Multiobjective Evolutionary Algorithms

Dr. Michael Lones
Room EM.G31
M.Lones@hw.ac.uk

Overview

- Multi-Objective Evolutionary Algorithms (MOEAs)
  - Are EAs that are used to solve problems with **multiple conflicting objectives**
  - Most real world problems have multiple conflicting objectives, so these are an important group of EAs
  - There are lots of different kinds of MOEA out there; this lecture will cover some of the best known of these

Example: Optimising a Car

Suppose you were using an EA to optimise a car

- What does an optimal car look like?

Example: Optimising a Car

Suppose you were using an EA to optimise a car

- What does an optimal car look like?
  - As fast as possible?

Example: Optimising a Car

Suppose you were using an EA to optimise a car

- What does an optimal car look like?
  - As fast as possible?
  - As many seats as possible?

Example: Optimising a Car

Suppose you were using an EA to optimise a car

- What does an optimal car look like?
  - As fast as possible?
  - As many seats as possible?
  - As survivable as possible?

Example: Optimising a Car

Suppose you were using an EA to optimise a car

- What does an optimal car look like?
  - As fast as possible?
  - As many seats as possible?
  - As survivable as possible?
  - As affordable as possible?

Example: Optimising a Car

- There is no single optimal design for a car
  - There are **multiple objectives** and the objectives are **conflicting**, meaning that different solutions represent different trade-offs between the objectives

Design that trades off capacity and affordability for speed

Design that trades off affordability and speed for survivability

Design that trades off speed for carrying capacity

Design that trades off speed and capacity for affordability

Objective Values

- In an MOEA, solutions have more than one measure of fitness, known as their objective values

Speed = 1.0
Affordability = 0.0
Survivability = 0.8
Capacity = 0.1

Speed = 0.3
Affordability = 0.7
Survivability = 0.5
Capacity = 1.0

Dominance

- A notion of dominance is used to compare solutions
  - A solution is said to **dominate** another solution if it is better in at least one objective and no worse in all others

Speed = **1.0**
Affordability = 0.0
Survivability = 0.8
Capacity = 0.1

Dominates

Speed = **0.9**
Affordability = 0.0
Survivability = 0.8
Capacity = 0.1

Dominance

- A notion of dominance is used to compare solutions
  - A solution is said to **dominate** another solution if it is better in at least one objective and no worse in all others

