,j}^{t})$$

Chicken Swarm Optimisation (CSO) (6/6)



Chicken Swarm Optimization Algorithm

```
Initialize a population of $TN$ total number of chickens and define the related parameters ($RN$, $HN$, $MN$, $CN$, $G$, Max_Generations, Number of Objectives, boundaries for each design variable).
Evaluate the fitness values of the $TN$ chickens, $t = 0$.
While $t <$ Max_Generation:
  If $t \ \% \ G = 0$:
    Rank the chickens' fitness values and establish a hierarchical order in the swarm.
    Divide the swarm into different groups and determine the relationship between the chicks and mother hens in a group.
  End if
  For $i = 1 : TN$:
    If $i$ is a rooster, update its solution/location using equation 3.3.
    If $i$ is a hen, update its solution/location using equation 3.6.
    If $i$ is a chick, update its solution/location using equation 3.7.
    Evaluate the new solution.
    If the new solution is better than its previous one, update it.
    Set the elite individuals.
    The whole is added to the database.
  End for
End while
```

Differential Evolution (DE) (1/7)

Basic features:

- **Agents (candidate solutions) move through the search (design) space, by combining the positions of other agents in the population.**
- **If the new position (solution) of the agent is an improvement compared to its previous solution, then this replaces the previous, otherwise this is discarded.**

Differential Evolution (DE) – In its Standard Form (2/7)

NP = population size

$\vec{X}_{i,G}$ **, with i=0,1,2,…,NP-1 are the population members in generation G.**

For each i=0,1,2,…,NP-1

Differential Evolution (DE) – In its Standard Form (3/7)

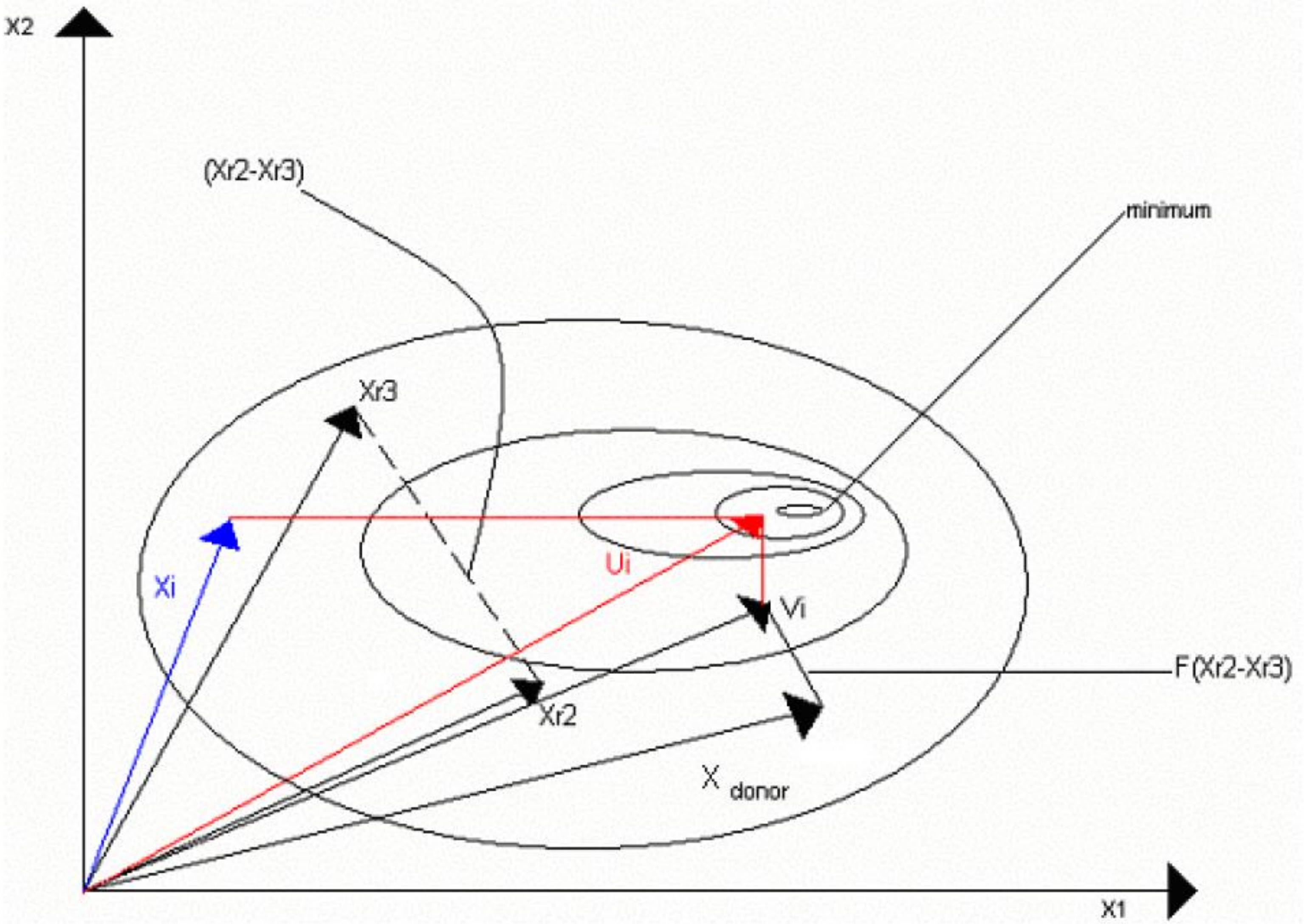
Mutation:

