OPTIMIZATION TECHNIQUES

Multi-objective optimization

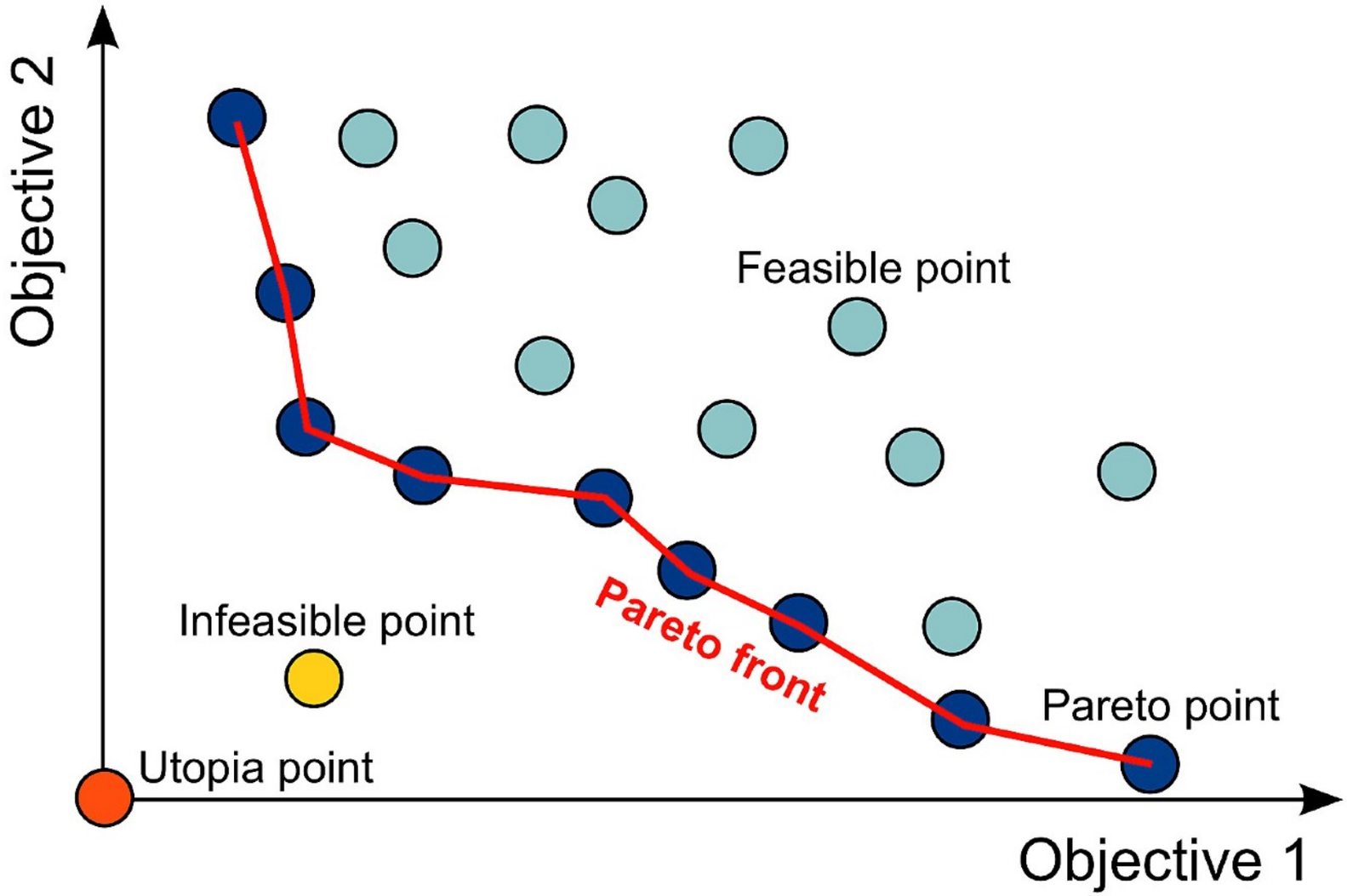
Prof. Giovanni Iacca

giovanni.iacca@unitn.it

UNIVERSITY OF TRENTO - Italy

Information Engineering
and Computer Science Department

Multi-Objective Optimization



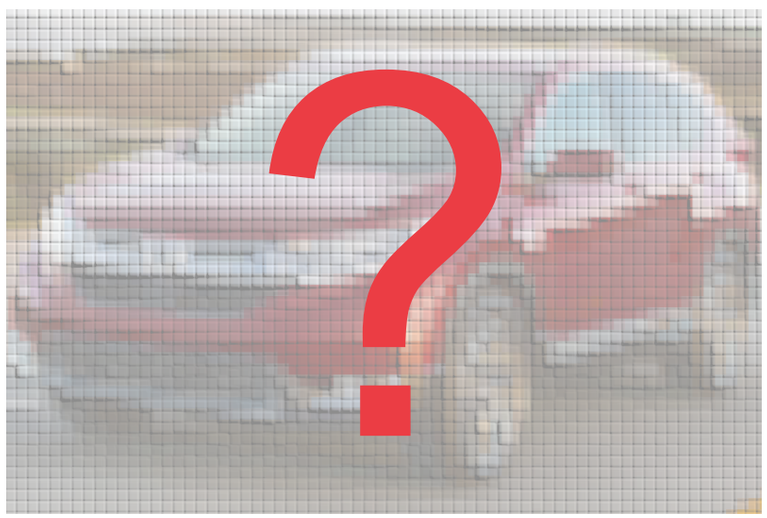
MULTI-OBJECTIVE OPTIMIZATION

MULTI-OBJECTIVE OPTIMIZATION PROBLEMS (MOPS)

A wide range of problems (virtually *all* real-world problems) are characterized by the presence of a number of *n* possibly conflicting objectives, e.g.:

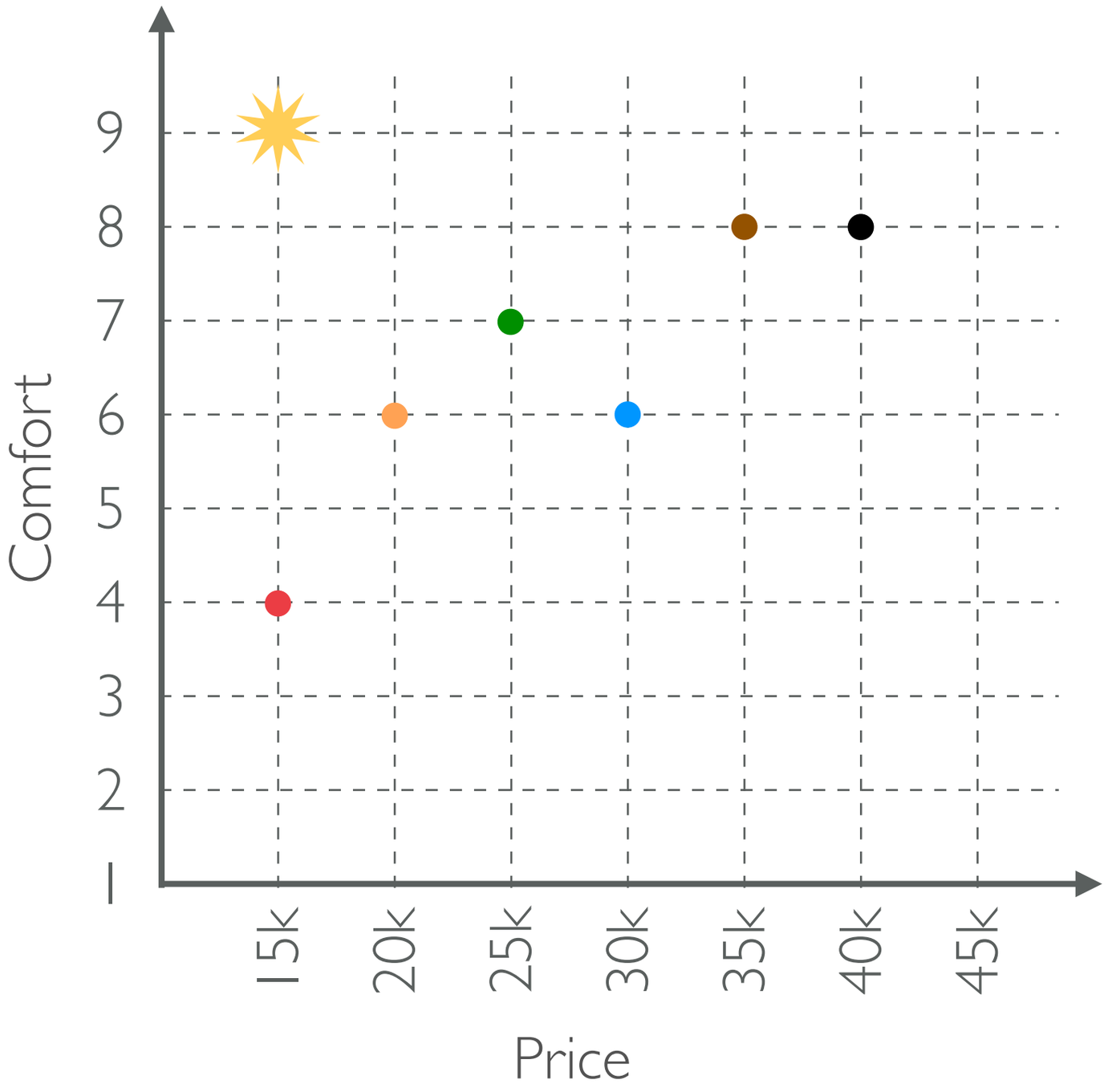
- buying a car: comfort vs price
- buying a house: location vs size vs quality vs price
- choosing a job: location vs salary vs flexibility vs duration of the contract
- engineering design: lightness vs strength

For instance, when we buy a car, we would like the car to be as cheap as possible (minimize cost) and as comfortable as possible (maximize comfort). If we consider the two objectives separately, we'll probably obtain two different optimal solutions. BUT, neither of them is likely what we want…



MULTI-OBJECTIVE OPTIMIZATION

MULTI-OBJECTIVE OPTIMIZATION PROBLEMS (MOPS)



Car	Price	Comfort
A	15k	4
B	20k	6
C	25k	7
D	30k	6
E	30k	8
F	40k	8

… what if there was a car with cost 15k and comfort 9?

MULTI-OBJECTIVE OPTIMIZATION

CONVENTIONAL APPROACHES ("A PRIORI" METHODS)

Characterized by the fact that some decisions must be made a priori, i.e. *before optimization*, in order to solve the MOP as (one or more) single-objective problems.

