UNIVERSITÄT PASSAU

Search-Based Software Engineering

Multi-Objective Optimisation

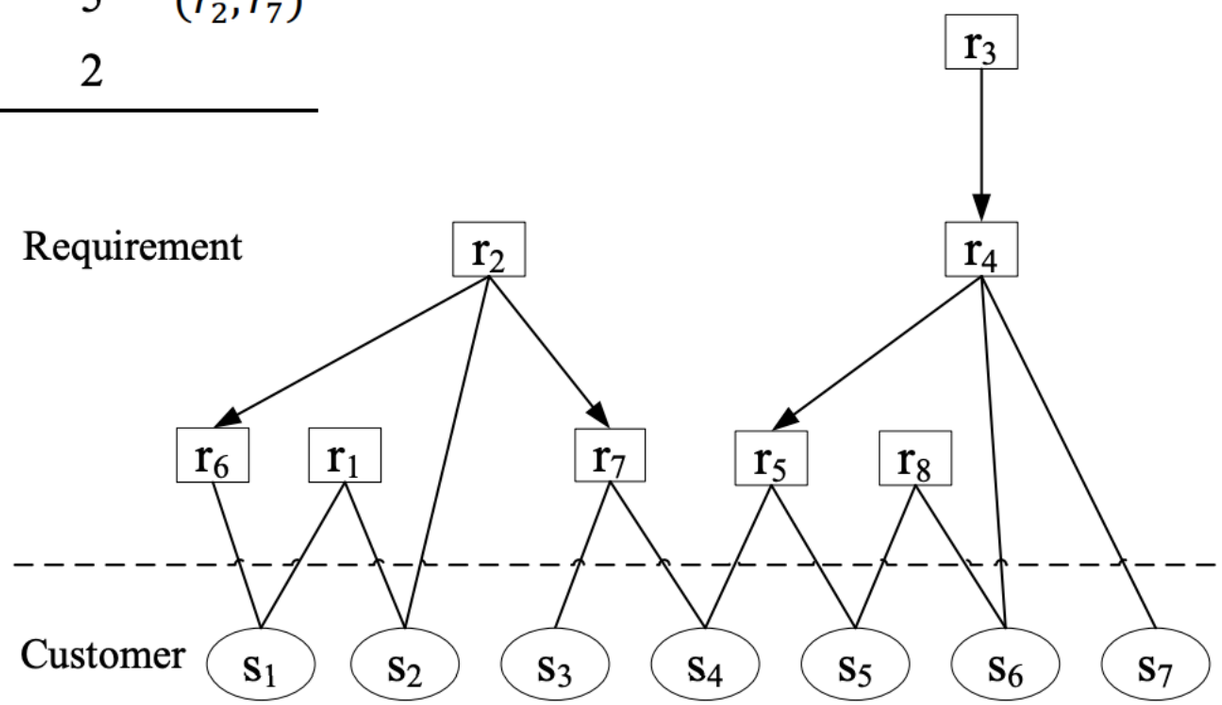
Gordon Fraser
Lehrstuhl für Software Engineering II

Example: Next Release Problem

- Given a set of requirements / user stories
- Which of these requirements to include in the next iteration / release?
- Each requirement has a *cost*
- Each requirement provides *value* to a customer
- We would like to minimise cost, but maximise value

Example: Next Release Problem

Requirement	Description	Cost	Arc
r_1	Expanding memory on BTS controller	2	
r_2	BTS variant	5	
r_3	Market entry feature 1	4	
r_4	Market entry feature 2	3	(r_3, r_4)
r_5	Market entry feature 3	8	(r_4, r_5)
r_6	Next generation BTS	1	(r_2, r_6)
r_7	Pole mount packaging	5	(r_2, r_7)
r_8	Software quality initiative	2	



More Than One Objective

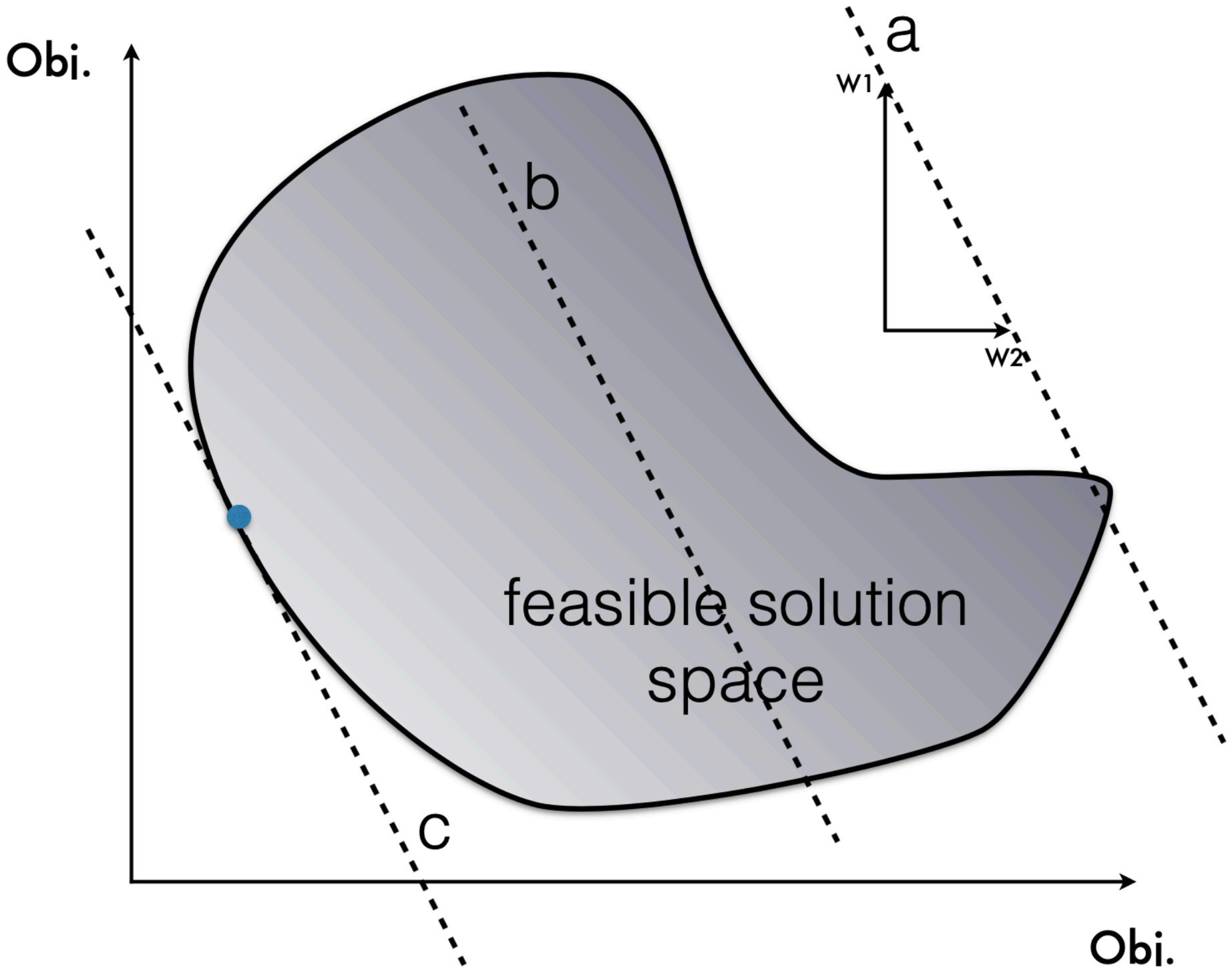
- If you have more than one objective, what would you do with your GA?
  - "I want to maximise profit while minimising costs for the next release"
  - "I want to minimise the number of test cases to execute, but maximise the coverage, detect as many previous faults as possible, minimise test execution time, while using the least amount of power" - ???

Weighted Sum Approach

- Add the objectives into a single fitness value using weights
- Advantages:
  - Simple and easy to use
  - For convex problems, it is guaranteed to find all solutions

$$f = \sum_{i=0}^{n} w_i \cdot f_i$$

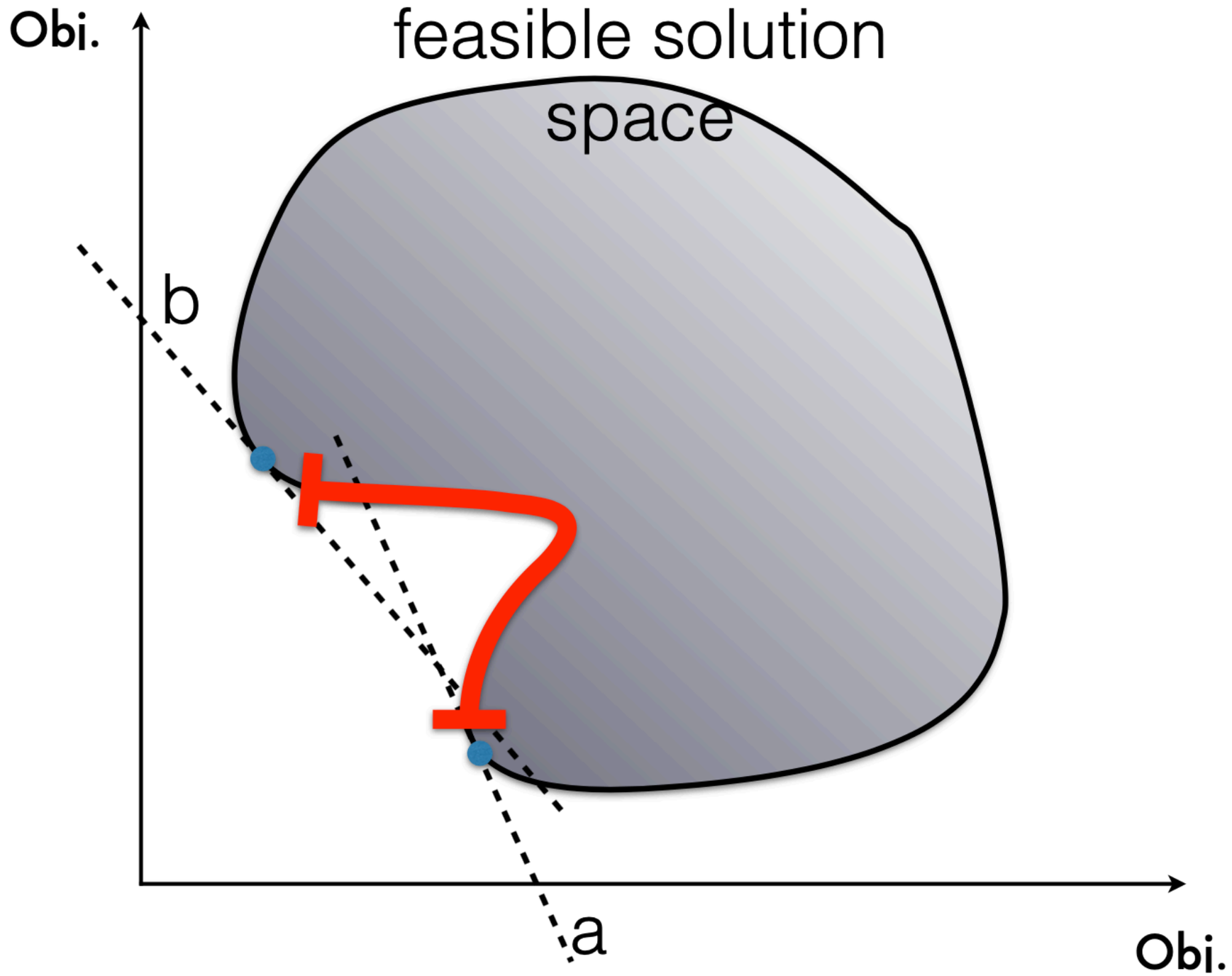
Weighted Sum Approach



Weighted Sum Approach

- Disadvantage:
  - All objectives should be either max or min
  - Yields a single global optimum w.r.t. weights
  - No a priori method to determine weights effectively
  - Non-convex problems

