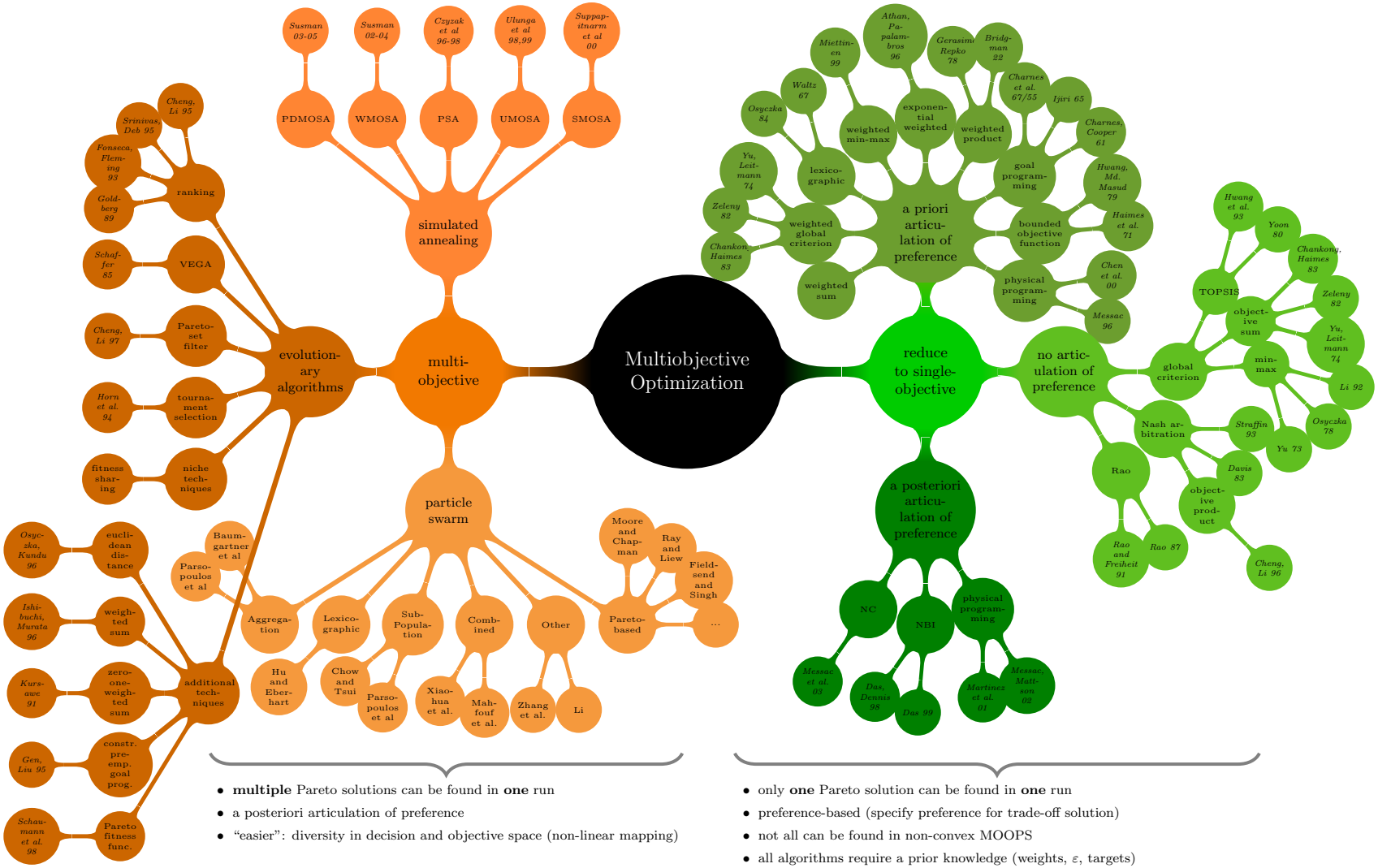
Multiobjective Optimisation Techniques

A. Adelmann (PSI-AMAS)

This talk is based largly on the PhD. thesis of Y. Ineichen

April 23, 2015



- **multiple** Pareto solutions can be found in **one** run
- a posteriori articulation of preference
- "easier": diversity in decision and objective space (non-linear mapping)

- only **one** Pareto solution can be found in **one** run
- preference-based (specify preference for trade-off solution)
- not all can be found in non-convex MOOPS
- all algorithms require a prior knowledge (weights, ε, targets)

Outline

1. History
2. A Simple but Instructive Example
3. Theoretical considerations
4. A Modern GA Implementation
5. Two Examples with the optPilot

History

[O.L. De Weck]

Rational people attempt to make the best decision within a specified set of possible alternatives.

- Multiobjective thinking originated in economics: the best referred to decisions taken by buyers and sellers (micro-economics) or governments (macro-economics), which simultaneously optimise or balance several criteria.
- Taxation: an optimal, average level of tax collected (% per $ of economic activity) maximizes the revenue available for the common good, while maintaining a sufficient incentive for individuals to earn income from their own work.

Francis Y. Edgeworth (1845-1926), King's College & Oxford

History cont.

Pareto on the other hand was a contemporary of Edgeworth, born in Paris in 1848, graduated from the University of Turin in 1870 (Civil Engineering) with a thesis: *The Fundamental Principles of Equilibrium in Solid Bodies*

- Pareto took up the study of philosophy and politics and was one of the first to analyse economic problems with mathematical tools
- In 1893, Pareto became the Chair of Political Economy at the University of Lausanne, where he created his two most famous theories:
  1. Circulation of the Elites
  2. The Pareto Optimum

Vilfredo Pareto (1848-1923)

History cont.

- The translation of Pareto's work into English in 1971 spurred the development of multiobjective methods in Applied Mathematics and Engineering.
- The growth of this field manifested itself particularly strongly in the United States with pioneering contributions by (Stadler 1979), (Steuer 1985) among many others.
- Theoretical aspects of multiobjective optimisation can be found in Japan (Sawaragi, Nakayama and Tanino, 1985).
- Over the last three decades the applications of multiobjective optimisation have grown steadily in many areas of Engineering and Design including the Particle Accelerator Community
- A particularly remarkable resource in this area is the website `http://www.lania.mx/~ccoello/EMOO/` created and maintained by C.A. Coello Coello.

Outline

1. History
2. A Simple but Instructive Example
3. Theoretical considerations
4. A Modern GA Implementation
5. Two Examples with the optPilot