1. **Scalarization**: different objectives are combined into one (non-)linear function.
   E.g. (for two objectives f_1 and f_2):
   $f=\alpha \times f_1-(1-\alpha)\times f_2$
   $f=(f_1+1.0/f_2)$
   $f=f_1^{\alpha}\times f_2^{(1-\alpha)}$
   …

The most common scalarization function is a weighted sum of the objectives: $f=w_1\times f_1+w_2\times f_2+\ldots$

Important to take into account:
- how each objective increases/decreases when f is minimized or maximized
- different orders of magnitudes for each objective → magnitude of weights is crucial!
- different weights correspond to different rankings (importance) of the objectives
- this approach corresponds to an a priori restriction of the fitness space, i.e., in order to find different trade-off solutions the problem must be solved with different weights w_i

MULTI-OBJECTIVE OPTIMIZATION

CONVENTIONAL APPROACHES ("A PRIORI" METHODS)

2. **Lexicographic ordering**: the objectives $< f_1, f_2, \ldots f_k >$ are ranked in a user-defined *order of importance*, so that f_1 is the most important and f_k is the least important. Then a sequence of single-objective optimization problems is solved, where first f_1 is optimized (ignoring $f_2, \ldots f_k$) to get an optimum f_1^*, then f_2 is optimized imposing a constraint $f_1 \leq f_1^*$ (in case of maximization), then f_3 is optimized with constraints $f_1 \leq f_1^*$ and $f_2 \leq f_2^*$, and so on so forth.

3. **ε-constraint method**: k single-objective optimization problems are solved separately, one for each objective, imposing for each problem k-1 constraints corresponding to the other k-1 problems, i.e. $f_j \leq \varepsilon_j$ where ε_j are user-defined thresholds.

Some questions to ask:
1) How should the single objectives be ranked?
2) What is the effect of ranking on the resulting trade-off?
3) Is it possible to consider all the objectives *simultaneously*?

→ Vilfredo Pareto, *Manual of Political Economy*, 1896

MULTI-OBJECTIVE OPTIMIZATION

MULTI-OBJECTIVE OPTIMIZATION PROBLEMS (MOPS)

SCALARIZATION (MAXIMIZATION)

f=(40k-price) + 10k × comfort

25k + 40k = 65k
20k + 60k = 80k
15k + 70k = 85k
10k + 60k = 80k
10k + 80k = 90k Optimum: E
0 + 80k = 80k

f=(40k-price) + 1k × comfort

25k + 4k = 29k Optimum: A
20k + 6k = 26k
15k + 7k = 22k
10k + 6k = 16k
10k + 8k = 18k
0 + 8k = 8k

Car	Price	Comfort
A	15k	4
B	20k	6
C	25k	7
D	30k	6
E	30k	8
F	40k	8

MULTI-OBJECTIVE OPTIMIZATION

MULTI-OBJECTIVE OPTIMIZATION PROBLEMS (MOPS)

LEXICOGRAPHIC ORDERING

Price more important than comfort
1. Minimize price: {A}
2. Maximize comfort in {A}: A

Optimum: A

Comfort more important than price
1. Maximize comfort: {E, F}
2. Minimize price in {E, F}: E

Optimum: E

Car	Price	Comfort
A	15k	4
B	20k	6
C	25k	7
D	30k	6
E	30k	8
F	40k	8

ε-CONSTRAINT METHOD

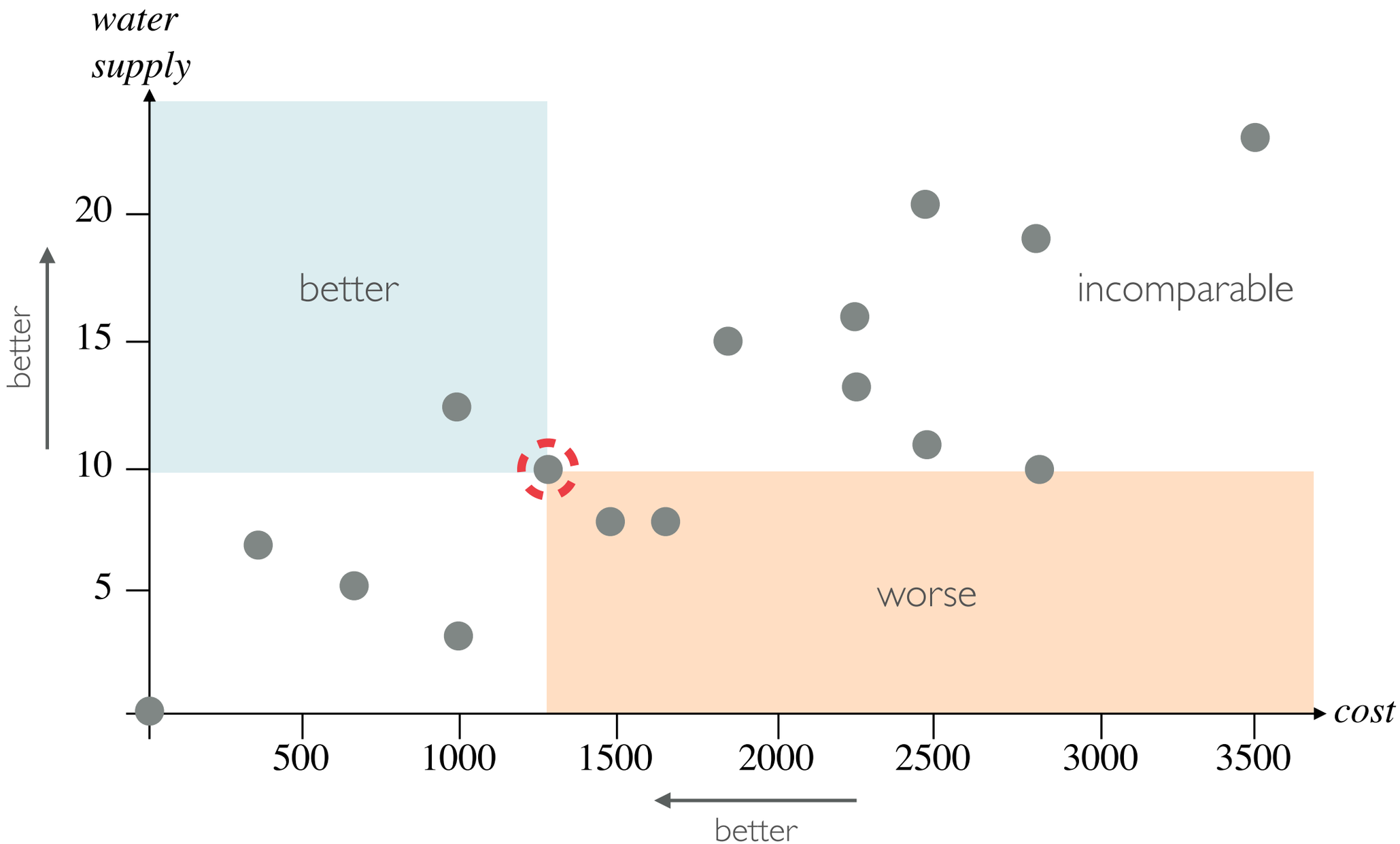
- Minimize price (s.t. comfort ≥ 7): {C, E, F}
- Maximize comfort (s.t. price < 30k): {A, B, C}

Optimum: C

MULTI-OBJECTIVE OPTIMIZATION

PARETO-DOMINANCE

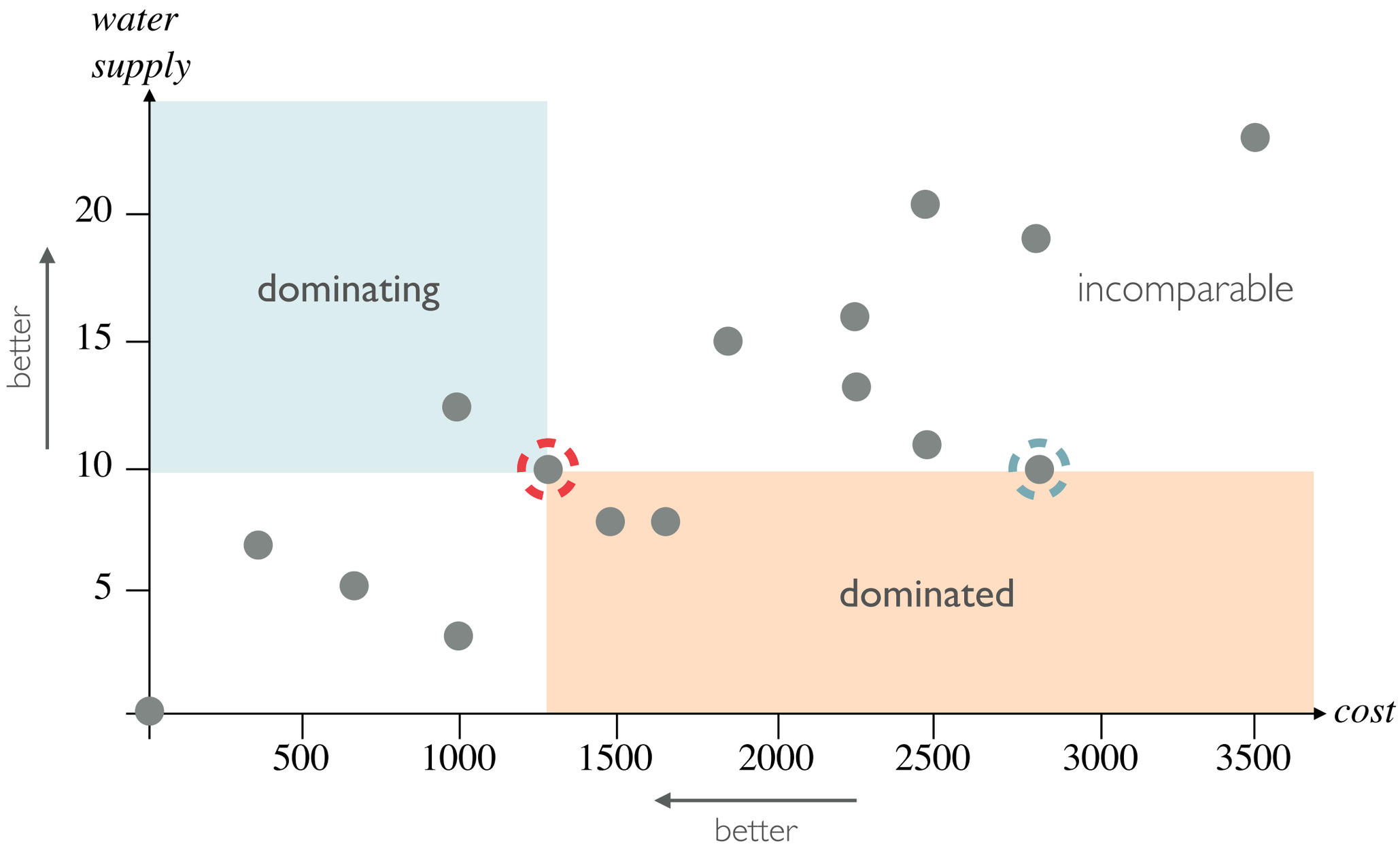
Given a certain solution, we can identify solutions that are certainly better (worse) than it, i.e. they are better (worse) for all objectives, or solutions that are "incomparable", i.e. they are better w.r.t. at least one objective, but worse w.r.t. to the others.



