Course: Natural Computing

6. Multi-Objective Optimisation

J. Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

Overview: Multiobjective optimisation

- Optimisation often asks for more than for a global optimum
- Multiobjective optimisation (MOO) implies several objective functions, the objectives of which can be conflicting.
- Diversity means here not only a good coverage for search, but also of the regions relevant for all the objectives.
- Population-based algorithms seem particularly suitable for MOO (Carlos A. Coello Coello, 2017)
  - simultaneity of search by a population
  - less susceptible to the shape or continuity of the Pareto front
- A fixed weighting of the objectives (*scalarisation*) requires background knowledge and may be unsuitable if the objectives interact in a non-trivial way
- Originally from economics: Maximise the wealth of a nation
- Interest in many application areas still growing

Multiobjective Optimization

Several objectives: (d dimensions, k objectives)

To each point in search space $x = (x_1, \ldots, x_d)$ corresponds a point in objective space $f = (f_1(x), \ldots, f_k(x))$

The concept of a *global optimum* is not obvious in MOO. We should rather define a set X of optimal points in the search space.

Example: $f_1 = 2^x$, $f_2 = -x^2$

Scalarise e.g. by equal weight: $f_{1/2} = \frac{1}{2} f_1 + \frac{1}{2} f_2$

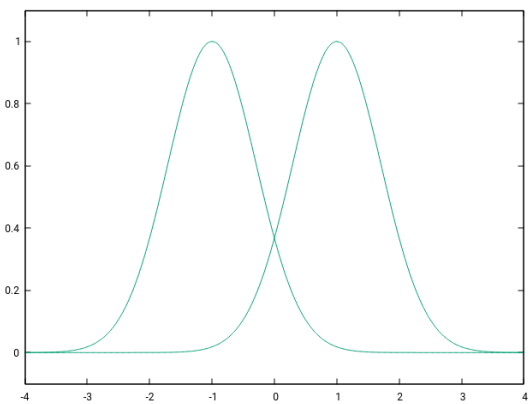
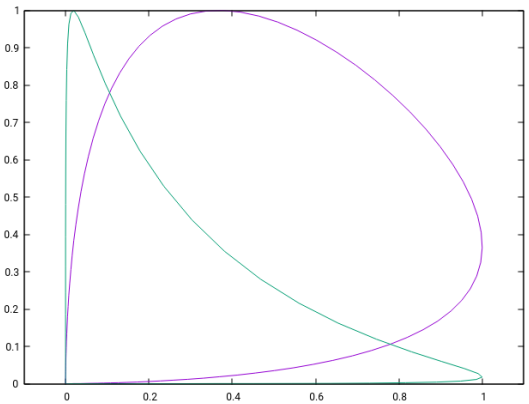
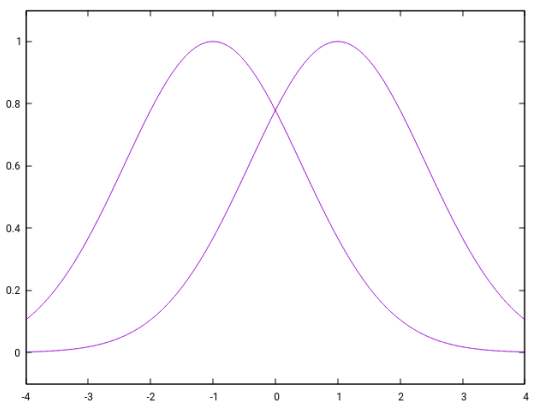
- Search space $x \in [0, 1]$:
  Global maximum of $f_{1/2}$ is at $\arg\max_x f_{1/2}(x) \approx 0.5$
  whereas: $2 = \arg\max_x f_1(x)$ and $0 = \arg\max_x f_2(x)$
- Search space $x \in [0, 100]$: f_2 is ignored for large x

Example: A machine is characterized by power and torque. A machine is better if – at equal torque – its power is higher.

