Multi-Objective Optimisation

CS454 AI-Based Software Engineering

Shin Yoo

More Than One Objectives

- If you have more than one objective, what would you do with your GA?
  - “I want to maximise travel distance of my EV but minimise power consumption” - ratio may work (distance per kW)
  - “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” - ???

Classical Methods

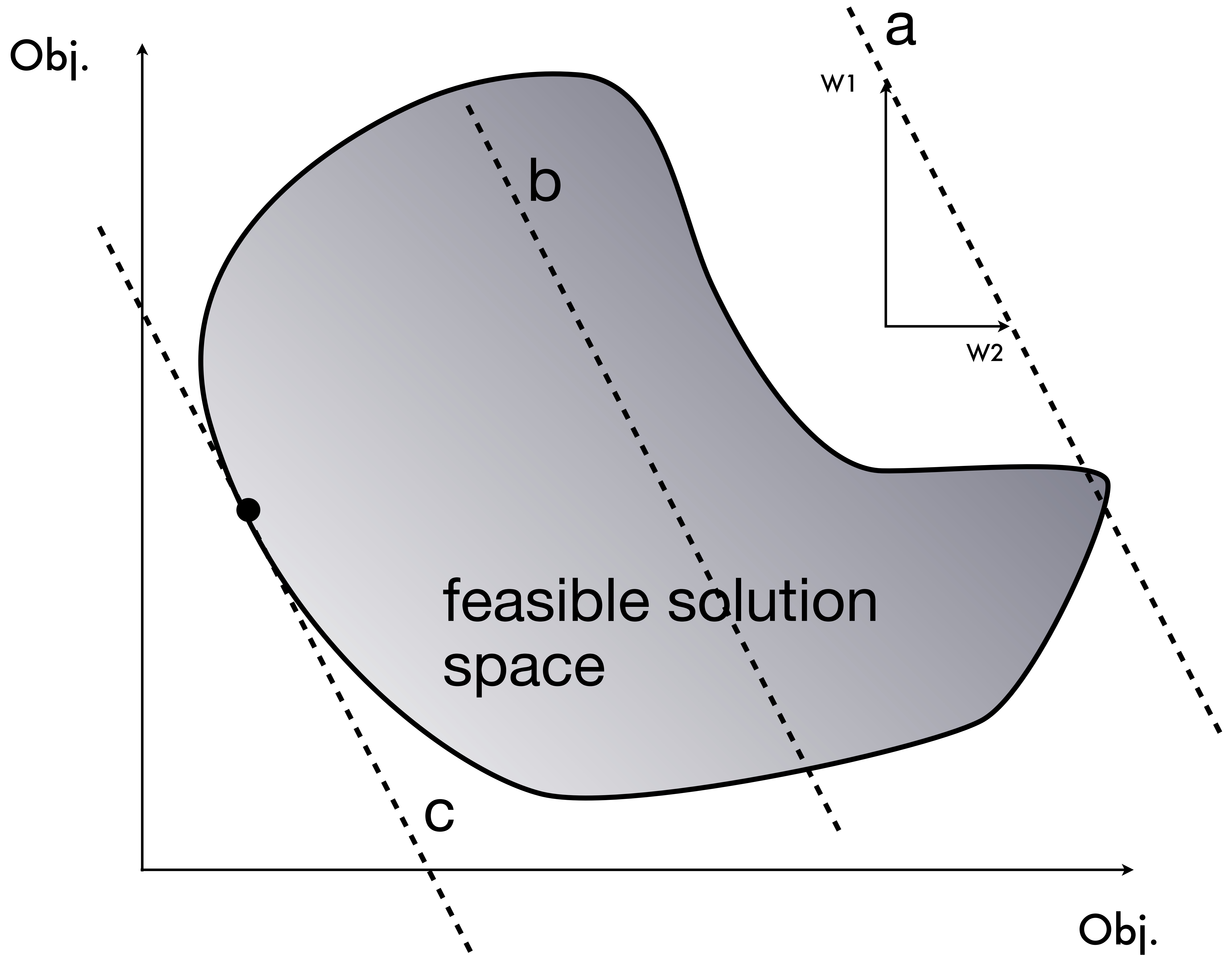
- Weighted Sum
- ε-Constraint Method
- ... and other assorted methods
- Each of these assumes we have a single objective solver (i.e. optimisation algorithm): either a precise algorithm (e.g. linear programming) or a single objective GA

Weighted-Sum Approach

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

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

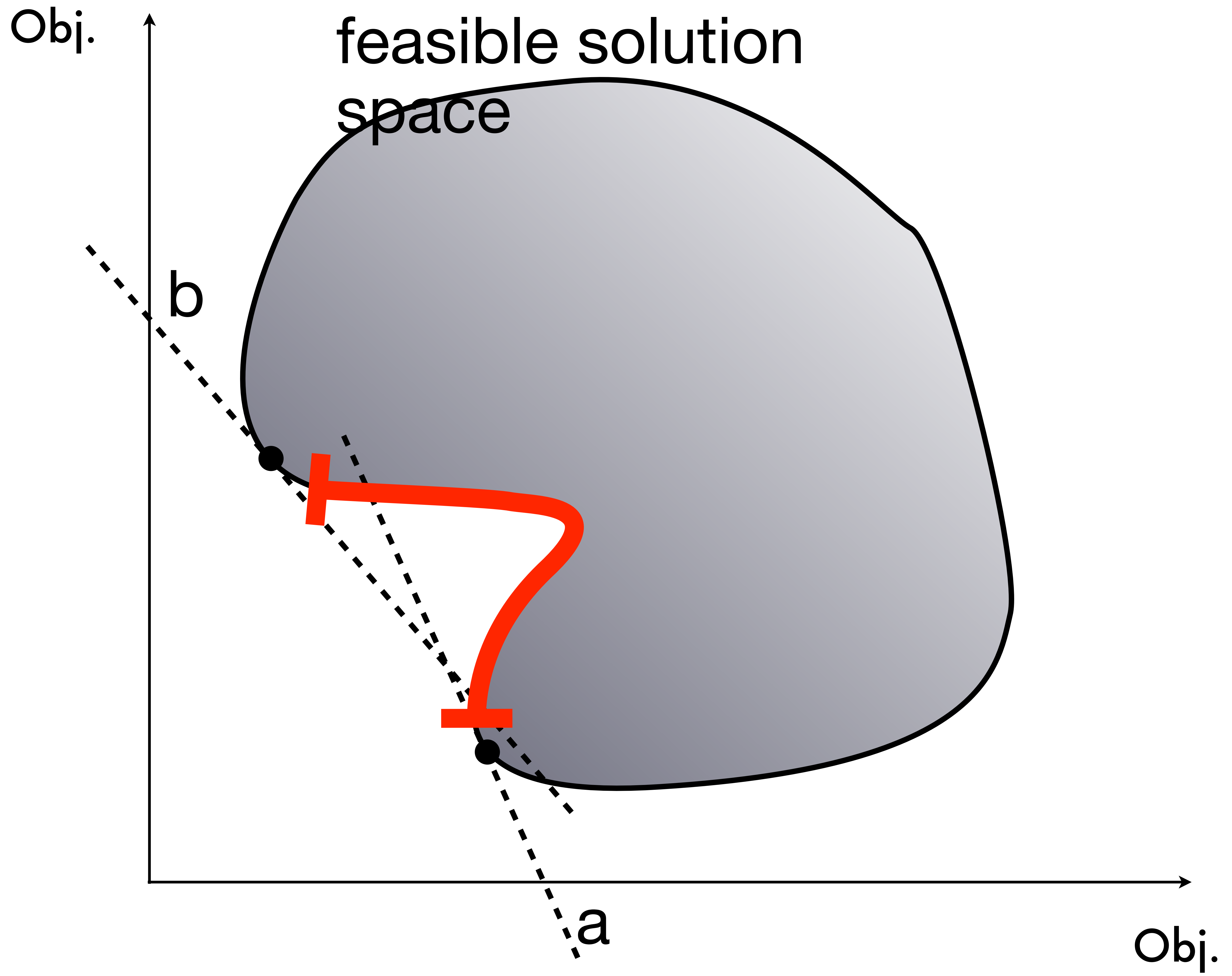
Weighted-Sum Approach



Weighted-Sum Approach

- Disadvantages:
  - 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

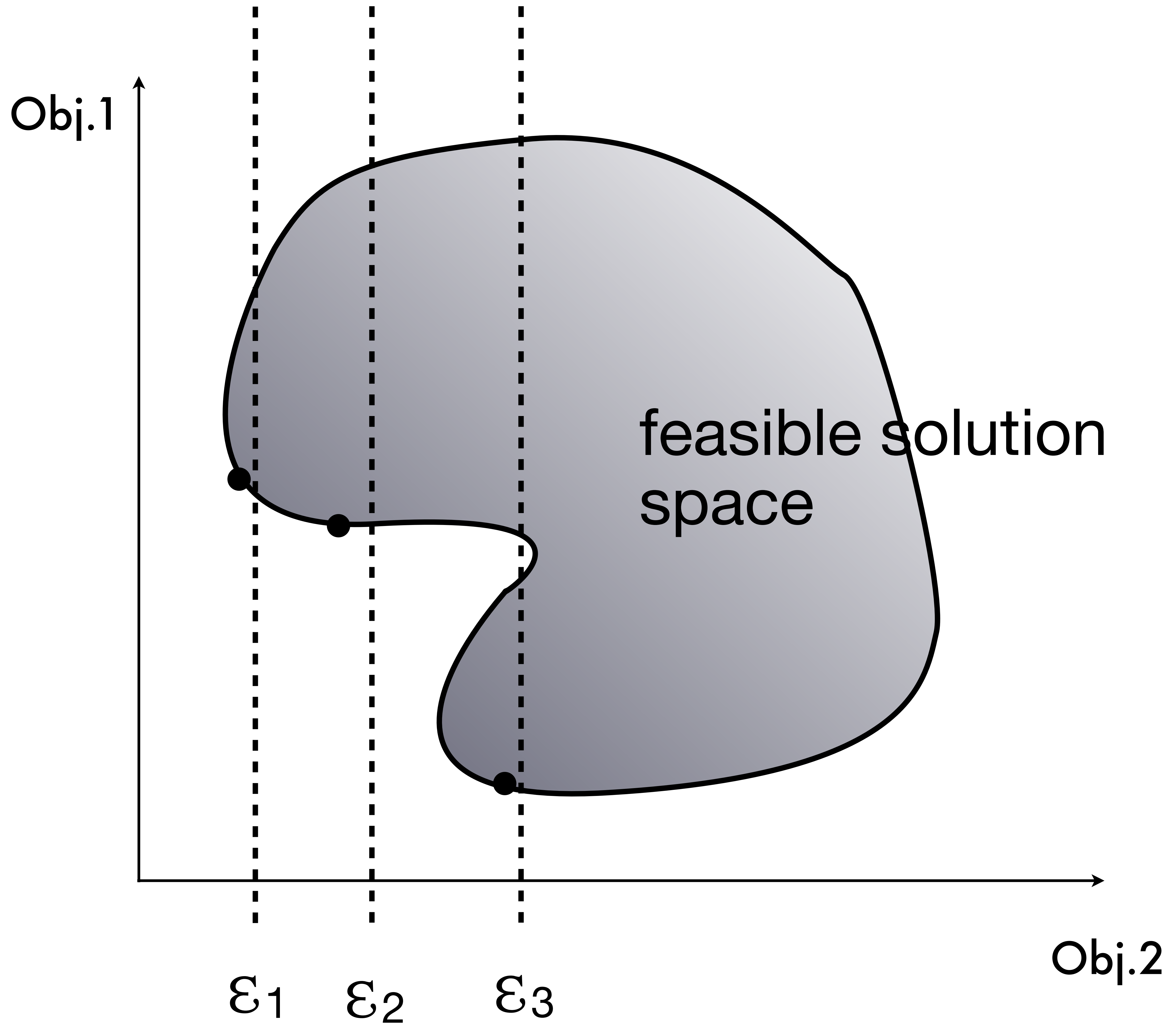


ε-Constraint Method

- Focus only on one objective; turn others into user-defined constraints:

$$\begin{array}{ll} \text{minimise} & f_i(x) \\ \text{subject to} & [f_1(x), \ldots, f_{i-1}(x), f_{i+1}(x), f_M(x)] \leq \epsilon \end{array}$$