MULTI-OBJECTIVE OPTIMIZATION

PARETO-DOMINANCE

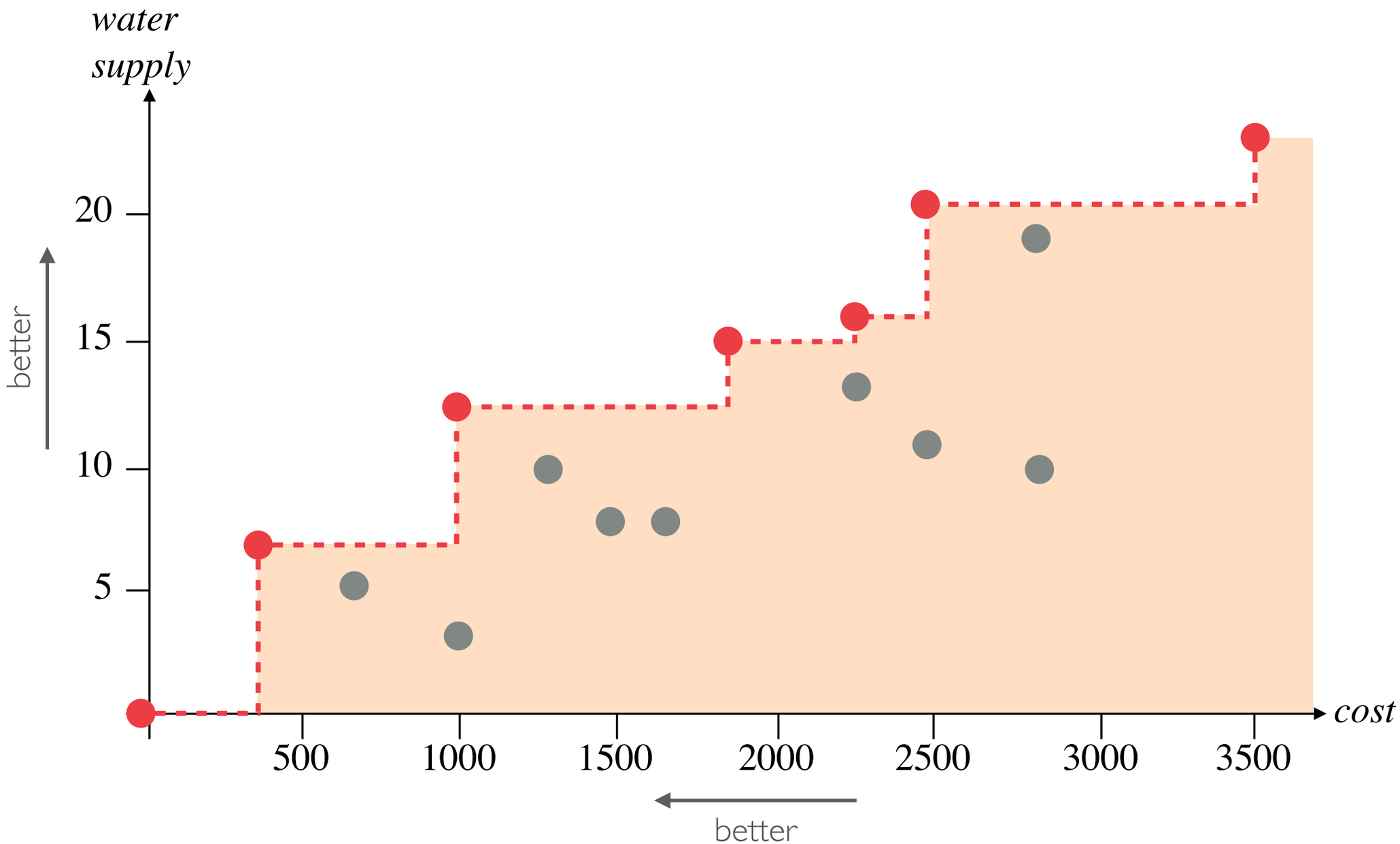
Definition: A solution *i* is said to "Pareto-dominate" a solution *j* if (both conditions must hold!):
- *i* no worse than *j* **on all objectives**
- *i* is better than *j* **on at least one objective**



MULTI-OBJECTIVE OPTIMIZATION

PARETO-DOMINANCE

There is no single optimal solution, BUT: some solutions (red dots) are better than all others (gray dots). These solutions are exactly those that are **not dominated** by any others. They form the so-called **non-dominated (Pareto) front**.

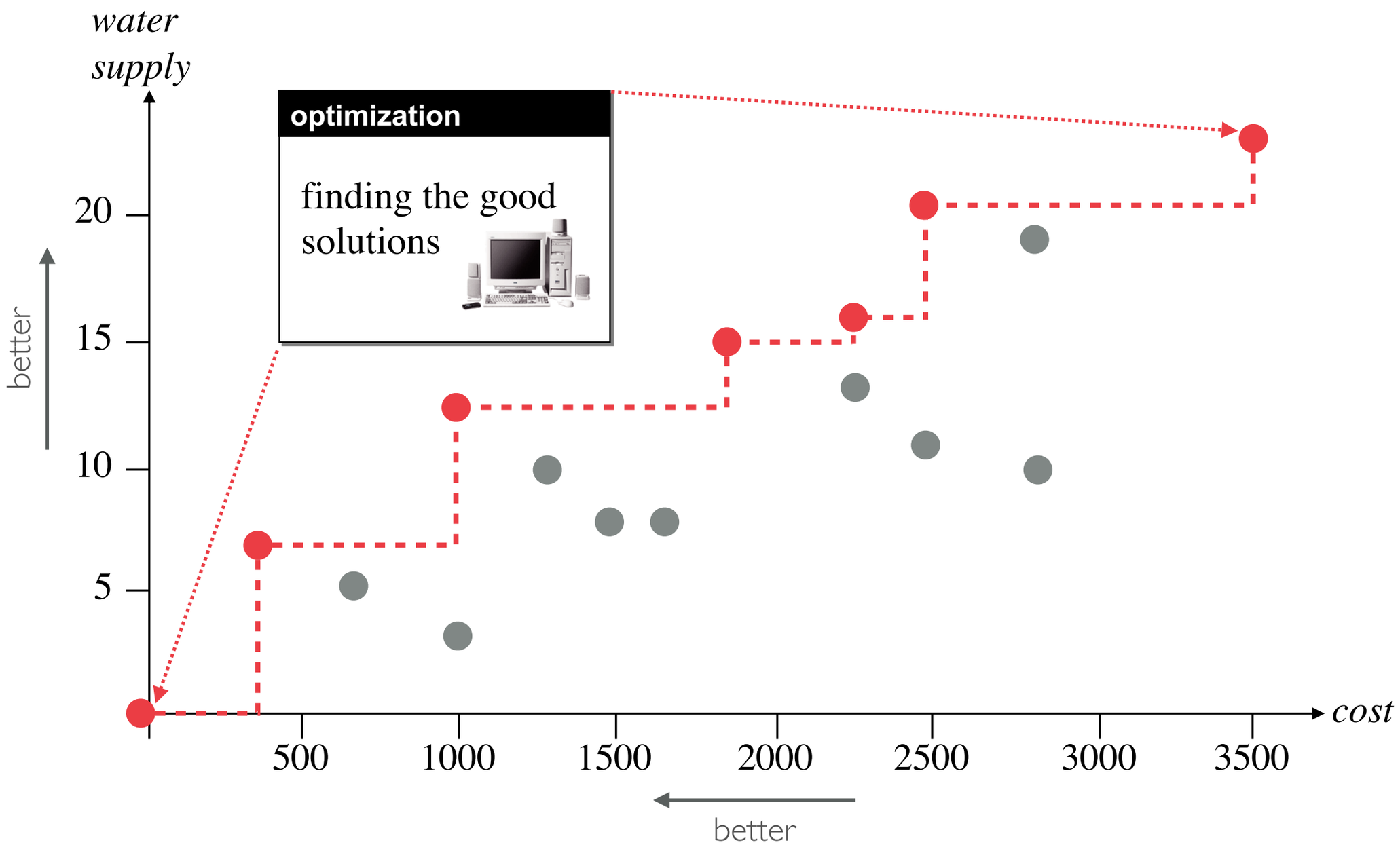


MULTI-OBJECTIVE OPTIMIZATION

MULTI-OBJECTIVE OPTIMIZATION (MOO) VS MULTI-CRITERIA DECISION MAKING (MCDM)

Multi-objective optimization problems (MOPs) are usually solved in two steps:

1. Find the "good" solutions, i.e. the solutions belonging to the Pareto front (optimization)

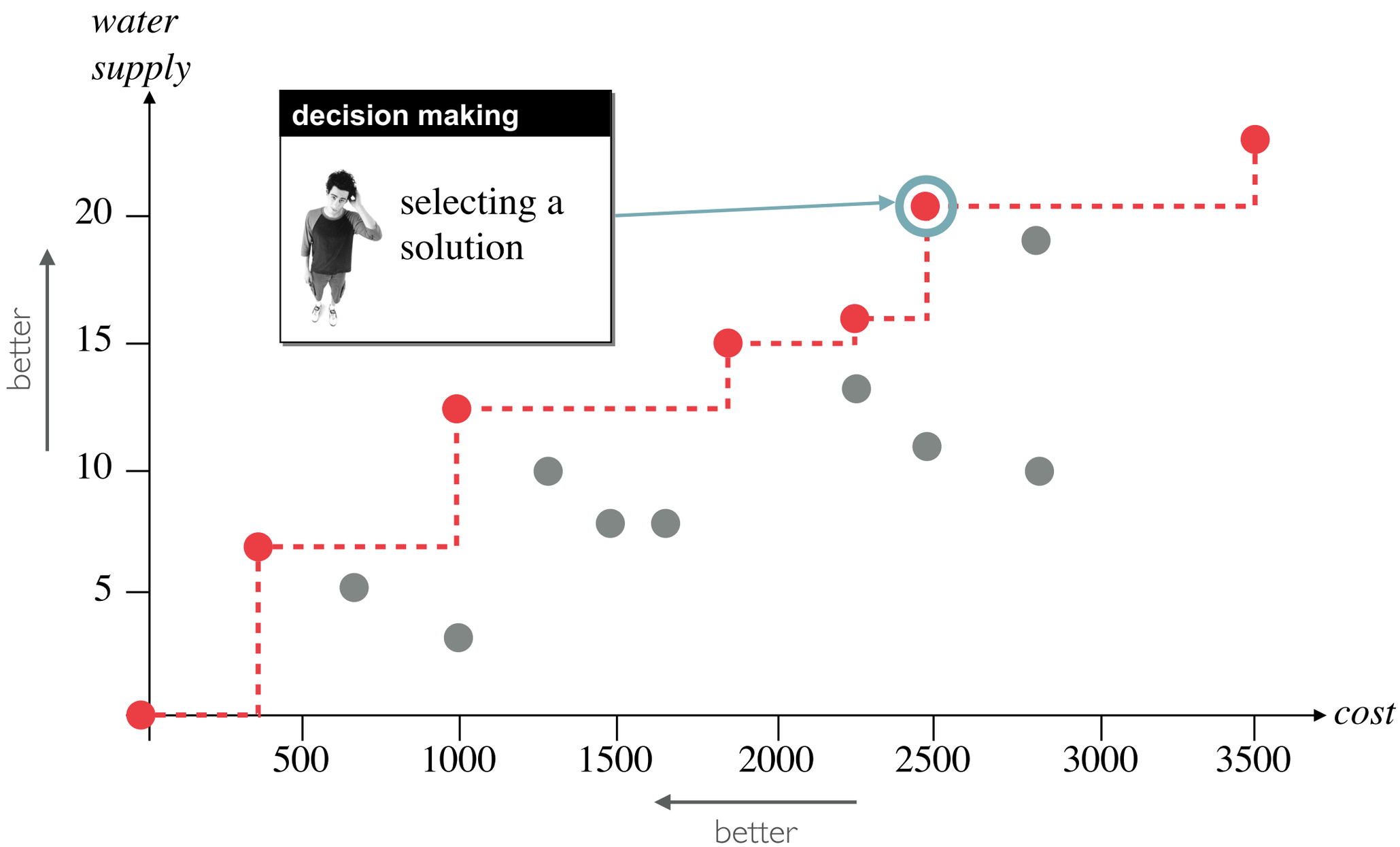


MULTI-OBJECTIVE OPTIMIZATION

MULTI-OBJECTIVE OPTIMIZATION (MOO) VS MULTI-CRITERIA DECISION MAKING (MCDM)