Select 3 individuals (a triplet) in the population. One of them randomly becomes the donor, the other are used to defining the perturbation to the donor. Generate a perturbed individual:

$$\vec{V}_i = \vec{X}_{r_1,G} + \mathrm{F} \cdot (\vec{X}_{r_2,\mathrm{G}} - \vec{X}_{r_3,G}),$$

with $r_1, r_2, r_3 \in [0, NP-1]$ **three different integers, with** $F \in [0,1+]$**.**

Differential Evolution (DE) – In its Standard Form (4/7)



Differential Evolution (DE) – In its Standard Form (5/7)

Crossover:

Combine test vector $\vec{\mathrm{V}}_{\mathrm{i}}$ and the population vector $\vec{\mathrm{X}}_{\mathrm{i,G}}$ and create offspring $\vec{\mathrm{U}}_{\mathrm{i}}$, for which:

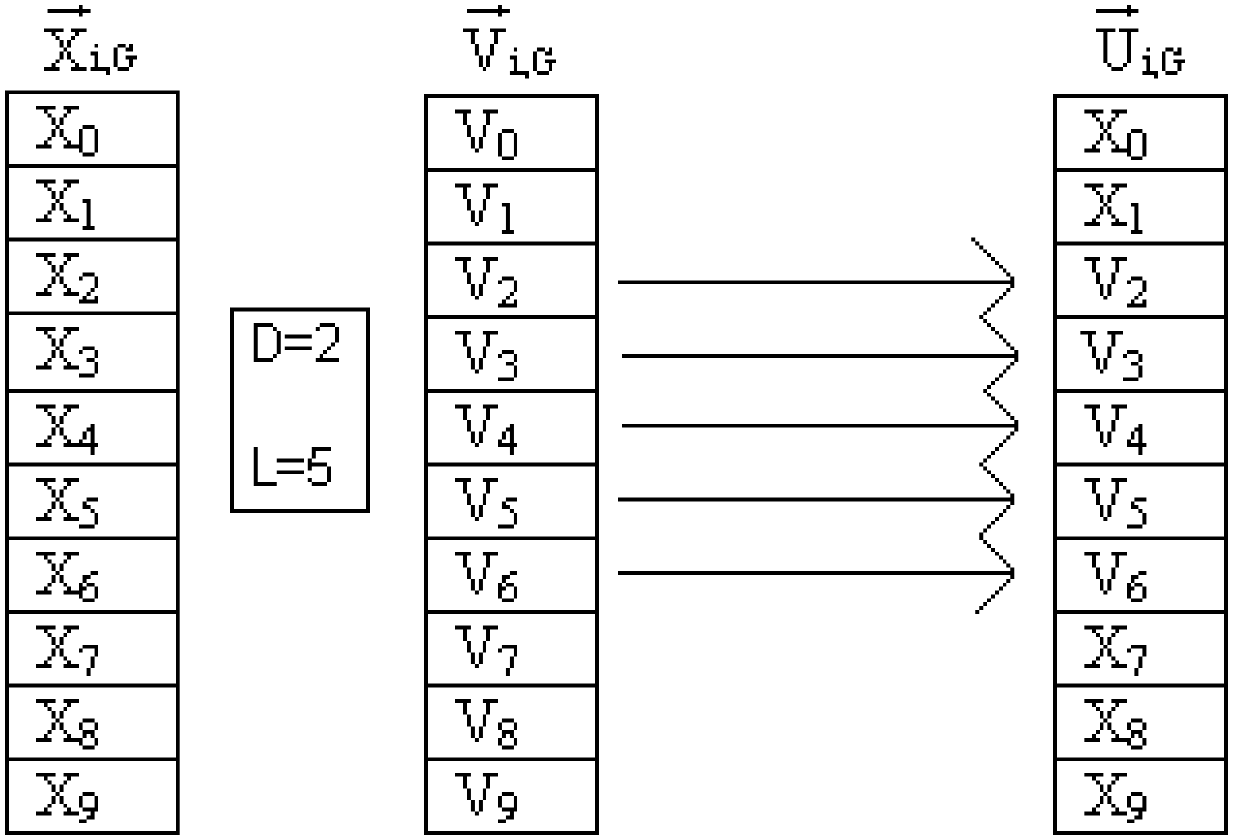
$$\vec{U}_i = (u_1, u_2, \ldots, u_n)^T \textbf{ with}$$

$$u_j = \begin{cases} v_j & \text{if} \quad \mathrm{Pr(L{=}v)} \le {C_r}^{v} \\ x_{j,G} & \quad\ \ \text{otherwise} \end{cases}$$

where C_r is the user-defined crossover factor.