ε-Constraint Method



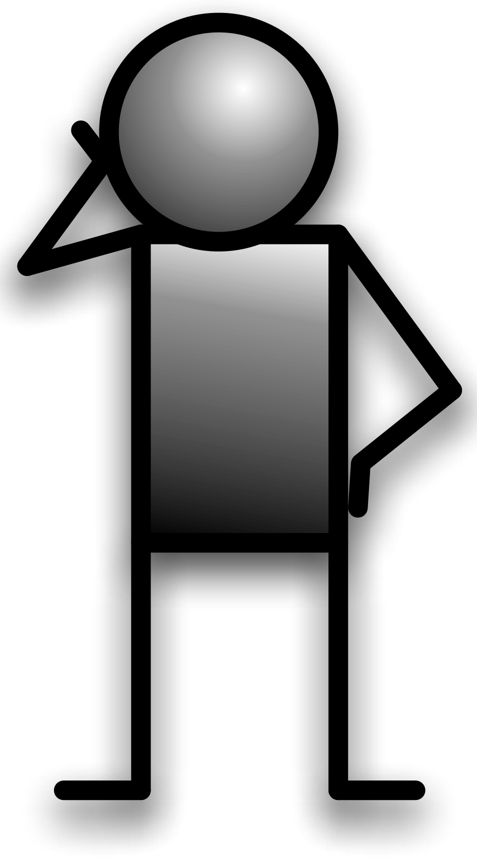
ε-Constraint Method

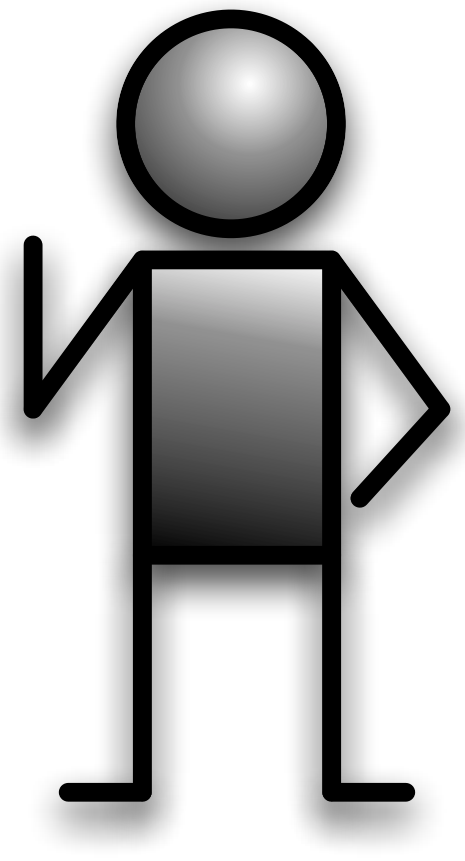
- Advantages:
  - Copes with non-convex solution space
- Disadvantages:
  - ε vector largely decides which part of Pareto front is obtained
  - The higher the dimension, the more input the human needs to provide (ε vector size increases)

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





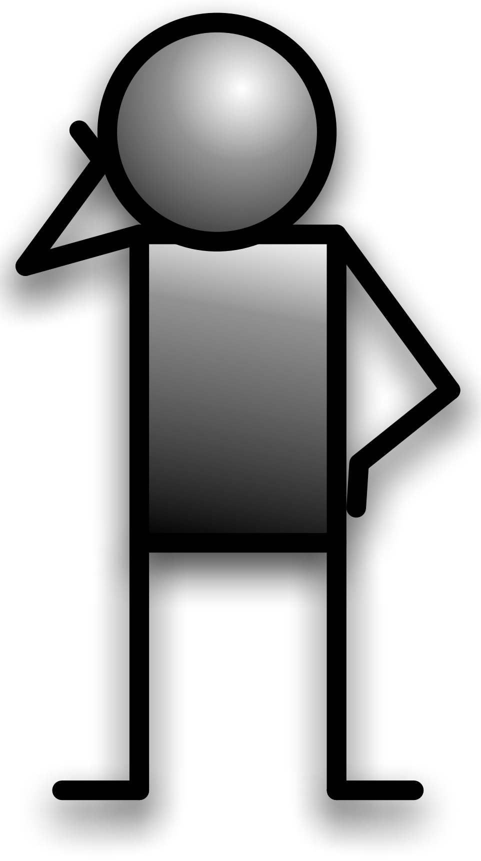
English: 80
Math: 60

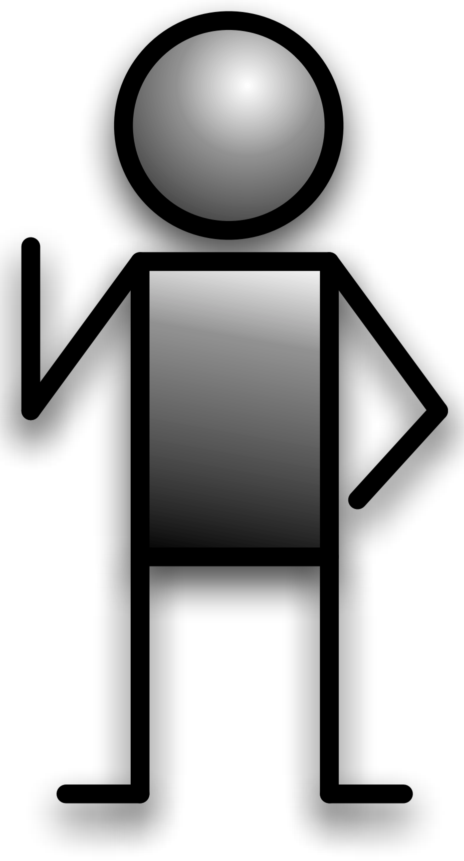
English: 70
Math: 90

Who is the better student?

Pareto Optimality

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





English: 80
Math: 60

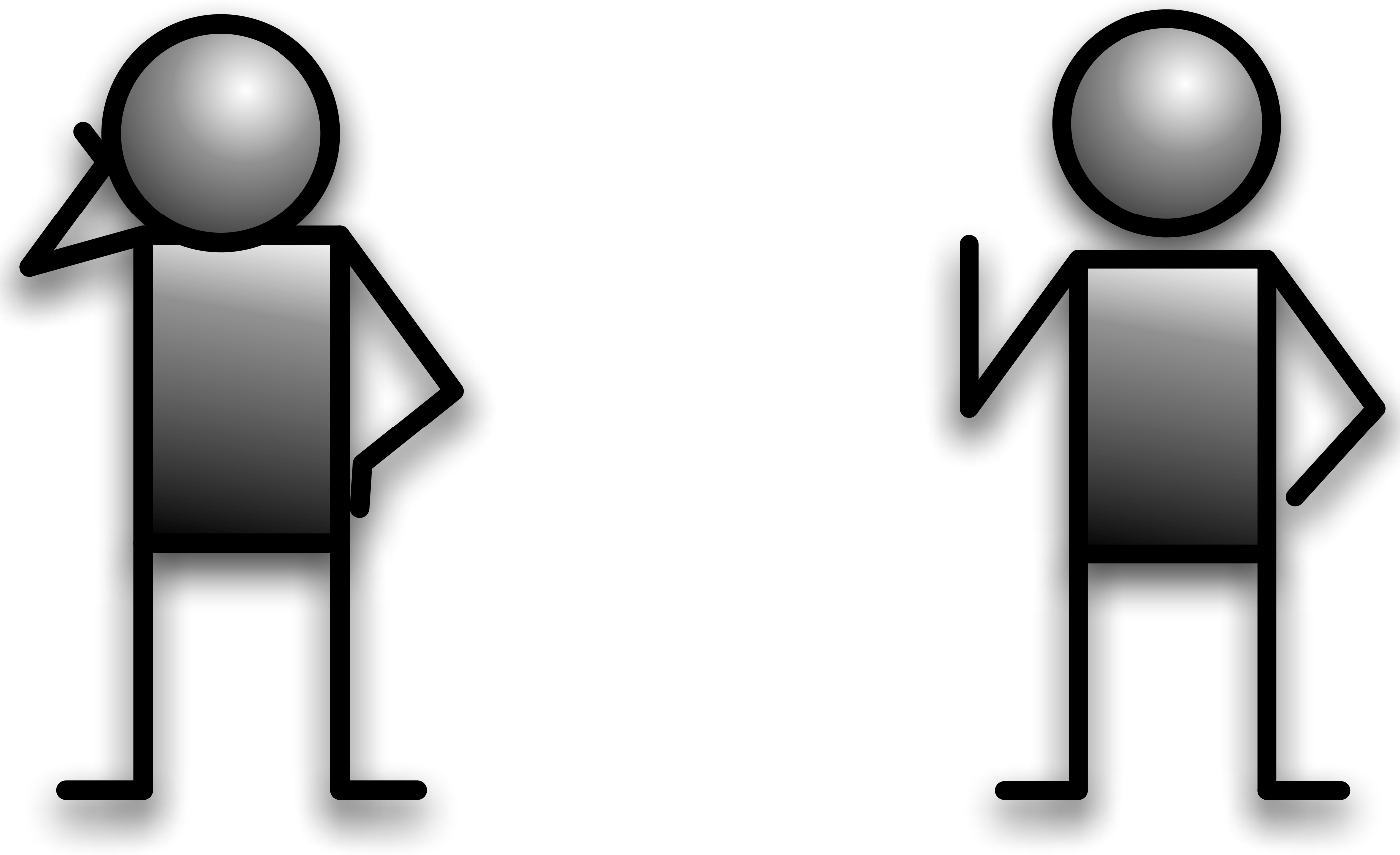
English: 70
Math: 90

Pareto Optimality

Vilfred F. D. Pareto (1848~1923)
Economist, Engineer & Philosopher

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

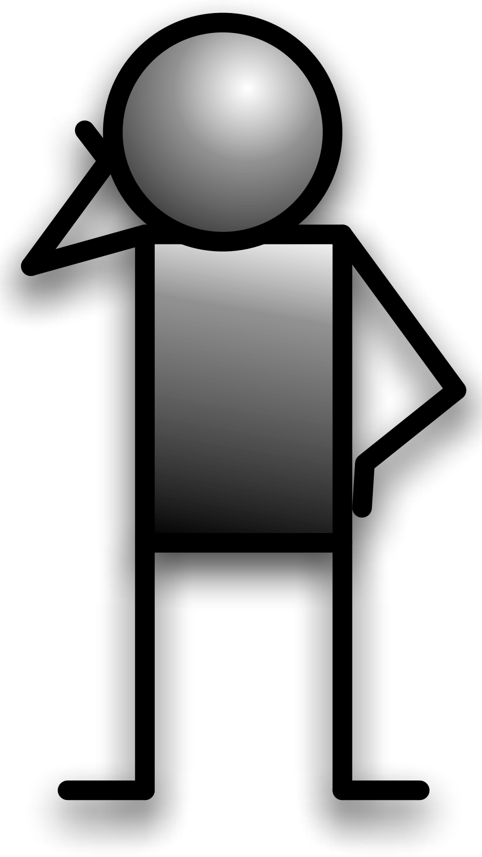


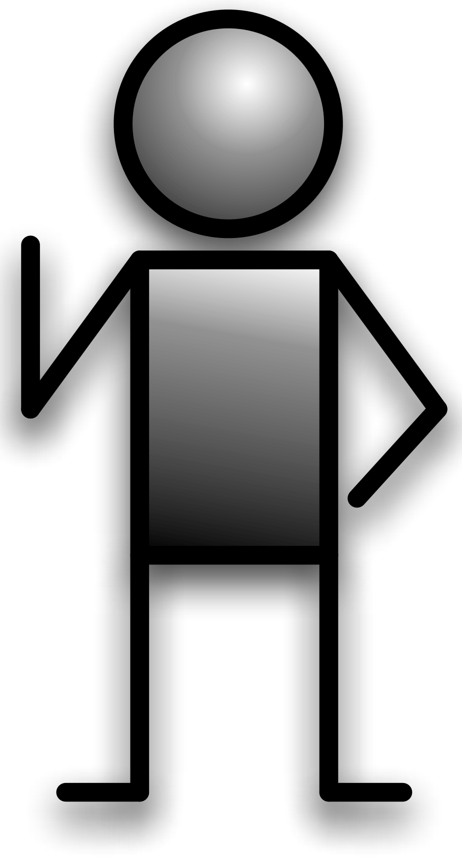
English: 80
Math: 60

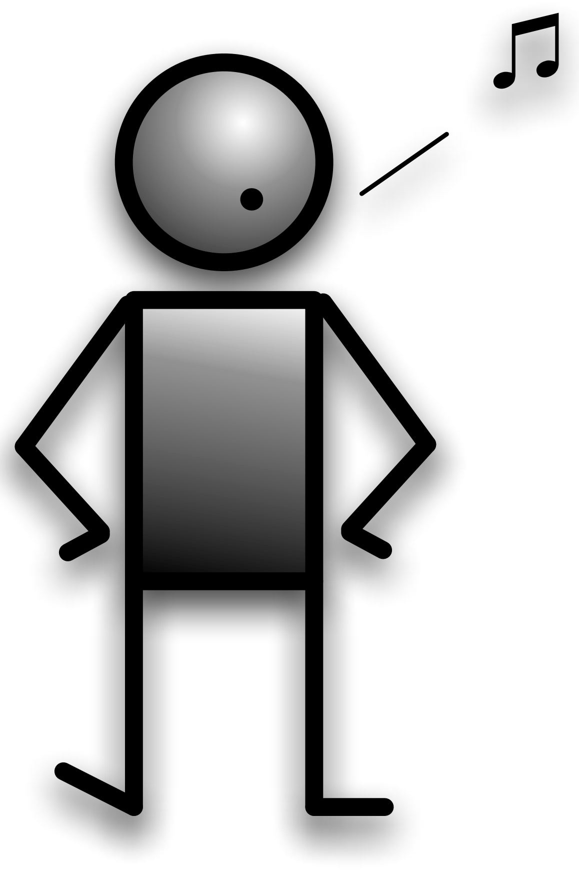
English: 70
Math: 90

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

English: 80
Math: 90

These two are non-dominating to each other.