Multiobjective Optimization

Approaches to MOO will include two mechanisms

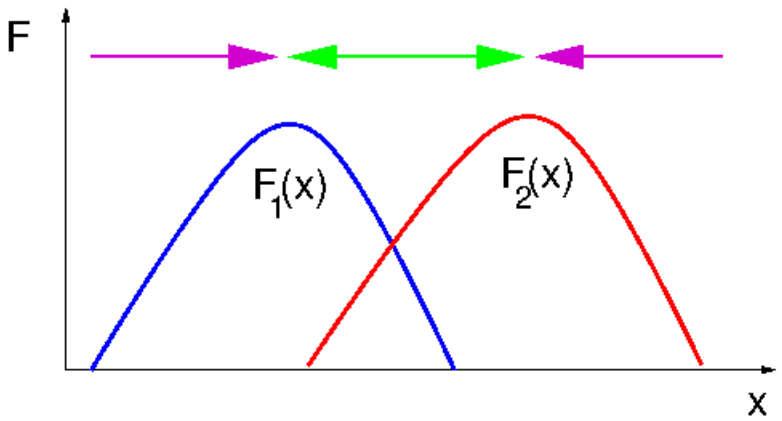
- Selection improvement w.r.t. to either objective
- Diversification: find solutions that correspond to different combinations of the fitnesses.
- Description in search space (as usual) or in fitness space

$\sigma = 1$ $\quad f_{\pm} = \exp\left(-\frac{(x \pm 1)^2}{\sigma^2}\right)$ $\quad \sigma = 2$

1D search space 2D fitness space 1D search space

GA for Multiobjective Optimization



Combination of fitness functions

$f(x) = \alpha f_1(x) + (1-\alpha) f_2(x)$

$f(x) = |f_1(x)|^\alpha + |f_2(x)|^\alpha$

How to set α?

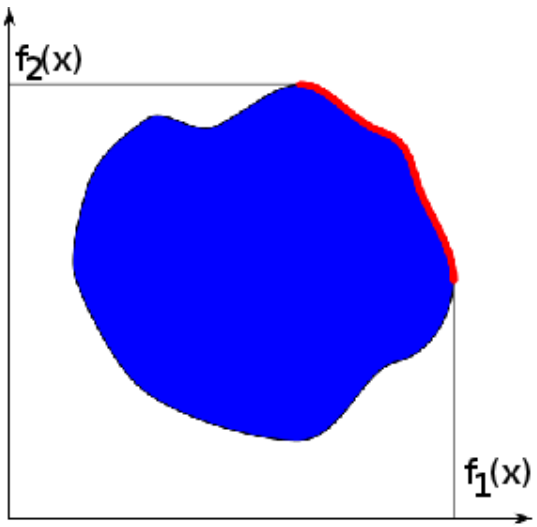
If α is not implied by the problem, then any value in between the two maxima is equally good.

If a comparison between the two quantities is not possible, a set of solutions should be considered as optimal (*Pareto-optimal*).

How to optimise one criterion without loosing on other criteria?

First relevant paper: J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. PhD thesis, Vanderbilt University, 1984, but earlier studies exist.

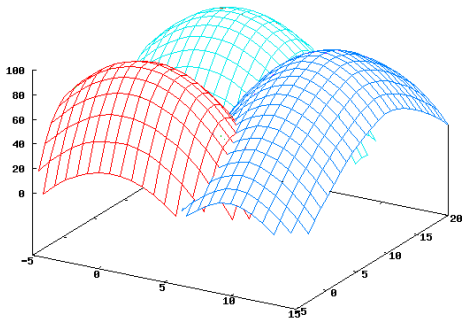
Multiobjective Optimization



x^* is Pareto optimal for a class of fitness functions $\{f_i\}$ if there exists **no** $x \neq x^*$ with $f_i(x) \geq f_i(x^*)$ for all i

or, equivalently, x^* is not **dominated** by any other x : $\sim\exists x \succ x^*$
(more specifically $\sim\exists x \succ_{\{f_i\}} x^*$)