Differential Evolution (DE) – In its Standard Form (6/7)

Crossover:



D=2

L=5

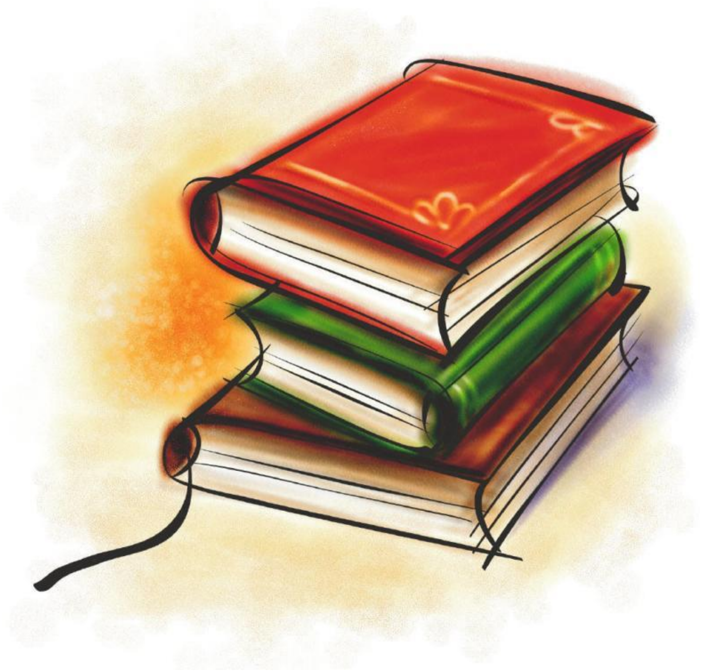
$\vec{X}_{iG}$	$\vec{V}_{iG}$	$\vec{U}_{iG}$
X_0	V_0	X_0
X_1	V_1	X_1
X_2	V_2	V_2
X_3	V_3	V_3
X_4	V_4	V_4
X_5	V_5	V_5
X_6	V_6	V_6
X_7	V_7	X_7
X_8	V_8	X_8
X_9	V_9	X_9

Differential Evolution (DE) – In its Standard Form (7/7)

Selection:

Select the members or the population of the next generation from the members of the current population and the corresponding trial vectors, according to the rule

- **If F($\overrightarrow{\mathrm{U}}_{\mathrm{i}}$)< F($\overrightarrow{\mathrm{X}}_{\mathrm{i,G}}$), $X_{i,G}$ will be replaced by** $\overrightarrow{\mathrm{U}}_{\mathrm{i}}$
- **Otherwise,** $\overrightarrow{\mathrm{X}}_{\mathrm{i,G}}$ **is retained in the next population.**



The Evolutionary Algorithm SYstem EASY

EASY – The Evolutionary Algorithm SYstem

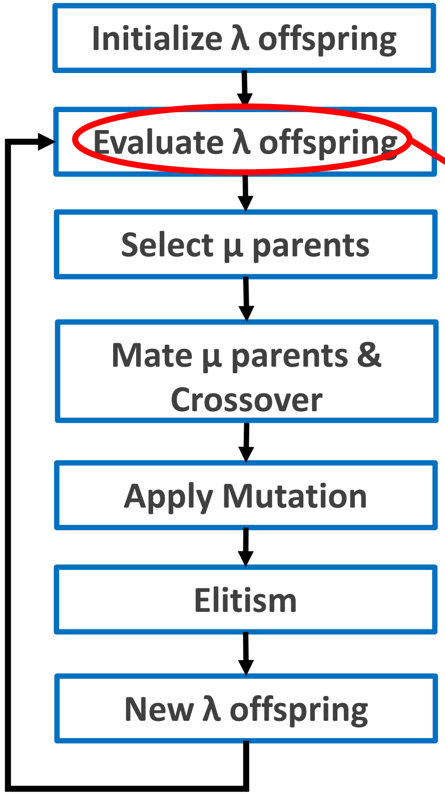
- ✓ **Problem Definition:**
  - Number of objectives M_o
  - Design variables
    - Number of design variables N
    - Bounds of each design variable: $b_i \in [b_i^{min}, b_i^{max}]$
    - Bits for the encoding $\Delta b_i = \dfrac{b_i^{max} - b_i^{min}}{2^{bits} - 1}$
  - Number of constraints M_c