This person dominates the other two.

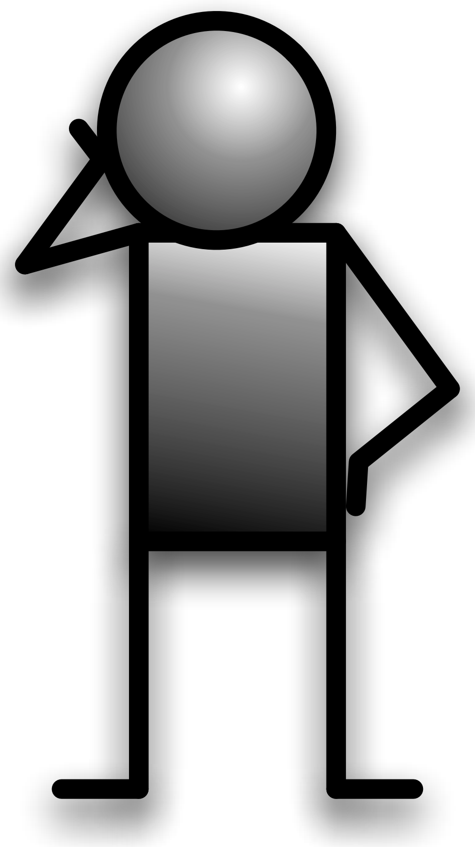
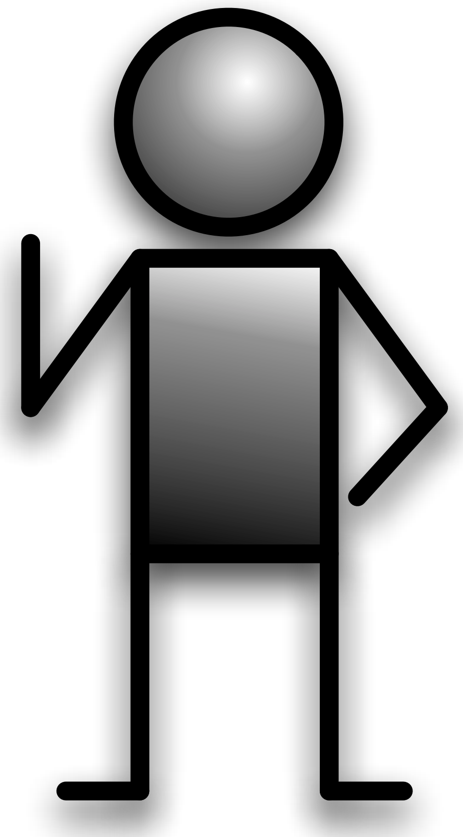
How do we optimise with this?

- With single-objective optimisation, we sort the population and pick the best solution.
- Now we may not have a single best solution, because there many be a non-dominating pair of solutions (or more!).

Pareto-fronts

- 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.

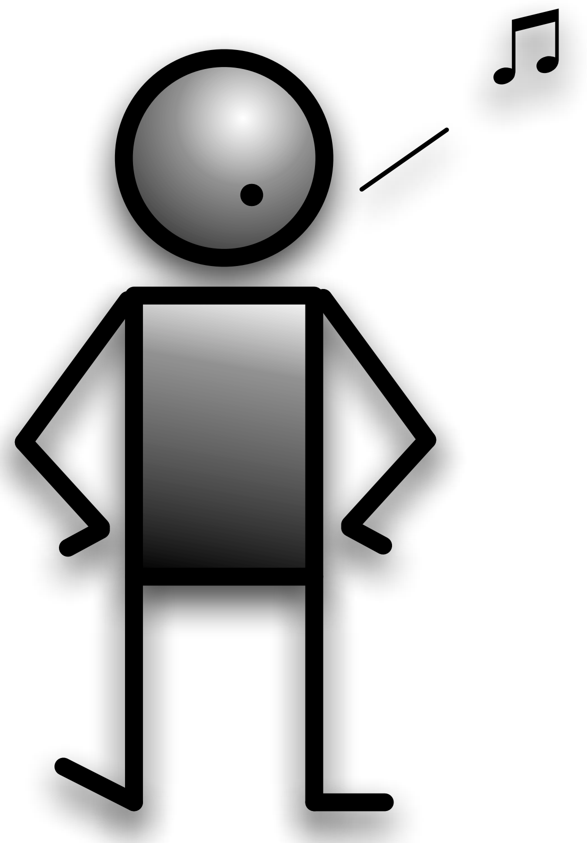
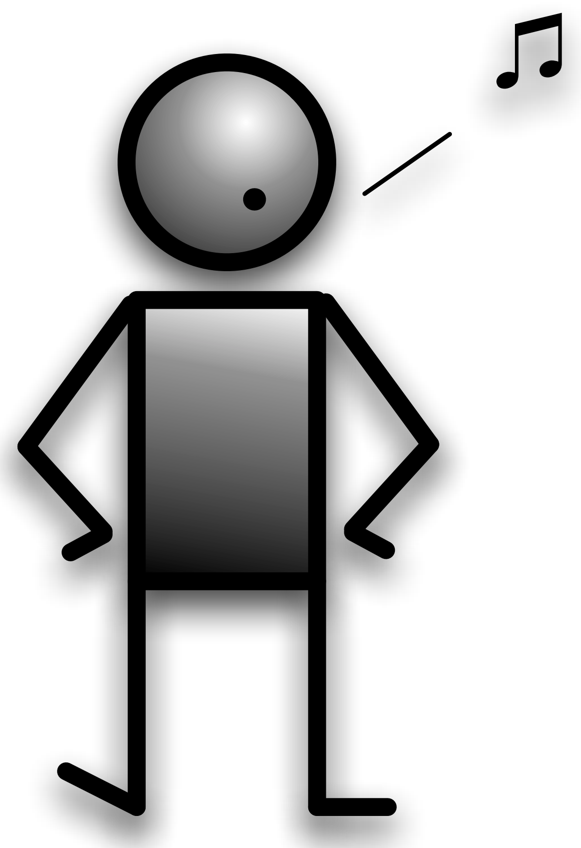
Pareto Optimality

English: 80
Math: 60

English: 60
Math: 80

These two are non-dominating to each other.

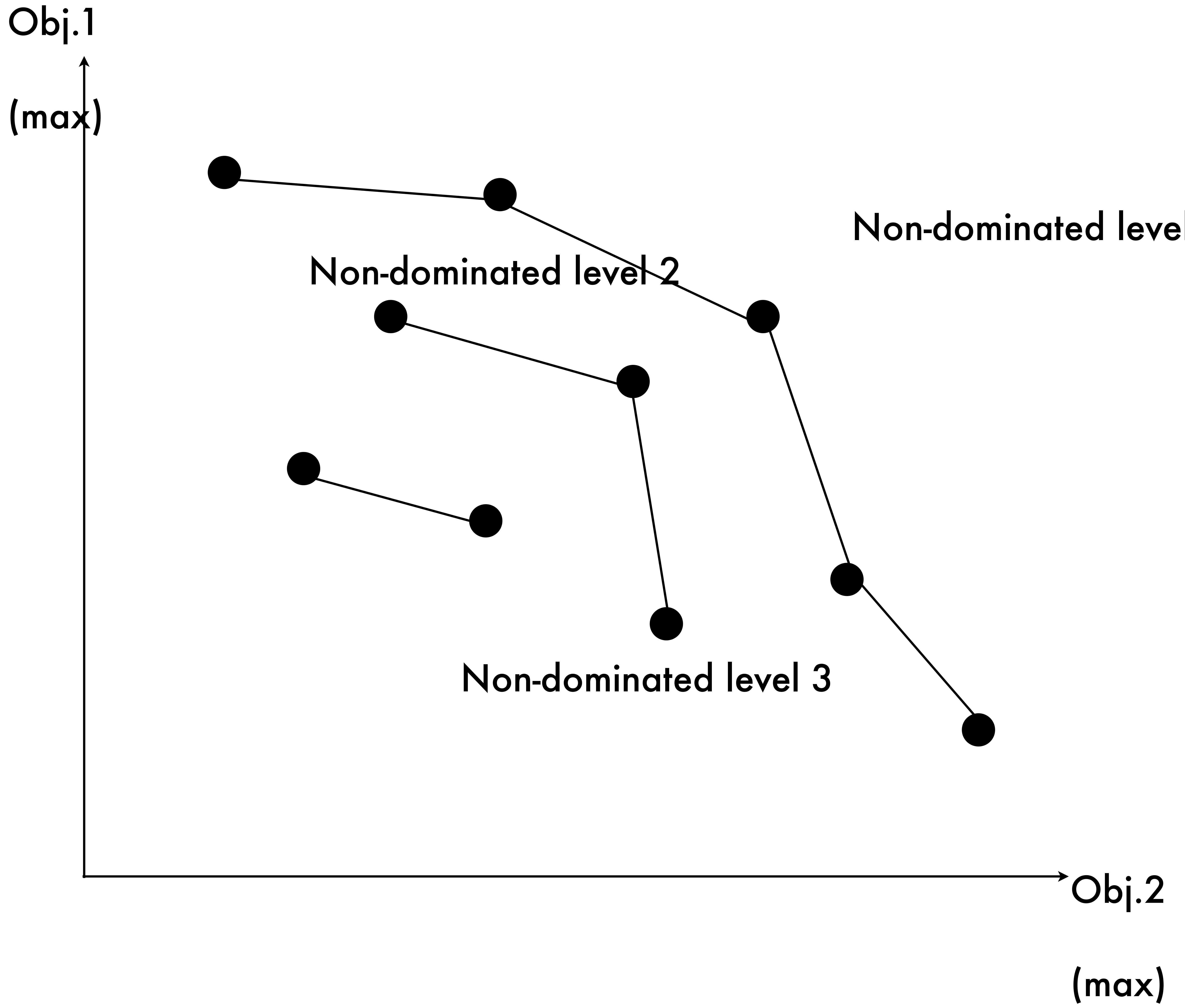



English: 90
Math: 80

English: 80
Math: 90

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 close to the true Pareto-front as possible (generally not measurable)
  - as uniformly distributed as possible
- Many variations of MOEAs regarding how to achieve convergence and diversity at the same time

NSGA-II (Deb et al., 2000)