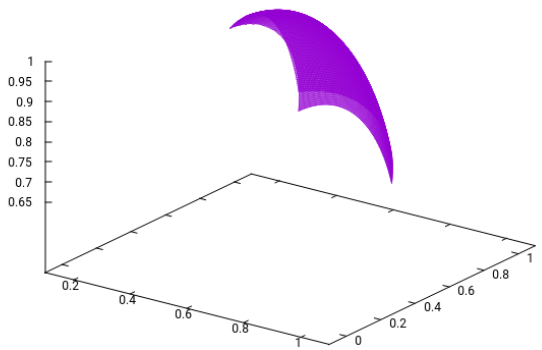
Multiobjective Optimization



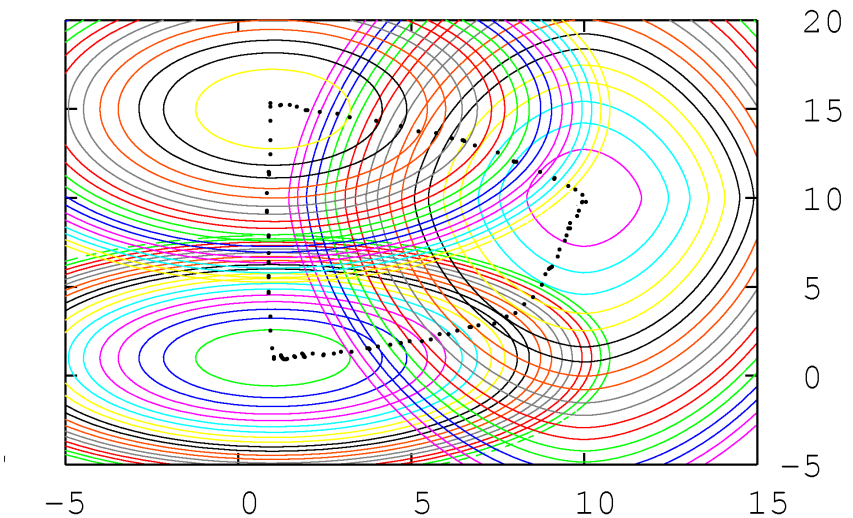
Example with three fitness functions

x^* is Pareto optimal for a class of fitness functions $\{f_i\}$ if there exists **no** $x \neq x^*$ with $f_i(x) \geq f_i(x^*)$ for all i

or, equivalently, x^* is not **dominated** by any other x : $\sim\exists x \succ x^*$
(more specifically $\sim\exists x \succ_{\{f_i\}} x^*$)



Part of the fitness space (similar illustration)



Similar example: Pareto area spanned by maxima in a shape-dependent way

GA for Multiobjective Optimization

Benefits:

- Collective search required for sampling the Pareto set
- Non-connected Pareto sets are possible
- Incorporation of constraints in fitness function

Problems:

- Selection of fit individuals?
- Elitism?
- Pareto-optimal diversity?
- Efficiency in high-dimensional problems?

Multiobjective Hill climbing and Simulated Annealing

- Set set of solutions $S = \emptyset$
- Start from a single starting state (as usual)
- Take a step
- Accept according to algorithm
- If new solution x is not dominated by any solution in S, then $S \leftarrow S \cup \{x\}$
- Apply anti-crowding mechanism in S
- For HC: Randomly restart when no improvement possible

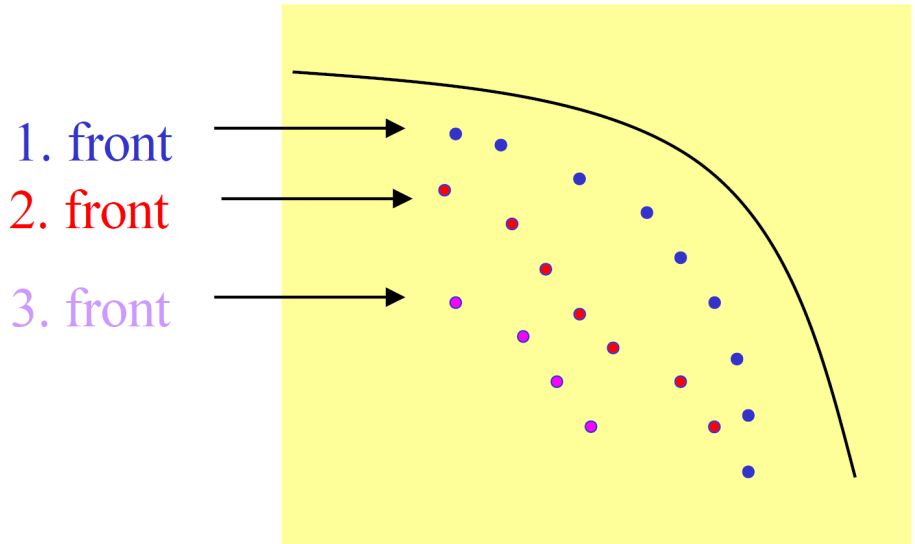
How does it work?