Cost

Power

Fuel Economy

Size

Buying a Car

Conflicting criteria → Trade-offs

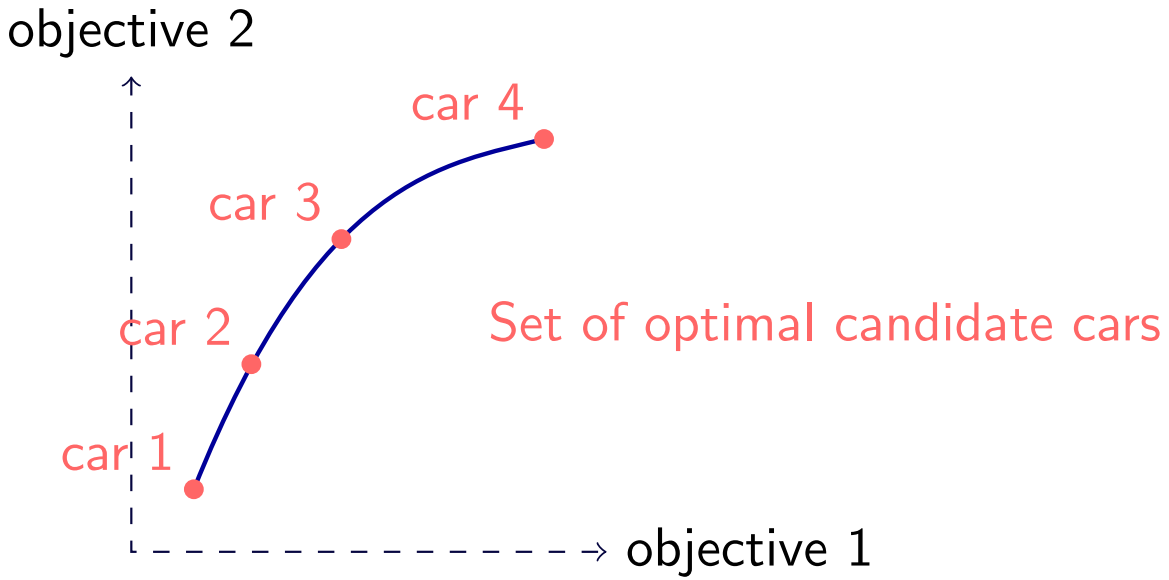
Conflicting criteria → Trade-offs

Subjective decision using
higher level information

Buying a Car

Conflicting criteria → Trade-offs

Conflicting criteria → Trade-offs

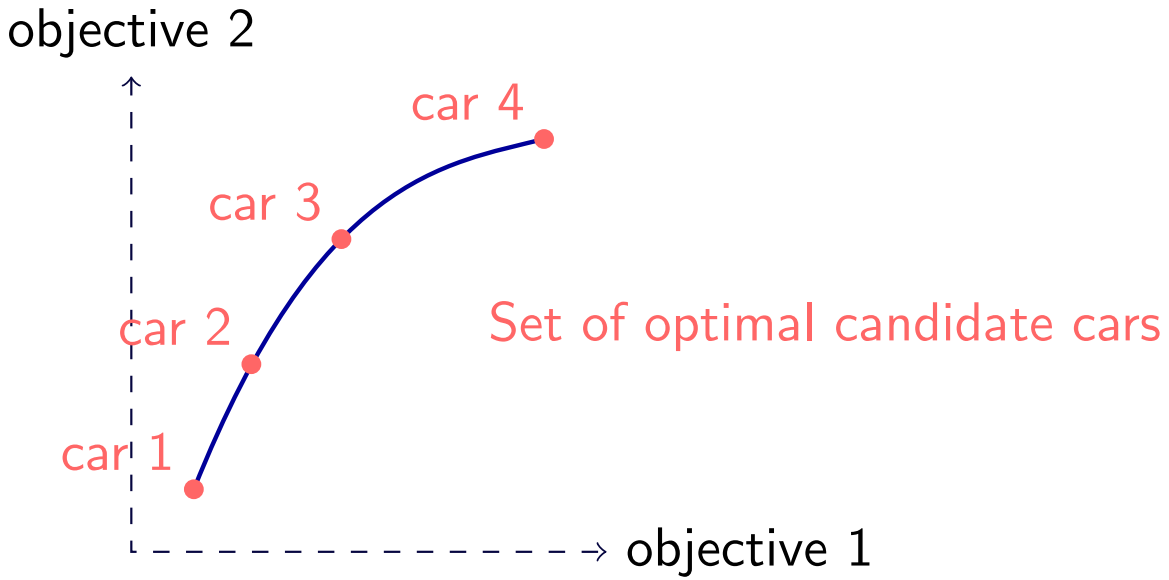


Subjective decision using
higher level information

Buying a Car

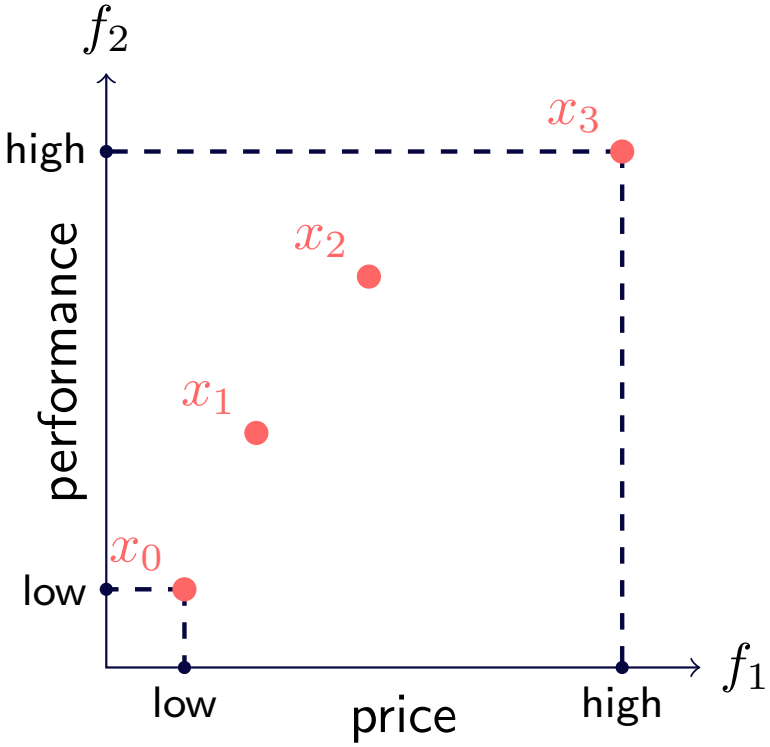
Conflicting criteria → Trade-offs

Conflicting criteria → Trade-offs



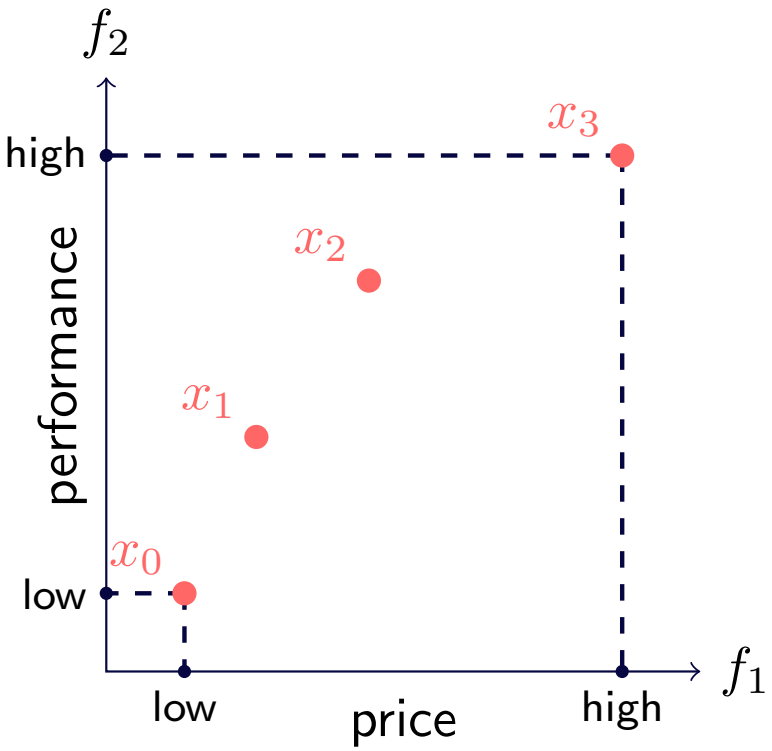
Subjective decision using
higher level information

Optimality?



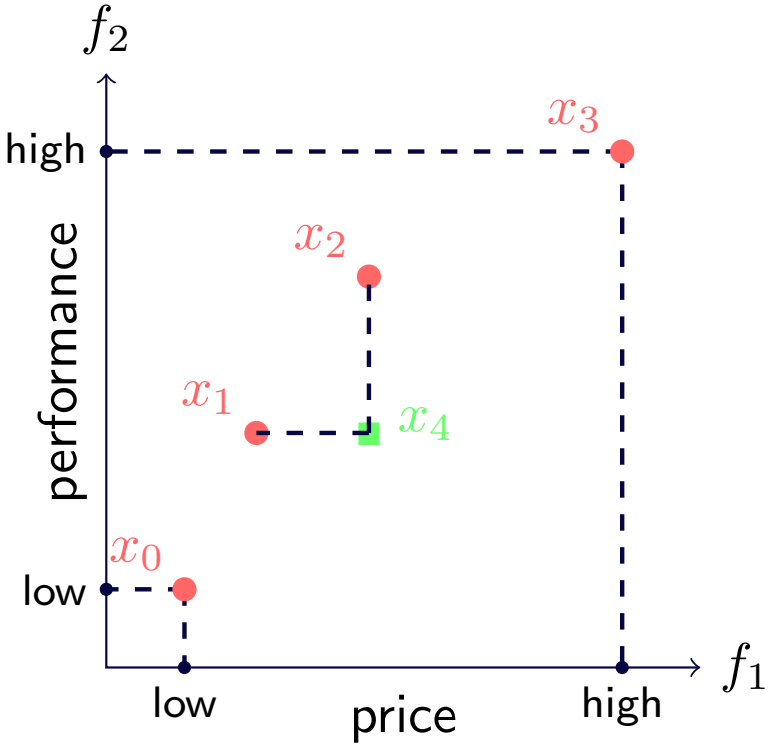
- conflicting objectives:
  minimize price
  maximize performance
- red points are "equally optimal": cannot improve one point without hurting at least one other solution $\rightarrow$ Pareto optimality
- x_4 is dominated by x_1 and x_2
- Pareto front

Optimality?



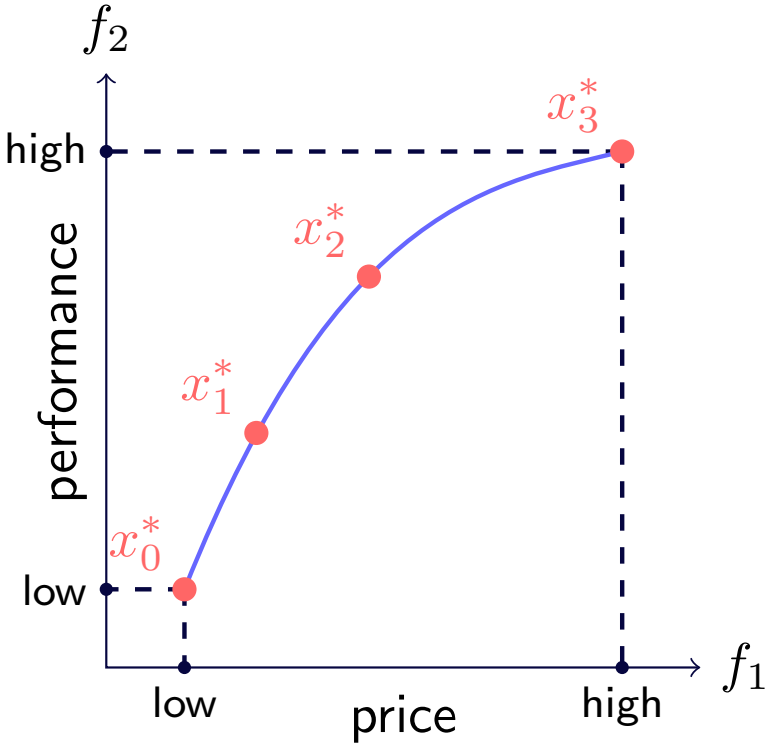
- conflicting objectives:
  minimize price
  maximize performance
- red points are "equally optimal":
  cannot improve one point without hurting at least one other solution
  $\rightarrow$ Pareto optimality
- x_4 is dominated by x_1 and x_2
- Pareto front

Optimality?



- conflicting objectives:
  minimize price
  maximize performance
- red points are "equally optimal": cannot improve one point without hurting at least one other solution
  $\rightarrow$ Pareto optimality
- x_4 is dominated by x_1 and x_2
- Pareto front