Speed = **1.0**
Affordability = 0.0
Survivability = 0.8
Capacity = 0.1

Does not dominate

Speed = **0.9**
Affordability = 0.0
Survivability = **0.9**
Capacity = 0.1

Dominance

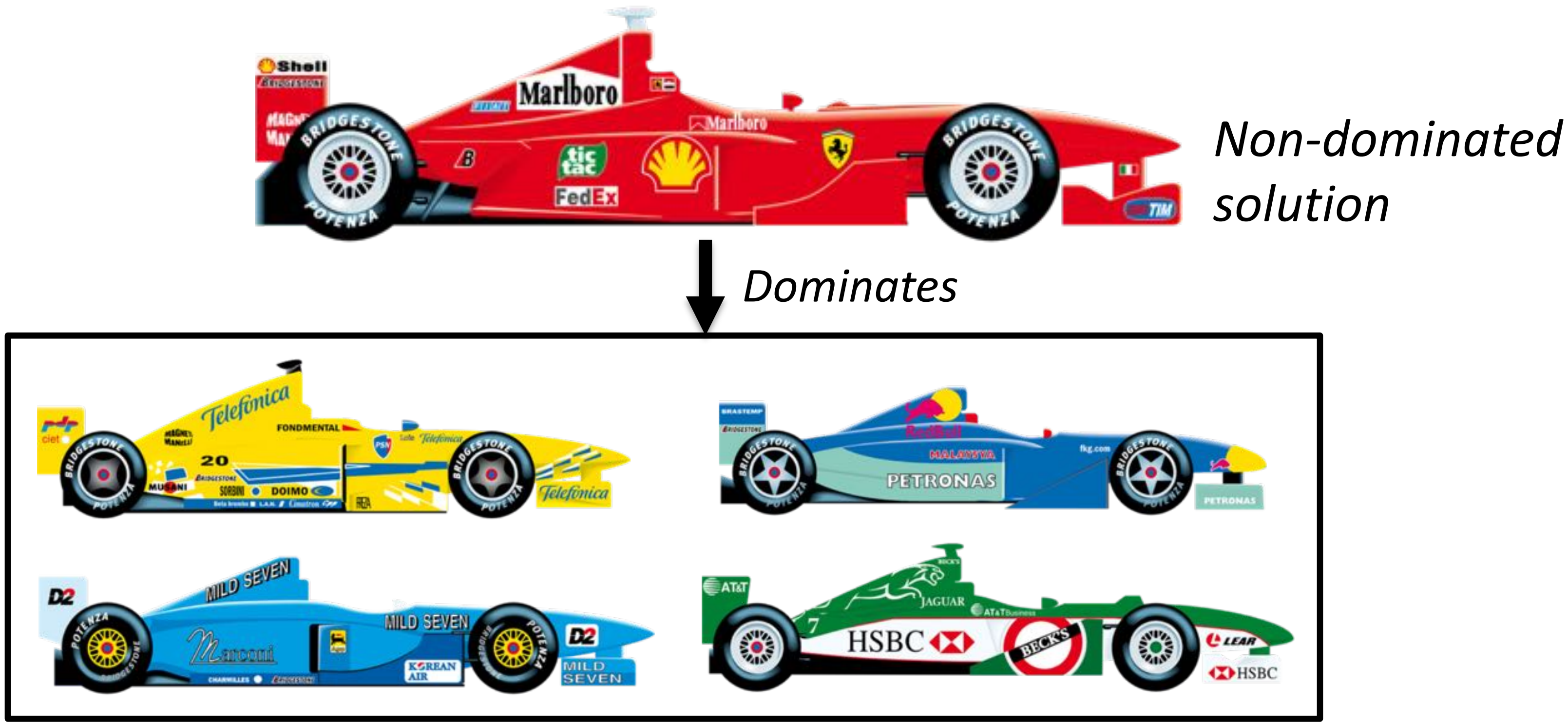
- Many solutions are incomparable
  - They're better at different things, so you can't say that one is better than the other