Multi-objective optimization problems (MOPs) are usually solved in two steps:

2. Select the solution that is needed for the particular interest (decision making)



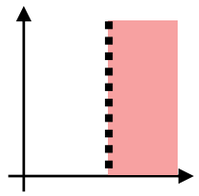
MULTI-OBJECTIVE OPTIMIZATION

WHEN TO MAKE A DECISION

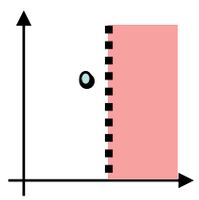
The main difference between "a priori" methods (scalarization, lexicographic method) and "a posteriori" methods (based on Pareto dominance) is when decisions about a problem are made: *before* or *after* optimization.

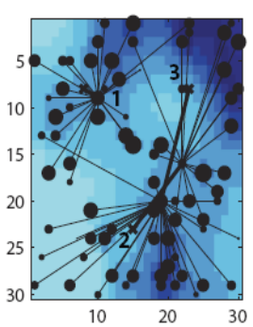
Before Optimization:

rank objectives, define constraints,…



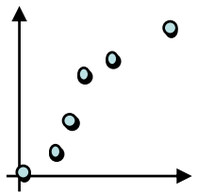
search for one solution



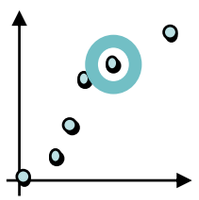


After Optimization:

search for a set of (Pareto-optimal) solutions



select one solution considering constraints, etc.



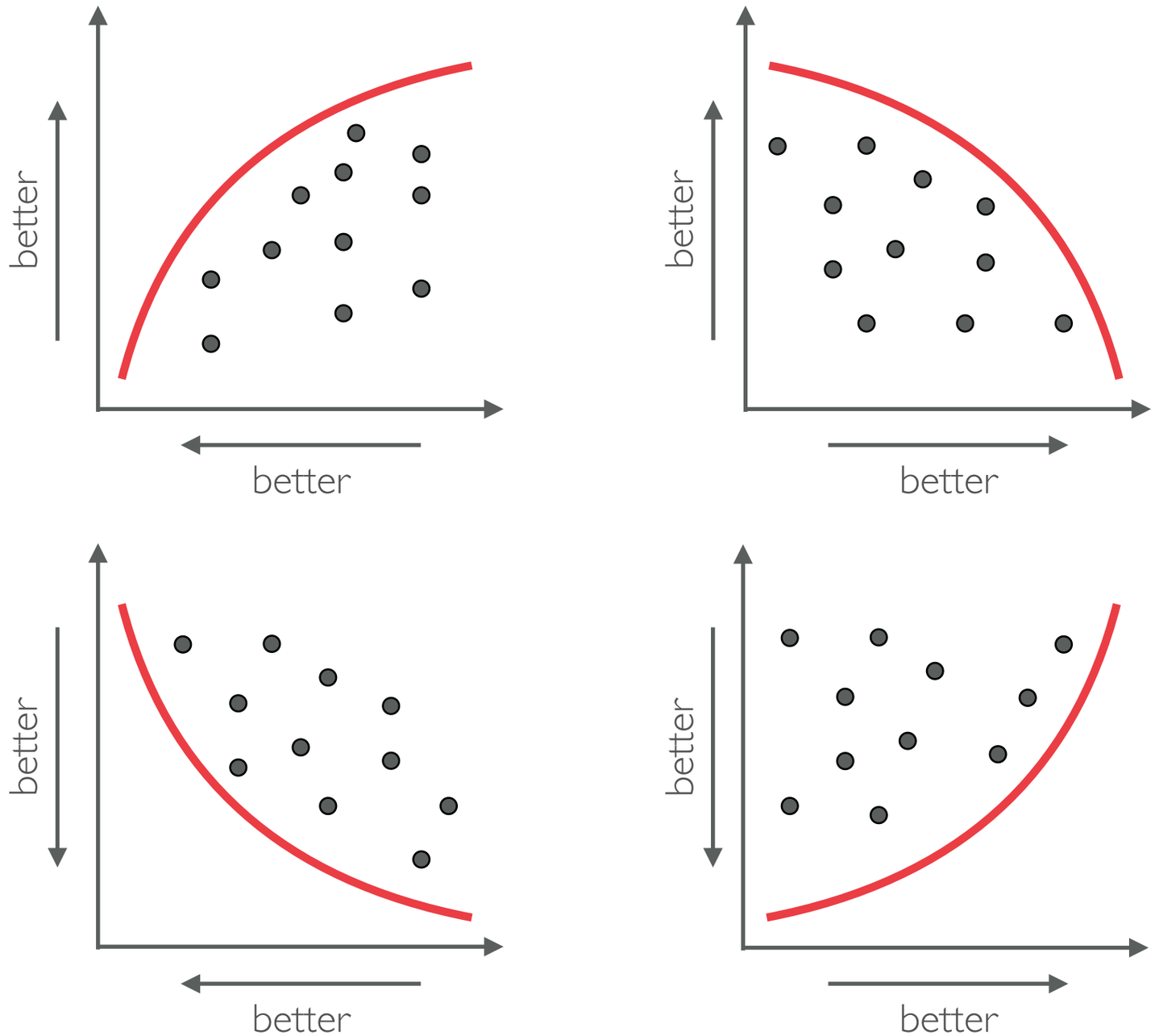
Focus: learning about a problem

- trade-off surface
- interactions among criteria
- structural information

MULTI-OBJECTIVE OPTIMIZATION

THE PARETO FRONT: DIFFERENT "SHAPES", DIFFERENT PROPERTIES

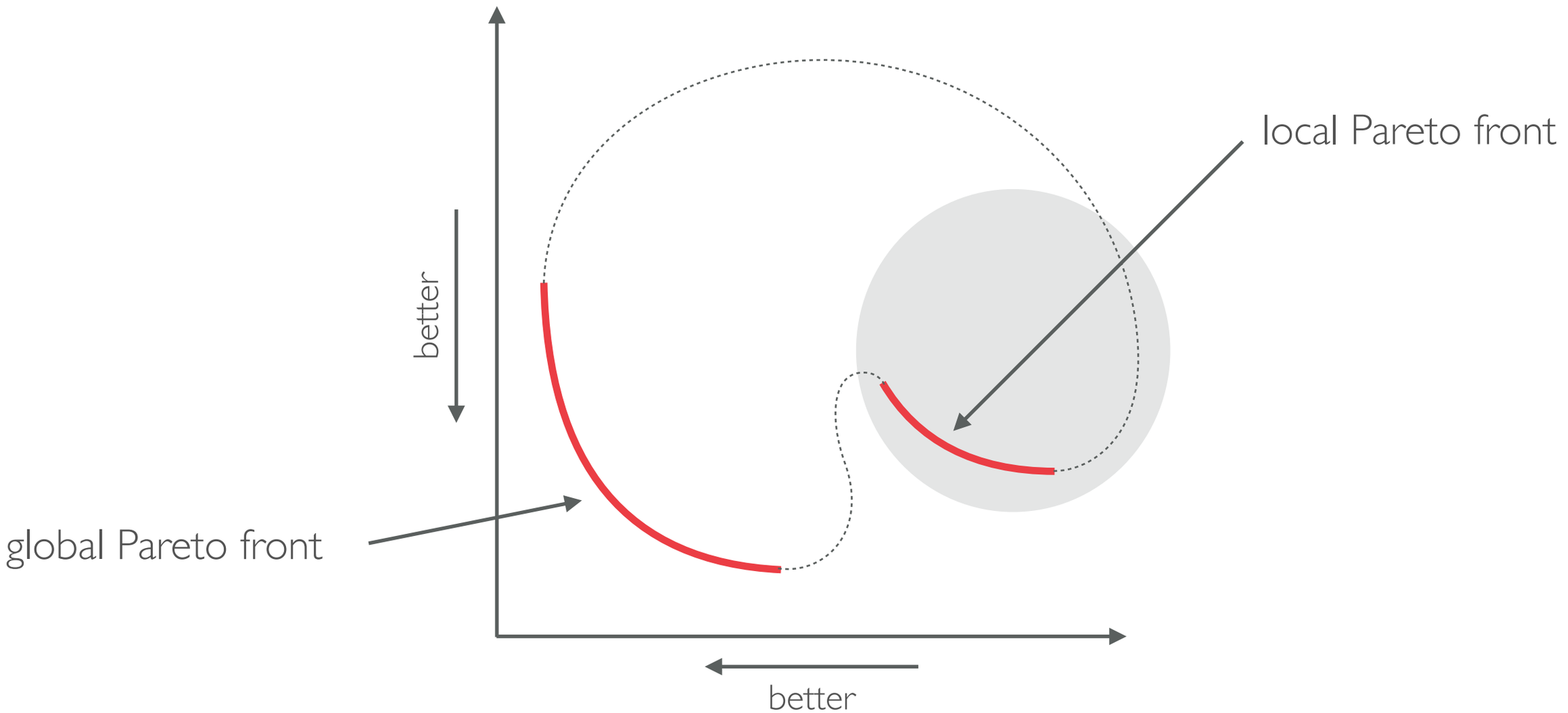
Depending on the specific problem and the minimization/maximization of each objective, the Pareto front assumes different shapes and properties (convex/non-convex, continuous/discontinuous, etc.).