- Non-dominated-sorting genetic algorithm (NSGA)
- Selection by non-dominated sorting (M fitness functions)
- Preserving diversity along the non-dominated front
- Use two populations P and P' (each with N individuals)
- "being dominated by", denotes a partial order induced by a set of fitness functions

P'=find-nondomminated front(P)	
$P' = \{1\}$	include first member into P'
for each $p \in P \wedge p \notin P'$	take on solution at a time
$P' = P' \cup \{p\}$	temporarily include p into P'
for each $q \in P' \wedge q \neq p$	compare p to other members of P'
if $q \prec p$ then $P' = p' \setminus \{q\}$	if p dominates a member q of P' then delete q
else if $p \prec q$ then $P' = p' \setminus \{p\}$	if p is dominated by another member then do not include p in P'

Complexity per step: $O(MN^2)$

Ranking



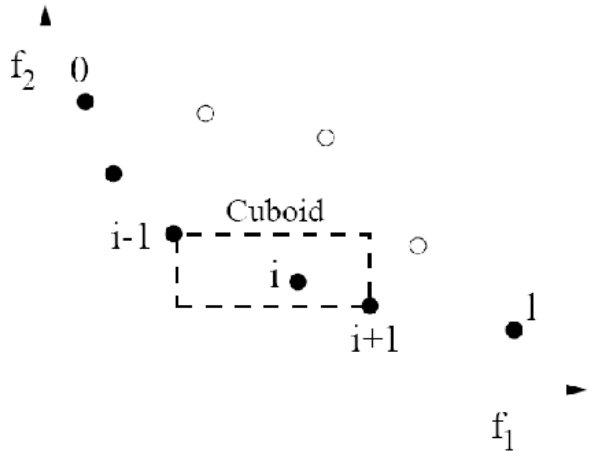
$\mathcal{F}$=fast-nondominated-sort(P); returns a set of nondominated fronts	
$i = 1$	i is the front counter
until $P \neq \emptyset$	temporarily include p into P'
$\mathcal{F}_i$=find-nondominated-front (P)	find the non-dominated front
$P = P \backslash \mathcal{F}_i$	remove nondominated solutions from P
$i = i + 1$	increment the front counter

Preserving density

New distance measure: first rank, then lowest density:

$i \succ_n j$ if $(i_{\text{rank}} < j_{\text{rank}})$ or

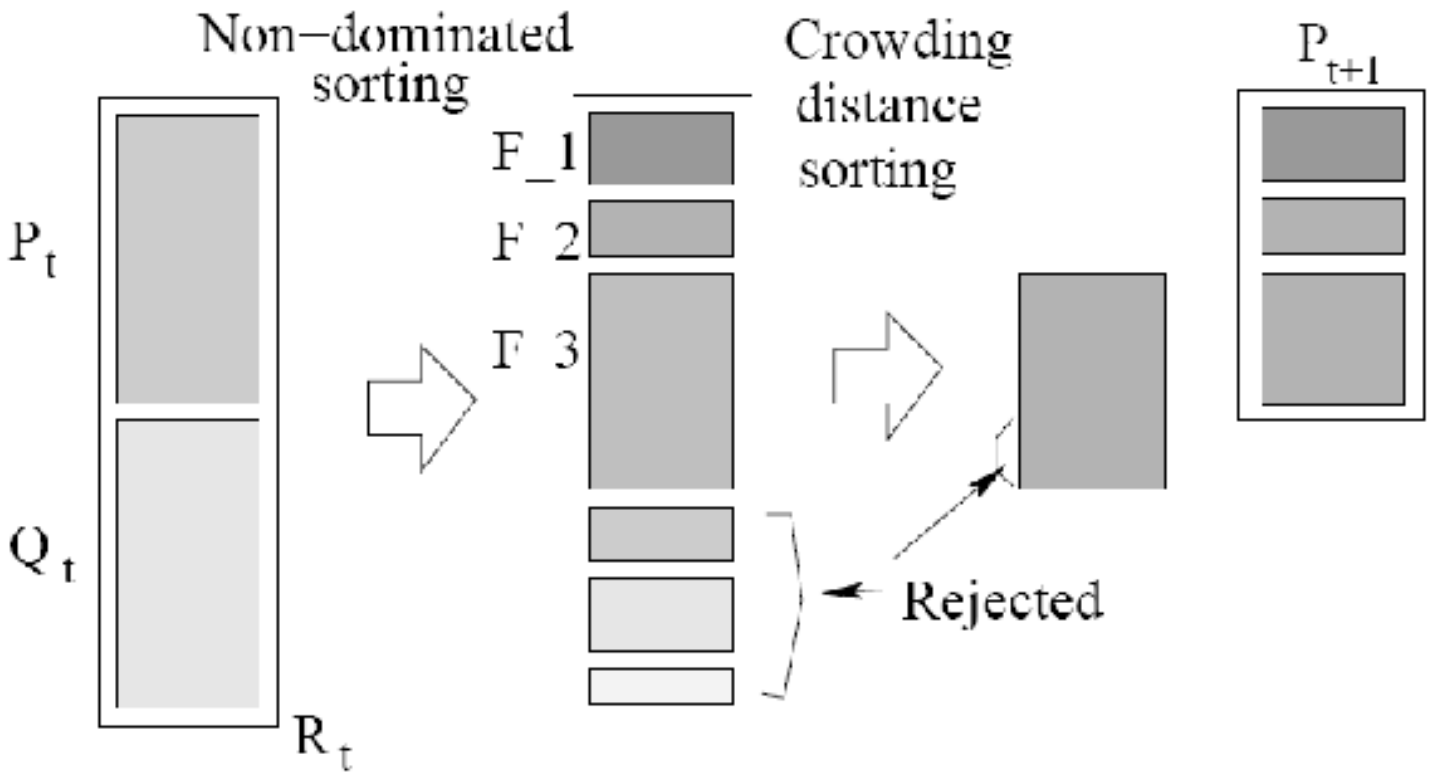
$((i_{\text{rank}} = j_{\text{rank}})$ and $i_{\text{dist}} > j_{\text{dist}})$



crowding-distance-assignment($\mathcal{I}$)

$l = \{\mathcal{I}\}$	number of solutions in $\mathcal{I}$
for each i set $\mathcal{I}[i]_{\text{dist}} = 0$	initialise distance
for each objective m	temporarily include p into P'
$\mathcal{I} = \text{sort}(\mathcal{I}, m)$	sort using each objective value
$\mathcal{I}[1]_{\text{dist}} = \mathcal{I}[l]_{\text{dist}} = \infty$	so that boundary points are always selected
for $i = 2$ to $l - 1$	for all non-boundary points:
$\mathcal{I}[i]_{\text{dist}} = \mathcal{I}[i]_{\text{dist}} + (\mathcal{I}[i+1]_m - \mathcal{I}[i-1]_m)^2$	