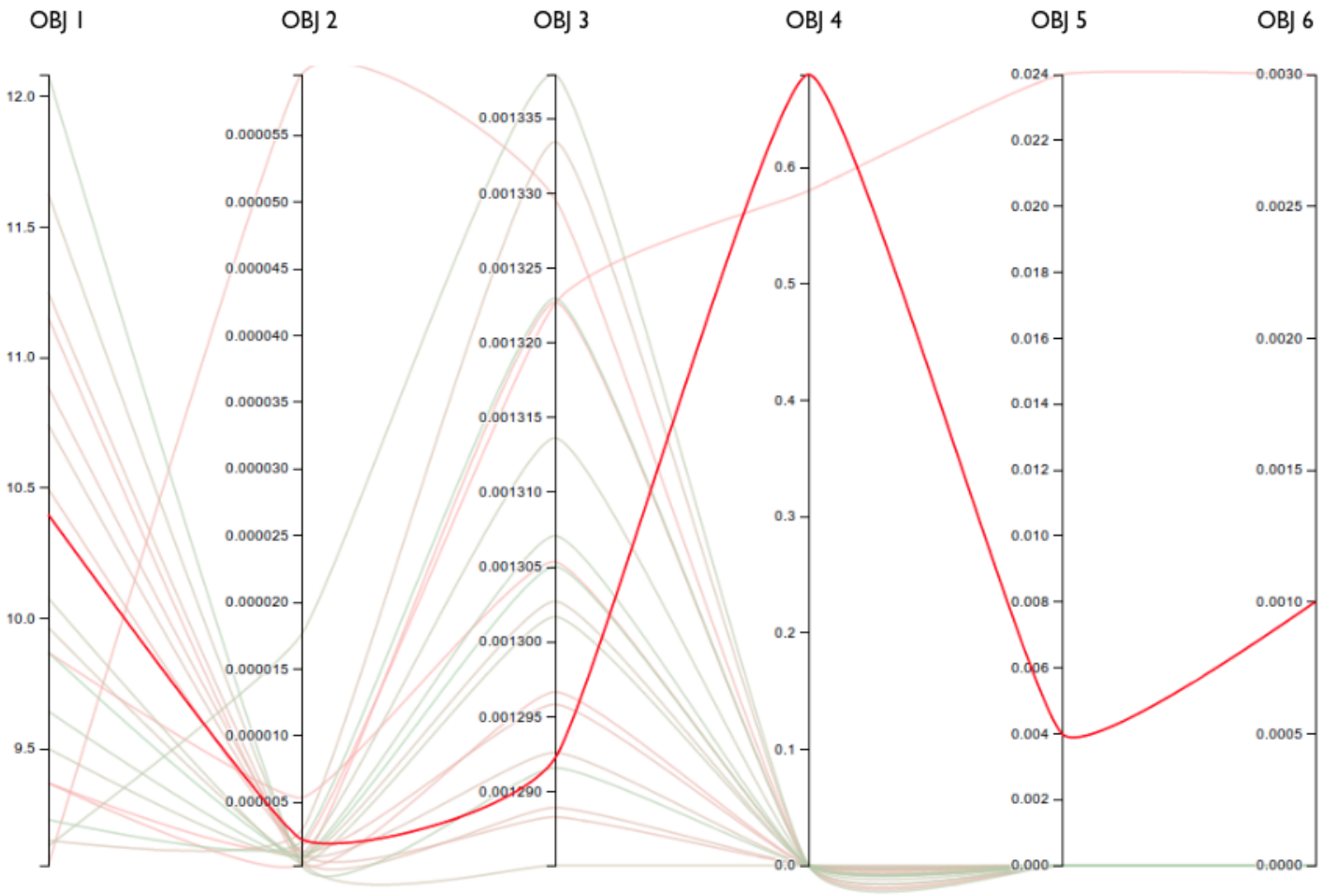
Optimality?



- conflicting objectives:
  minimize price
  maximize performance
- red points are "equally optimal": cannot improve one point without hurting at least one other solution
  $\rightarrow$ Pareto optimality
- x_4 is dominated by x_1 and x_2
- Pareto front

High Dimensional Data

Java/d3.js based Vis Tool, [Y. Ineichen, ETH Ph.D Thesis (2013)]



Outline

1. History
2. A Simple but Instructive Example
3. Theoretical considerations
4. A Modern GA Implementation
5. Two Examples with the optPilot

Formulation of the Multiobjective Optimisation Problem

[Parsopoulos,Vrahatis (2008)]

Denoting the feasible domain by $\mathbf{S} \in \mathbb{R}$, the problem is to minimise – **simultaneously** – all elements of the objective vector,

Objectives

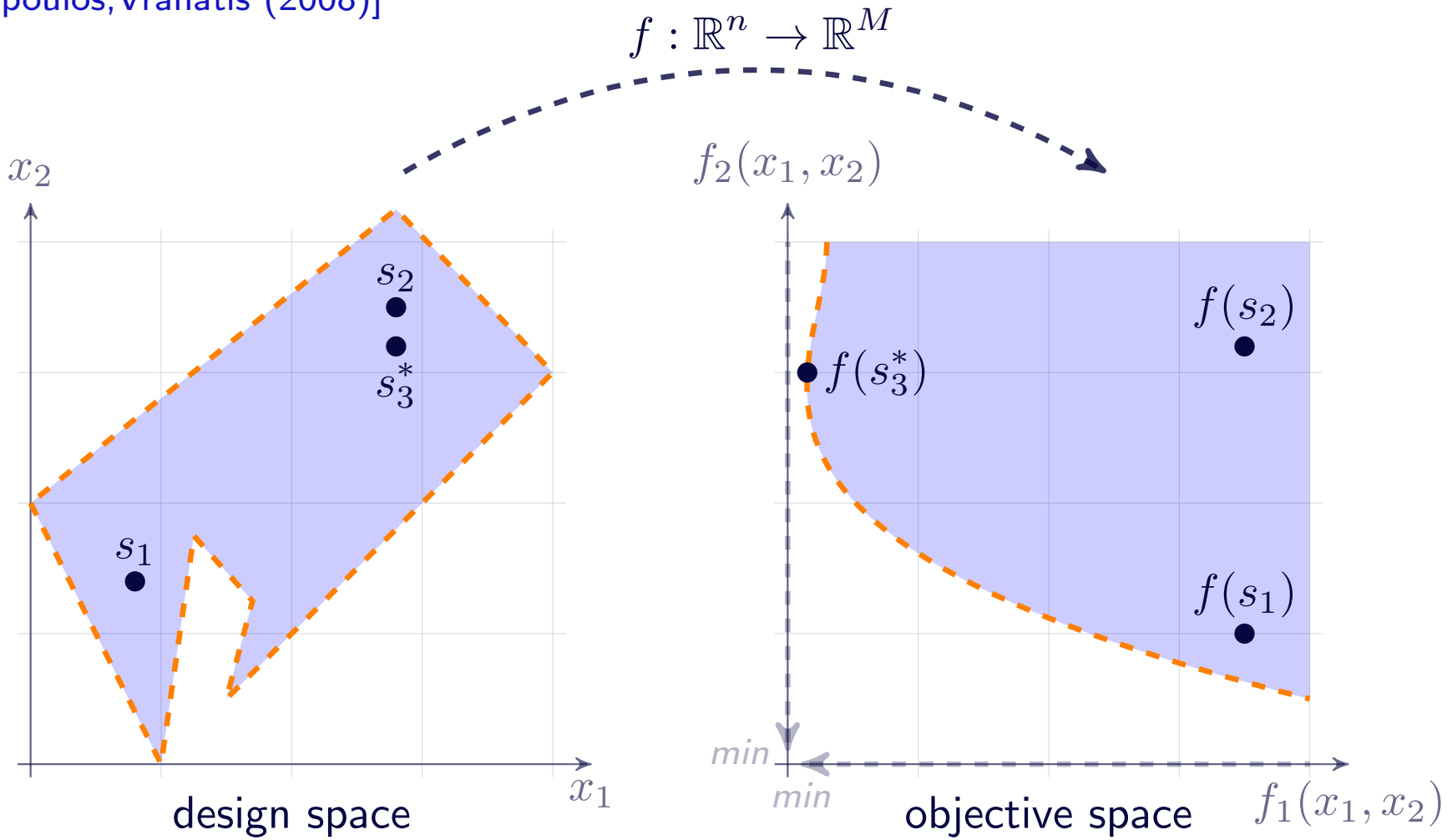
Design variables

$$
\begin{array}{lll}
\text{min} & f_m(\mathbf{x}) \in \mathbb{R} \text{ and } \mathbf{x} \in \mathbf{S}, & m = 1 \ldots M \\
\text{s.t.} & g_j(\mathbf{x}) \geq 0, & j = 0 \ldots J \\
 & h_k(\mathbf{x}) = 0, & k = 0 \ldots K \\
 & x_i^L \leq \mathbf{x} = x_i \leq x_i^U. & i = 0 \ldots n
\end{array}
$$

Constraints

Formulation of the Multiobjective Optimisation Problem

[Parsopoulos,Vrahatis (2008)]



The (non-linear) mapping $f : \mathbb{R}^n \to \mathbb{R}^M$ from design to objective space.

Formulation of the Multiobjective Optimisation Problem

[Parsopoulos,Vrahatis (2008)]

- Pareto front: the set of all solutions in the search space that are non-dominated by any solutions.
- Goal of multiobjective optimisation is obtain the Pareto front for further analysis.

Multiobjective optimisation methods can be broadly decomposed into two categories

1. Scalarisation approaches: the multiobjective problem is solved by translating it back to a single (or a series of) objective, scalar problems. This requires the formation of an overarching objective function which contains contributions from the sub-objectives in vector J.
2. Pareto approaches

Scalarization I

Weighted Sum Approach

Scalarization methods are based on the assumptions that

1. designer or decision-maker preferences are known before design solutions are found and that
2. the M objectives can be meaningfully combined to express a utility, U, dimensionless scalar quantity expressing the goodness of a particular design.

$$
\begin{aligned}
\text{min} \quad & U\{f_m(\mathbf{x})\} \in \mathbb{R} \text{ and } \mathbf{x} \in S, && m = 1 \ldots M \\
\text{where} \quad & U = \sum_{q=1}^{M} w_q f_q(\mathbf{x}), \text{with } w_q > 0 \text{ and } \sum_{q=1}^{M} w_q = 1 \\
\text{s.t.} \quad & g_j(\mathbf{x}) \geq 0, && j = 0 \ldots J \\
& h_k(\mathbf{x}) = 0, && k = 0 \ldots K \\
& x_i^L \leq \mathbf{x} = x_i \leq x_i^U. && i = 0 \ldots n
\end{aligned}
$$

Scalarization II

Weighted Sum Approach

- Formulated in this way the aggregate objective U always forms a strictly convex combination of objectives
- One of the issues in this method is the appropriate choice of λ
- In the case of two equally scaled objectives we get

$$U = \lambda J_1 + (1 - \lambda) J_2. \tag{1}$$

Finding optima for U as λ is changed gradually, in equal intervals, from $0 \ldots 1$ reveals a set of optimal solutions as the weight is gradually shifted from one objective to another.