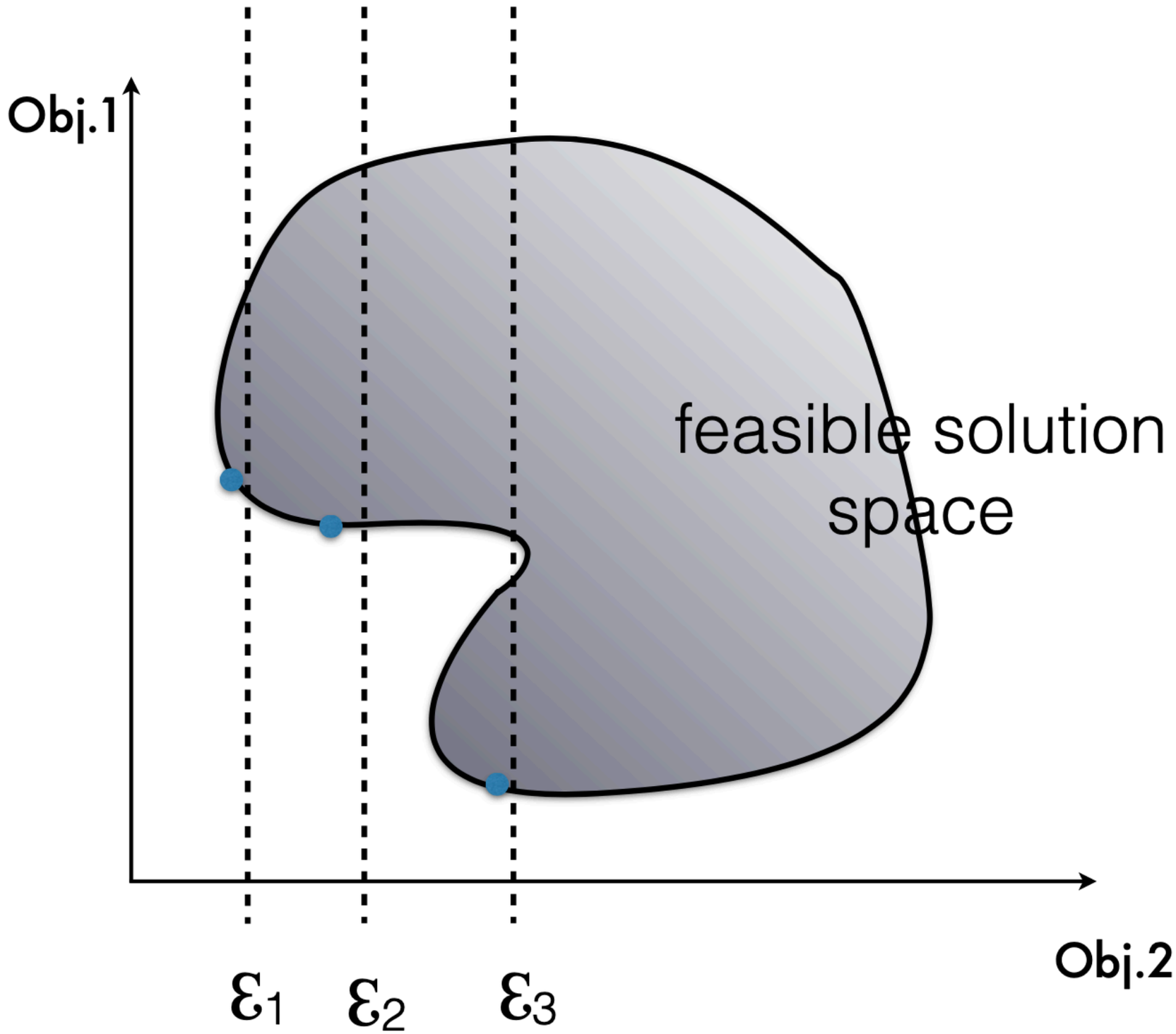
Weighted Sum Approach



ε-Constraint Method

- Focus only on one objective; turn others into user-defined constraints:
- minimise $f_i(x)$
  subject to $[f_1(x), \ldots, f_{i-1}(x), f_{i+1}(x), f_M(x)] \leq \varepsilon$

ε-Constraint Method

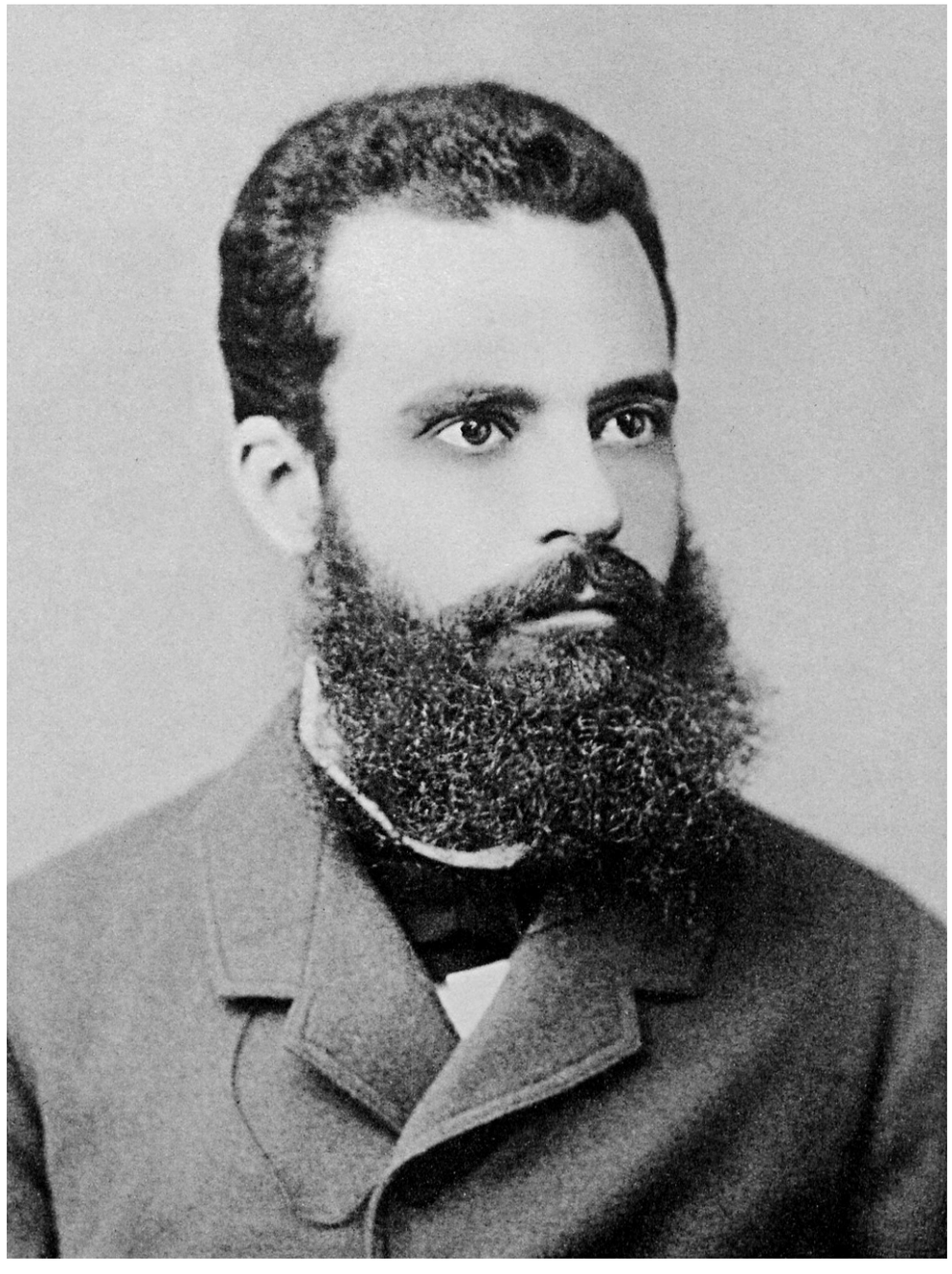


ε-Constraint Method

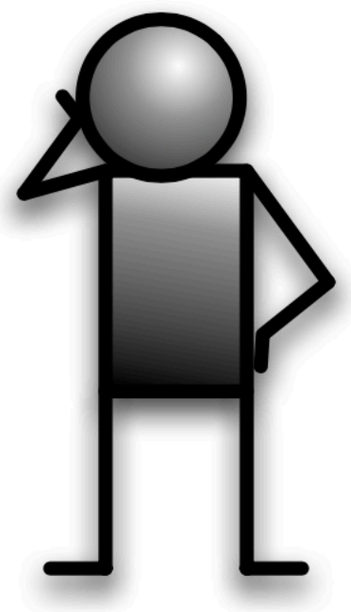
- Advantages:
  - Copes with non-convex solution space
- Disadvantages:
  - ε vector largely decides which part of Pareto front is obtained
  - The higher the dimension, the more input the human needs to provide (ε vector size increases)

Pareto Optimality

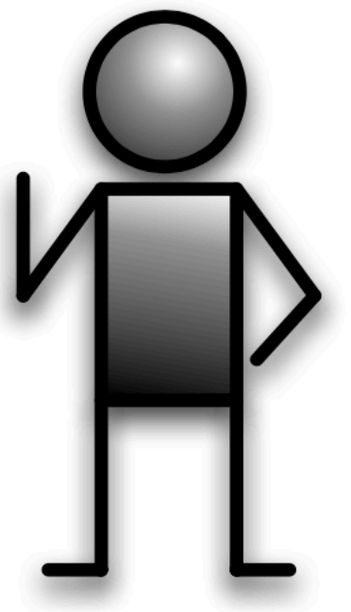
- Vilfredo Pareto
- Economist/Philosopher, 1848 - 1923
- Given a set of alternative allocations and a set of individuals, a movement is Pareto optimal when it makes at least one individual better off without making anyone else worse off.



Pareto Optimality



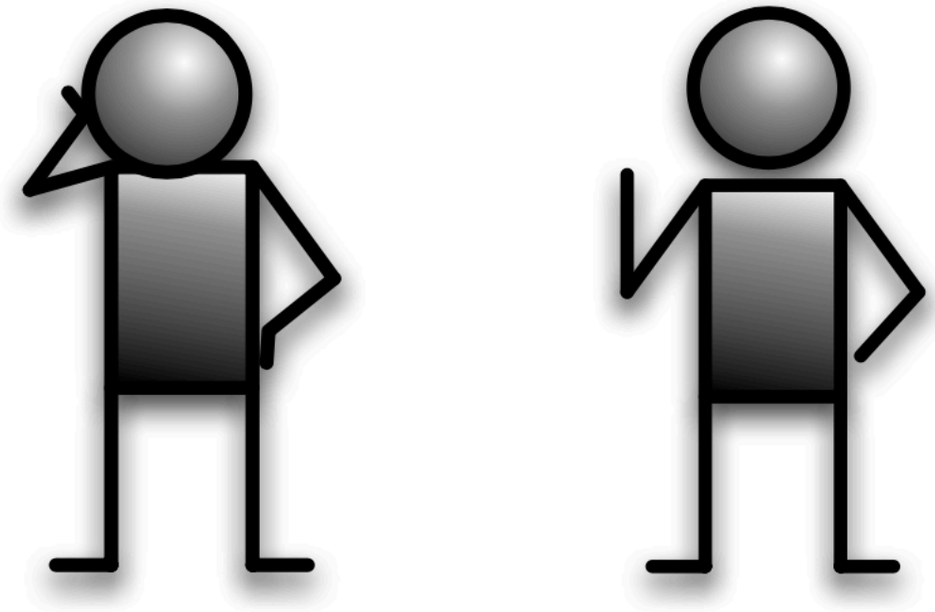
English: 80
Math: 60



English: 70
Math: 90

Who is the better student?