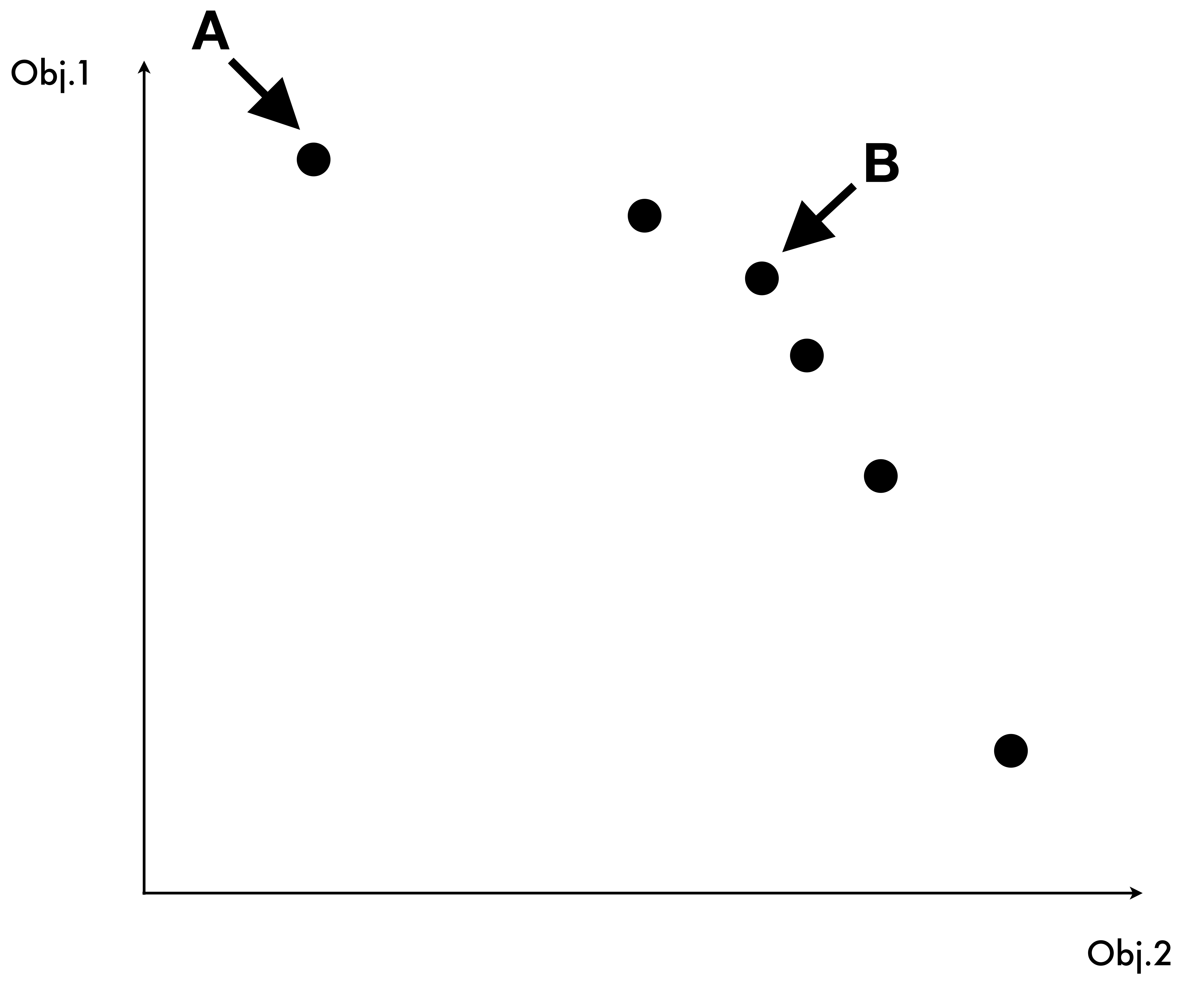
- Non-dominated Sorting Genetic Algorithm
    - Non-dominated sorting in $O(MN^2)$ complexity
        - (M: number of objectives, N: population size)
    - Crowding distance to encourage diversity
    - Elitism (the best front always carries over to the next gen.)

Fast Non-dominated Sort

```
fast-non-dominated-sort($P$)
for each $p \in P$
    $S_p = \emptyset$
    $n_p = 0$
    for each $q \in P$
        if $(p \prec q)$ then                 If $p$ dominates $q$
            $S_p = S_p \cup \{q\}$            Add $q$ to the set of solutions dominated by $p$
        else if $(q \prec p)$ then
            $n_p = n_p + 1$                   Increment the domination counter of $p$
    if $n_p = 0$ then                         $p$ belongs to the first front
        $p_{\mathrm{rank}} = 1$
        $\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$
$i = 1$                                       Initialize the front counter
while $\mathcal{F}_i \neq \emptyset$
    $Q = \emptyset$                           Used to store the members of the next front
    for each $p \in \mathcal{F}_i$
        for each $q \in S_p$
            $n_q = n_q - 1$
            if $n_q = 0$ then                 $q$ belongs to the next front
                $q_{\mathrm{rank}} = i + 1$
                $Q = Q \cup \{q\}$
    $i = i + 1$
    $\mathcal{F}_i = Q$
```

Crowding Distance

If you have to discard one solution, which one would you choose?



Crowding Distance

- A solution that is farther away from the others is rarer and, therefore, more valuable.
- If selecting from non-dominating pair, crowding distance can help differentiating solutions.

$$I_1 = I_n = \infty$$

$$I_i = \frac{I_{i+1} - I_{i-1}}{f_{max} - f_{min}} (2 \leq i \leq n-1)$$

Crowding Distance

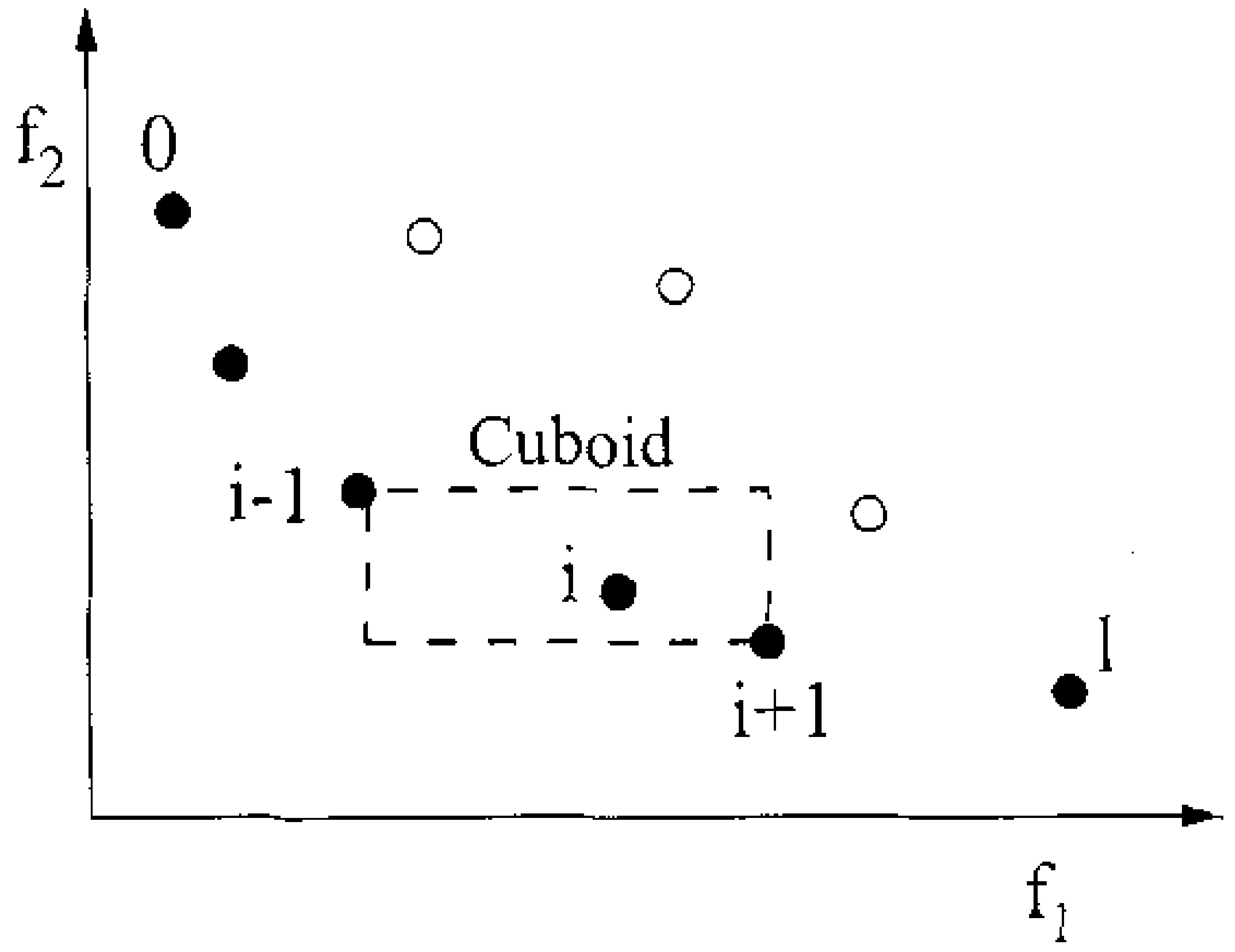
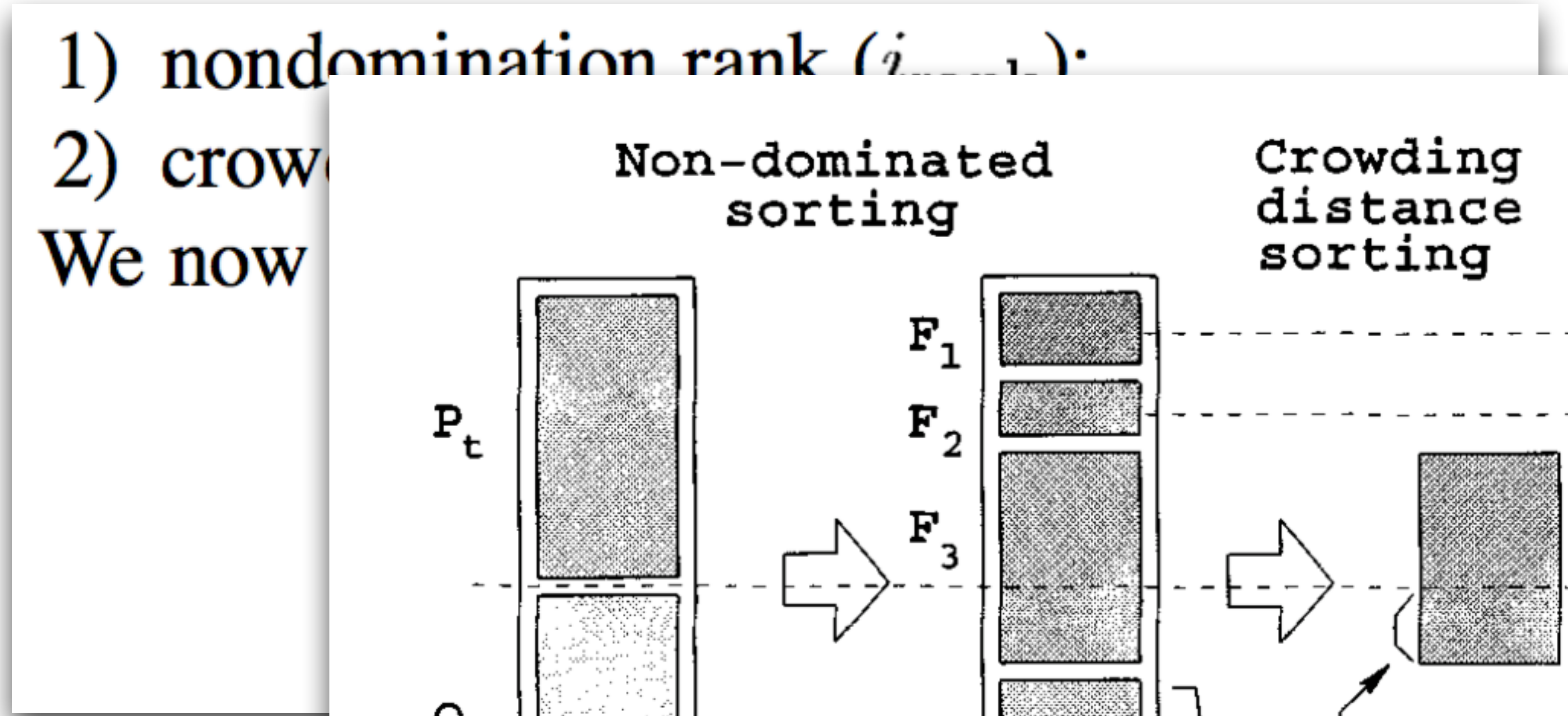


Fig. 1. Crowding-distance calculation. Points marked in filled circles are solutions of the same nondominated front.