EASY solves minimisation problems (SOO or MOO)

- ✓ **Evaluation Tool Definition:**
  - Script or batch file used for evaluating each candidate solution

- ✓ **Settings:**
  - Evolution settings
  - Metamodels settings /see upcoming lectures/
  - etc

EASY – Communication with the Evaluation Tool (1/2)



Evaluation Pseudocode

```
for j=1 to λ
    write task.dat
    call evaluationScript
    read task.res
    read task.cns
```

Basic/hardcoded filenames:

✓ task.dat	(design vector)
✓ task.res	(objective vector)
✓ task.cns	(constraint vector)

User-defined filename:

evaluationScript.sh or task.bat

EASY – Communication with the Evaluation Tool (2/2)

Evaluation Pseudocode

```
for j=1 to λ
   write task.dat
   call evaluationScript
   read task.res
   read task.cns
```

task.dat

$$\text{N+1 lines}\left\{\begin{array}{l} N \\ b_1 \\ b_2 \\ \vdots \\ b_N \end{array}\right.$$

Output, i.e. provided by EASY

evaluationScript

preprocessing.exe
evaluation.exe
postprocessing.exe

User's responsibility

task.res

$$\text{M}_\text{o}\ \text{lines}\left\{\begin{array}{l} f_1 \\ f_2 \\ \vdots \\ f_{M_o} \end{array}\right.$$

task.cns

$$\text{M}_\text{c}\ \text{lines}\left\{\begin{array}{l} c_1 \\ c_2 \\ \vdots \\ c_{M_c} \end{array}\right.$$

Input, i.e. EASY must read data from these files

EASY – The Evaluation Script

evaluationScript:

- **preprocessing.exe**

  Reads task.dat and creates the input file of the PSM

- **evaluation.exe**

  The PSM

- **postprocessing.exe**

  Reads the output files of the PSM and writes the task.res and (if required) task.cns files

Constraint Handling in EAs – Escalated Penalties

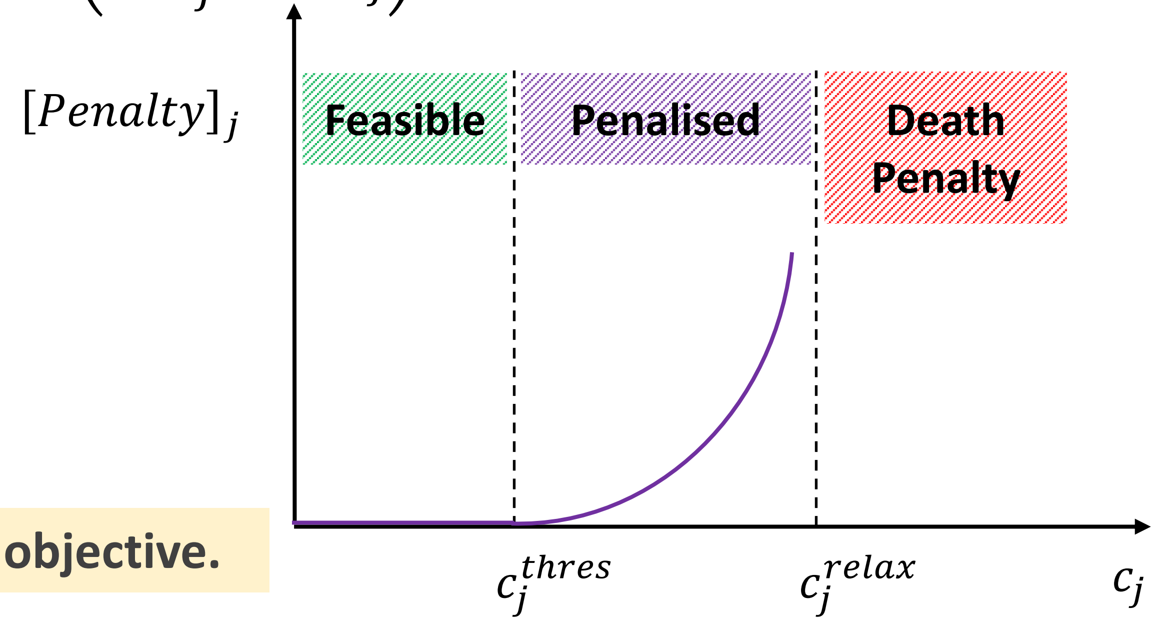
Exponential penalty if

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

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

Death penalty if

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



(* minimisation problem)

In MOO, this technique applies separately to each objective.

EASY - Output Files

EA_L1.log	: Log file
out_L1.log	: Output file that contains the optimal solution(s)
out_L1_GYY.log	: Intermediate output file @ generation YY
DB	: database of all evaluated individuals (binary/not editable file)