Pareto Optimality



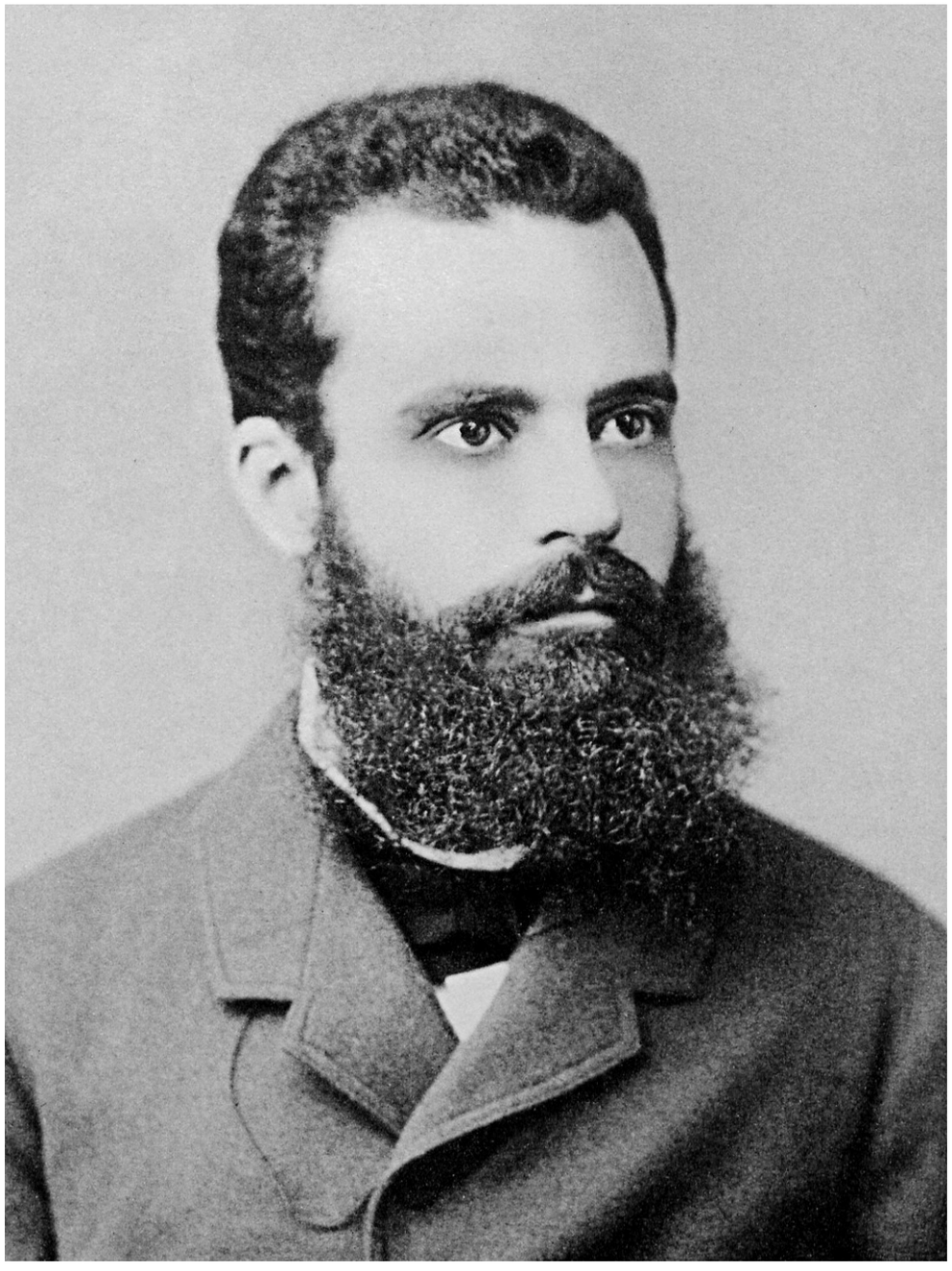
English: 80
Math: 60

English: 70
Math: 90

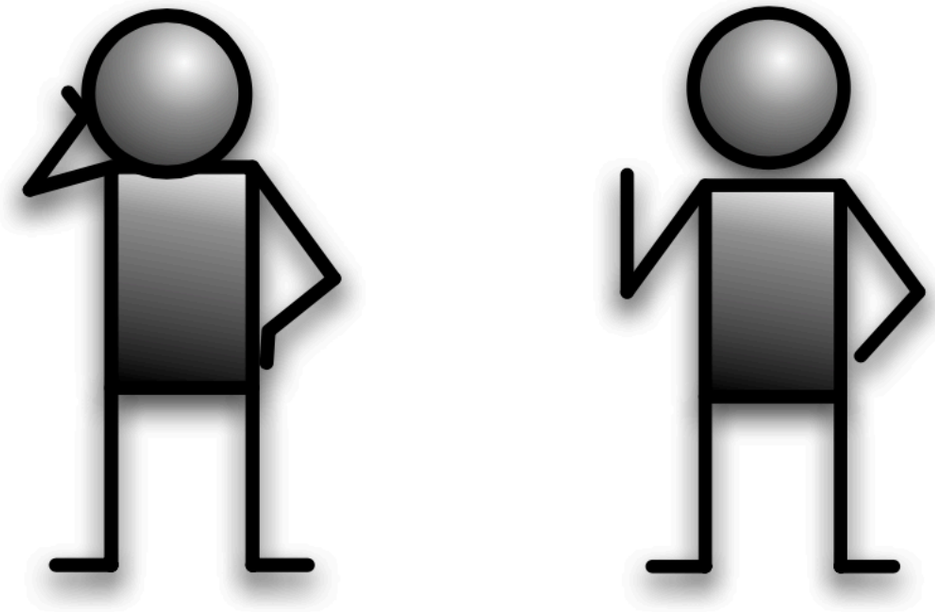
- Average: 70 vs. 80
- What if you are hiring a copywriter?
- What if you are hiring a CS PhD candidate?
- Comparing average assumes that two subjects are equally important, i.e., weights 0.5 and 0.5.

Pareto Optimality

A change from one allocation of wealth to another is Pareto-optimal if and only if that makes at least one individual better off without making any other individual worse off.



Pareto Optimality

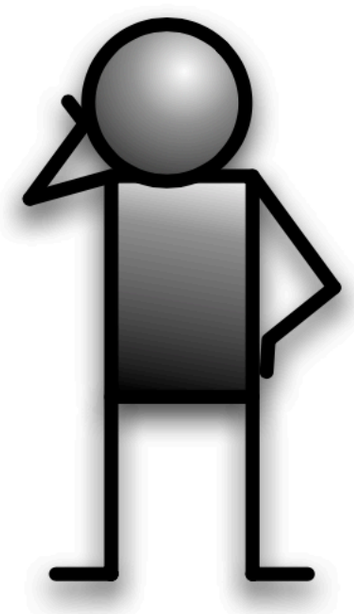


English: 80
Math: 60

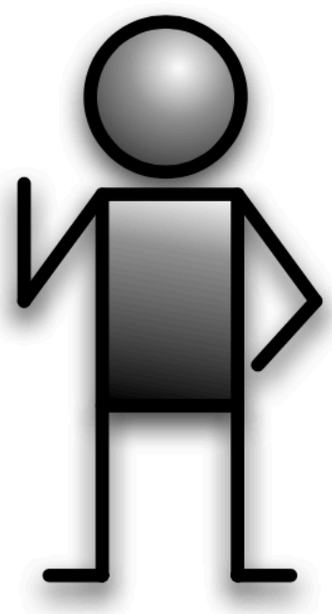
English: 70
Math: 90

A student from one allocation of marks compared to another is Pareto-optimal if and only if he/she excels in at least one subject without being inferior in any other subject.

Pareto Optimality

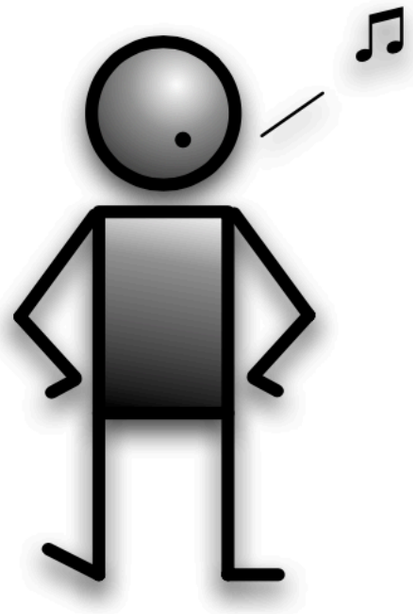


English: 80
Math: 60



English: 70
Math: 90

These two are non-dominating to each other.



English: 80
Math: 90

This person dominates the other two.

How do we optimise with this?

- With single-objective optimisation, we sort the population and pick the best solution.

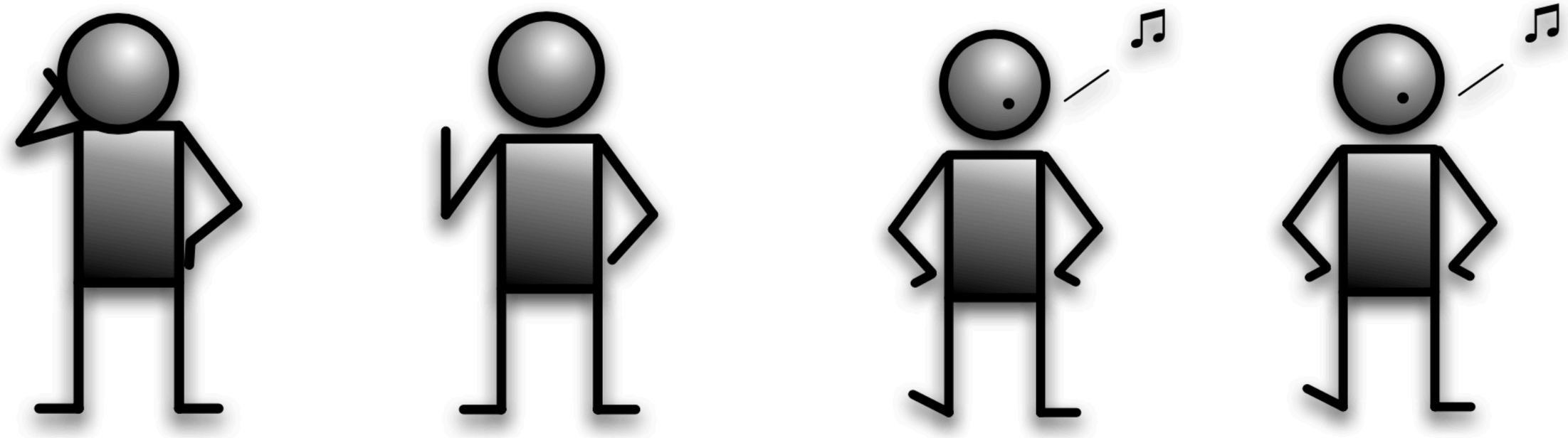
- Now we may not have a single best solution, because there may be a non-dominating pair of solutions (or more!).

- As a result, what you get as a "result" of your optimisation is not a single solution, but a set of non-dominated solutions — called Pareto-fronts.

- The true Pareto-front represents the real trade-offs between the objectives.

- With real-world applications, the true Pareto-front is often unknown.