Partial Order in Population



1) nondomination rank (i_{rank});

2) crow

We now

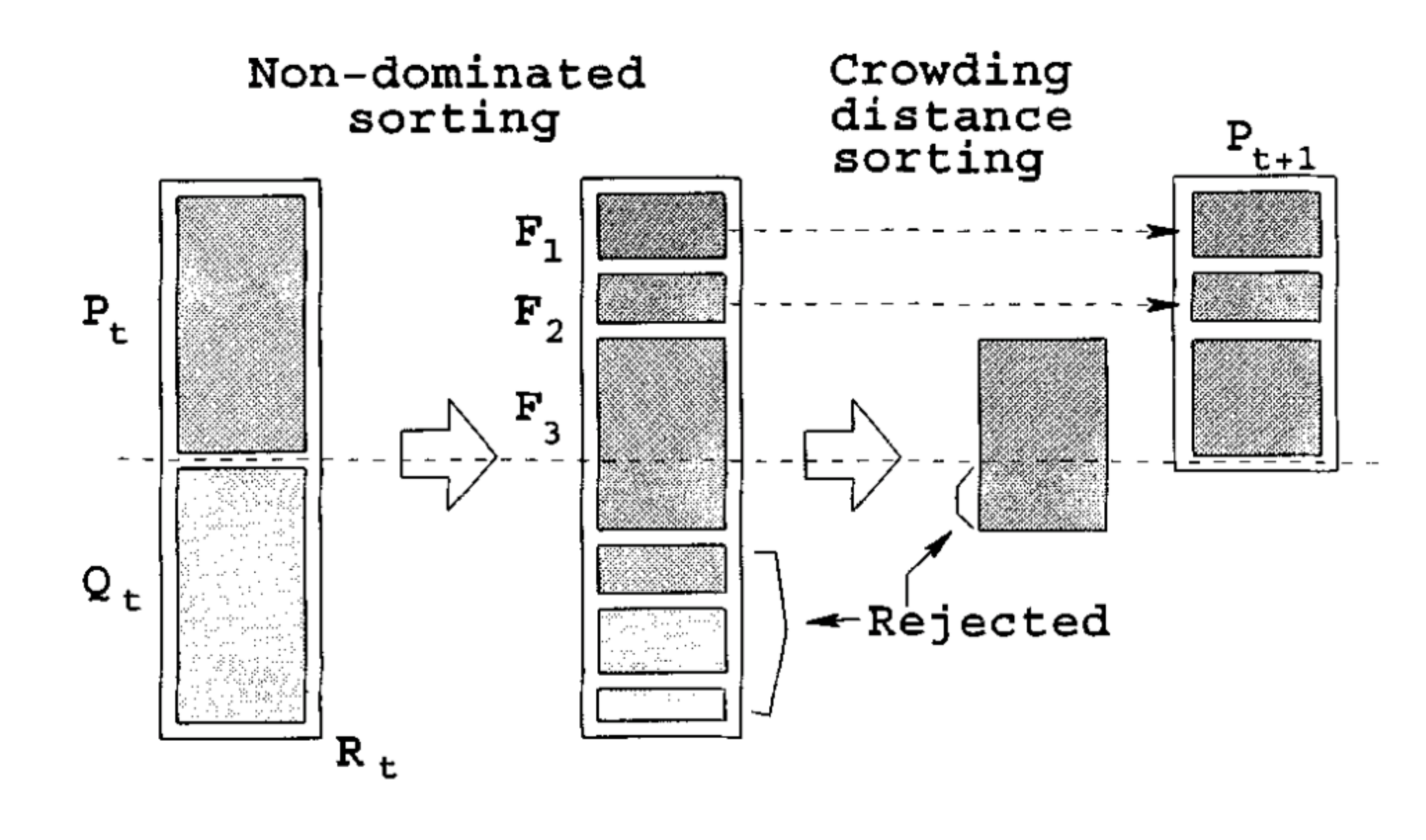


Fig. 2. NSGA-II procedure.

SPEA2 (Zitzler et al., 2001)

- 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., 2001)

Population *Archive*

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

Raw Fitness: $R(\boldsymbol{i}) = \sum_{\boldsymbol{j} \in \boldsymbol{P}_t + \overline{\boldsymbol{P}}_t, \boldsymbol{j} \succ \boldsymbol{i}} S(\boldsymbol{j}) \longrightarrow$ *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 sym

SPEA2 (Zitzler et al., 2001)

Updating Archive

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

(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

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

SPEA2 (Zitzler et al., 2001)

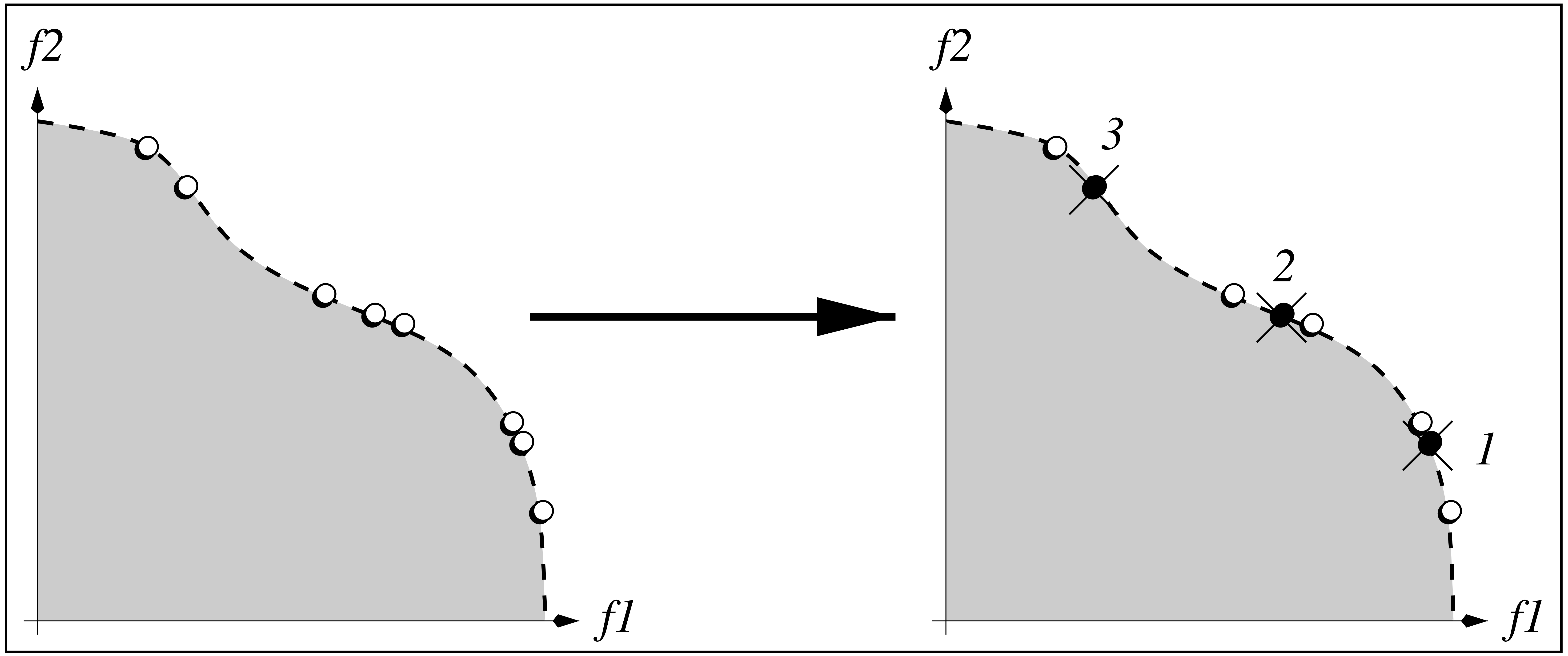


Figure 2: Illustration of the archive truncation method used in SPEA2. On the right, a nondominated set is shown. On the left, it is depicted which solutions are removed in which order by the truncate operator (assuming that $\overline{N} = 5$).

PESA2 (Corne et al., 2001)

- Spatial approach for selecting solutions:
  - Defines hyper-boxes in the multi-dimensional search space.
  - Each occupied hyper-box gets assigned a fitness based on the crowdedness.
  - Select hyper-box instead of individual solutions; then pick a random solution from the box.
- Maintain a non-dominated archive.

PESA2 (Corne et al., 2001)

- Probabilistic Arguments: assume that we have two hyper-boxes, one containing 9 solutions and the other just 1. The latter happens to be the most isolated. We do binary tournament selection.
- With NSGA-II: chance of selecting the most isolated solution is 1 - (9/10)^2 = 0.19, therefore chance of choosing one of the more crowded solutions is 0.81
- With PESA2: chance of choosing the less crowded box is 1 - (1/2)^2 = 0.75, therefore chance of choosing one of the more crowded solutions is 0.25.

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.

Multi-Objective Optimisation

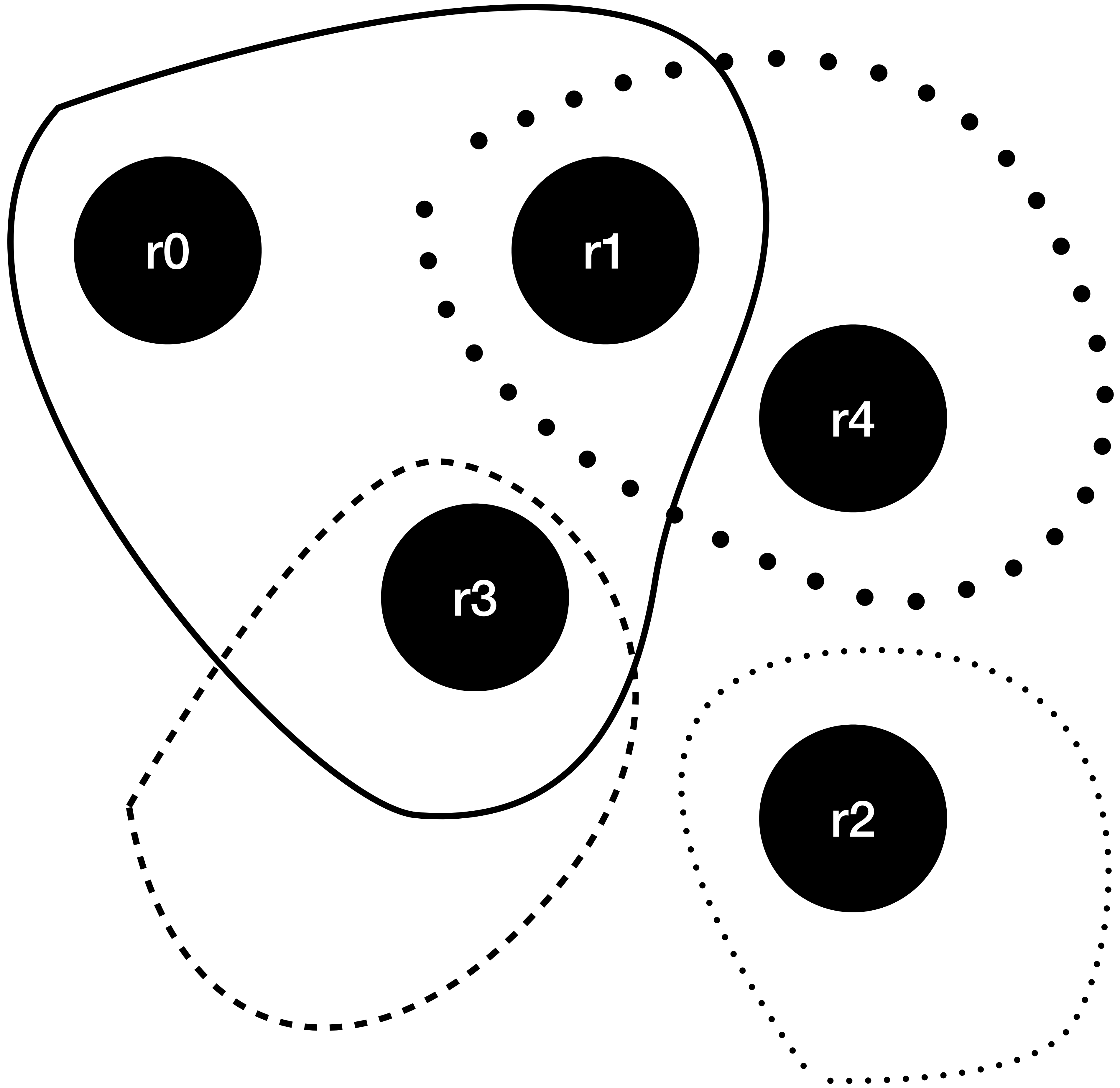
- Engineering is all about fine-tuning trade-offs.
- Software engineering is no exception!
- e.g. binary size vs. compiler optimisation, time to execute test vs. confidence in quality, cost of implementing requirements vs. expected revenue...

Case Study 1: Test Suite Minimisation

Pareto-Efficient Multi-Objective Test Case Selection, Yoo & Harman, ISSTA 2007

Test Suite Minimisation

- The Problem: Your regression test suite is too large.
- The Idea: There must be some redundant test cases.
- The Solution: Minimise (or reduce) your regression test suite by removing all the redundant tests.


r0
r1
r4
r3
r2

Test Suite Minimisation

Usually the information you need can be expressed as a matrix.

Things to tick off

(branches, statements,

DU-paths, etc)

	r0	r1	r2	r3	Time
t0	1	1	0	0	2
t1	0	1	0	1	3
t2	0	0	1	1	7
t3	0	0	1	0	3

Cost of ticking things off

Your tests

Now the problem becomes the following: what is the subset of rows (i.e. tests) that, when combined, will cover (i.e. put '1' on) the most of the columns (i.e. things to tick off)?