Speed = 1.0
Affordability = 0.0
Survivability = 0.8
Capacity = 0.1

Speed = 0.3
Affordability = 0.7
Survivability = 0.5
Capacity = 1.0

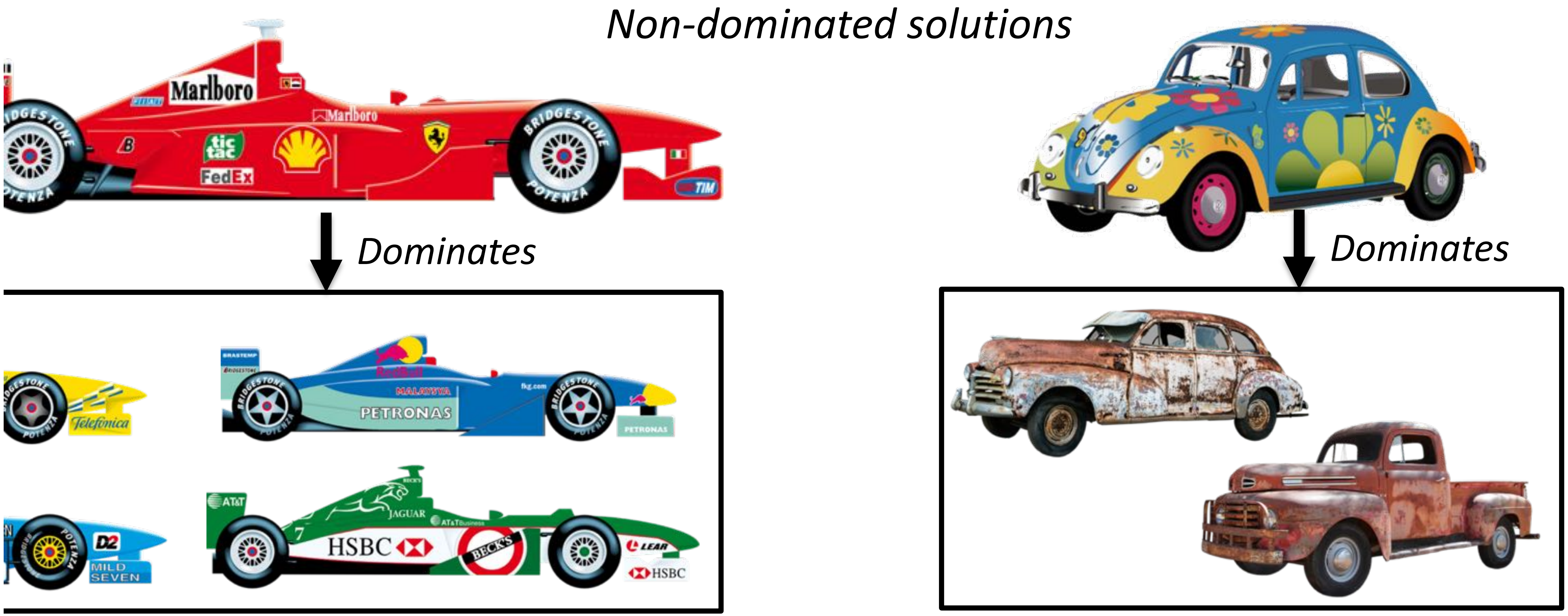
Non-dominated solution

- A non-dominated solution is dominated by no others
  - It is optimal within a particular part of the design space



Non-dominated solution

- A non-dominated solution is dominated by no others
  - Other non-dominated solutions can be found elsewhere



Pareto optimal set

- The non-dominated solutions found in a particular search space are known as its **Pareto optimal set**
  - E.g. all these are optimal for some kind of activity:

Any Questions?