MULTI-OBJECTIVE OPTIMIZATION

THE PARETO FRONT: DIFFERENT "SHAPES", DIFFERENT PROPERTIES

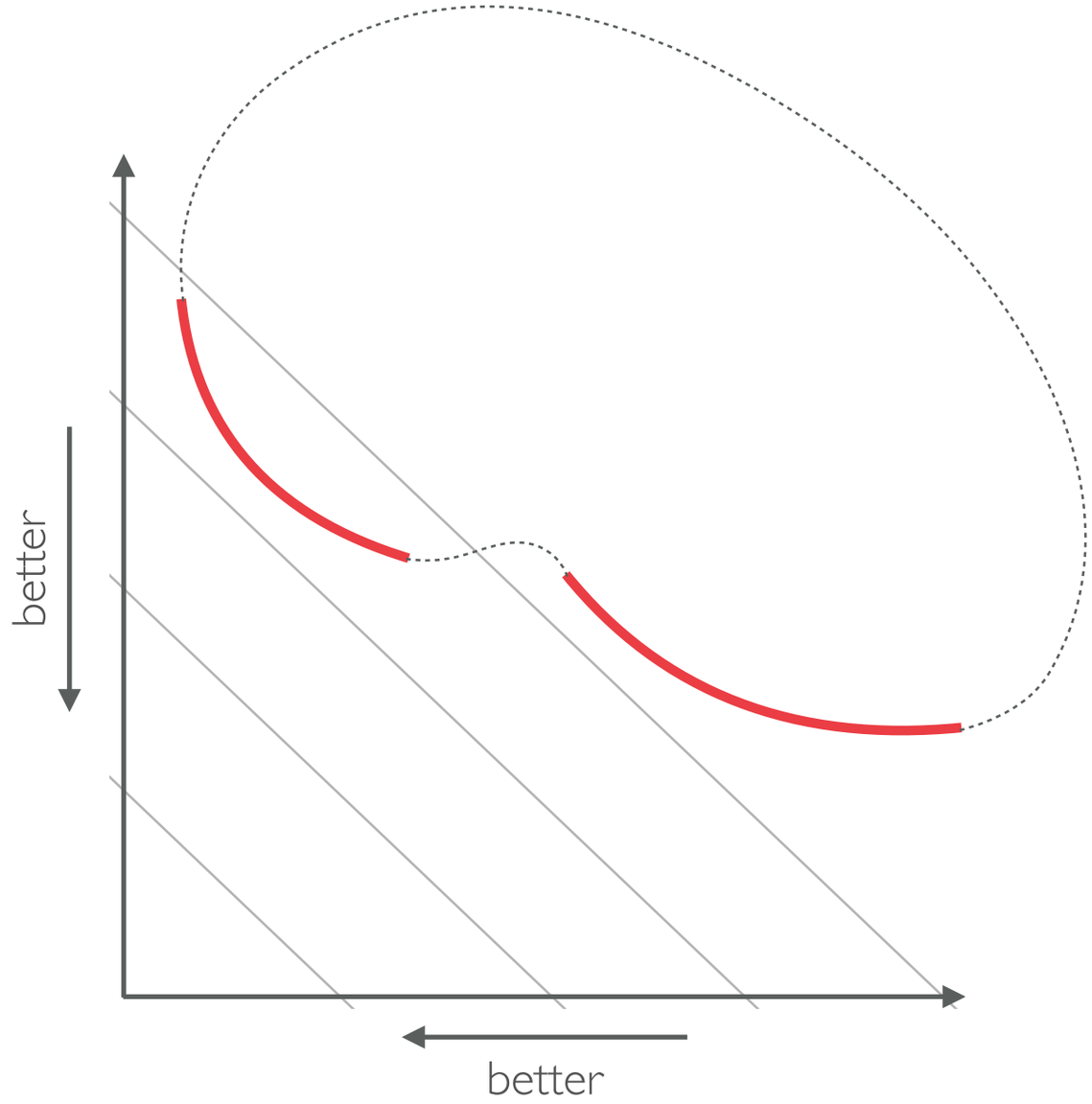
Some solutions can be"locally" non-dominated i.e., they form a Pareto front w.r.t. a local neighborhood in the fitness space, that may be different from the global Pareto front.



MULTI-OBJECTIVE OPTIMIZATION

PROBLEMS WITH LINEAR SCALARIZATION

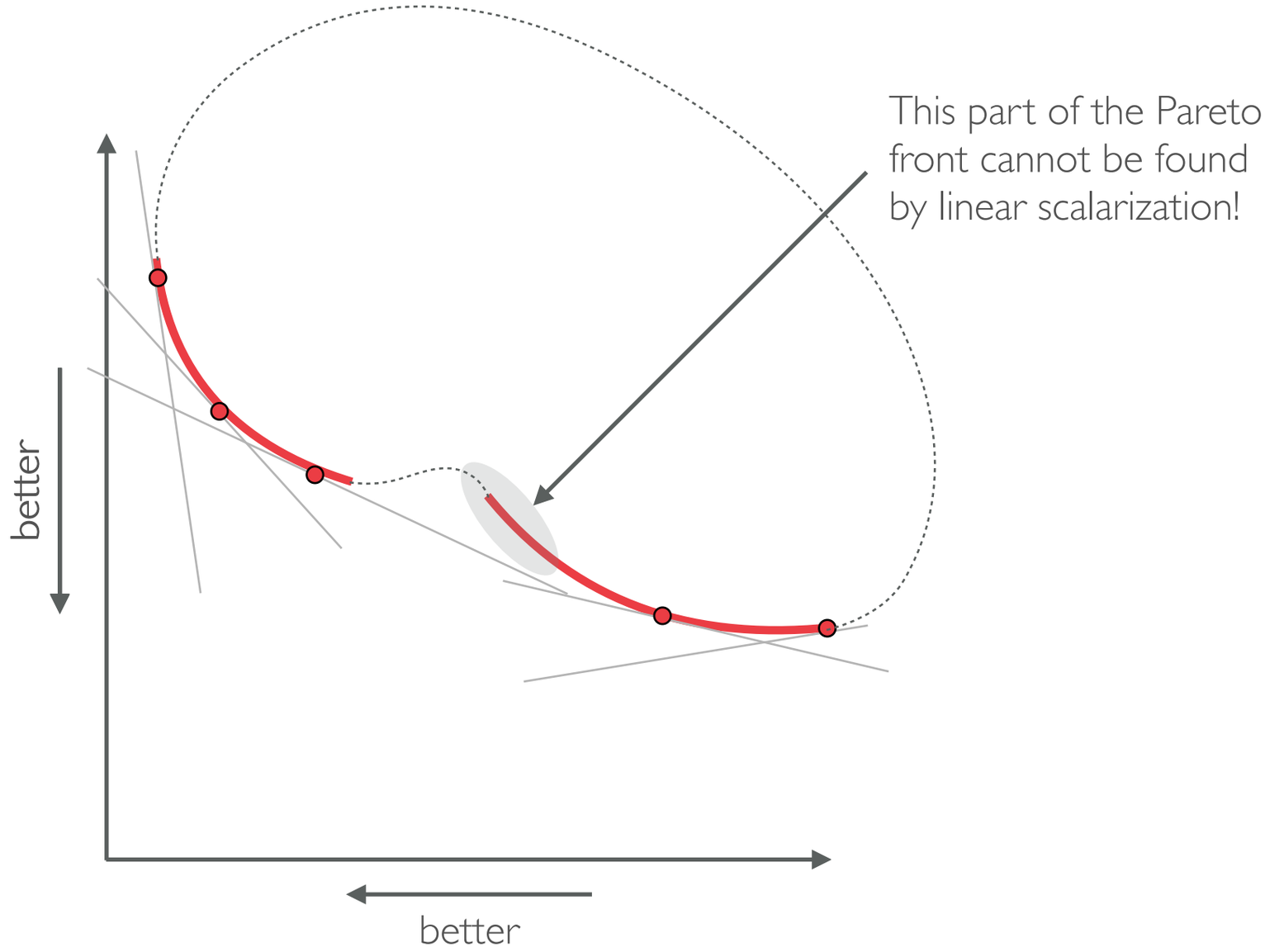
If the Pareto front is non-convex, some regions of the Pareto front cannot even be found by linear scalarization (weighted sum)!



MULTI-OBJECTIVE OPTIMIZATION

PROBLEMS WITH LINEAR SCALARIZATION

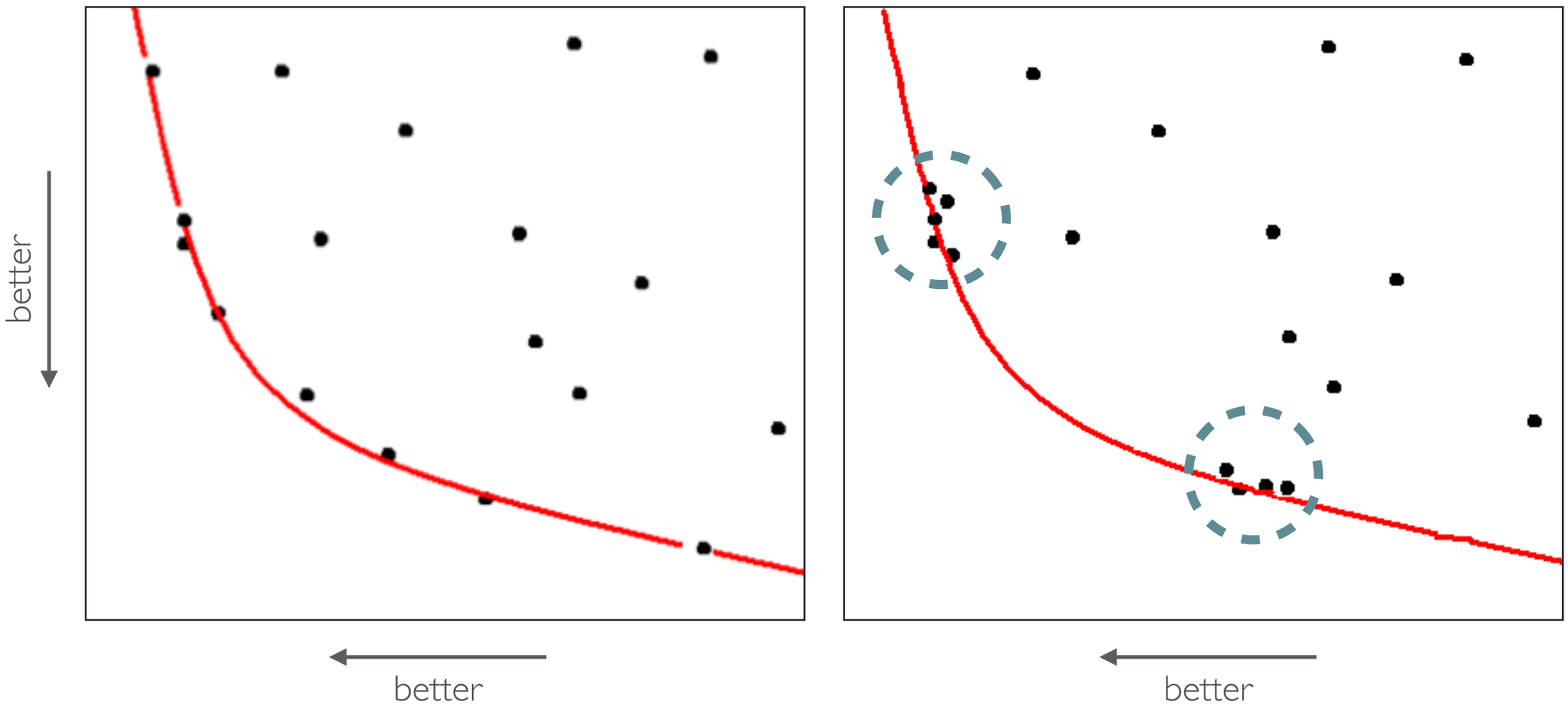
If the Pareto front is non-convex, some regions of the Pareto front cannot even be found by linear scalarization (weighted sum)!