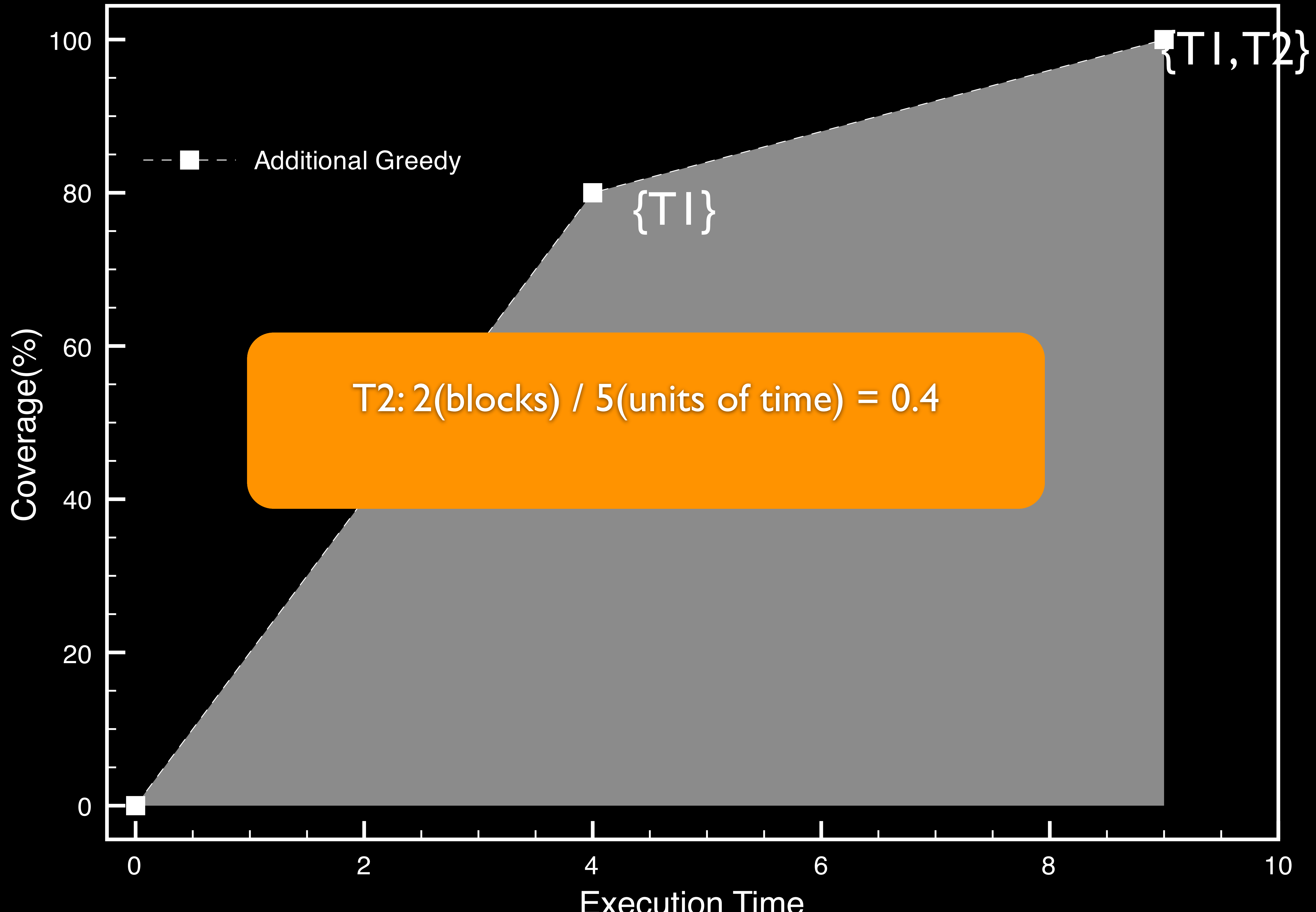
Test Suite Minimisation

- The problem definition you just saw maps nicely into “set-cover” problem (http://en.wikipedia.org/wiki/Set_cover).
- Unfortunately, the problem is known to be NP-complete, meaning that there is no known efficient AND exact algorithm.
- We rely on approximated heuristics.

Multi-Objective Optimisation

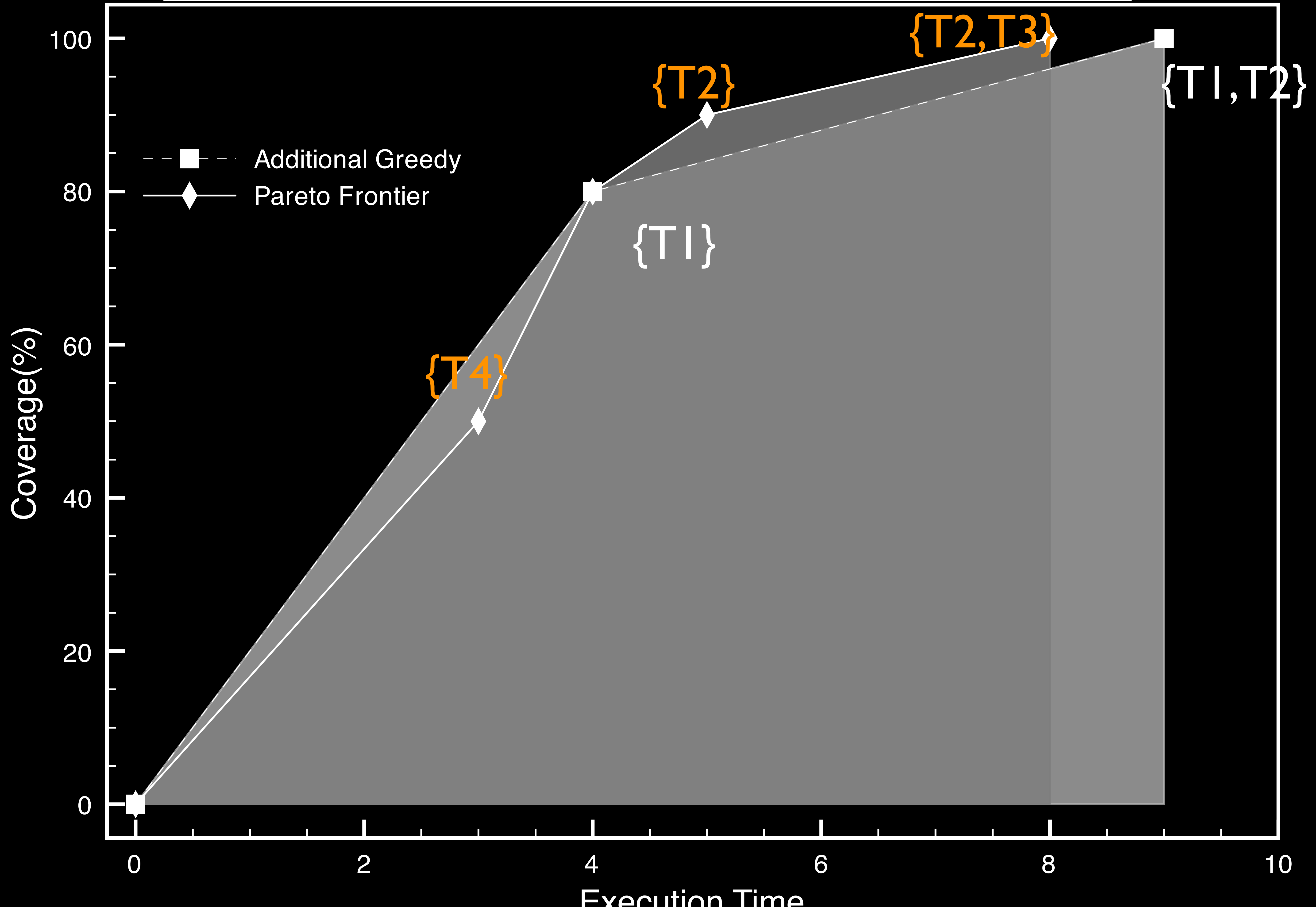
- We want to select a subset of tests that:
  - minimises the cost
  - maximises the coverage (ticking-off)
  - and possibly maximises other desirable things, such as inclusion of tests that previously detected faults
- The 2007 ISSTA paper studies the use of NSGA-2

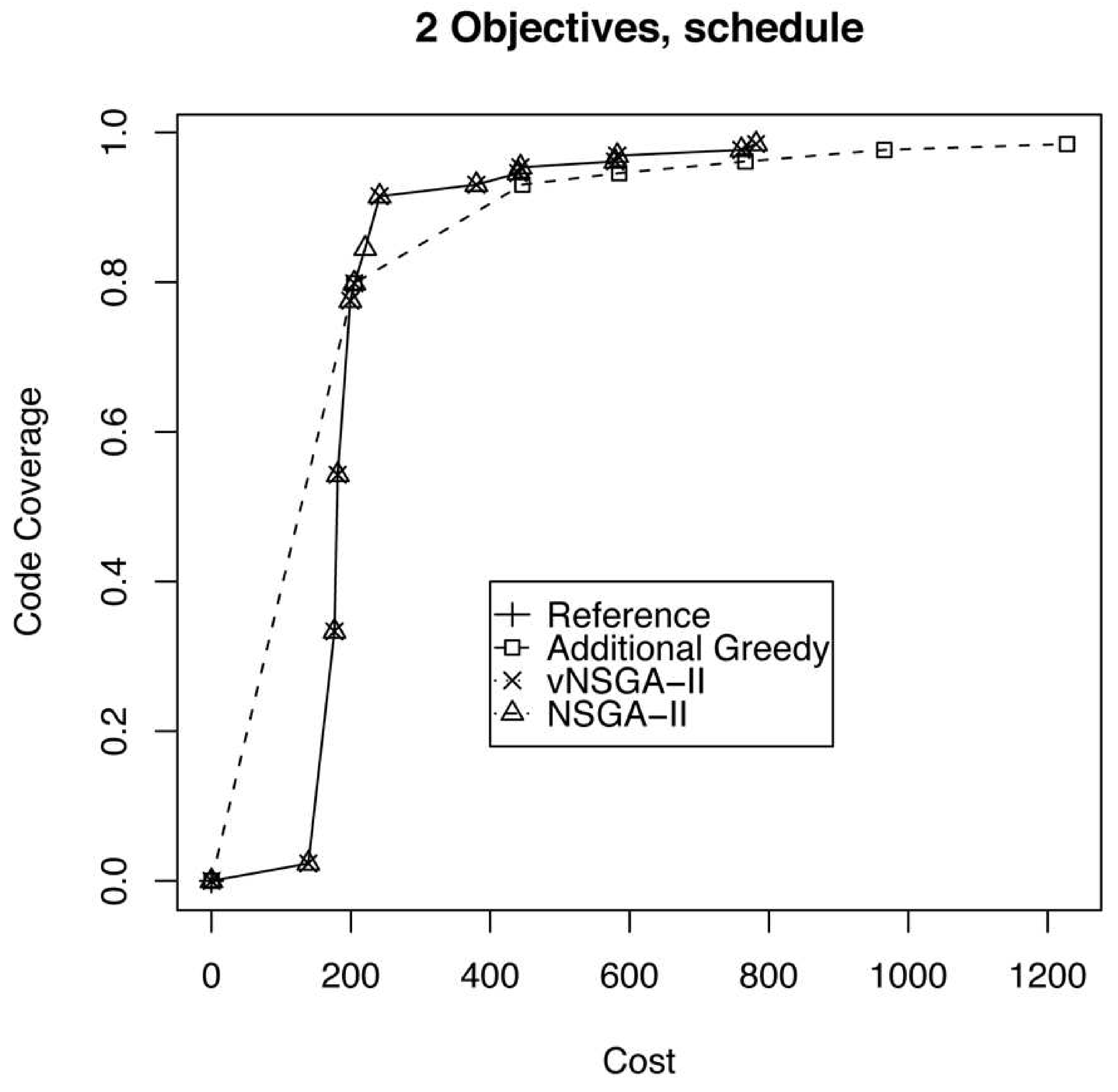
Test Case	Program Blocks										Time
	1	2	3	4	5	6	7	8	9	10	
T1	x	x	x	x	x	x	x	x			4
T2	x	x		x	x	x	x	x	x	x	5
T3	x	x	x						x		3
T4	x	x	x	x	x						3