NSGA-II: Main Loop



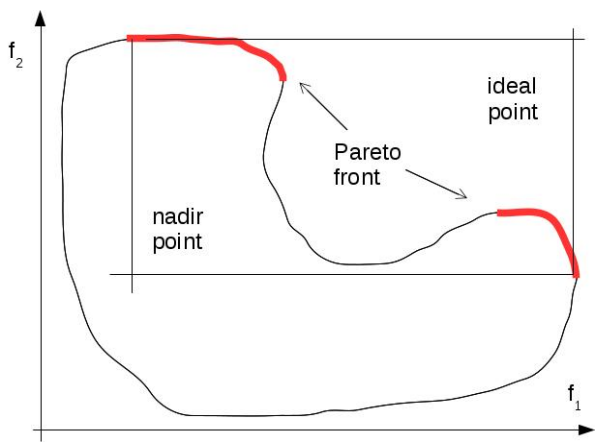
NSGA-II: Main Loop

$R_t = P_t \cup Q_t$	combine parents and children
$\mathcal{F}$=fast-nondominated-sort(R_t)	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$, all nondominated fronts of R_t
$P_{t+1} = \emptyset$ and $i = 1$	
until $\lvert P_{t+1}\rvert + \lvert\mathcal{F}_i\rvert \leq N$	till the parent population is filled
crowding-distance-assignment($\mathcal{F}_i$)	calculated crowing distance in $\mathcal{F}_i$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	include the ith front into parent population
$i = i + 1$	check next front for inclusion
sort($\mathcal{F}_i, \prec_n$)	take part of the following front
$P_{t+1} =$ $P_{t+1} \cup \mathcal{F}_i\,[1 : (N - \lvert P_{t+1}\rvert)]$	choose the first $(N - \lvert P_{t+1}\rvert)$ elements of $\mathcal{F}_i$
Q_{t+1} =make-new-pop(P_{t+1})	use selection, crossover and mutation to create a new population Q_{t+1} (standard GA)
$t = t + 1$	increment the generation counter

Other approaches: Scalarisation

- Scalar approaches: Transformation into a single-objective problem
  - weighted sum
  - weighted product
  - distance from ideal solution, i.e. the point $(\max f_1, ..., \max f_k)$
  - achievements: weighted improvement w.r.t. a user-defined point

- If Pareto front $\mathcal{P}$ is known (e.g. in competitions):
  - Distance of solutions from the Pareto front integrated over true Pareto front
  - Volume dominated by solution w.r.t. to *nadir* point, i.e. for $x \in \mathcal{P}$ $(\min f_1(x), ..., \min f_k(x))$



Other approaches

- Criterion-based: Treating the various noncommensurable objectives separately
- Priority-based: Order results lexicographically based on goal priority
- Achievement-based: Define lower bounds for all criteria and compare advantage of solutions
- ϵ-constraint method: Bounds on other objectives as constraints
- Ranking-based: Average single-objective ranks
- Stepping-stone-based: Consider a scalarised problem as a Pareto problem in order to optimise first, what is most easily optimisable
- Dominance-based approaches: Guiding the search process by Pareto optimality
- Indicator-based approaches: Using performance quality indicators to drive the search toward the Pareto front (e.g. distance from nadir point)
- Repulsion-based or cooperation-based (see PSO lecture).

Conclusion on MOO

- MOO is related to constraint optimisation
- MOO is related to neutral evolution
- MOO provides an approach to extend and to generalise academic problems into practical application
- MOO is applicable also if fitness evaluation is costly and several partial models exist
- MOO provide an approach to analyse practical problems where various objectives have been identified by the customer

Further reading: M. T. M. Emmerich and A. H. Deutz: A tutorial on multiobjective optimization: Fundamentals and evolutionary methods. *Natural Computing* **17** (2018) 585.

Beyond MHO (?)

- Heuristics
- Metaheuristics
- Memetic Algorithms (1st level MA)
- Hybrid algorithms
- Hyperheuristic (2nd level MA)
- Co-evolution and self-generating (3rd level MA)

Classification according to Chen, Ong, & Lim (2010)

Memetic algorithms (Moscato, 1989)

- Metaphor based on social evolution $\rightarrow$ *cultural algorithm*
- Includes both genetic and individual learning (similar to the Baldwin effect and Lamarckian evolution)
- In the context of MOO this can include a preference for certain solutions or a measure of diversity.
- Can be as simple as GA with ES for local search
- In principle, the memetic component of the MH needs to be developed in a social context (different from Baldwin effect and Lamarckian evolution, t.b.d. later)
- Can be considered as a type of *hyperheuristic algorithms*, see below.

see e.g. Neri & Cotta (2012) Memetic algorithms and memetic computing optimization: A literature review.