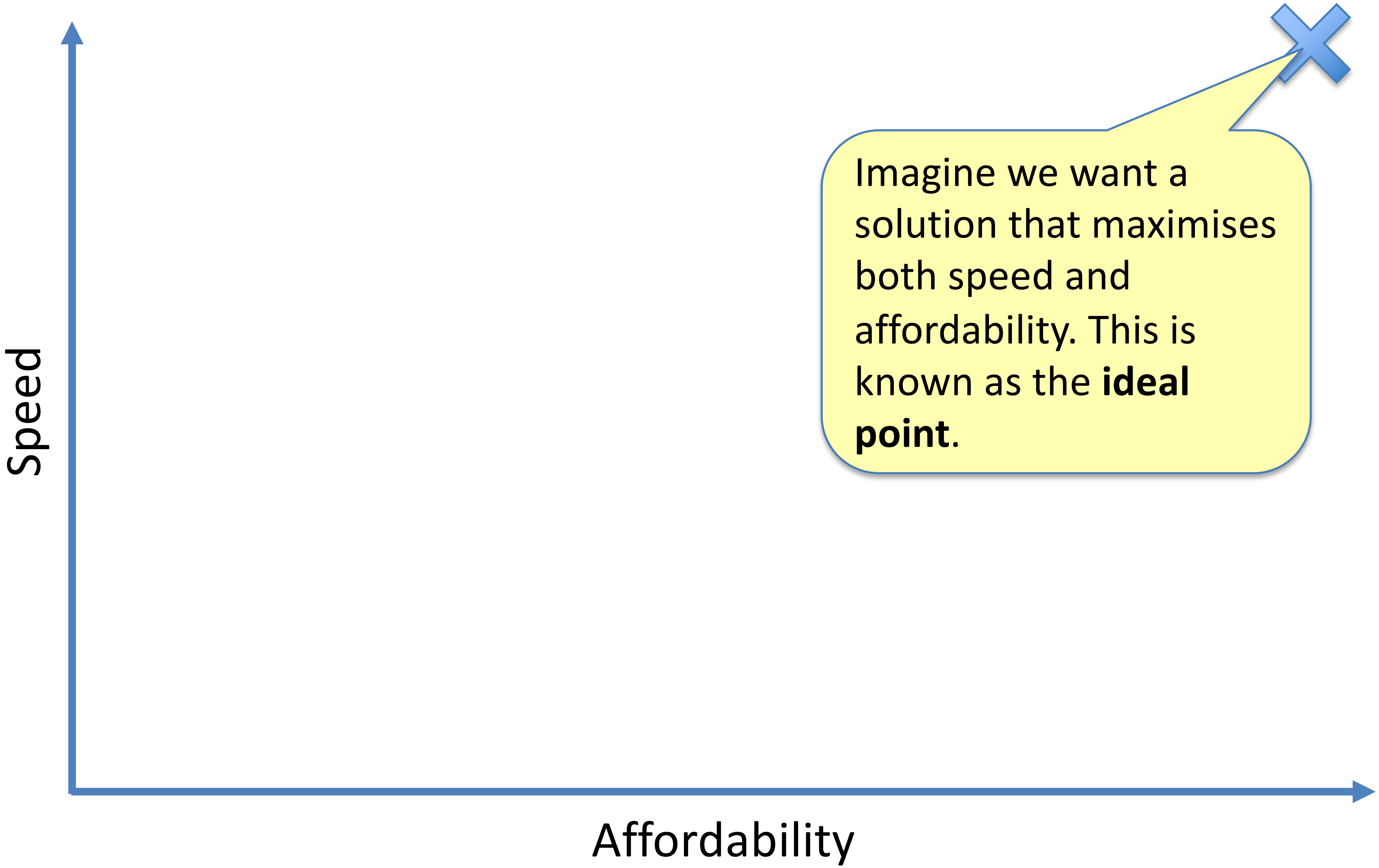
Objective Space

- When discussing MOEAs, it is common to talk about the **objective space**
  - This has one axis for each objective
  - If there are two objectives, it can be plotted as a 2D graph
  - If there are more than three dimensions, it can be very hard to visualise!