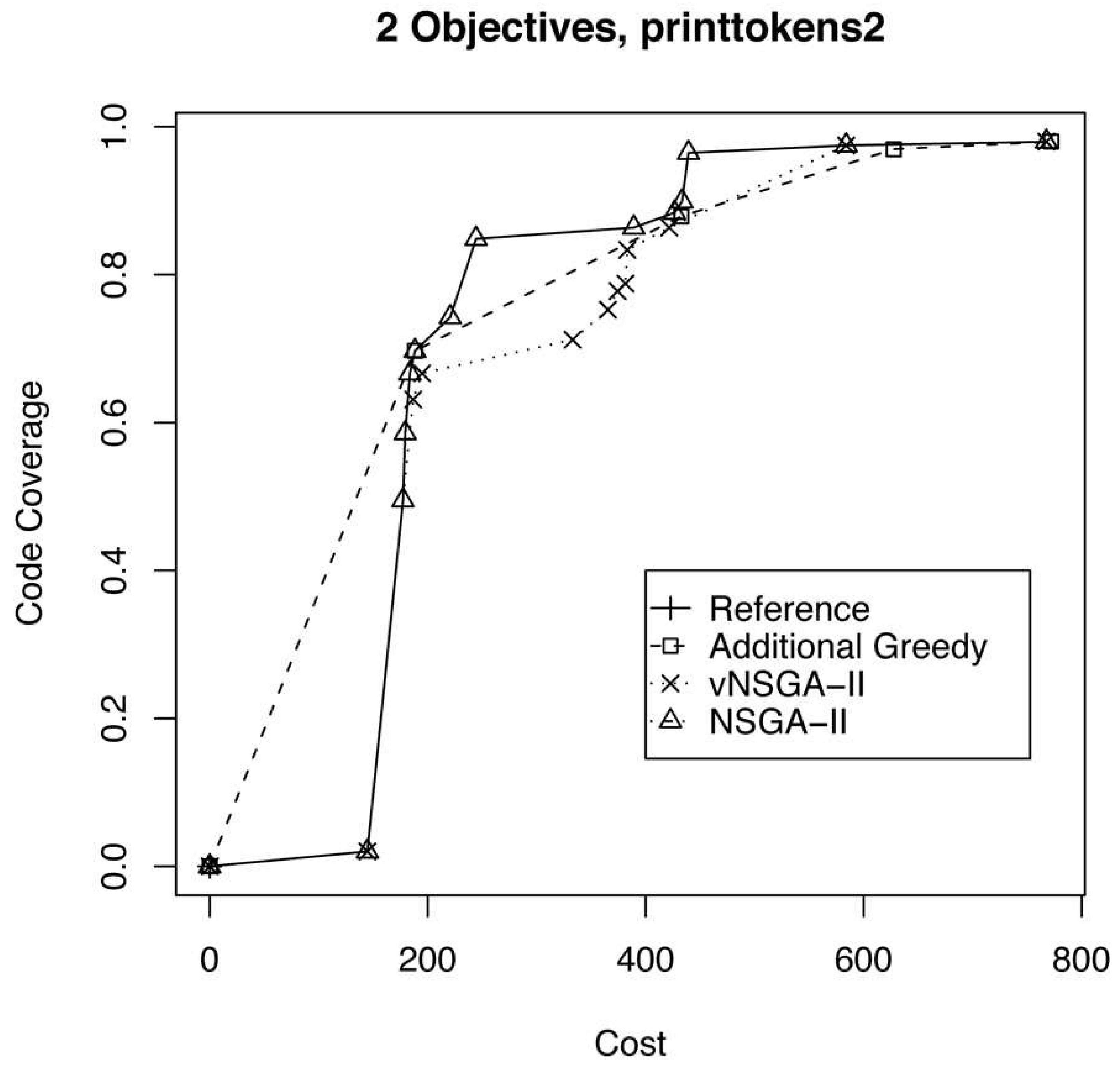


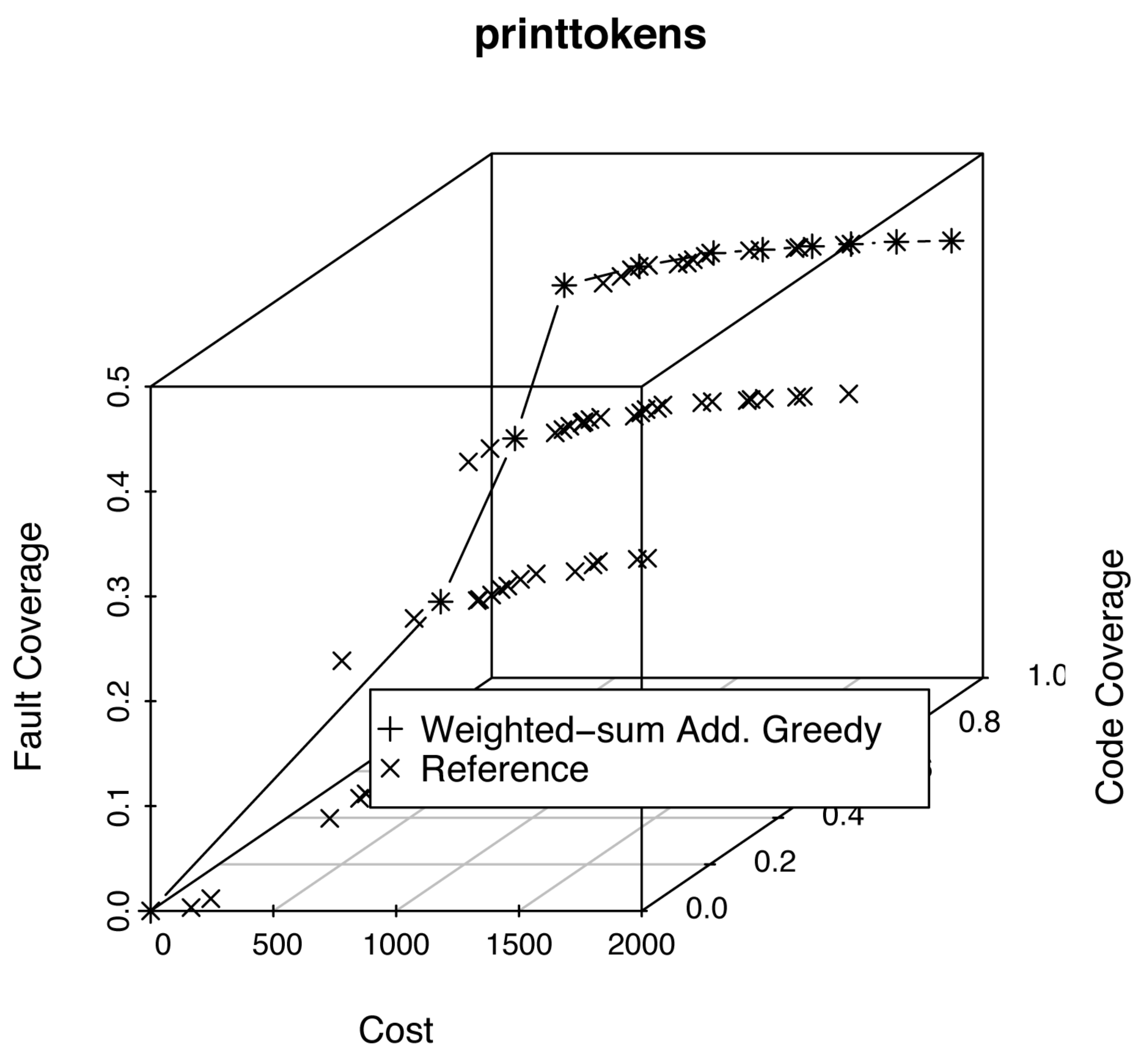
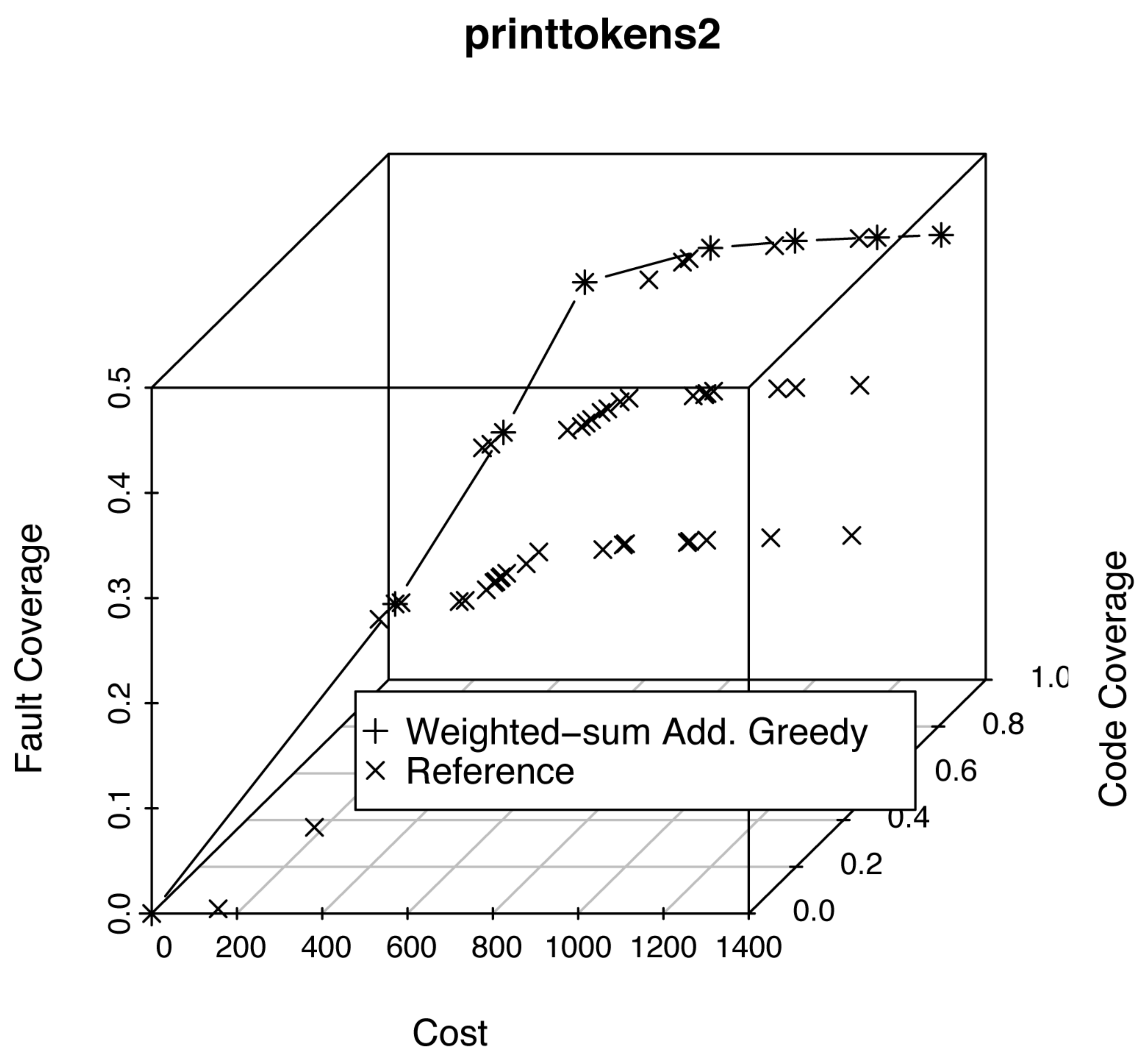
T2: 2(blocks) / 5(units of time) = 0.4

Test Case	Program Blocks										Time
	1	2	3	4	5	6	7	8	9	10	
T1	x	x	x	x	x	x	x	x			4
T2	x	x		x	x	x	x	x	x	x	5
T3	x	x	x						x		3
T4	x	x	x	x	x						3




2 Objectives, schedule
Code Coverage
Cost
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
1200
Reference
Additional Greedy
vNSGA−II
NSGA−II


2 Objectives, printtokens2
Code Coverage
Cost
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
Reference
Additional Greedy
vNSGA−II
NSGA−II


printtokens
Fault Coverage
Cost
Code Coverage
0.0
0.1
0.2
0.3
0.4
0.5
0
500
1000
1500
2000
0.0
0.2
0.4
0.8
1.0
+ Weighted−sum Add. Greedy
× Reference

printtokens2
Fault Coverage
Cost
Code Coverage
0.0
0.1
0.2
0.3
0.4
0.5
0
200
400
600
800
1000
1200
1400
0.0
0.2
0.4
0.6
0.8
1.0
+ Weighted−sum Add. Greedy
× Reference