Formulation of the Pareto Optimal Condition

A point $\mathbf{x}_1$ is dominating $\mathbf{x}_2$

1. the solution $\mathbf{x}_1$ is no worse than $\mathbf{x}_2$ in all objectives
2. the solution $\mathbf{x}_1$ is strytictly better than $\mathbf{x}_2$ in at least one objective.

$$\mathbf{x}_1 \preceq \mathbf{x}_2 \text{ iff } \begin{cases} f_m(\mathbf{x}_1) \geq f_m(\mathbf{x}_2), \ \forall m \in 1 \ldots M \\ f_j(\mathbf{x}_1) > f_j(\mathbf{x}_2), \ \exists j \in 1 \ldots M \end{cases}$$

The properties of the dominance relation include transitivity

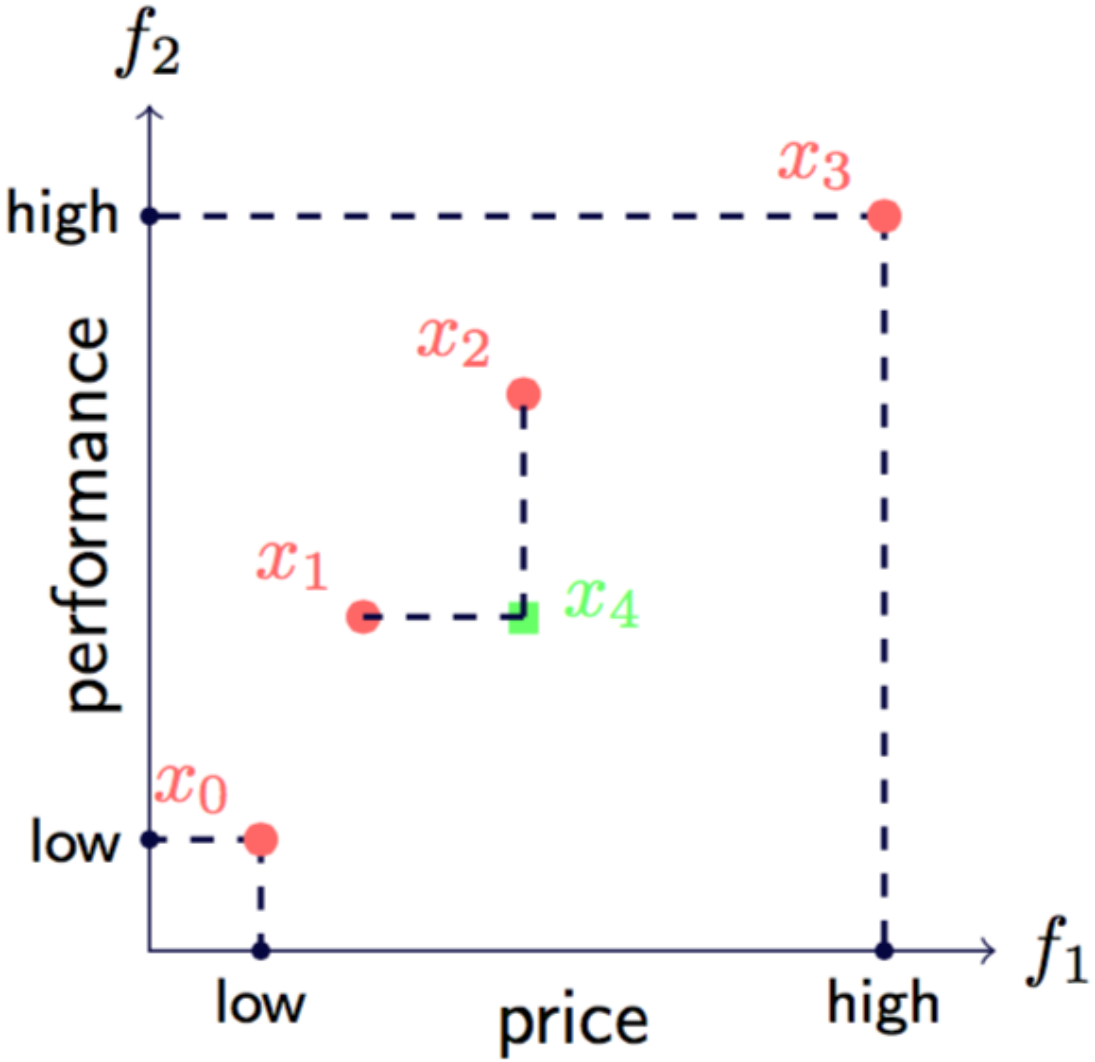
$$x_1 \preceq x_2 \wedge x_2 \preceq x_3 \Rightarrow x_1 \preceq x_3,$$

and asymmetry, which is necessary for an unambiguous order relation

$$x_1 \preceq x_2 \Rightarrow x_2 \not\preceq x_1.$$

Using the concept of dominance, the sought-after set of Pareto optimal solution points can be approximated iteratively as the set of non-dominated solutions.

Formulation of the Pareto Optimal Condition



Remarks on Pareto Optimality I

- Deciding if a point truly belongs to the set of Pareto optimal solutions is NP-hard[1] however many efficient heuristics exists.
- A comprehensive or full-factorial evaluation of the design space is often impossible due to the n-dimensionality of the design vector, $\mathbf{x}$, and the required computational effort for obtaining f, g and h.

Solutions obtained are mere approximations of the Pareto Front.

Among the Pareto approaches two in particular have gained increased acceptance and use in recent years:

1. Multiobjective Genetic Algorithms
2. Multiobjective Swarm Optimisation Algorithms

[1] A problem is NP-hard if an algorithm for solving it can be translated into one for solving any NP-problem (nondeterministic polynomial time) problem. NP-hard therefore means "at least as hard as any NP-problem," although it might, in fact, be harder

Evolutionary Algorithms I

- Evolutionary algorithms (EA) are loosely based on nature's evolutionary principles to guide a population of individuals towards an improved solution by honoring the "survival of the fittest" practice.
- This "simulated" evolutionary process preserves entropy (or diversity in biological terms) by applying genetic operators, such as mutation and crossover, to remix the fittest individuals in a population.

A generic evolutionary algorithms consists of the following components:

- *Genes*: traits defining an individual (design variables)
- *Fitness*: a mapping from genes to a set of numeric values (evaluating each objective function) describing the fitness of an individual,
- *Selector*: selecting the k fittest individuals of a population based on some sort of ordering,
- *Variator*: recombination (mutations and crossover) operators for offspring generation.

Evolutionary Algorithms in OptPilot I

Non-dominated sorting

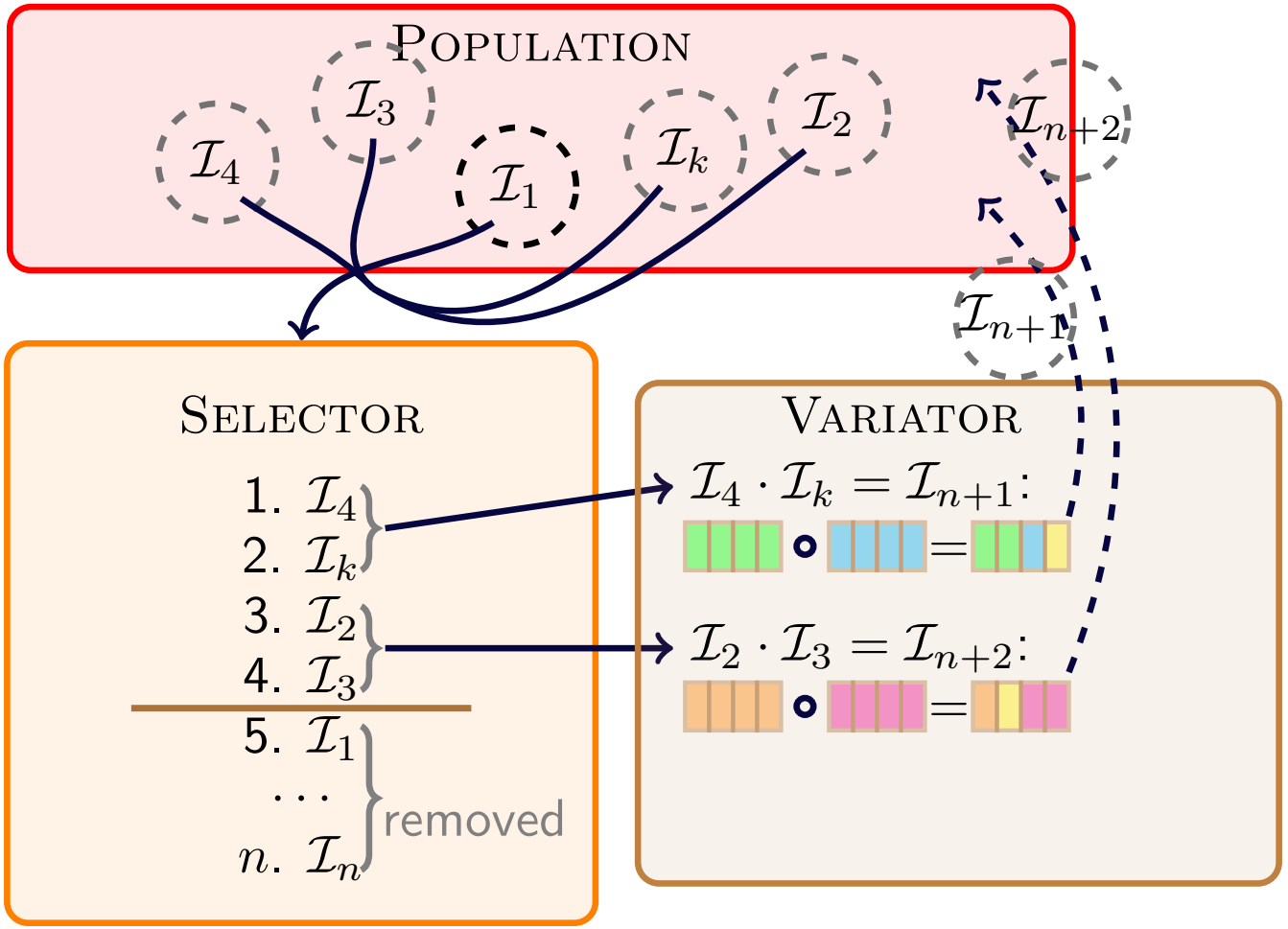
Algorithm: $\forall$ generations

1. initially random population of individuals I_i with a unique set of genes and corresponding fitness
2. In a next step the population is processed by the SELECTOR determining the k fittest individuals.
3. While the k fittest individuals are passed to the VARIATOR, the remaining $n - k$ individuals are eliminated from the population.
4. The VARIATOR mates the k fittest individuals to generate new offspring and applies the recombination operators.
   - Check convergence
5. After evaluating the fitness of all the freshly born individuals a *generation* cycle has completed and the process can start anew.