Pareto Optimality



English: 80
Math: 60

English: 60
Math: 80

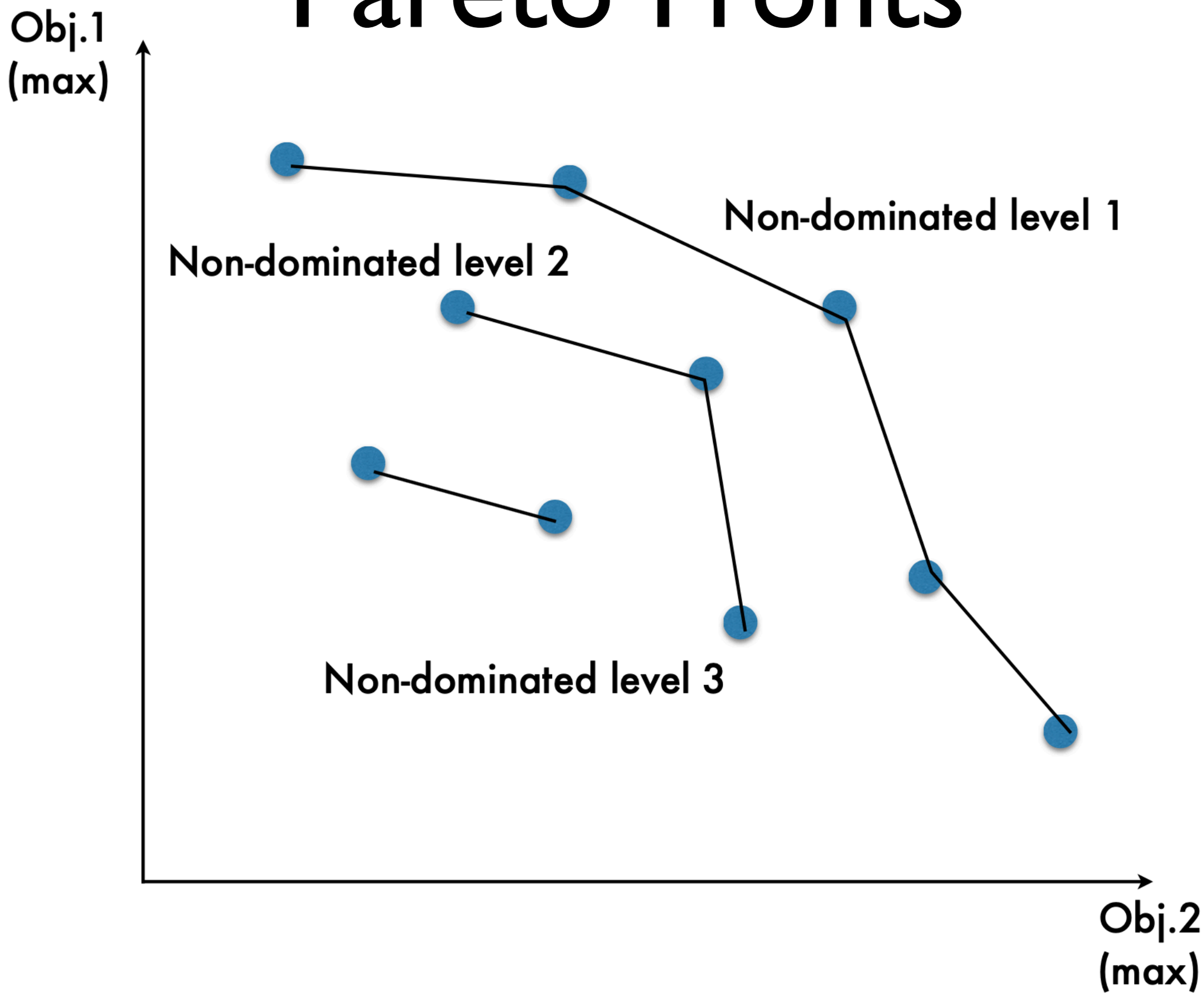
English: 90
Math: 80

English: 80
Math: 90

These two are non-dominating to each other.

These two are non-dominating to each other.

Pareto Fronts



Implications for Optimisation

- Any comparisons (e.g. for selection) should be based on Pareto Optimality

- A good result is one that is:
  - as uniformly distributed as possible
  - as close to the true Pareto-front as possible (generally not measurable)

- Many variations of MOEAs regarding how to achieve convergence and diversity at the same time

NSGA-II (Deb et al., 2002)

- K. Deb, A. Pratap, S. Agarwal, T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182-197. 2002.

- NSGA: "Non-dominated Sorting Genetic Algorithm"

- Non-dominated sorting in $O(MN^2)$ complexity
  - (M: number of objectives, N: population size)

- Crowding distance to encourage diversity

- Elitism (the best front always carries over to the next gen.)

Algorithm

$R_t = P_t \cup Q_t$ combine parent and offspring population
$\mathcal{F} = \texttt{fast-non-dominated-sort}(R_t)$ $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all nondominated fronts of R_t
$P_{t+1} = \emptyset$ and $i = 1$
until $|P_{t+1}| + |\mathcal{F}_i| \leq N$ until the parent population is filled
  $\texttt{crowding-distance-assignment}(\mathcal{F}_i)$ 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 $\prec_n$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$ choose the first $(N - |P_{t+1}|)$ elements of $\mathcal{F}_i$
$Q_{t+1} = \texttt{make-new-pop}(P_{t+1})$ use selection, crossover and mutation to create a new population Q_{t+1}
$t = t + 1$ increment the generation counter

Partial Order in Population

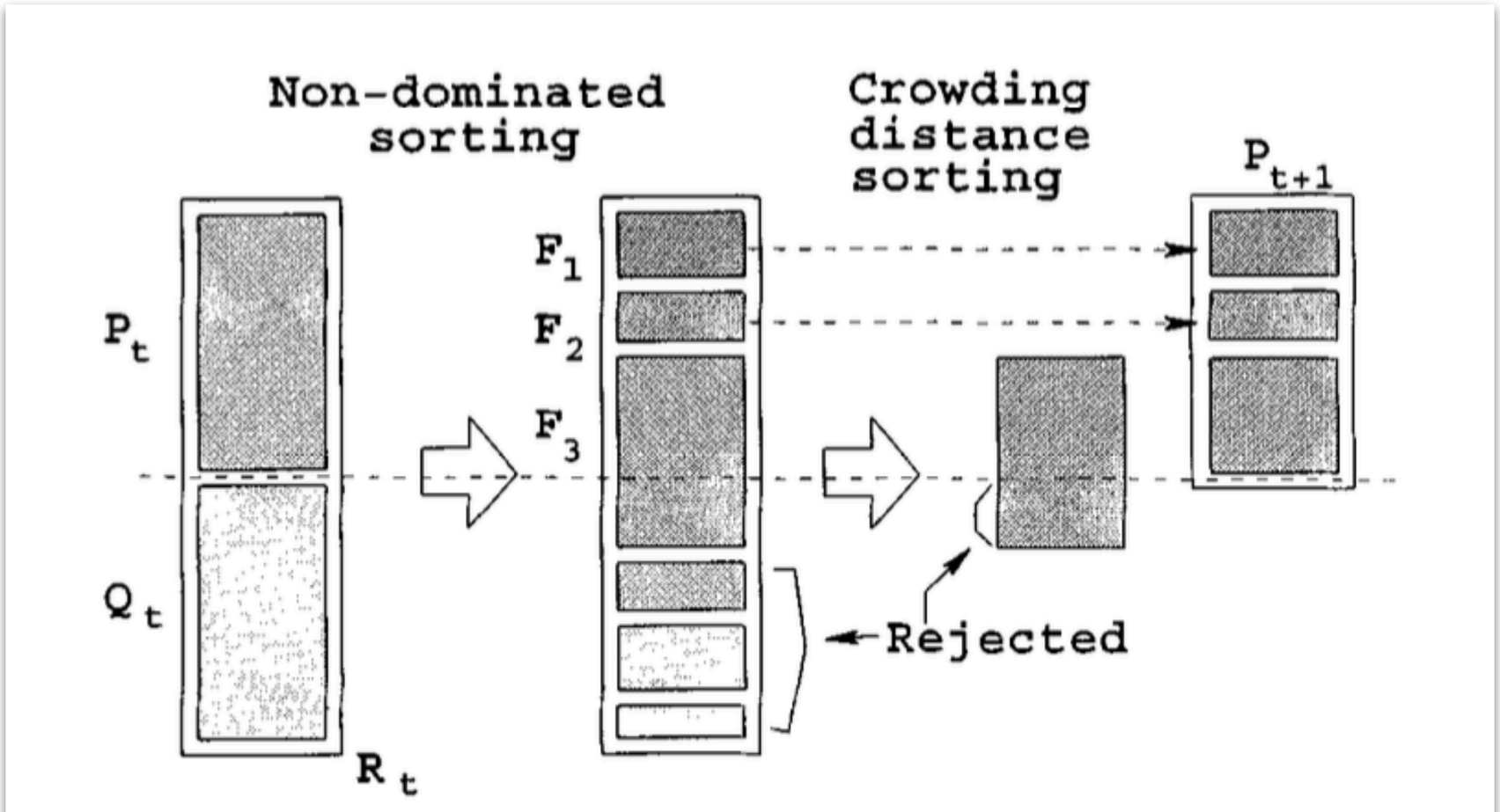
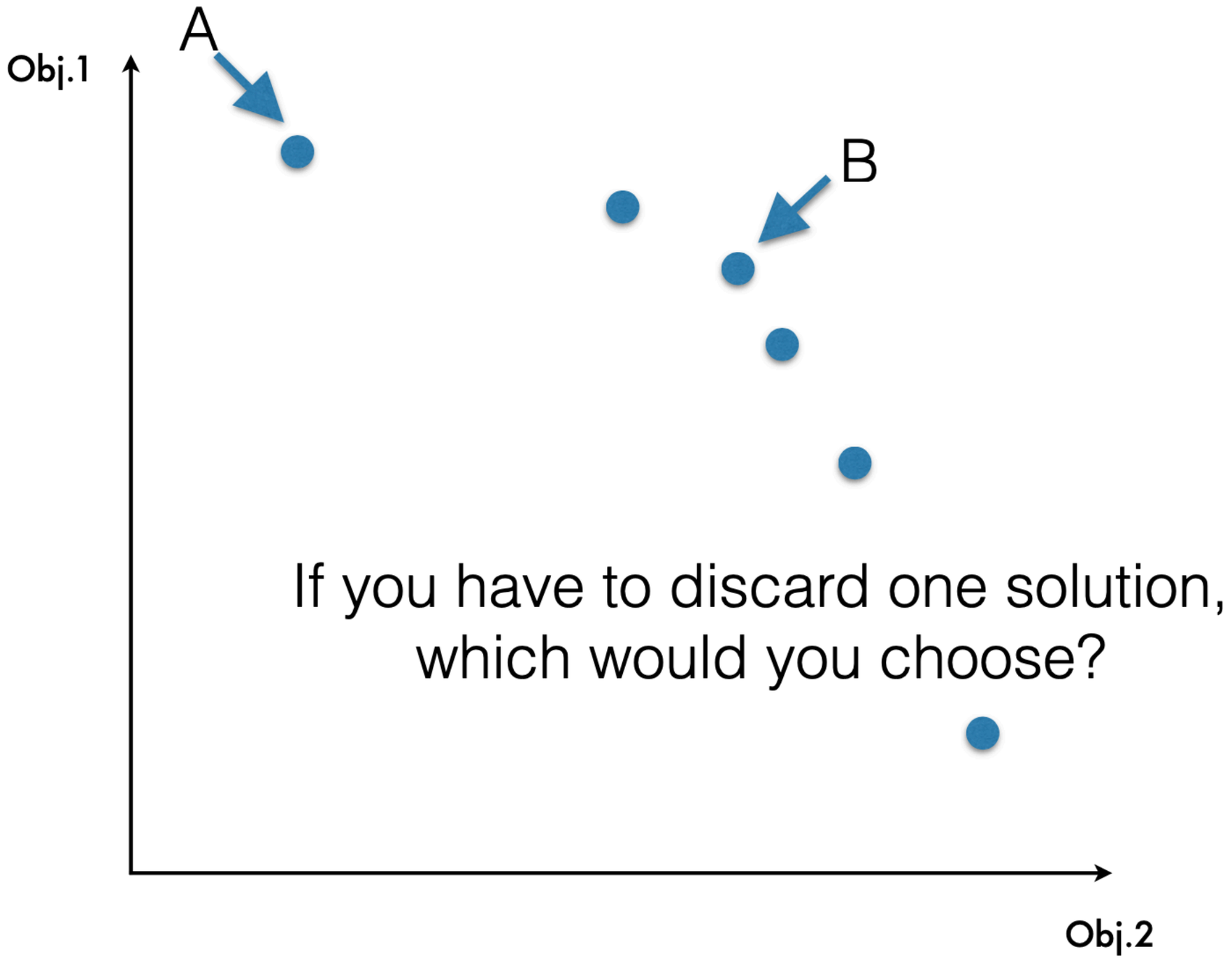


Fig. 2. NSGA-II procedure.