MULTI-OBJECTIVE OPTIMIZATION

NOT ALL PARETO SOLUTIONS ARE BORN EQUAL

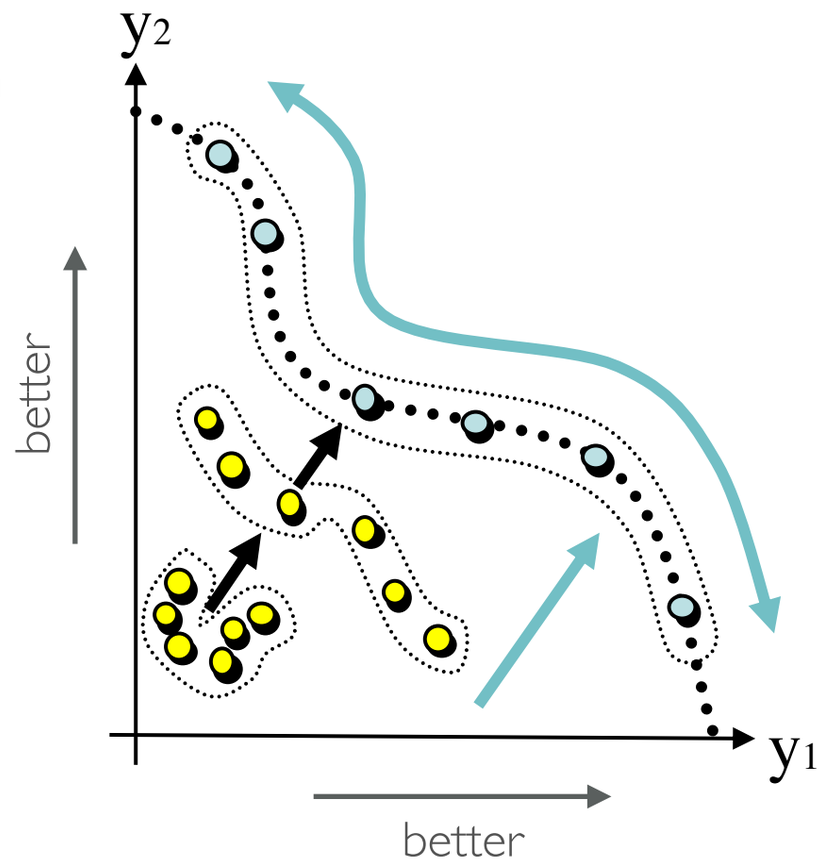
An important concept: **crowding** (how close are the non-dominated solutions to each other).
Ideally, one wants to cover all the Pareto front as evenly as possible (i.e., with a uniform density).



MULTI-OBJECTIVE OPTIMIZATION

NON-DOMINATED SORTING GENETIC ALGORITHM-II (NSGA-2 or NSGA-II)

- Originally proposed by Deb et al. in 2002, it represents a milestone in multi-objective optimization.
- Main features
  - Typically used for binary or real-valued representations
  - The genetic operators (mutation and crossover) are essentially the same used for single-objective optimization
  - The selection mechanism (with **elitism**) is composed of two parts:
    1) Pareto rank based on **non-dominated sorting**, to select the non-dominated solutions according to their dominance levels
    2) **crowding-distance sorting** to prefer "isolated" solutions to better represent the Pareto front
       → acts also as a diversity preservation mechanism

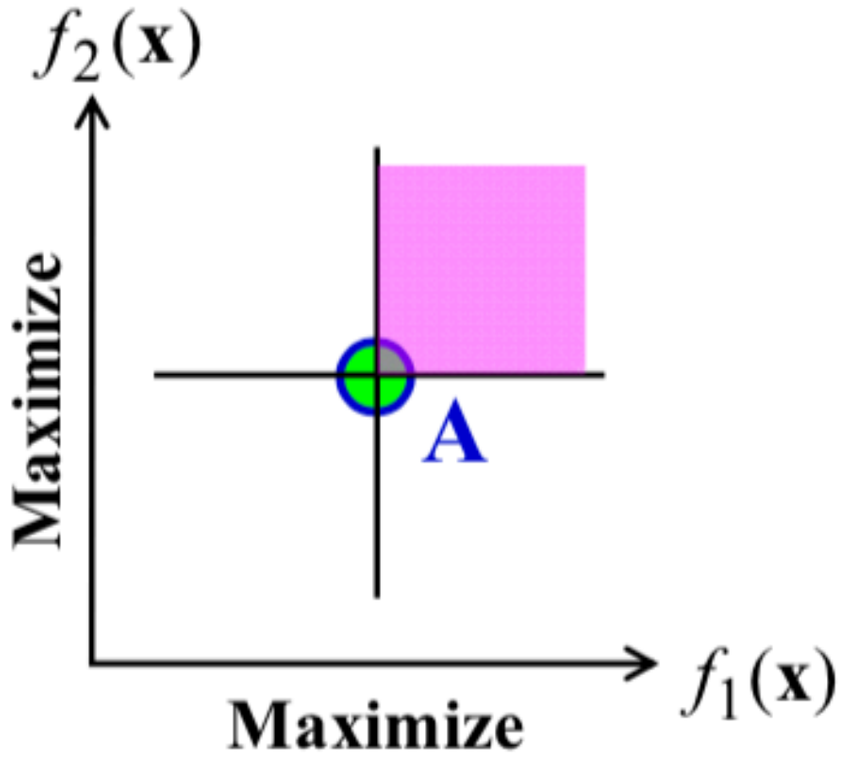


- Applied widely and successfully, but like other multi-objective evolutionary algorithms its performance breaks down as the number of objectives increases (curse of dimensionality!)

MULTI-OBJECTIVE OPTIMIZATION

CURSE OF DIMENSIONALITY

The more objectives are added, the harder it becomes to find non-dominated solutions (i.e., the probability of generating a non-dominated solution becomes increasingly smaller)



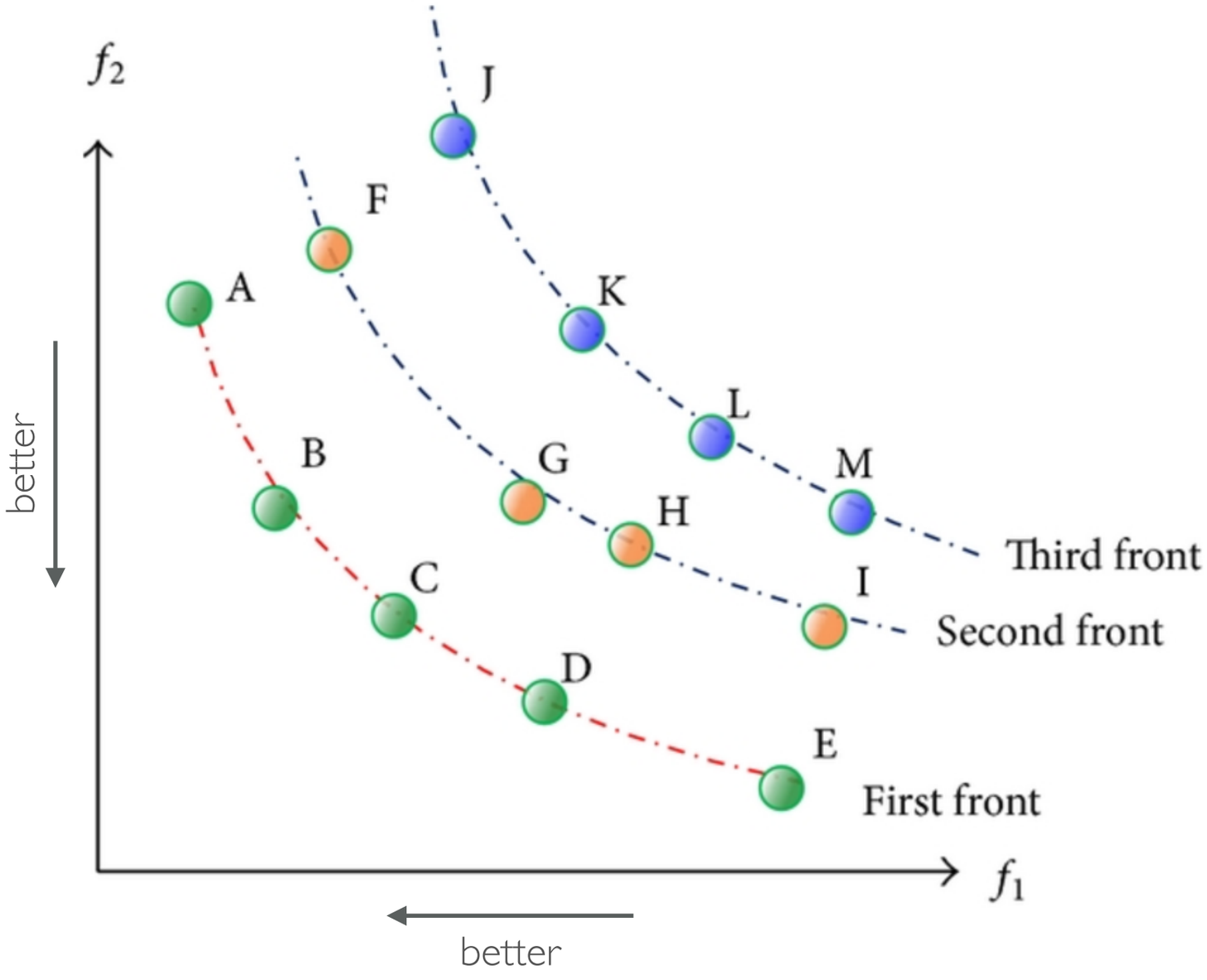
Percentage of the better region

2 objectives	1/4	25%
3 objectives	1/8	13%
4 objectives	1/16	6%
5 objectives	1/32	3%
10 objectives	1/1024	0.1%
15 objectives	1/32768	0.003%
20 objectives	1/1048576	0.0001%