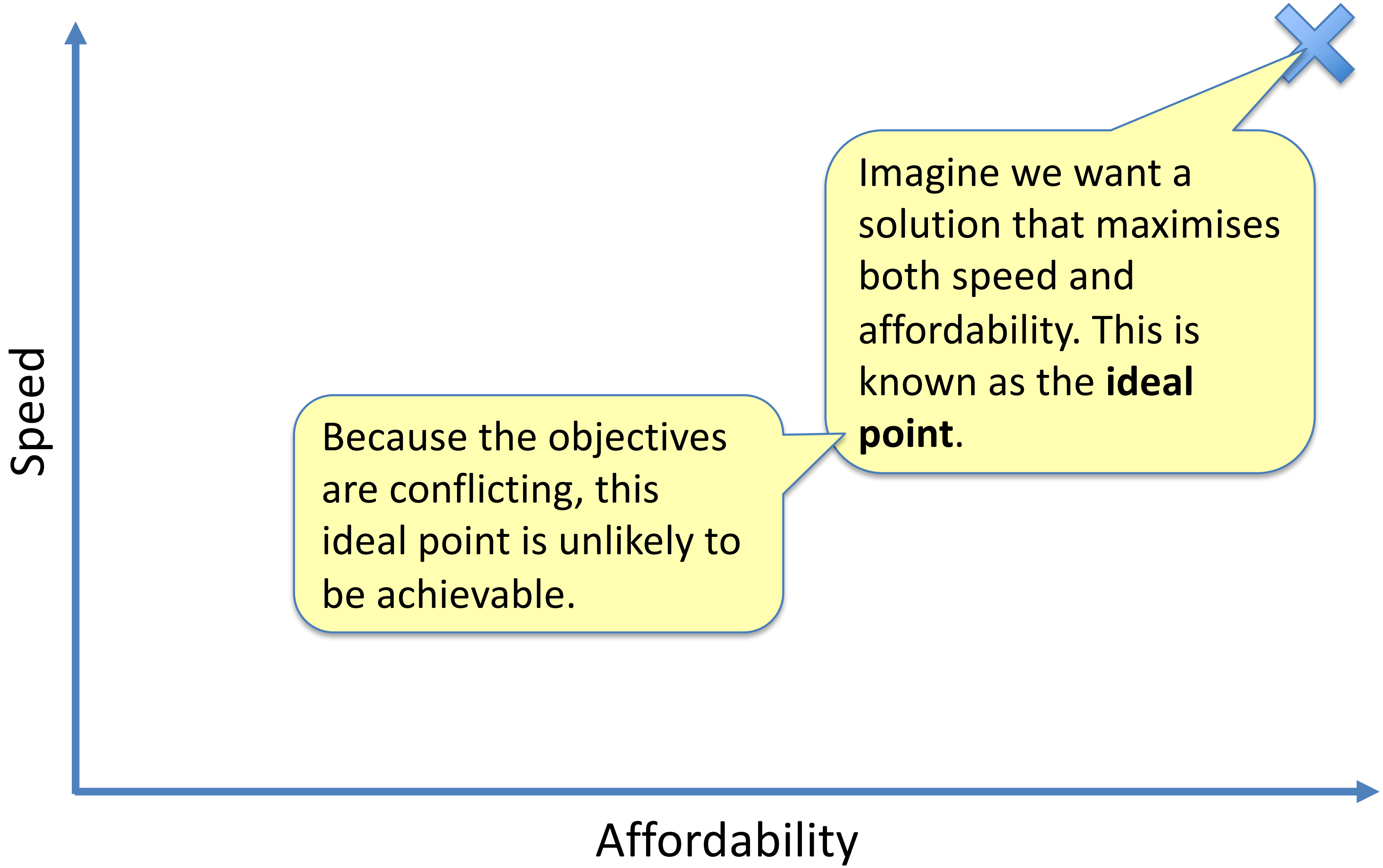
Objective 2

Objective 1

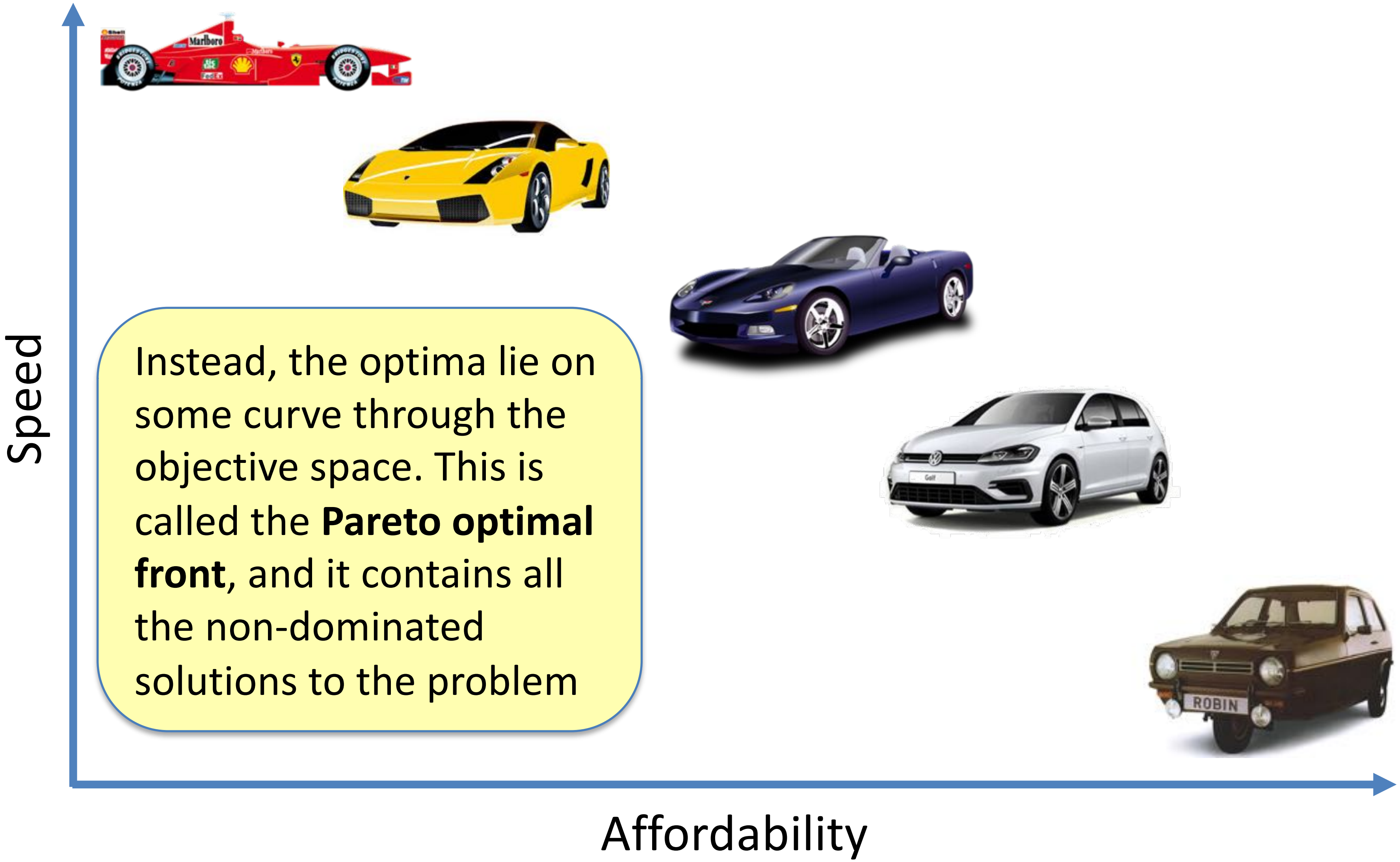