Complexity: $\mathcal{O}(MN^2)$ with M number of genes and N the population size.

Evolutionary Algorithms

A Platform and Programming Language Independent Interface for Search Algorithms [2]



[2] NSGA-II: `http://www.tik.ee.ethz.ch/pisa/`

Particles Swarm Optimisation (PSO) I

[Sedighizadeh,Masehian (2009)]

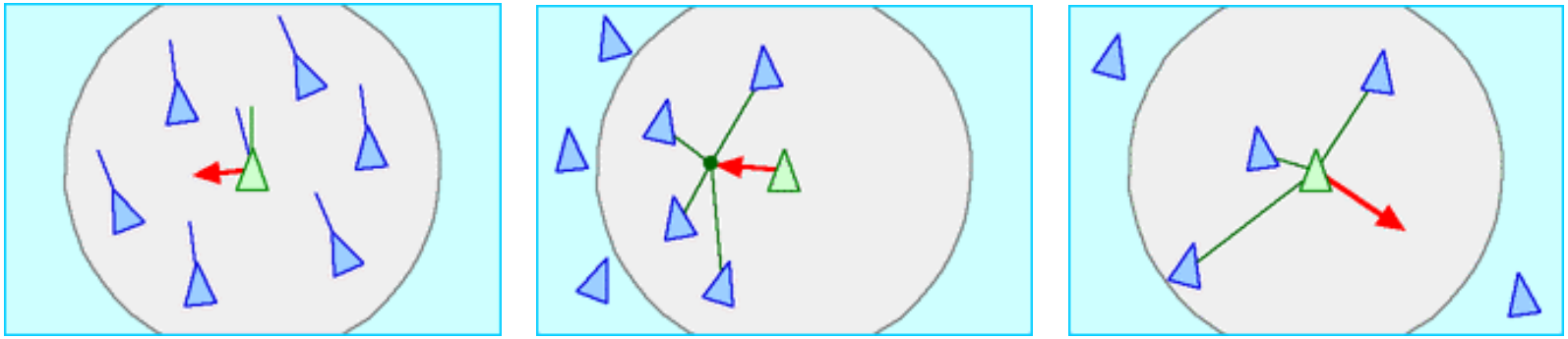
Eberhart and Kennedy (1995) developed PSO as an expansion of an animal social behavior simulation system that incorporated concepts such as nearest-neighbor velocity matching and acceleration by distance (Kennedy & Eberhart, 1995).

Similarly to evolutionary algorithms,

- PSO exploits a population, called a **swarm**
- of potential solutions, called **particles or agents**
- each particle keeps track:
  - its best solution, personal best, p_{best}
  - the best value of any particle, global best, g_{best}
- which are modified **stochastically** at each iteration of the algorithm.

Particles Swarm Optimisation Algorithm I

However, the manipulation of swarm differs significantly from that of evolutionary algorithms, promoting a cooperative rather than a competitive model.



- Separation: avoid crowding local flockmates
- Cohesion: move towards the average position of local flockmates
- Allignment: move towards the average heading of local flockmates

Particles Swarm Optimisation Algorithm II

1. Create a 'population' of particles uniformly distributed over S
2. Evaluate each particle's position according to the objective function
3. If a particle's current position is better than its previous best position, update it
4. Determine the best particle (according to the particle's previous best positions)
5. Update particles' velocities

$$\begin{aligned} v_i^{t+1} &= v_i^t + \text{personal influence} + \text{social influence} \\ &= v_i^t + \underbrace{c_1 \mathbf{U}(p_{i,\text{best}}^t - p_i^t)}_{\text{personal influence}} + \underbrace{c_2 \mathbf{U}(g_{i,\text{best}}^t - p_i^t)}_{\text{global influence}} \end{aligned}$$

6. Move particles to their new positions $p_i^{t+1} = p_i^t + v_i^{t+1}$
7. Go to step 2 until stopping criteria are satisfied

Advantages & Disadvantages

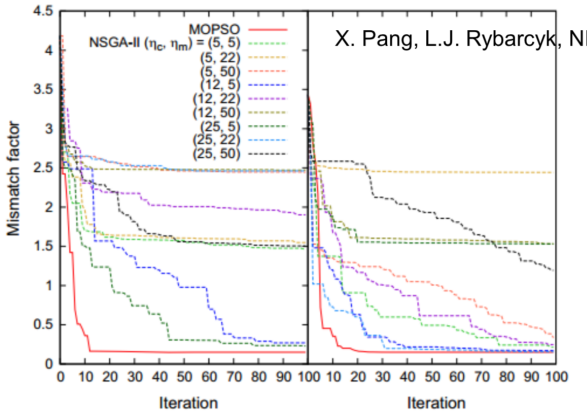


Fig. 2. The two plots depict simulations started with two different initial random populations. Both NSGA-II and MOPSO converged to similar final solution, but MOPSO did it in fewer iterations. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

- Advantages
  - insensitive to scaling of design variables
  - simple implementation
  - easily parallelised
  - derivative free
  - very efficient global search algorithm
- Disadvantages
  - tendency to a fast and premature convergence local solution
  - slow convergence in refined search stage (weak local search ability)
  - need to say something about the c_i's.

Advantages & Disadvantages

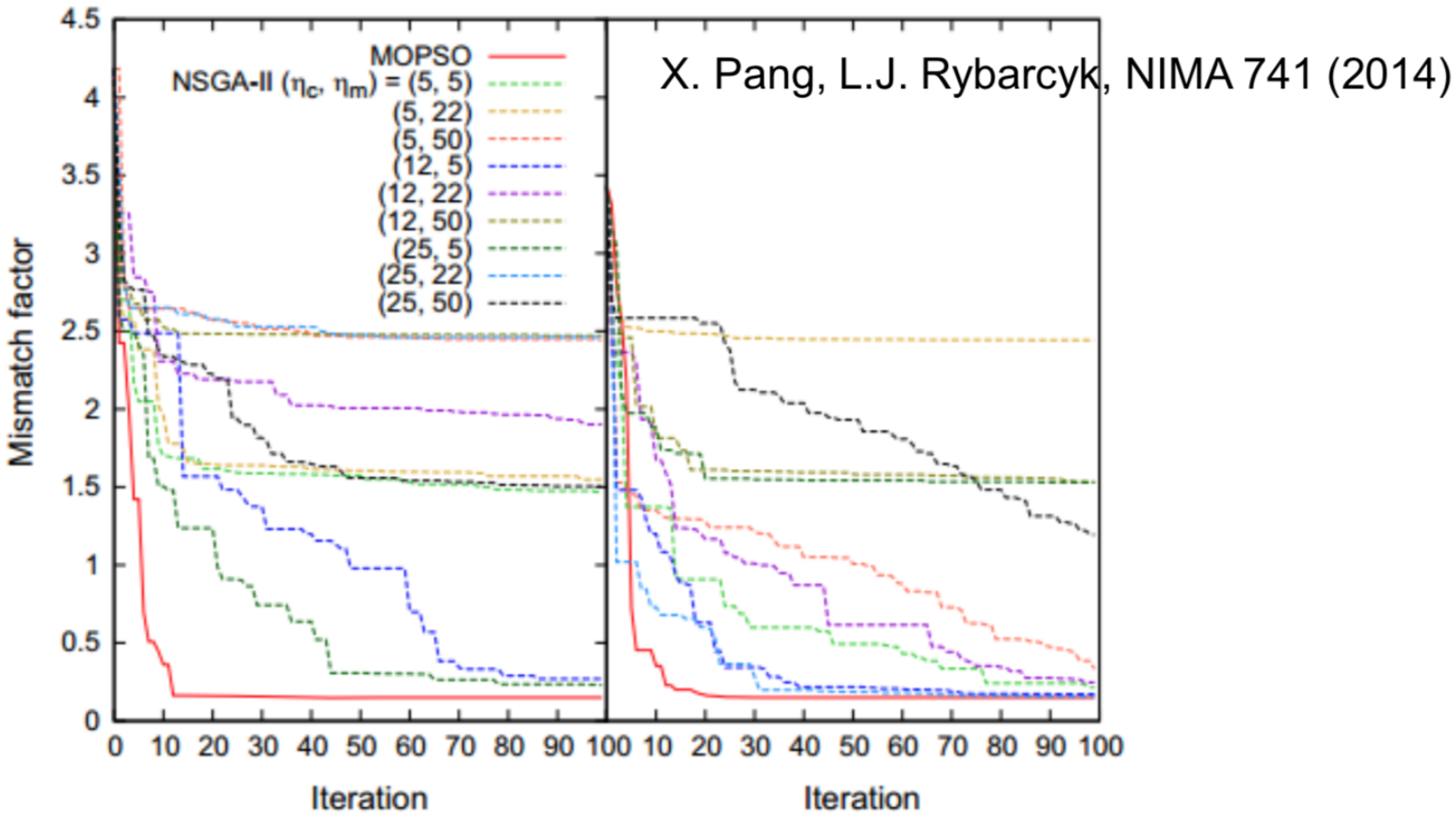


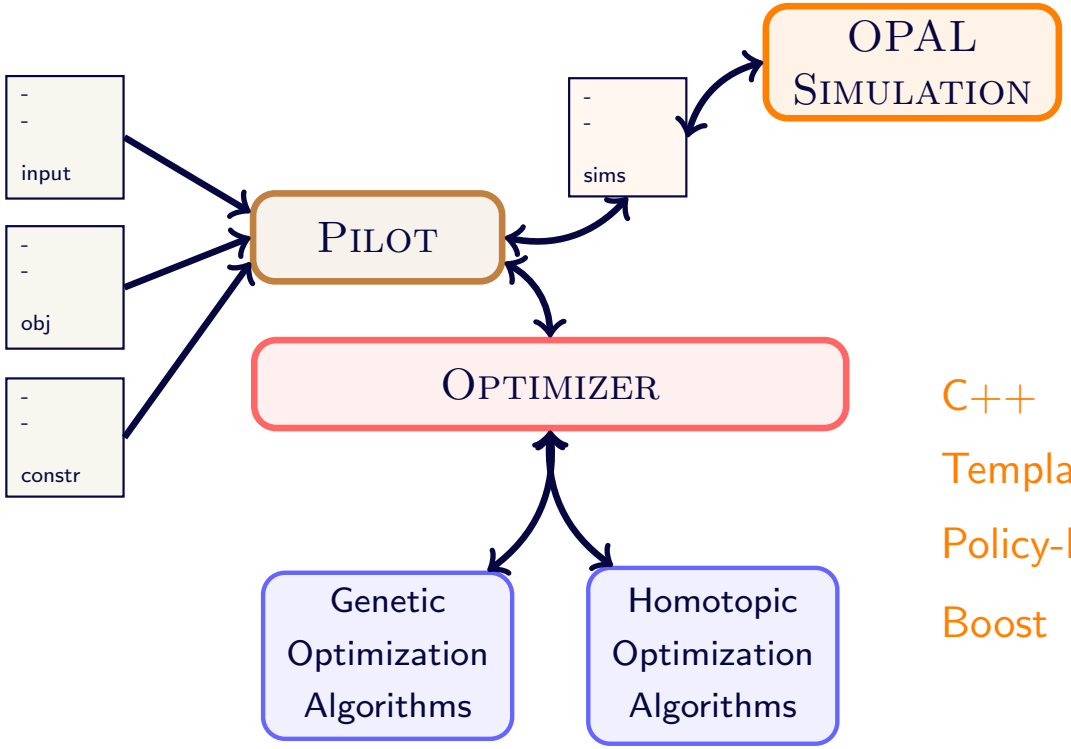
Fig. 2. The two plots depict simulations started with two different initial random populations. Both NSGA-II and MOPSO converged to similar final solution, but MOPSO did it in fewer iterations. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Outline