MULTI-OBJECTIVE OPTIMIZATION

PARETO FRONT LEVELS

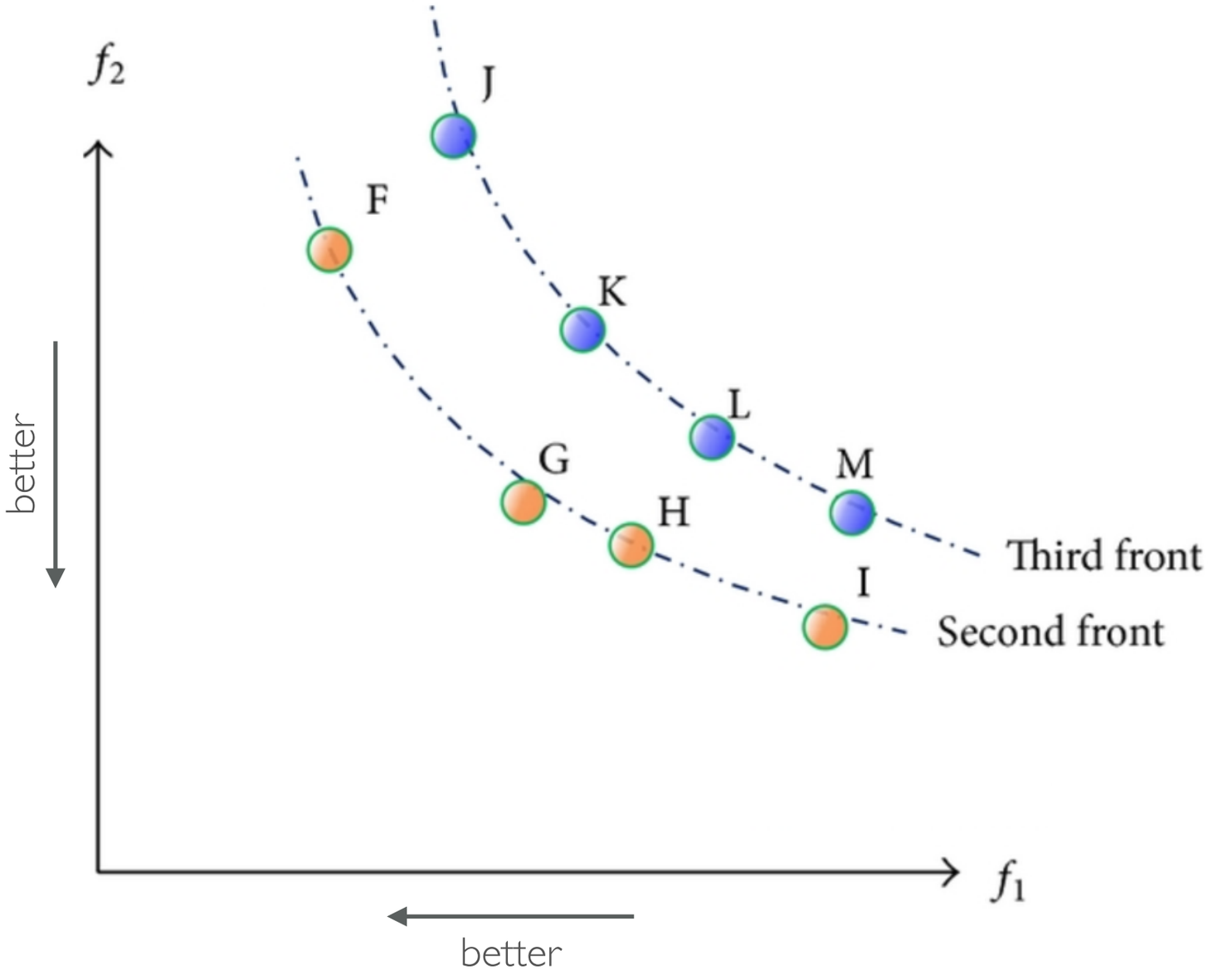
An important concept: **dominance levels.** Solutions can be ranked according to their level of dominance: non-dominated solutions (Pareto front) have rank R1, solutions that are dominated only by R1-solutions have rank R2, and so on (recursively) → **multiple fronts can be identified!**



MULTI-OBJECTIVE OPTIMIZATION

PARETO FRONT LEVELS

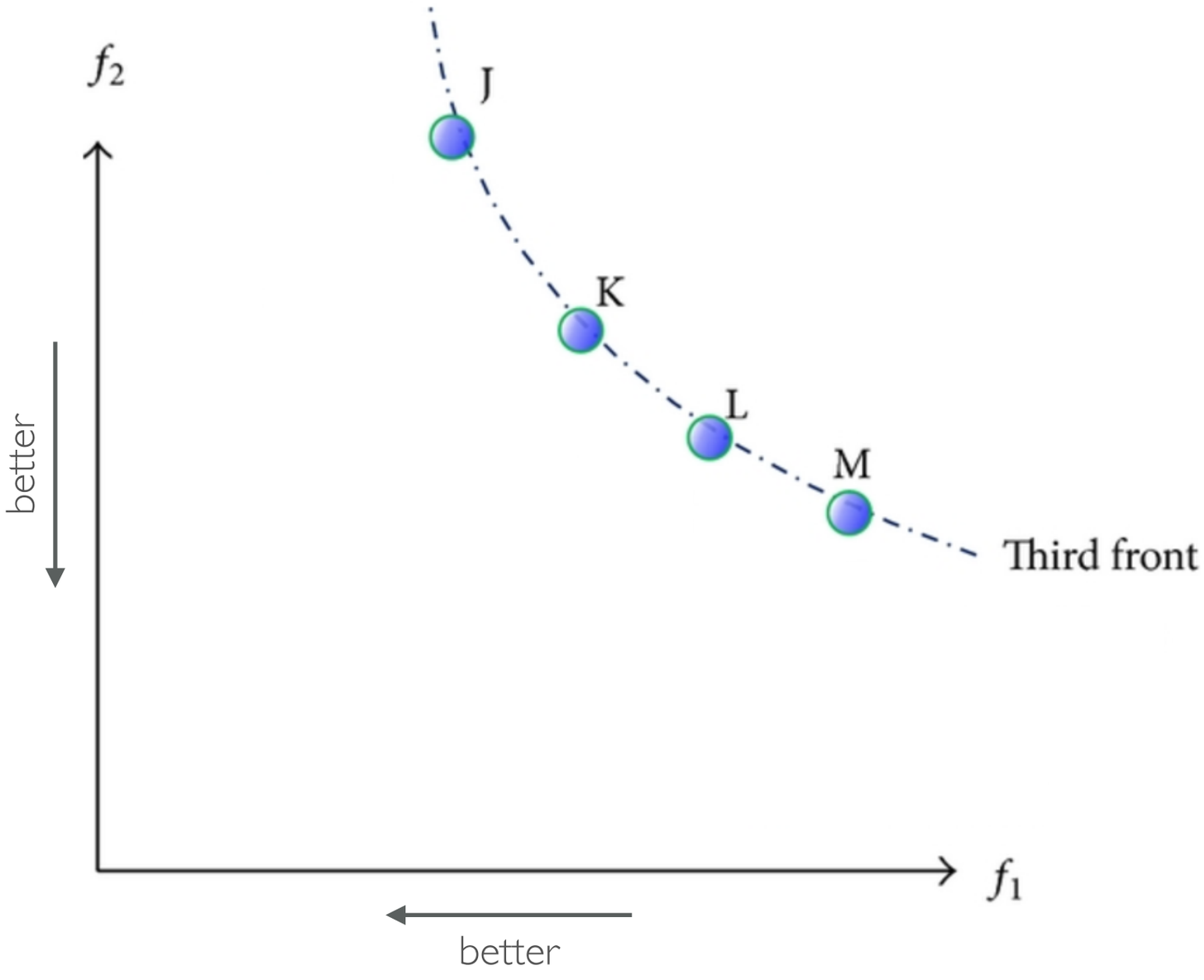
An important concept: **dominance levels.** Solutions can be ranked according to their level of dominance: non-dominated solutions (Pareto front) have rank R1, solutions that are dominated only by R1-solutions have rank R2, and so on (recursively) → **multiple fronts can be identified!**



MULTI-OBJECTIVE OPTIMIZATION

PARETO FRONT LEVELS

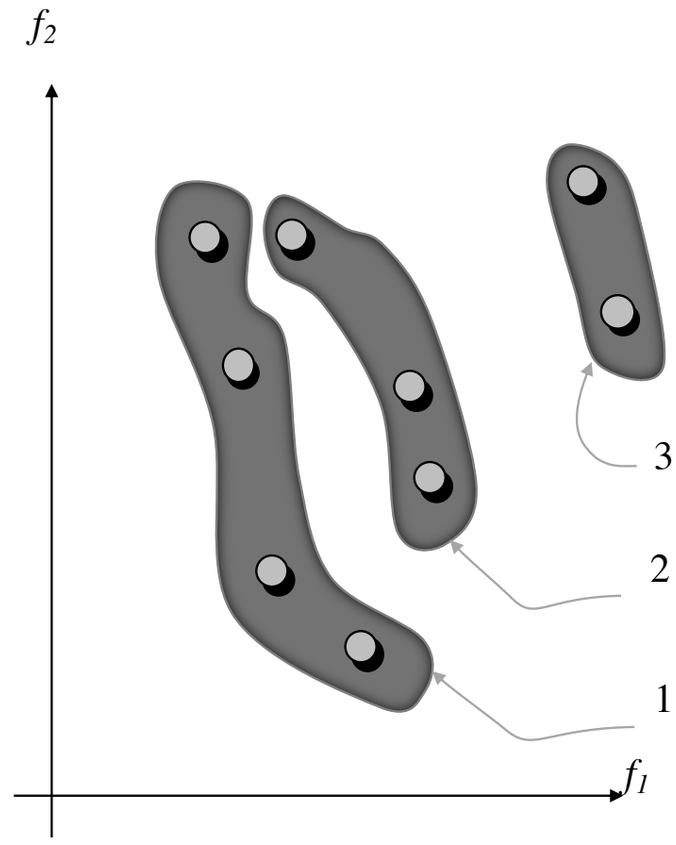
An important concept: **dominance levels.** Solutions can be ranked according to their level of dominance: non-dominated solutions (Pareto front) have rank R1, solutions that are dominated only by R1-solutions have rank R2, and so on (recursively) → **multiple fronts can be identified!**