Crowding Distance



Crowding Distance

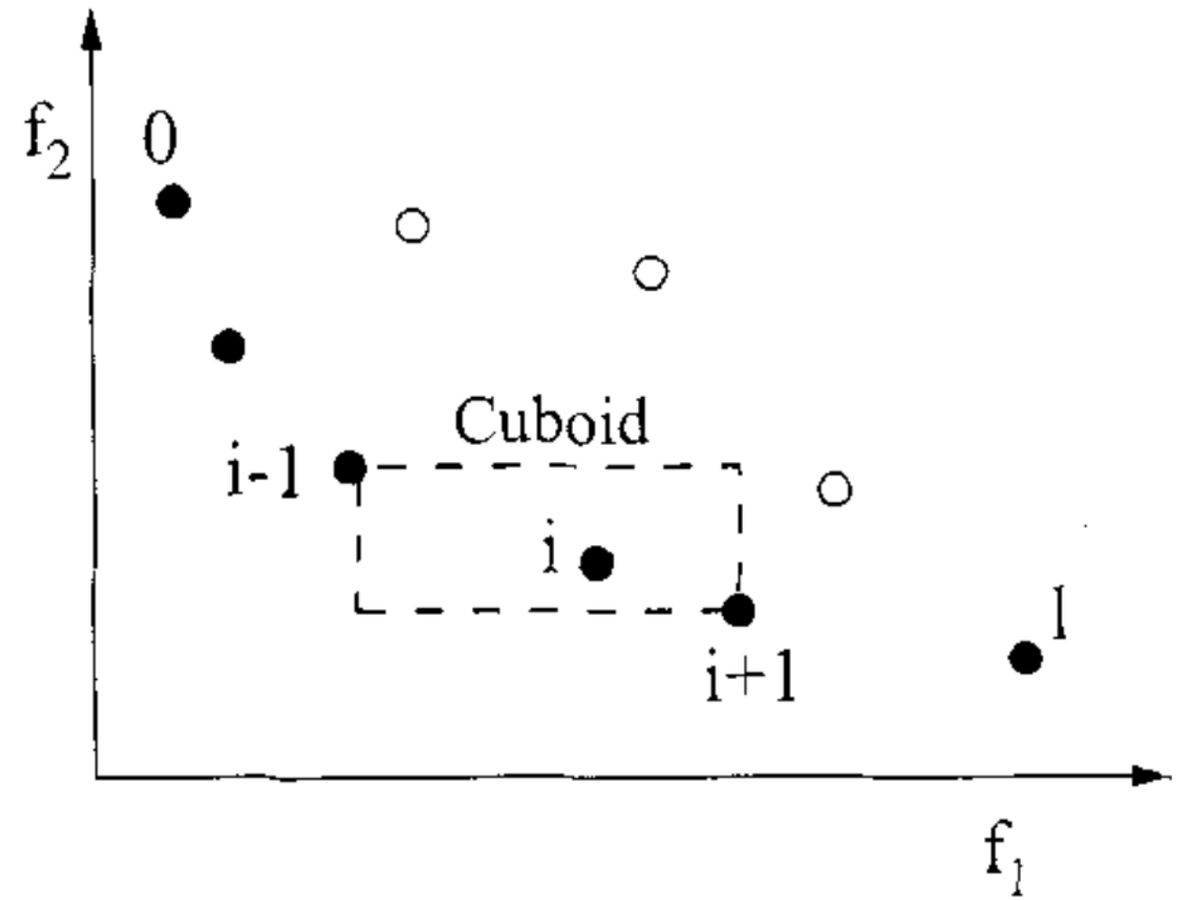


Fig. 1. Crowding-distance calculation. Points marked in filled circles are solutions of the same nondominated front.

Crowding Distance

- A solution that is farther away from the others is rarer and, therefore, more valuable.

- If selecting from a non-dominating pair, crowding distance can help differentiating solutions.

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

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

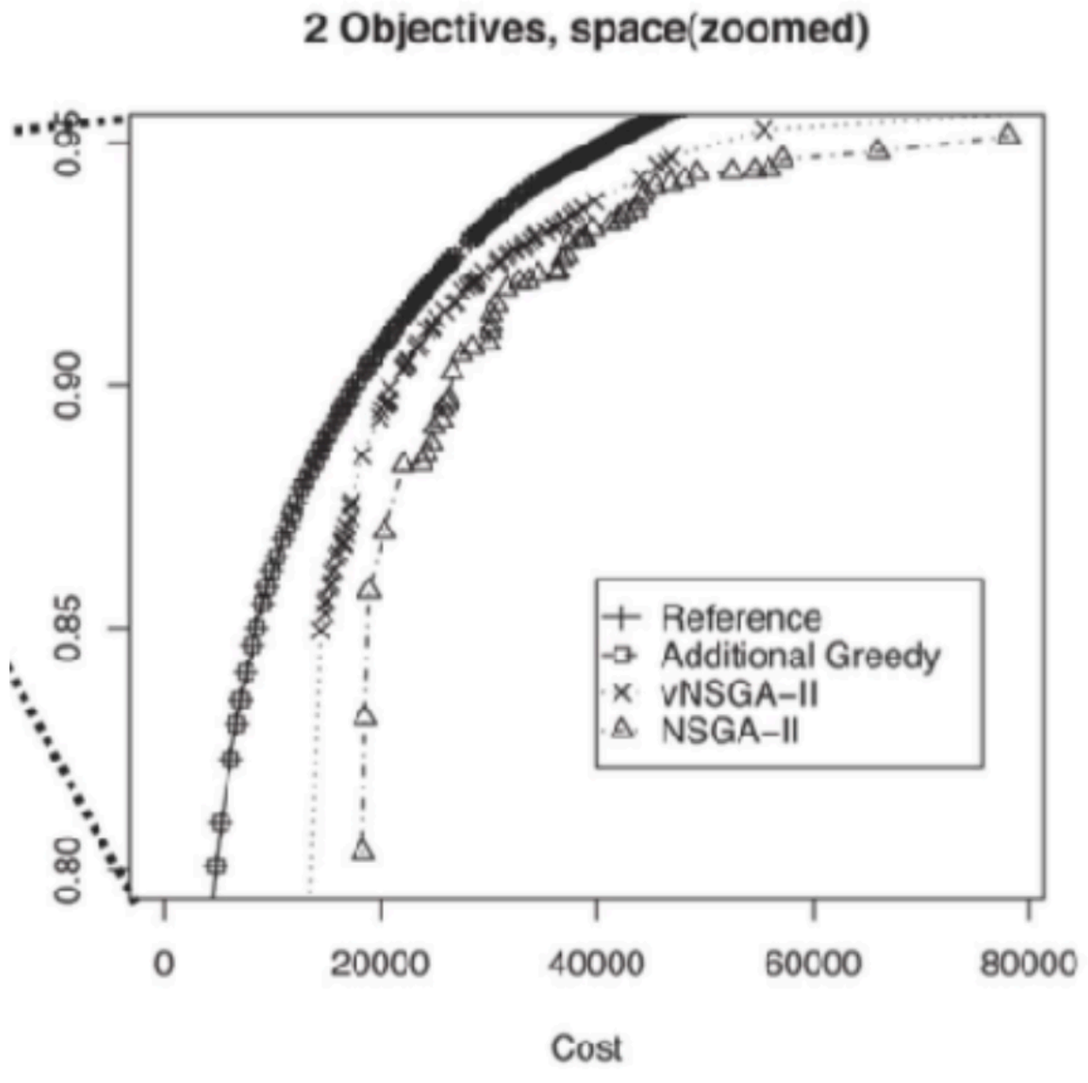
Fast Non-dominated Sort

```
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
```

Comparing Pareto Fronts

- Comparison itself is not as straightforward as comparing two scalar values.

- There is no reference point, as the true Pareto front is usually not known



Empirical Evaluation

- Empirical evaluation of different MOEAs becomes a meta-comparison: it is not only about the domain-specific quality, but it is also about the quality of the front itself.

- Properties that we want to evaluate:

  - Closeness to the true Pareto front

  - Diversity of the solutions on the Pareto front

Closeness to true front

- There are cases where the true Front is known:
  - for example, benchmark optimisation problems.
- For cases where the true front is not known, we can use a "reference front":
  - Collect all known solutions from all MOEAs involved.
  - Extract a single Pareto front from the collected solutions.
- Reference Pareto Front will include solutions contributed by different MOEAs.

Generational Distance (GD) and Inverted Generational Distance (IGD)

- Given a reference front $\Lambda = \{y_1, y_2, \ldots, y_r\}$ (even if approximated) we can calculate the closeness of a solution set V obtained by a MOEA using one of the following two metrics:

- IGD: average distance from each reference point to its closest solution.
  - $IGD(V, \Lambda) = \frac{1}{r}\left(\sum_{i=1}^{r} d(y_i, V)^2\right)^{1/2}$
  - The smaller, the better.
- GD: average distance from each solution to its closest reference point.
  - $GD(V, \Lambda) = IGD(\Lambda, V)$

Weaknesses of GD

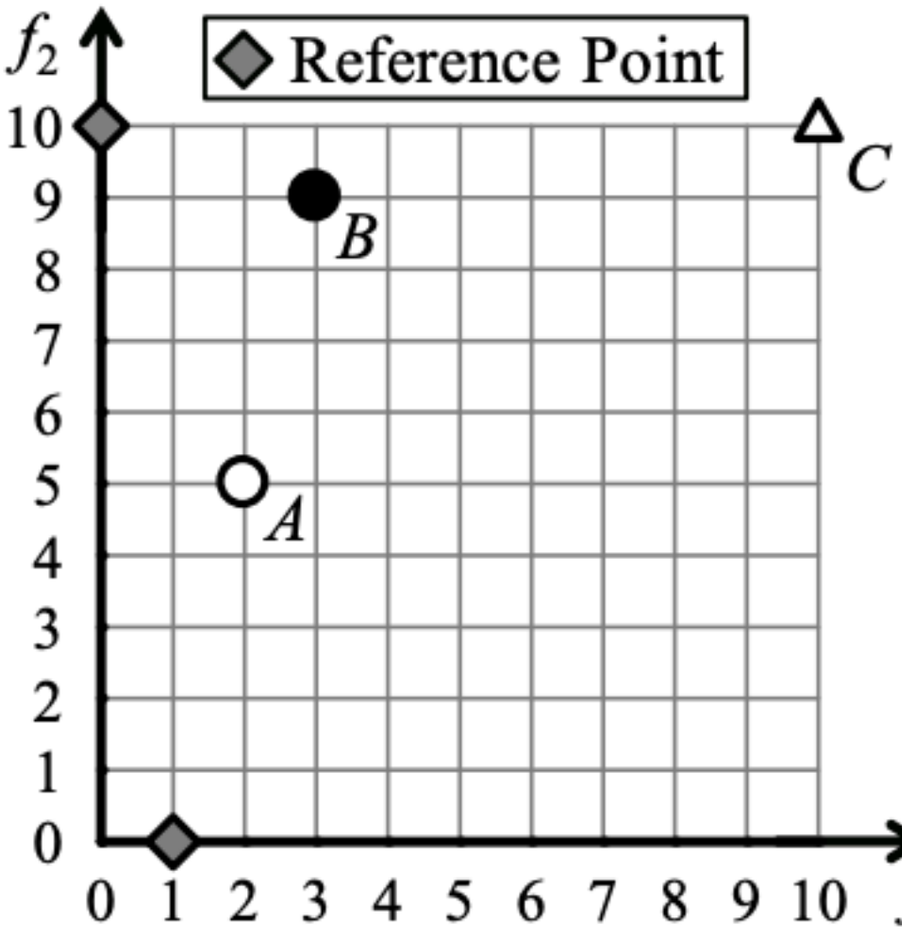


Fig. 1. Example 1 (Zitzler et al. [26]).