1. History
2. A Simple but Instructive Example
3. Theoretical considerations
4. A Modern GA Implementation
5. Two Examples with the optPilot

MOOP Framework with OPAL & Standalone

[Y. Ineichen et al., CS-R&D (2012), Y. Ineichen et al., arXiv:1302.2889]



C++

Templates

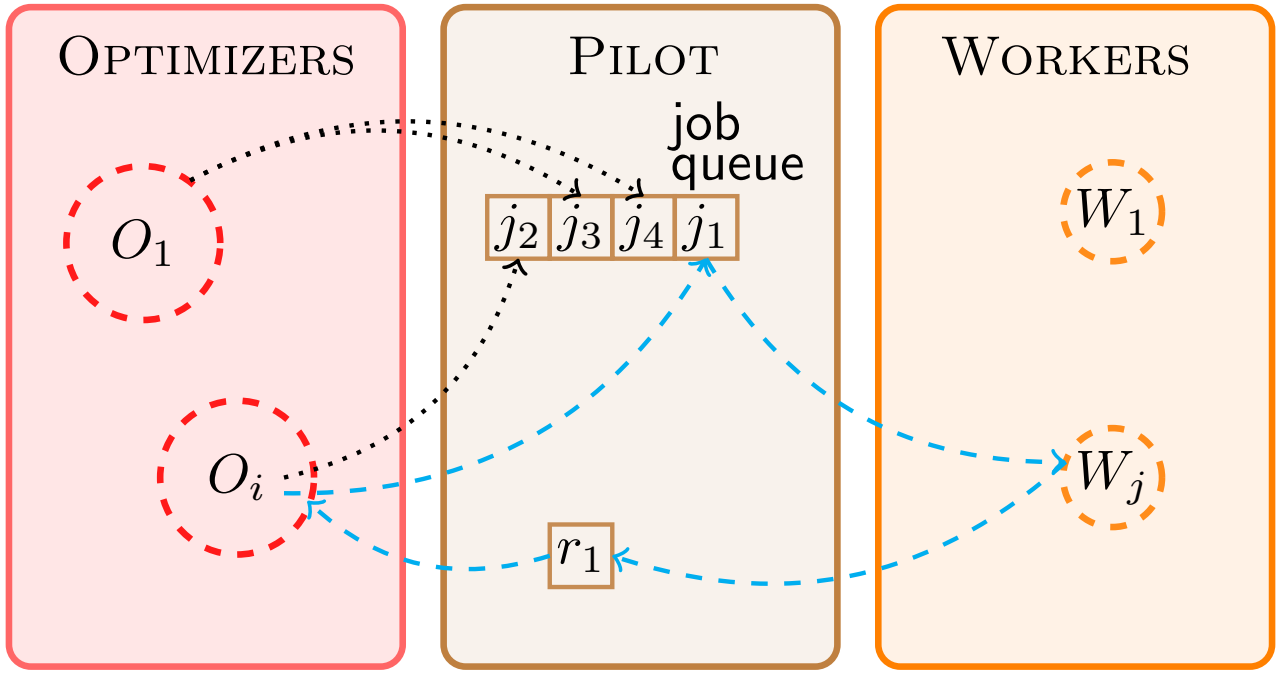
Policy-based design

Boost

Compared to Durillo et al., Halim, Leon et al., (..):

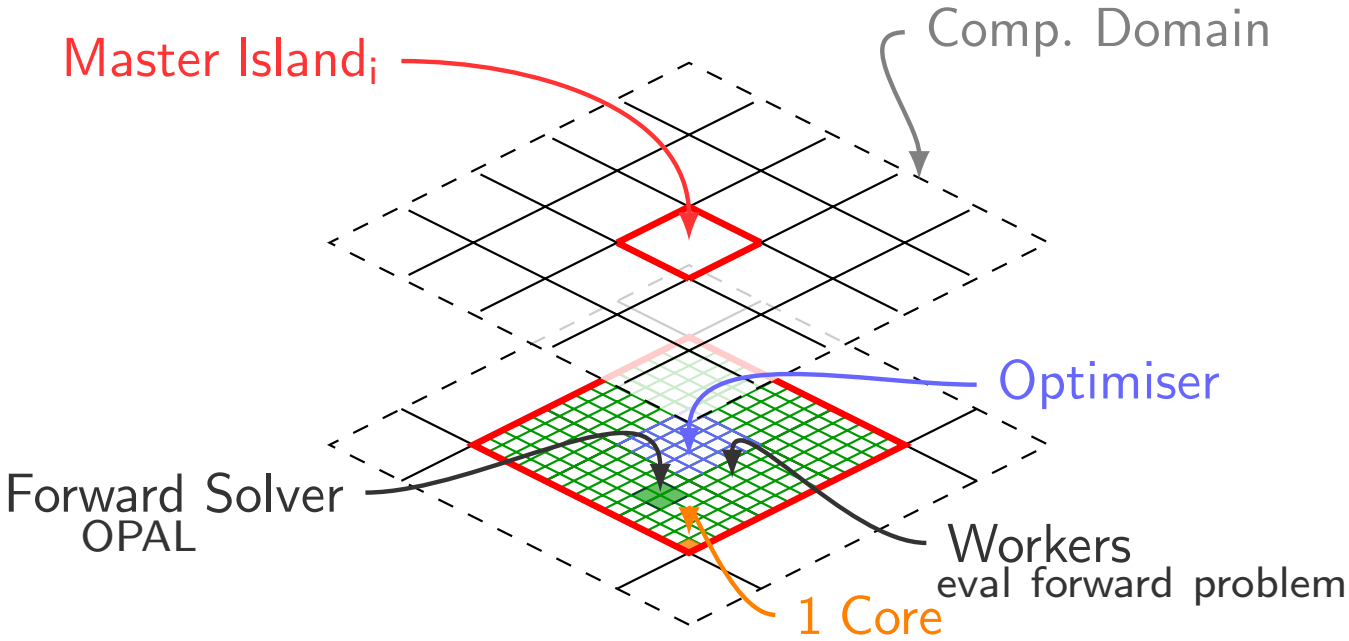
- no tight coupling to optimisation algorithm, and
- parallelization mechanism.

Master/Worker Model



- Asynchronous finite state machine (MPI)
- Multi-Scale optimisation

Island-based Master Model



- using techniques from social network theory
- can solve very challenging problems using largest HPC resources
- PRACE [3] award 2012

[3] Partnership for Advanced Computing in Europe

Outline

1. History
2. A Simple but Instructive Example
3. Theoretical considerations
4. A Modern GA Implementation
5. Two Examples with the optPilot

Standalone use of the Framework I

`pisa-standalone`

The FON problem (from Pereyra 2009)

$$\min \quad \begin{bmatrix} 1-\exp\left(-1\left(\left(x_1-\frac{1}{\sqrt{3}}\right)^2+\left(x_2-\frac{1}{\sqrt{3}}\right)^2+\left(x_3-\frac{1}{\sqrt{3}}\right)^2\right)\right) \\ 1-\exp\left(-1\left(\left(x_1+\frac{1}{\sqrt{3}}\right)^2+\left(x_2+\frac{1}{\sqrt{3}}\right)^2+\left(x_3+\frac{1}{\sqrt{3}}\right)^2\right)\right) \end{bmatrix}$$