MULTI-OBJECTIVE OPTIMIZATION

ALGORITHM DETAILS: NON-DOMINATED SORTING

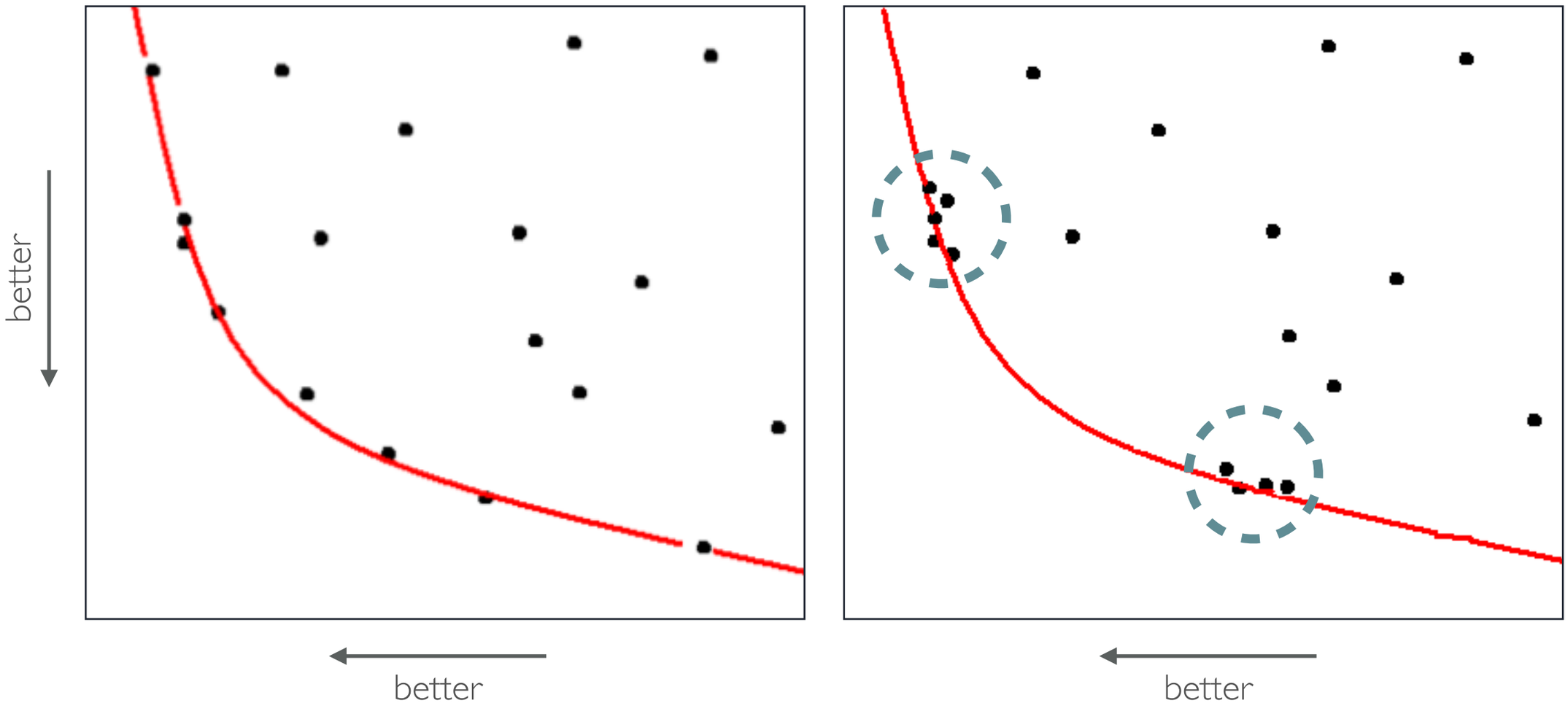
```
fast-non-dominated-sort(P)
for each p ∈ P
    S_p = ∅
    n_p = 0
    for each q ∈ P
        if (p ≺ q) then                    If p dominates q
            S_p = S_p ∪ {q}                Add q to the set of solutions dominated by p
        else if (q ≺ p) then
            n_p = n_p + 1                  Increment the domination counter of p
    if n_p = 0 then                        p belongs to the first front
        p_rank = 1
        F_1 = F_1 ∪ {p}
i = 1                                      Initialize the front counter
while F_i ≠ ∅
    Q = ∅                                  Used to store the members of the next front
    for each p ∈ F_i
        for each q ∈ S_p
            n_q = n_q − 1
            if n_q = 0 then                q belongs to the next front
                q_rank = i + 1
                Q = Q ∪ {q}
    i = i + 1
    F_i = Q
```



MULTI-OBJECTIVE OPTIMIZATION

REMEMBER: CROWDING SHOULD BE AVOIDED!

An important concept: **crowding** (how close are the non-dominated solutions to each other). Ideally, one wants to cover all the Pareto front as evenly as possible (i.e. with a uniform density).



MULTI-OBJECTIVE OPTIMIZATION

ALGORITHM DETAILS: CROWDING-DISTANCE SORTING

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

$l = |\mathcal{I}|$ — number of solutions in $\mathcal{I}$

for each i, set $\mathcal{I}[i]_{\text{distance}} = 0$ — initialize distance

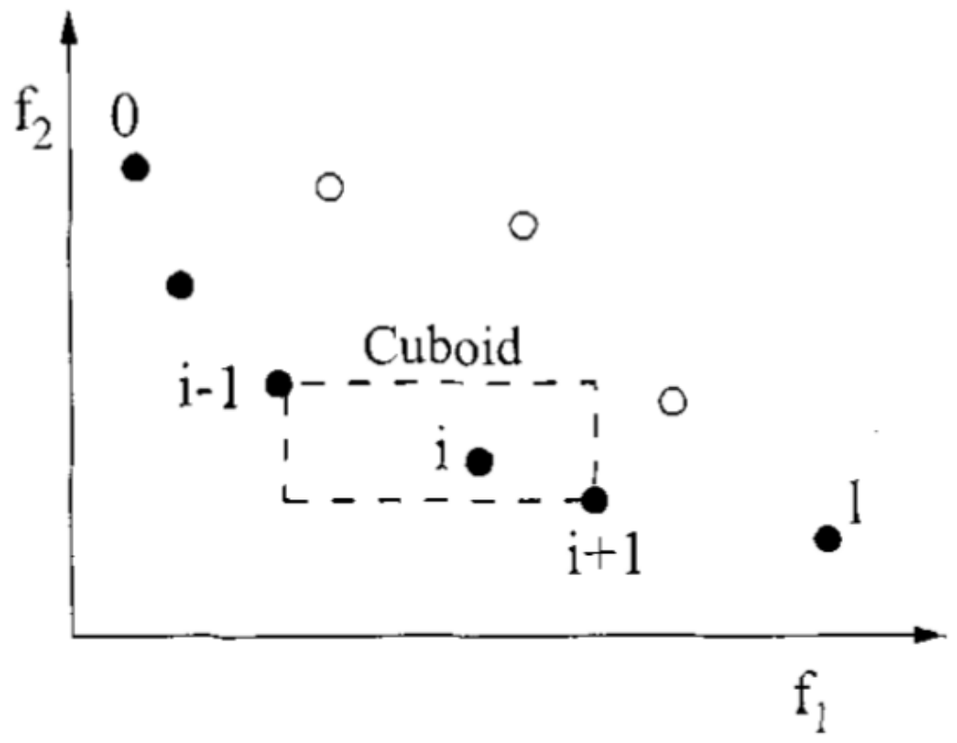
for each objective m

$\mathcal{I} = \text{sort}(\mathcal{I}, m)$ — sort using each objective value

$\mathcal{I}[1]_{\text{distance}} = \mathcal{I}[l]_{\text{distance}} = \infty$ — so that boundary points are always selected

for $i = 2$ to $(l - 1)$ — for all other points

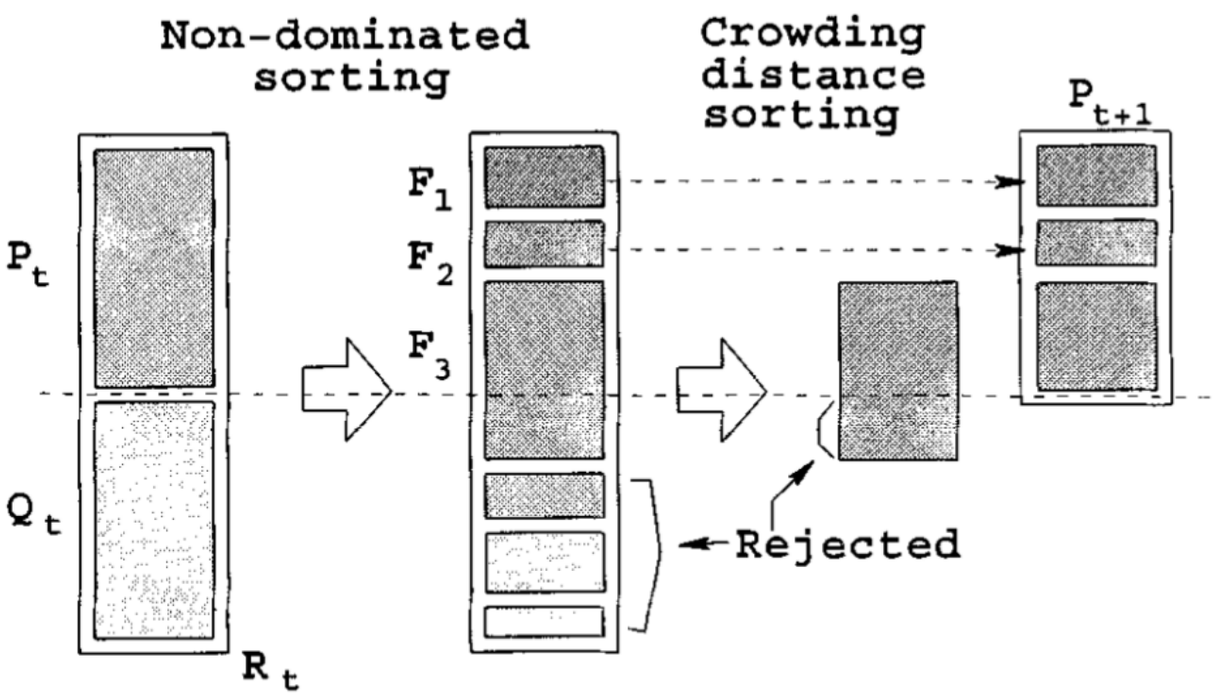
$$\mathcal{I}[i]_{\text{distance}} = \mathcal{I}[i]_{\text{distance}} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)/(f_m^{\max} - f_m^{\min})$$



MULTI-OBJECTIVE OPTIMIZATION

ALGORITHM DETAILS: PUTTING ALL TOGETHER

$R_t = P_t \cup Q_t$	combine parent and offspring population
$\mathcal{F} =$ `fast-non-dominated-sort`(R_t)	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all nondominated fronts of R_t
$P_{t+1} = \emptyset$ and $i = 1$	
until $\lvert P_{t+1}\rvert + \lvert\mathcal{F}_i\rvert \leq N$	until the parent population is filled
`crowding-distance-assignment`$(\mathcal{F}_i)$	calculate crowding-distance in $\mathcal{F}_i$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	include ith nondominated front in the parent pop
$i = i + 1$	check the next front for inclusion
Sort$(\mathcal{F}_i, \prec_n)$	sort in descending order using $\prec_n$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - \lvert P_{t+1}\rvert)]$	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}
$t = t + 1$	increment the generation counter



- Parent and offspring populations are combined before sorting, i.e. (μ+λ) strategy → **elitism**
- First preference is given to Pareto rank, then to crowding-distance

Questions?