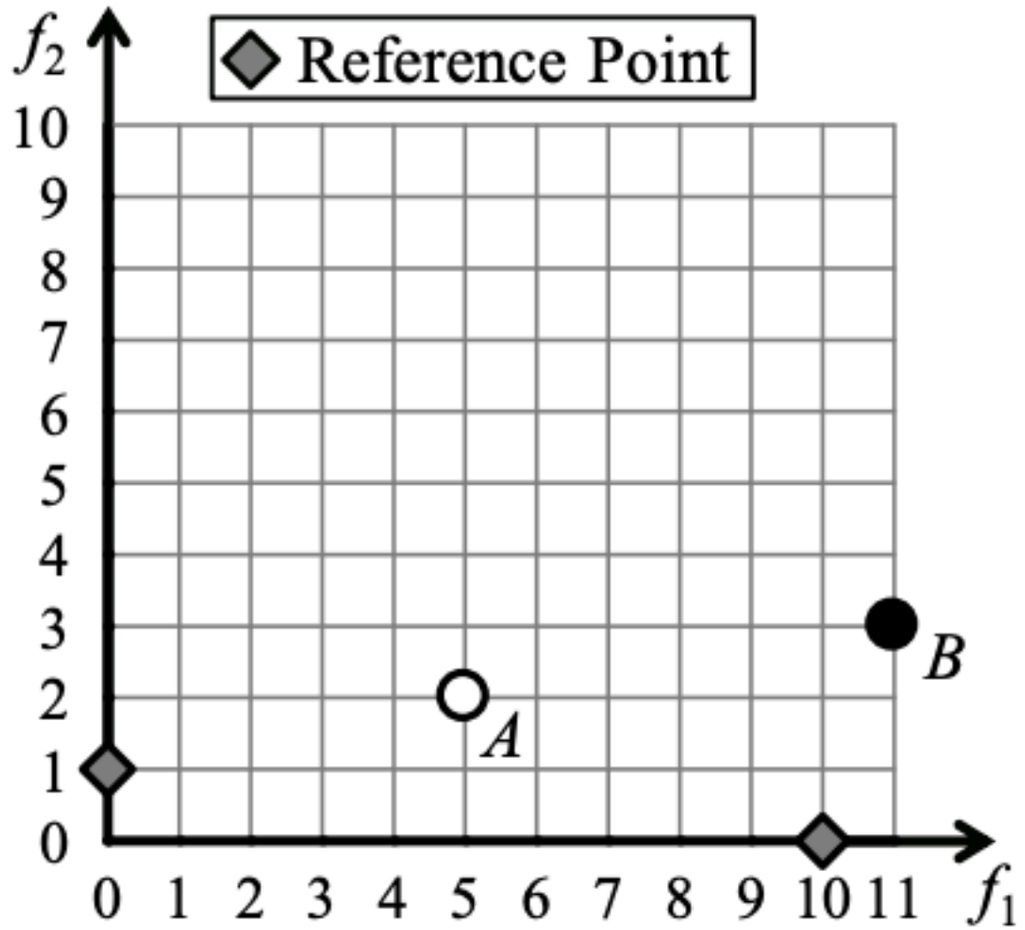


Fig. 2. Example 2 (Schütze et al. [18]).

B has a shorter distance to its closest reference point, but arguably A is a better solution (A dominates B).

Ishibuchi, H., Masuda, H., Tanigaki, Y., & Nojima, Y. (2015, March). Modified distance calculation in generational distance and inverted generational distance. In International conference on evolutionary multi-criterion optimization (pp. 110-125). Springer, Cham.

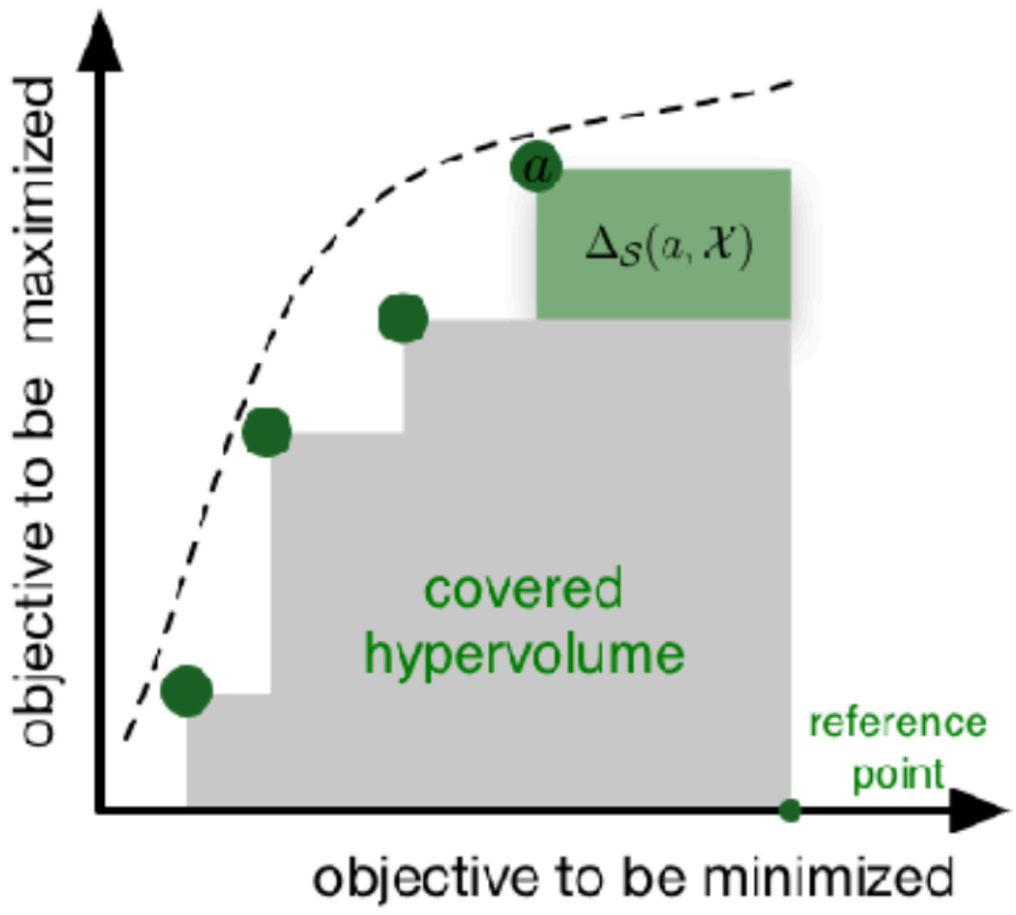
Hypervolume

- Intuitively, hypervolume measures the area (space) dominated by a given Pareto front.

- A unary indicator: does not need a reference front.

- Can be thought to measure both convergence and diversity.

- Requires reference point.



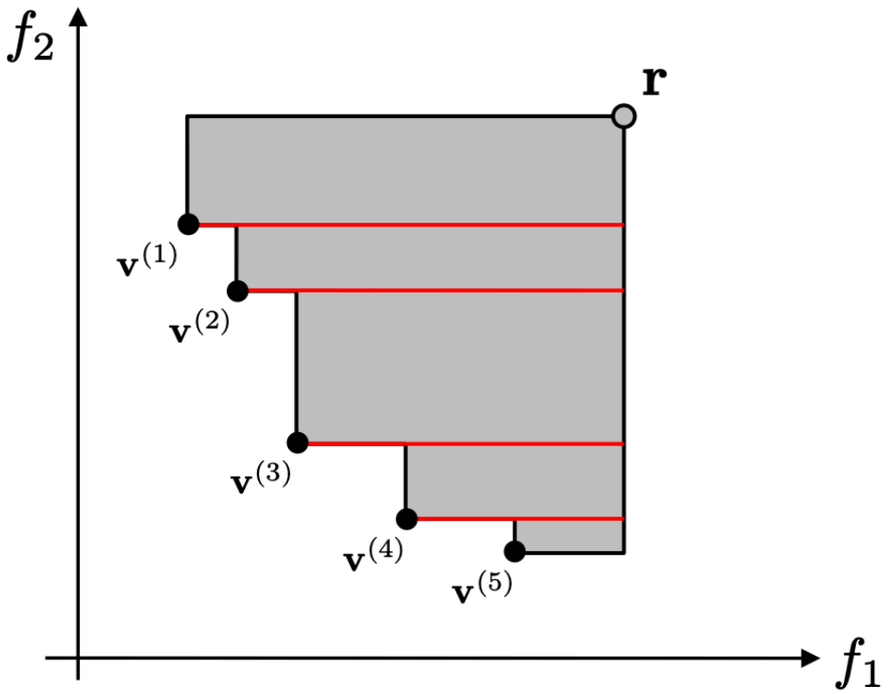
Hypervolume

- Naive Inclusion-exclusion algorithm: Calculates size of union of sets
  - Sums volumes dominated by each point in the set individually
  - Then subtracts volumes dominated by the intersection of each pair of points
  - Then adds back volumes dominated by the intersection of each triple of points
  - And so on, until all subsets of the original set have been accounted for
  - Exponential in the number of points
    - m points, each calculation takes O(n) — O(n*2^m)
  - Many better algorithms have been proposed

Hypervolume

- In 2-dimensional spaces (d = 2), the hypervolume can easily be calculated as the sum of areas of rectangles.

- A rectangle is spanned by a point in M and bounded in one dimension by a neighbouring point, and in the other by the reference point



- Sorting M = {v (1) , . . . , v (m)} ascending regarding the first component, i.e., $v^{(1)}_1 < v^{(2)}_1 < \ldots < v^{(m)}_1$ it also holds $v^{(1)}_2 > v^{(2)}_2 > \ldots > v^{(m)}_2$.

- Then the hypervolume of M with M < r is

$$H(M, \mathbf{r}) = (r_1 - v_1^{(1)})\,(r_2 - v_2^{(1)}) + \sum_{i=2}^{m} (r_1 - v_1^{(i)})\,(v_2^{(i-1)} - v_2^{(i)}).$$

Scaling and Normalisation

- The concept of Pareto optimality itself is independent from scale and normalisation: it is strictly based on partial order only.

- For quality indicators, normalisation may be necessary:

  - so that multiple objectives contribute equally to the indicators

Normalisation of Fitness

- "[with obj.] Mathematics multiply (a series, function or item of data) by a factor that makes the norm or some associated quantity such as an integral equal to a desired value (usually 1)."

- Seems innocuous, but the choice of normalisation method can affect optimisation itself.

Normalisation

- Linear: requires bounds.
- Widely used: $\omega_0(x) = 1 - \alpha^{-x}, \text{ where } \alpha > 1$
- Suggested by Arcuri 2010:

$$\omega_1(x) = \frac{x}{x + \beta}, \text{ where } \beta > 0$$

Requirements

- Should preserve the partial order of the raw values.

$$\omega : \mathbb{R}^+ \to [0, 1]$$

$$x_i < x_j \Leftrightarrow \omega(x_i) < \omega(x_j)$$

$$x_i = x_j \Leftrightarrow \omega(x_i) = \omega(x_j)$$

Direct Impact on Runtime

- The choice of normalisation function has a side-effect on the difference of energy level when deciding whether to accept a sub-optimal solution.

Figure 4. Toy program.