$$\text{s.t.} \quad -1 \le x_i \le 1, \quad i = 1,2,3.$$

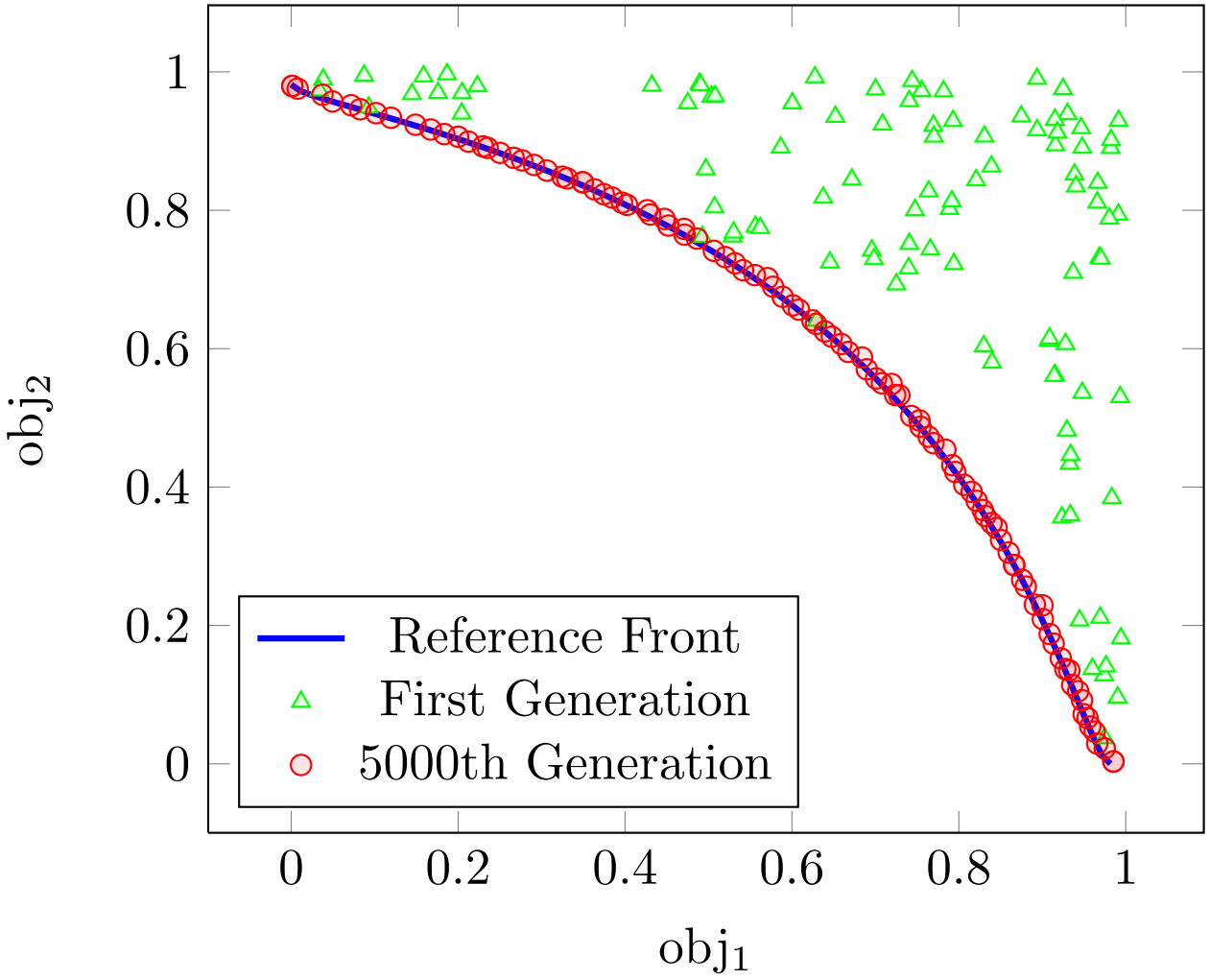
```
d1: DVAR, VARIABLE="x1", LOWERBOUND="-1.0", UPPERBOUND="1.0";
d2: DVAR, VARIABLE="x2", LOWERBOUND="-1.0", UPPERBOUND="1.0";
d3: DVAR, VARIABLE="x3", LOWERBOUND="-1.0", UPPERBOUND="1.0";

obj1: OBJECTIVE, EXPR="1.0-exp(-1.0*(sq(x1-1.0/sqrt(3.0))
                              +sq(x2-1.0/sqrt(3.0))
                              +sq(x3-1.0/sqrt(3.0)) ))";
```

Standalone use of the Framework II

pisa-standalone

```
obj2: OBJECTIVE, EXPR="1.0-exp(-1.0*(sq(x1+1.0/sqrt(3.0))
                          +sq(x2+1.0/sqrt(3.0))
                          +sq(x3+1.0/sqrt(3.0)) ))";

objs: OBJECTIVES = (obj1, obj2);

dvars: DVARS = (d1, d2, d3);
constrs: CONSTRAINTS = ();
opt: OPTIMIZE, OBJECTIVES=objs, DVARS=dvars, CONSTRAINTS=constrs;
```

Validation


1
0.8
0.6
0.4
0.2
0
obj2
0
0.2
0.4
0.6
0.8
1
obj1
Reference Front
First Generation
5000th Generation

OPAL Integration

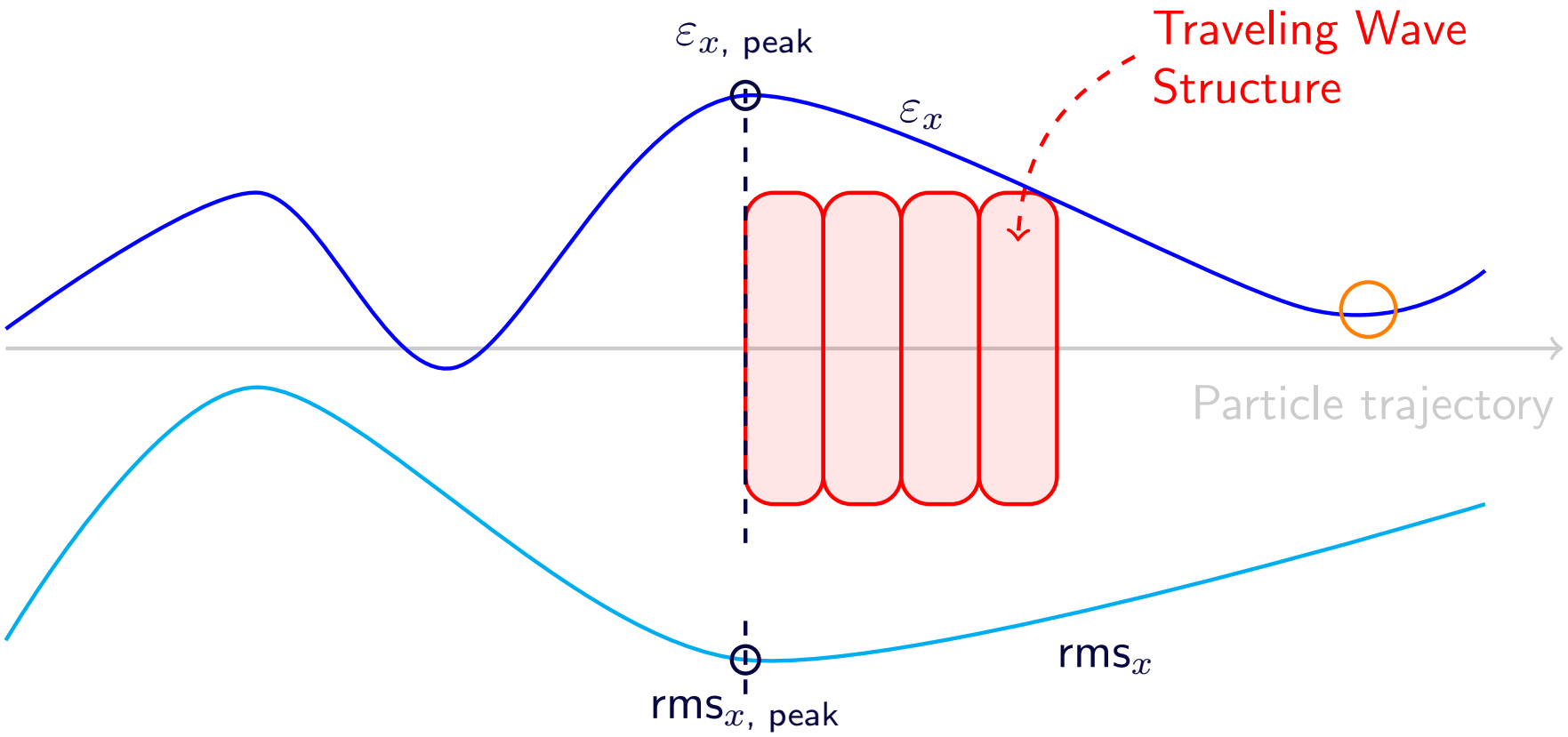
https://amas.psi.ch/OPAL/raw-attachment/wiki/DownloadPresetations/optBend.pdf

- definition of the MOOP problem in the OPAL input file
- OPAL als library is linked to the framework

```
dv1: DVAR, VARIABLE="c1", LOWERBOUND="0.1", UPPERBOUND="0.9";
dv2: DVAR, VARIABLE="c2", LOWERBOUND="1", UPPERBOUND="2";

dvars: DVARS = (dv1,dv2);

drmsx: OBJECTIVE, EXPR="fabs(sddsVariableAt("rms_x", ...
dex:   OBJECTIVE, EXPR="fabs(sddsVariableAt("emit_x", ...

objs: OBJECTIVES = (drmsx, dex);

co: CONSTRAINTS = ();
opt: OPTIMIZE, OBJECTIVES=objs, DVARS=dvars, CONSTRAINTS=co;
```