Adventure in Google

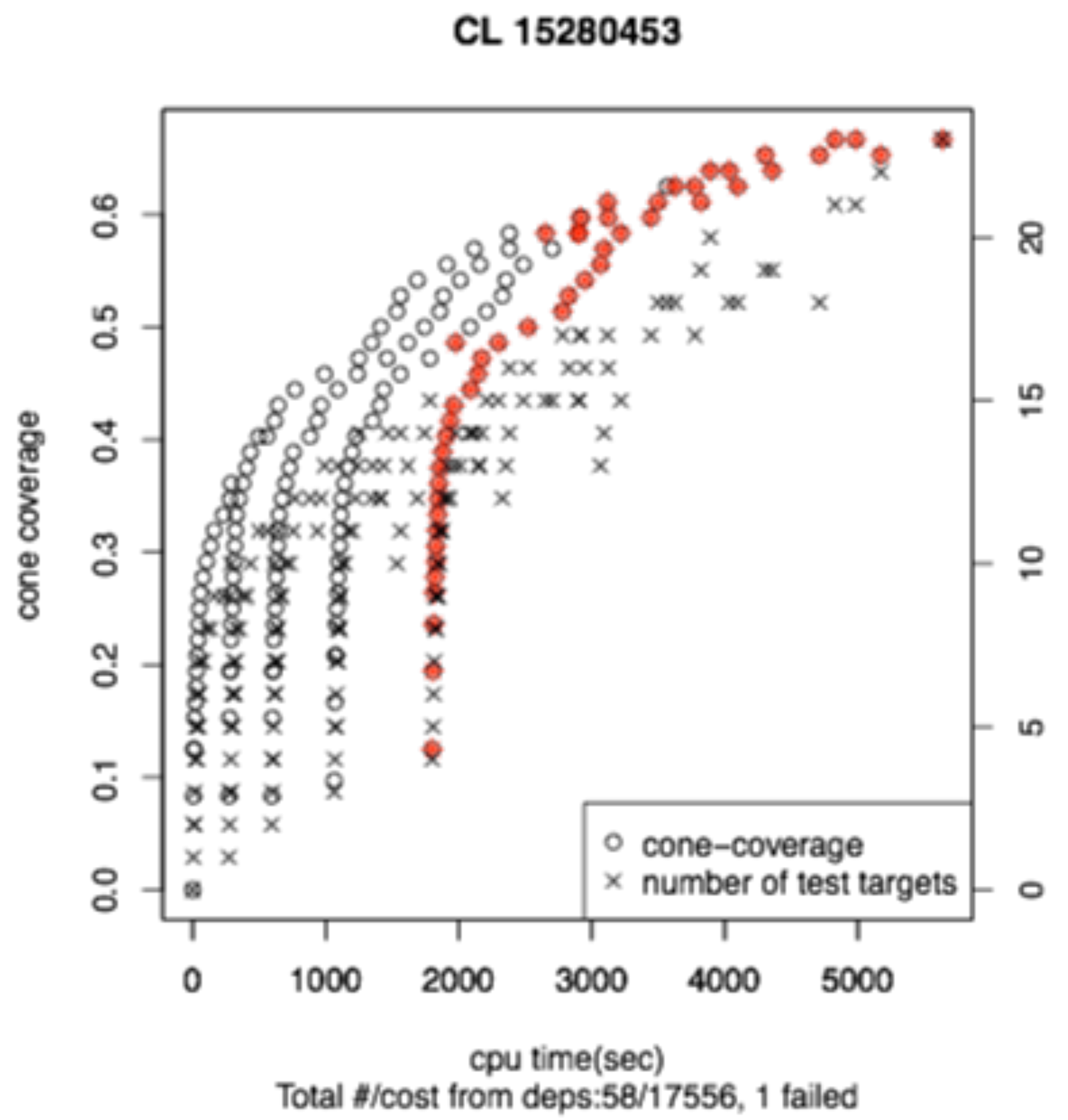
- An automated testing infrastructure that executes 120M tests per day against all code submitted to the single repository
- For each project/module, a list of “must-run” tests is maintained and executed

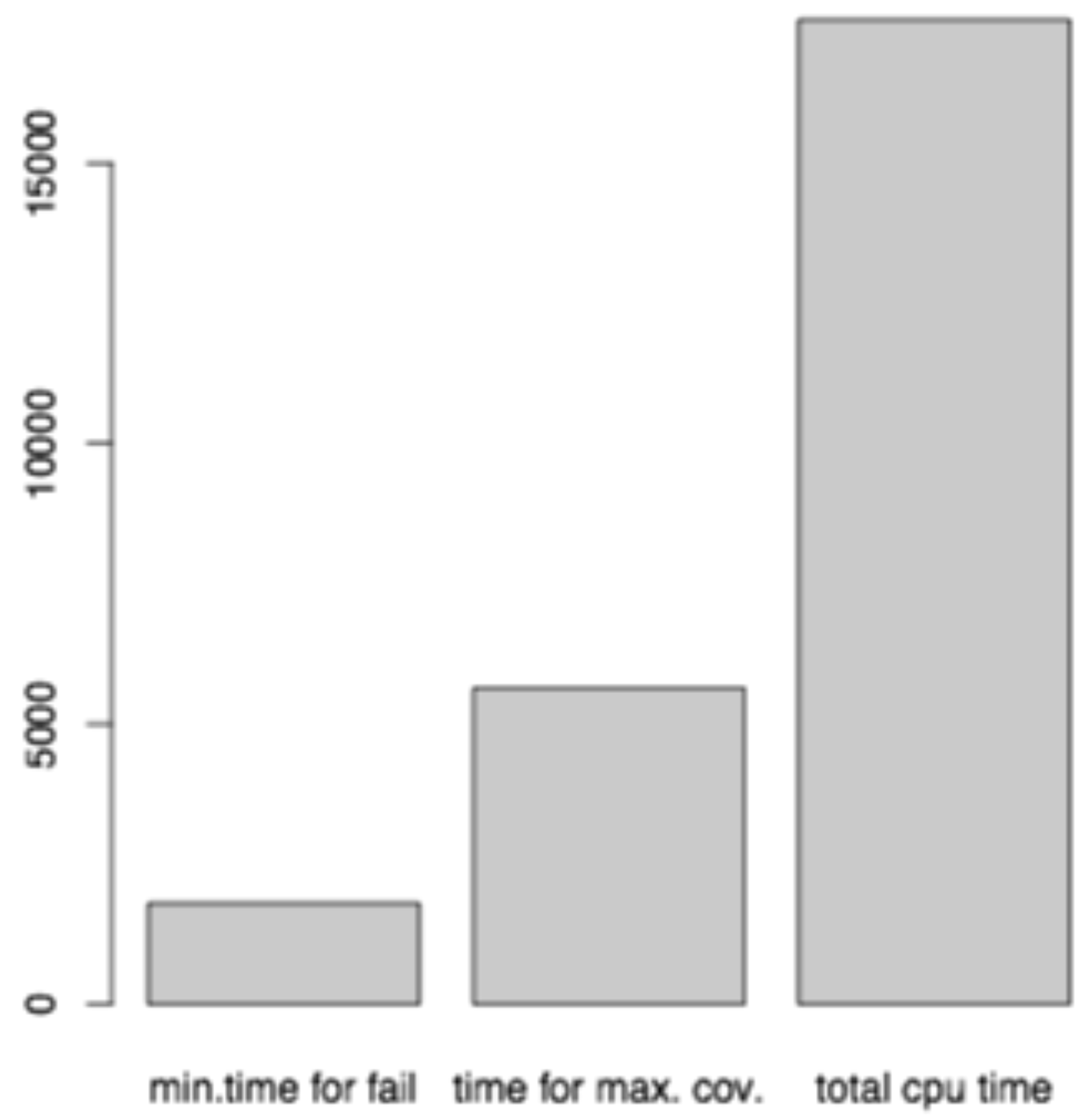
Copland. Google’s Innovation Factory, ICST 2010 Keynote

Adventure in Google

- Need for early feedback: better understanding of expected behaviour, less waiting, better fault localisation

Multi-Objective Optimisation





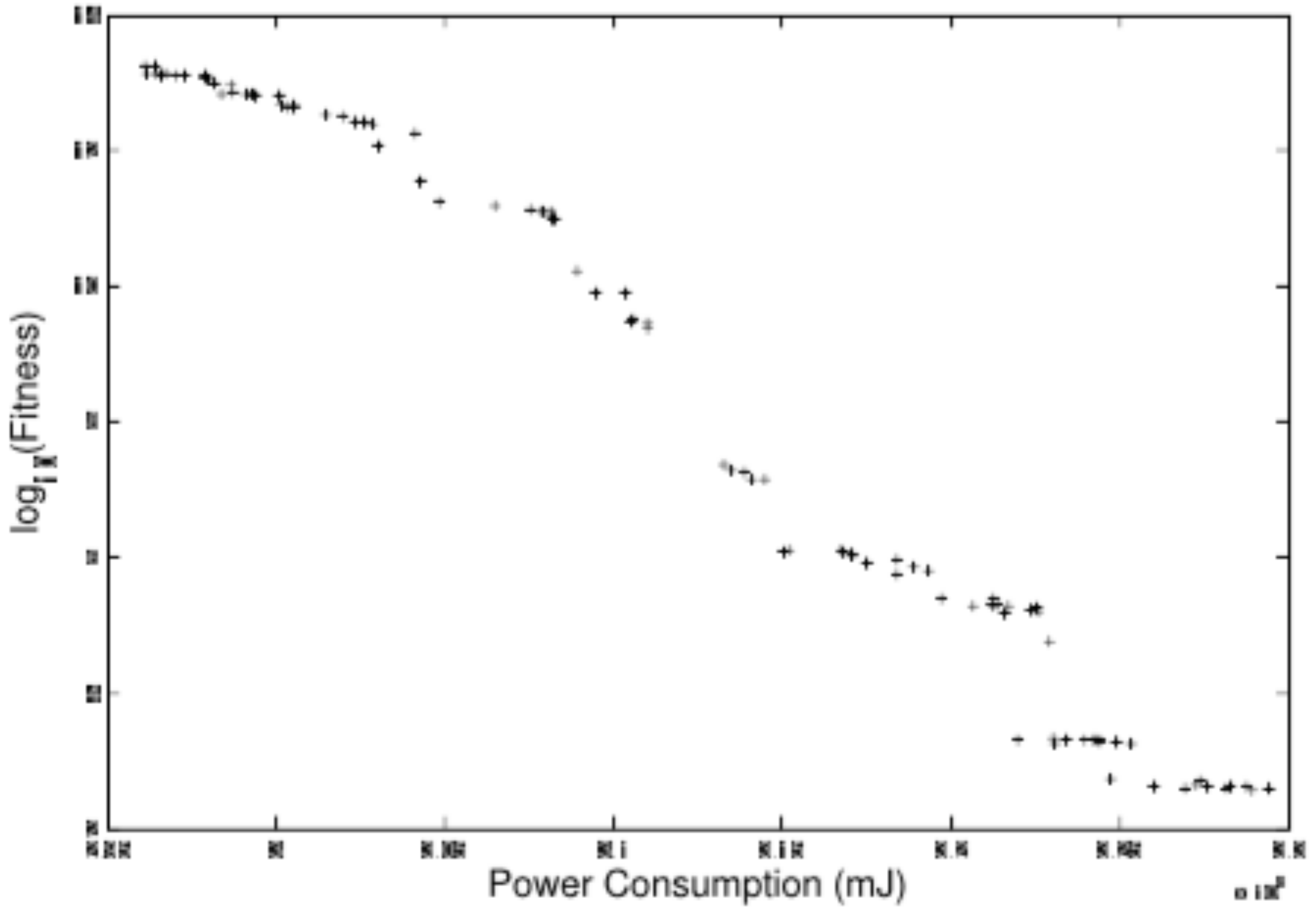
Two-Archive algorithm

Objectives: dependency coverage, execution cost, fault history

(invited to GTAC 2010, Yoo, Harman & Nilsson)

Even weirder tradeoff

- Can we trade functional and non-functional properties?
  - Your software is 100% correct, but the laptop battery runs out after 3 hours.
  - Your software is ever so slightly wrong (or crashes every now and then), but the battery lasts 6 hours.
- Energy-efficient random number generator: if you allow a little less randomness (?), you can save energy.



D. R. White, J. Clark, J. Jacob, and S. M. Poulding. Searching for resource-efficient programs: Low- power pseudorandom number generators. In Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO '08, pages 1775–1782, New York, NY, USA, 2008. ACM.

Criticisms to MO formulations

- In the end, don’t we have to pick **one** solution? Which one do we pick? If there are too many solutions on the Pareto front, it does not mean good optimisation, it means bad news for the user.
- One counter-claim is that the insight lies not in single solutions but in the shape of the Pareto front. But, this is still a reasonable critique, to which no real answer exists.

Summary

- Pareto optimality results in a group of non-dominated solutions, rather than a single solution.
- Preserving and promoting diversity becomes even more important!
- Observing trade-offs is often important in SE context, but just producing more solutions is not the goal in itself.