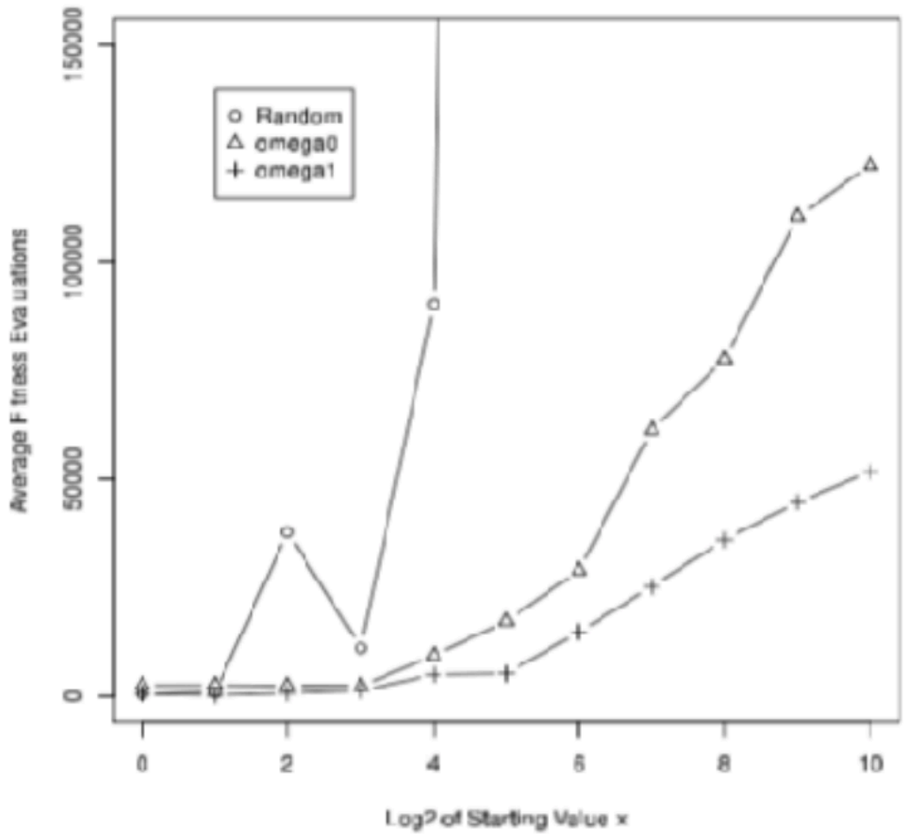


Figure 5. Average number of fitness evaluations to find input $x = 0$.

A. Arcuri. It does matter how you normalise the branch distance in search based software testing. In Software Testing, Verification and Validation (ICST), 2010 Third International Conference on, pages 205–214, April 2010.

PAES (Knowles & Corne 1999)

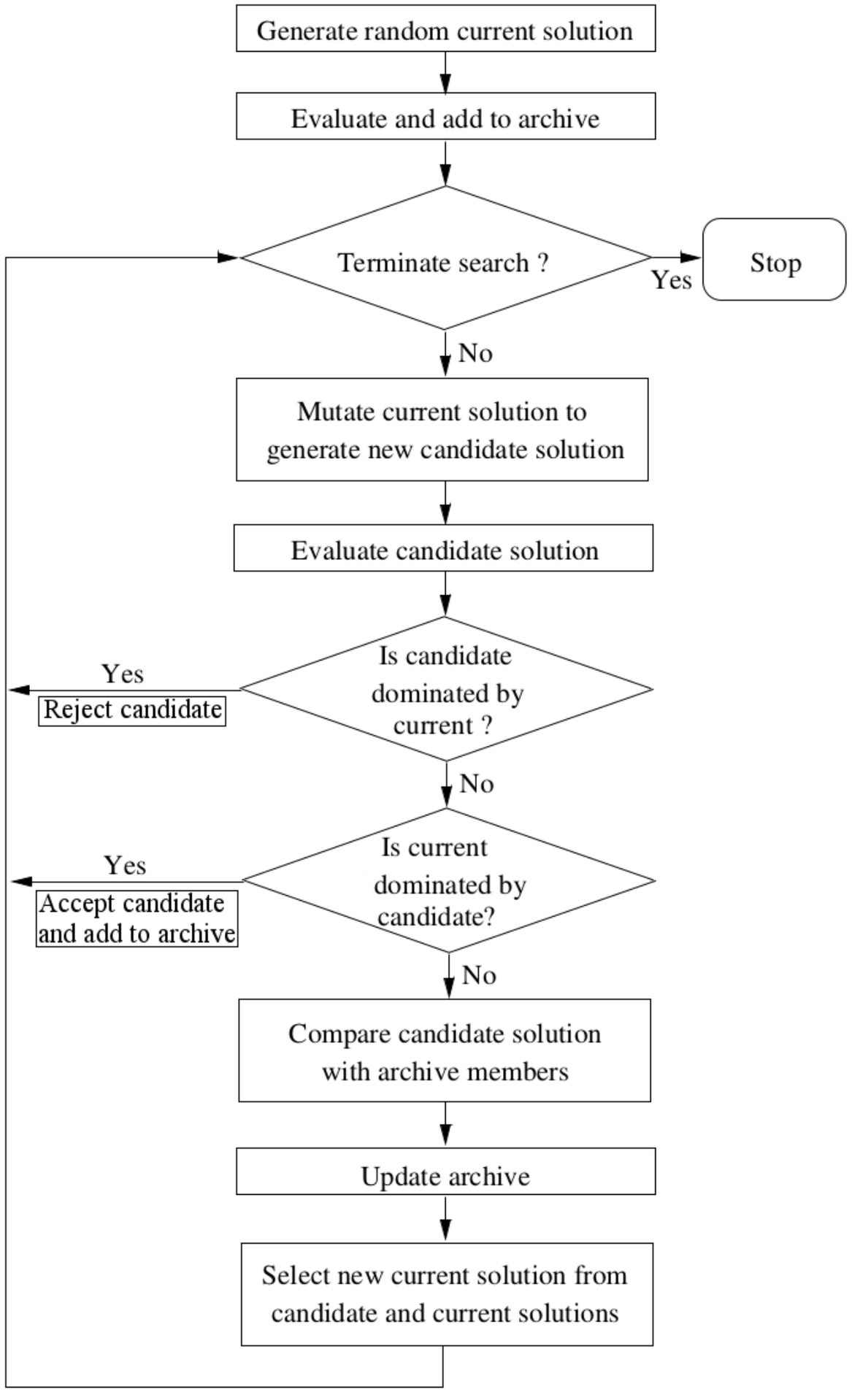
- J. Knowles, D. Corne. "The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation." In Congress on Evolutionary Computation (CEC99) (Vol. 1, pp. 98-105). 1999

- Pareto Archived Evolution Strategy

- Simplest possible non-trivial algorithm capable of generating diverse solutions in the Pareto optimal set.

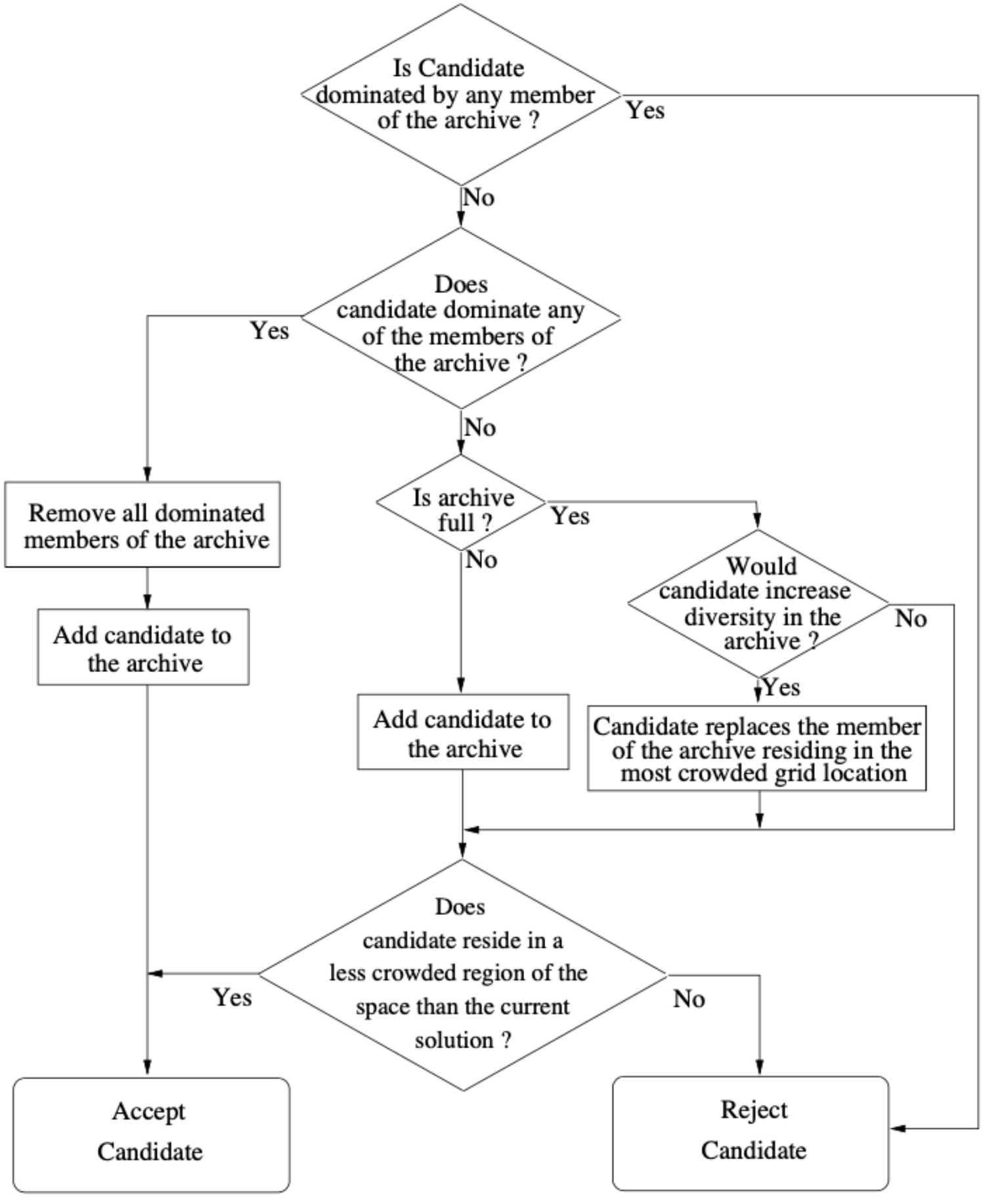
  - (1+1)EA adapted for multi-objective search

- Intended as baseline approach for evaluation


Generate random current solution
Evaluate and add to archive
Terminate search ?
Yes
Stop
No
Mutate current solution to generate new candidate solution
Evaluate candidate solution
Is candidate dominated by current ?
Yes
Reject candidate
No
Is current dominated by candidate?
Yes
Accept candidate and add to archive
No
Compare candidate solution with archive members
Update archive
Select new current solution from candidate and current solutions

PAES (Knowles & Corne 1999)

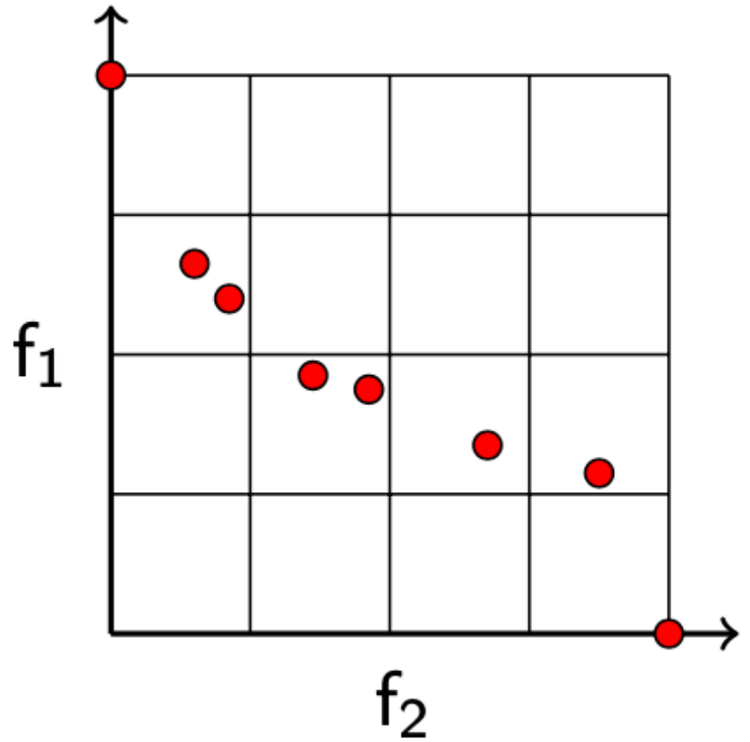
- Archive
    - Stores nondominated solutions (subject to diversity criteria) generated, ready for presentation at the end of a run.
    - Has a maximum size (*refpop*)
        - Set by the user to reflect the required number of final solutions desired.
- Helps to decide acceptance
    - Candidates which dominate the comparison set are always accepted and archived.
    - Candidates which are dominated by the comparison set are always rejected.
    - Non-dominated are accepted/archived based on crowding in their grid location


Is Candidate dominated by any member of the archive ?
Yes
No
Does candidate dominate any of the members of the archive ?
Yes
No
Remove all dominated members of the archive
Add candidate to the archive
Is archive full ?
Yes
No
Would candidate increase diversity in the archive ?
No
Yes
Add candidate to the archive
Candidate replaces the member of the archive residing in the most crowded grid location
Does candidate reside in a less crowded region of the space than the current solution ?
Yes
No
Accept Candidate
Reject Candidate

PAES (Knowles & Corne 1999)

- The archive also keeps track of the solutions' position in the search space

- Archive is split n times in each dimension to determine location density



SPEA2 (Zitzler et al., 2002)

- E. Zitzler, M. Laumanns and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization", in Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems, 2002.