Application: Ferrario Matching in FEL context

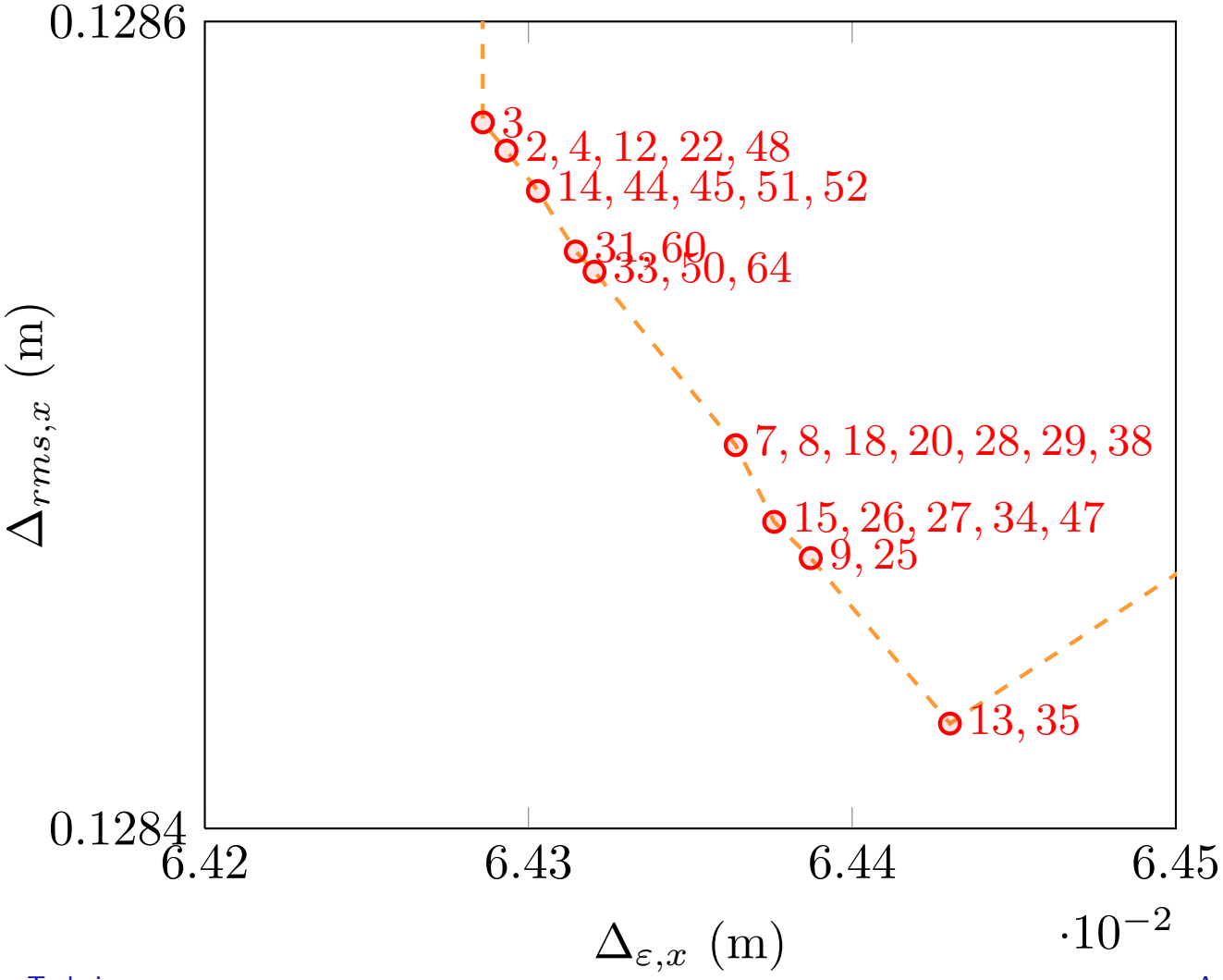
Partial problem description



Goal: Maximizes emittance dumping

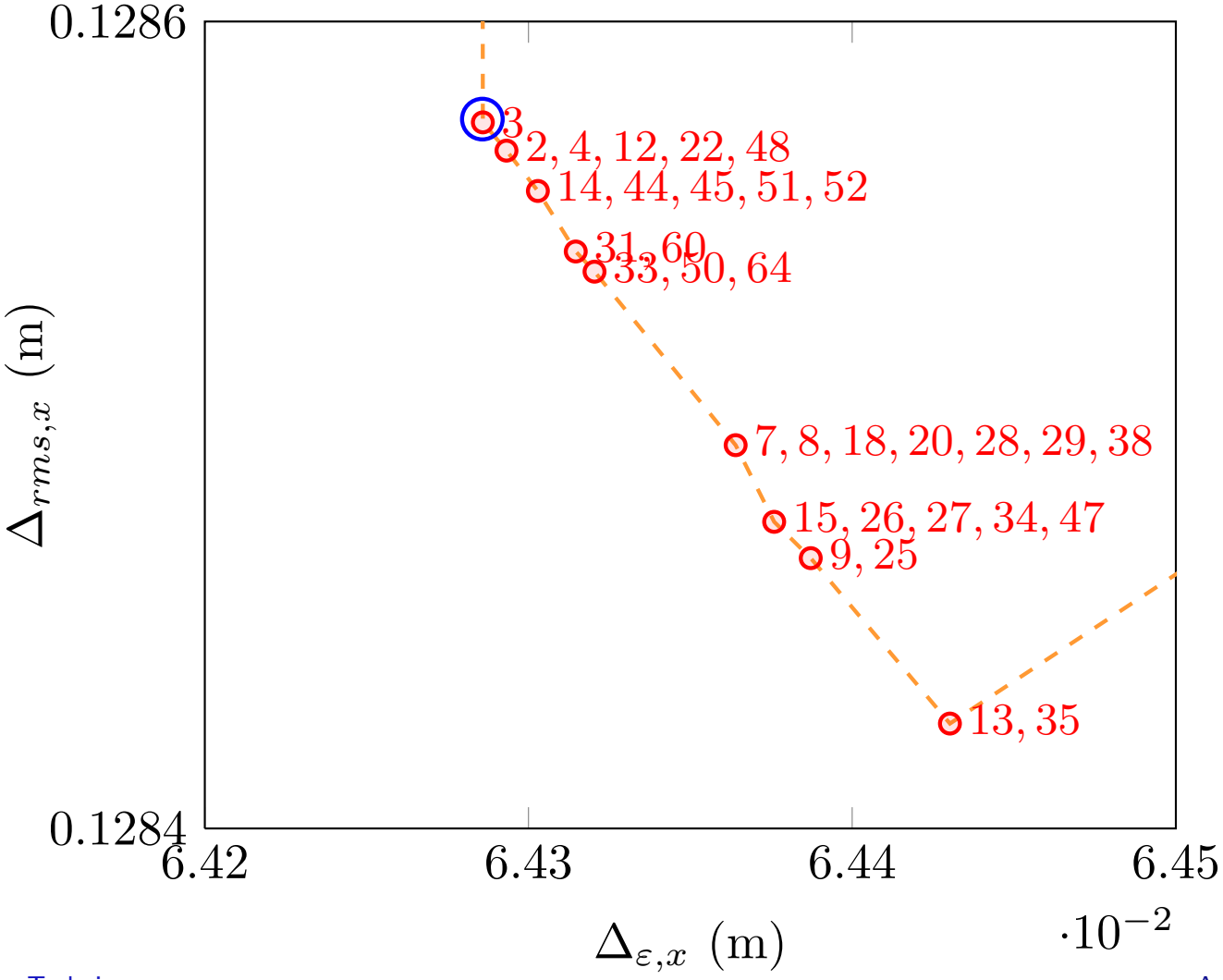
Pareto front after 1'000 generations

30 minutes on 32 cores

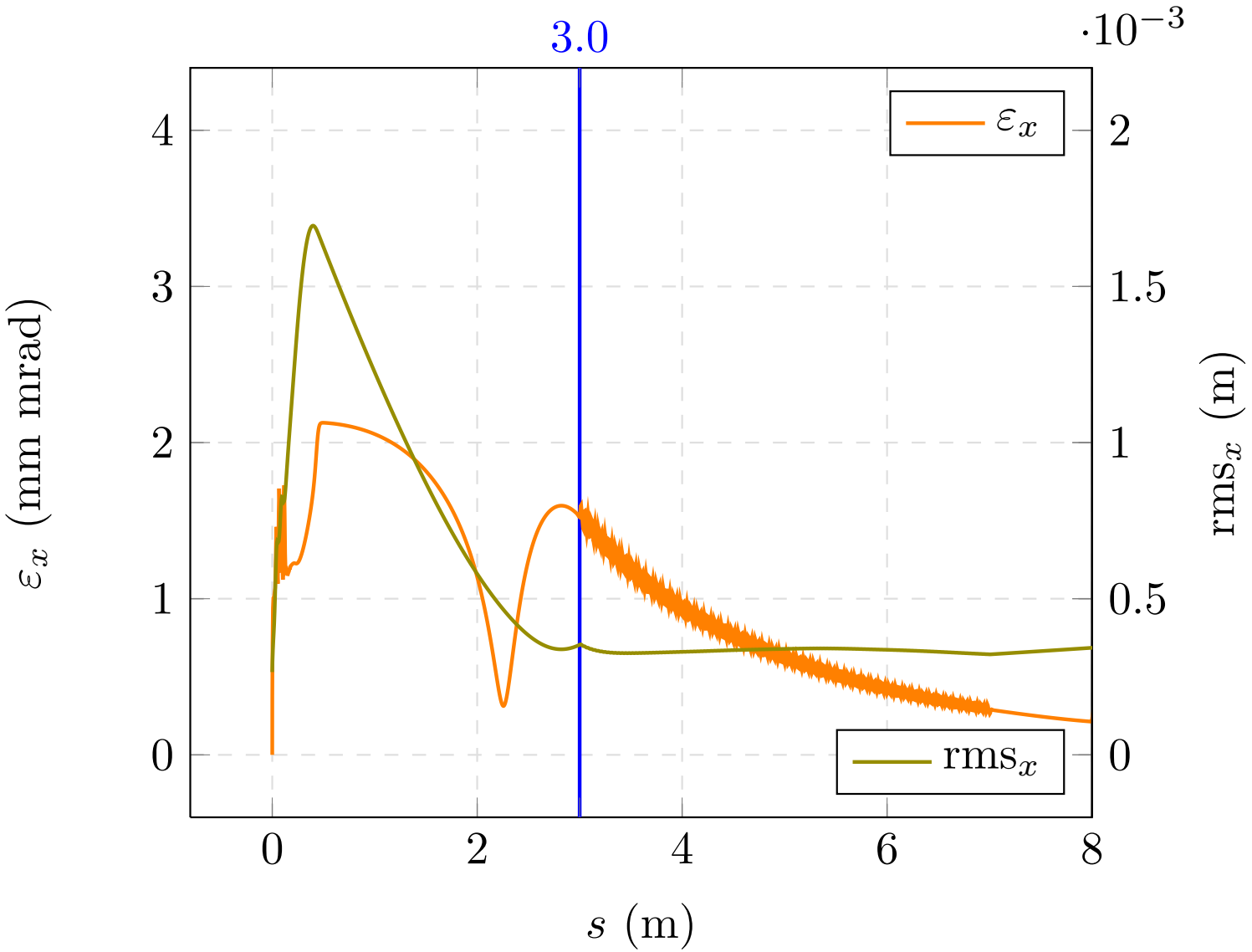


Pareto front after 1'000 generations

30 minutes on 32 cores



Simulation results for individual 3



References

- A Parallel General Purpose Multi-Objective Optimization Framework, with Application to Beam Dynamics Y. Ineichen, A. Adelmann, A. Kolano, C. Bekas, A. Curioni, P. Arbenz, arXiv:1302.2889, 2013
- Y. Ineichen ETH-Diss 21114, 2013
- Y. Ineichen, A. Adelmann et al., Computer Science - Research and Development, pp. 1-8. Springer, Heidelberg, 2012.
- Davoud Sedighizadeh and Ellips Masehian, "Particle Swarm Optimisation Methods, Taxonomy and Applications". International Journal of Computer Theory and Engineering, Vol. 1, No. 5, December 2009
- Konstantinos E. Parsopoulos and Michael N. Vrahatis, Multi-Objective Particles Swarm Optimisation Approaches Chapter II, 2008, IGI Global
- Multiobjective Optimisation: History and Promisses, CJK-OSM3 `http://strategic.mit.edu/docs/3_46_CJK-OSM3-Keynote.pdf`, 2004