- Assigns fitness of x based on the strength of x's dominators

- SPEA2 minimises: we want solutions that are not dominated

- Density function for niching

SPEA2 (Zitzler et al., 2002)

Algorithm 1 (SPEA2 Main Loop)

Input: N *(population size)*
$\overline{N}$ *(archive size)*
T *(maximum number of generations)*
Output: $\boldsymbol{A}$ *(nondominated set)*

Step 1: ***Initialization***: *Generate an initial population* $\boldsymbol{P}_0$ *and create the empty archive (external set)* $\overline{\boldsymbol{P}}_0 = \emptyset$. *Set* $t = 0$.

Step 2: ***Fitness assignment***: *Calculate fitness values of individuals in* $\boldsymbol{P}_t$ *and* $\overline{\boldsymbol{P}}_t$ *(cf. Section 3.1).*

Step 3: ***Environmental selection***: *Copy all nondominated individuals in* $\boldsymbol{P}_t$ *and* $\overline{\boldsymbol{P}}_t$ *to* $\overline{\boldsymbol{P}}_{t+1}$. *If size of* $\overline{\boldsymbol{P}}_{t+1}$ *exceeds* $\overline{N}$ *then reduce* $\overline{\boldsymbol{P}}_{t+1}$ *by means of the truncation operator, otherwise if size of* $\overline{\boldsymbol{P}}_{t+1}$ *is less than* $\overline{N}$ *then fill* $\overline{\boldsymbol{P}}_{t+1}$ *with dominated individuals in* $\boldsymbol{P}_t$ *and* $\overline{\boldsymbol{P}}_t$ *(cf. Section 3.2).*

Step 4: ***Termination***: *If* $t \geq T$ *or another stopping criterion is satisfied then set* $\boldsymbol{A}$ *to the set of decision vectors represented by the nondominated individuals in* $\overline{\boldsymbol{P}}_{t+1}$. *Stop.*

Step 5: ***Mating selection***: *Perform binary tournament selection with replacement on* $\overline{\boldsymbol{P}}_{t+1}$ *in order to fill the mating pool.*

Step 6: ***Variation***: *Apply recombination and mutation operators to the mating pool and set* $\boldsymbol{P}_{t+1}$ *to the resulting population. Increment generation counter* ($t = t + 1$) *and go to Step 2.*

SPEA2 (Zitzler et al., 2002)

Population *Archive*

Score: $S(\boldsymbol{i}) = |\{\boldsymbol{j} \mid \boldsymbol{j} \in \boldsymbol{P}_t + \overline{\boldsymbol{P}}_t \wedge \boldsymbol{i} \succ \boldsymbol{j}\}|$ → *# sols. that i dominates*

Raw Fitness: $R(\boldsymbol{i}) = \sum_{\boldsymbol{j} \in \boldsymbol{P}_t + \boldsymbol{P}_t, \boldsymbol{j} \succ \boldsymbol{i}} S(\boldsymbol{j})$ → *sum of dominators' strength*

Density: $D(\boldsymbol{i}) = \frac{1}{\sigma_{\boldsymbol{i}}^k + 2} \quad (k = \sqrt{N + \bar{N}})$

$\sigma_i^k = \text{distance to nearest } k\text{-th neighbour}$

Fitness: $F(i) = R(i) + D(i)$

*Note that Deb et al. and Zitzler et al. are using the precedence symbol in two different ways

SPEA2 (Zitzler et al., 2002)

Updating Archive

$$\overline{\boldsymbol{P}}_{t+1} = \{\boldsymbol{i} \mid \boldsymbol{i} \in \boldsymbol{P}_t + \overline{\boldsymbol{P}}_t \wedge F(\boldsymbol{i}) < 1\}$$

(means R(i) = 0, i.e., no dominators)

$$\begin{cases} \text{add } \bar{N} - |\bar{P}_{t+1}| \text{ solutions from } P_t & \text{if } |\bar{P}_{t+1}| < \bar{N} \\ \text{truncate } i \text{ such that } \forall j \in \bar{P}_{t+1}, i \leq_d j & \text{if } |\bar{P}_{t+1}| > \bar{N} \end{cases}$$

Truncating Archive

$$\begin{aligned} \boldsymbol{i} \leq_d \boldsymbol{j} \quad :\Leftrightarrow \quad & \forall\, 0 < k < |\overline{\boldsymbol{P}}_{t+1}| \;:\; \sigma_{\boldsymbol{i}}^k = \sigma_{\boldsymbol{j}}^k \quad \vee \\ & \exists\, 0 < k < |\overline{\boldsymbol{P}}_{t+1}| \;:\; \left[\left(\forall\, 0 < l < k \;:\; \sigma_{\boldsymbol{i}}^l = \sigma_{\boldsymbol{j}}^l\right) \wedge\; \sigma_{\boldsymbol{i}}^k < \sigma_{\boldsymbol{j}}^k\right] \end{aligned}$$

SPEA2 (Zitzler et al., 2002)

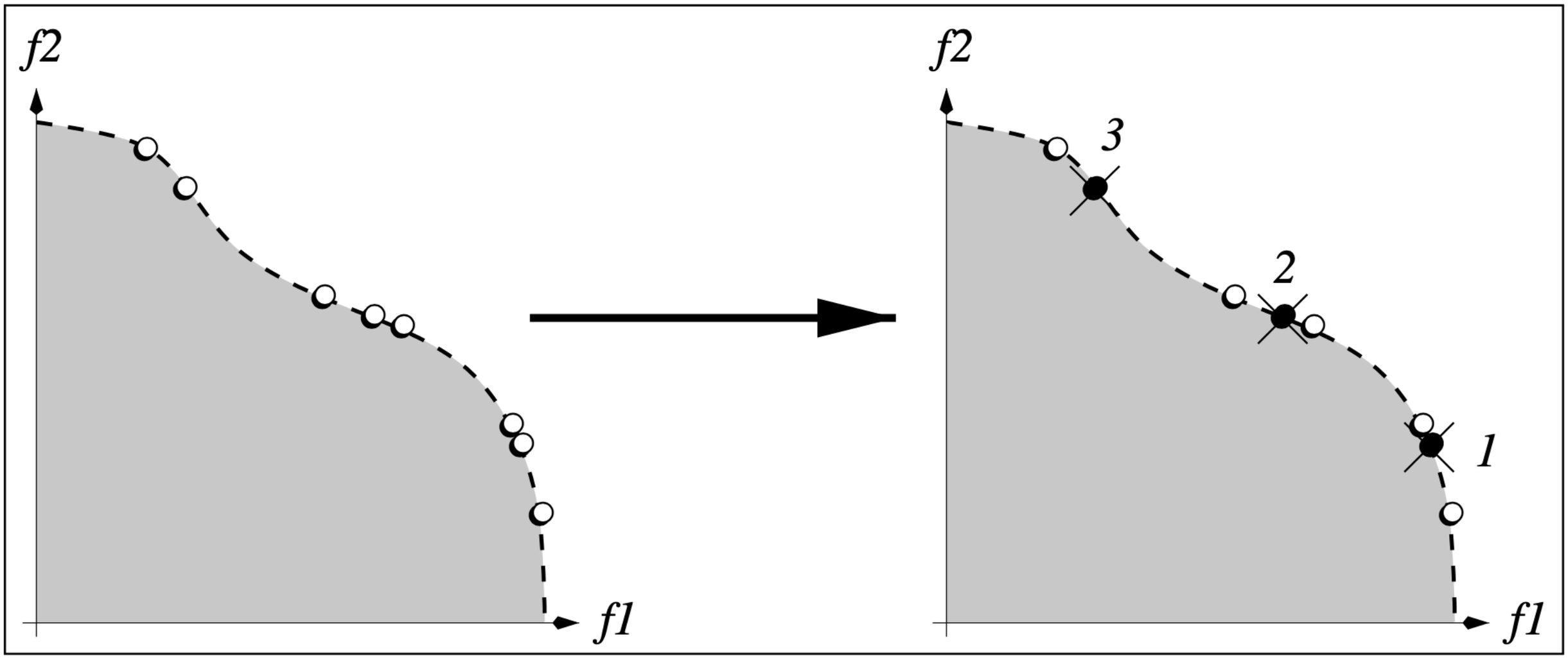


Figure 2: Illustration of the archive truncation method used in SPEA2. On the right, a nondominated set is shown. On the left, it is depicted which solutions are removed in which order by the truncate operator (assuming that $\overline{N} = 5$).

Two Archive (Praditwong & Yao, 2006)

- Maintains two different archives: one for convergence, the other for diversity.

  - If a new solution is not dominated by both archives and dominates at least one solution in either archive, it goes into the *convergence archive*.

  - If a new solution is not dominated by both archives but fails to dominate any solution in either archive, it goes into the *diversity archive*.

  - When archive gets full, convergence archive is preserved while diversity archive gets pruned based on crowding distance.

Two Archive (Praditwong & Yao, 2006)

Algorithm 1 The Two-Archive Algorithm

```
1:  Initialise the population
2:  Initialise archives to the empty set
3:  Evaluate initial population
4:  Set t=0
5:  repeat
6:      Collect non-dominated individuals to archives
7:      Select parents from archives
8:      Apply genetic operators to generate a new population
9:      Evaluate the new population
10:     t = t + 1
11: until t == MAX_GENERATION
```

Two Archive (Praditwong & Yao, 2006)

- Selection:
  - An archive is chosen with a probability.
    - The probability is a pre-defined parameter that is a ratio to choose members from the convergence archive to the diversity archive.
- A member in the chosen archive should be selected uniformly at random.
- Finally, the chosen parent goes to the mating population.

Two Archive (Praditwong & Yao, 2006)

- Archive update:
  - For each individual of the population:
    - If it is non-dominated, it goes to the next step. Otherwise, it is discarded
  - The non-dominated solution from the population is compared with all members in the current archives.
    - If it is dominated by a member of the archives, it is discarded.
    - Otherwise it is a new member of the archives
- The remainder of the archives are compared with the new member and two cases are possible:
  - If the new member dominates a member of the archives: The dominated member is removed and the new member is received by the *convergence* archive
  - If the new member does not dominate any members and is not dominated by any archive member: The new member becomes a member of the diversity archive, and the size of the *diversity* archive is increased.

Two Archive (Praditwong & Yao, 2006)

- If the total size of the archives overflows, the removal operation should be performed:
  - The removal operator deletes only members in the diversity archive. This operator has no impact on the convergence archive.
- All members in the diversity archive calculate the shortest distance to the nearest member in the convergence archive.
  In other words, each member of the diversity archive calculates its Euclidean distance to all members of the convergence archive.
- The member with the shortest distance among the diversity members is deleted until the total size equals the capacity.

SMS-MOEA

- Beume, N., Naujoks, B., & Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research, 181(3), 1653-1669.