Objective Space Example



Objective Space Example

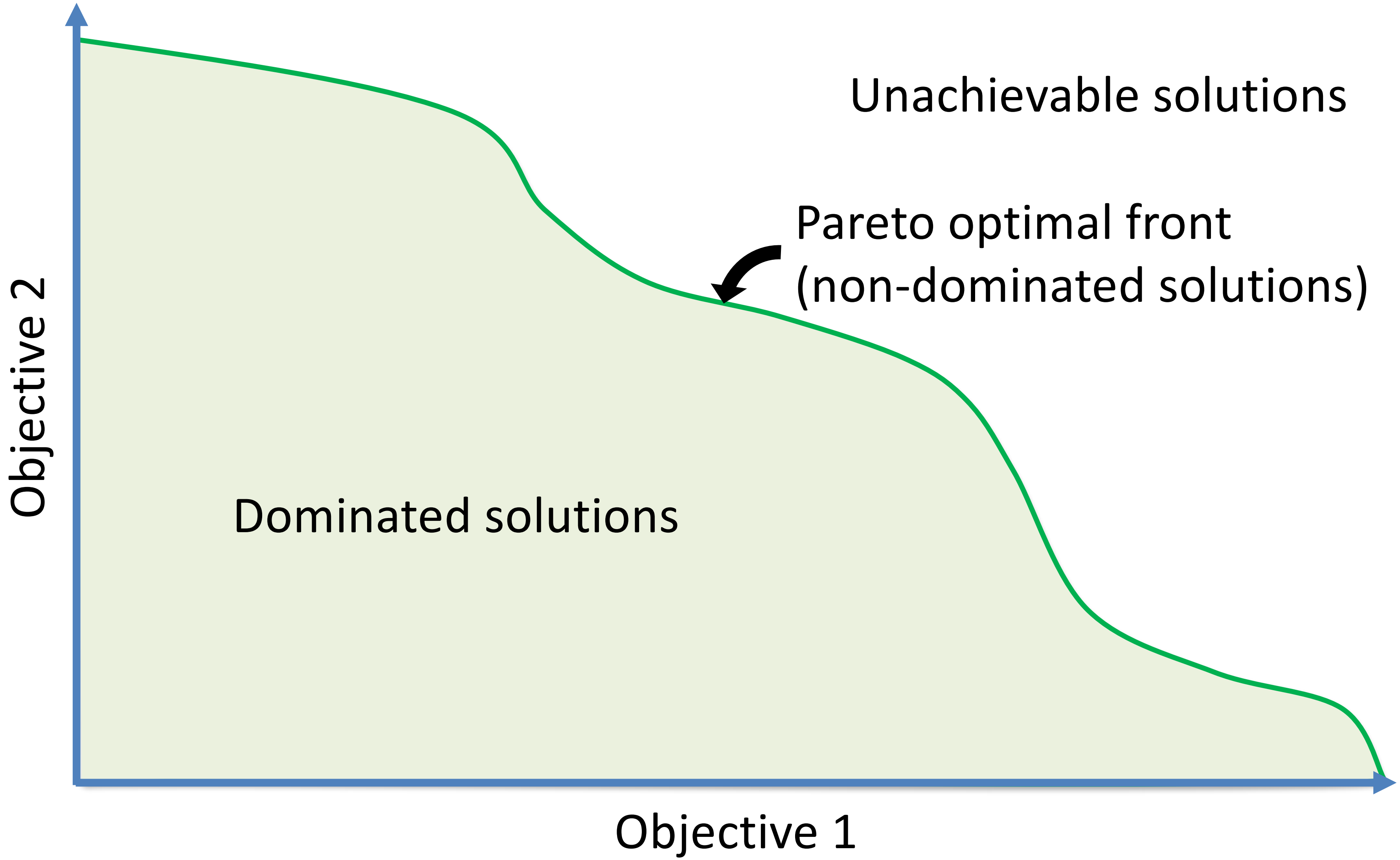


Objective Space Example

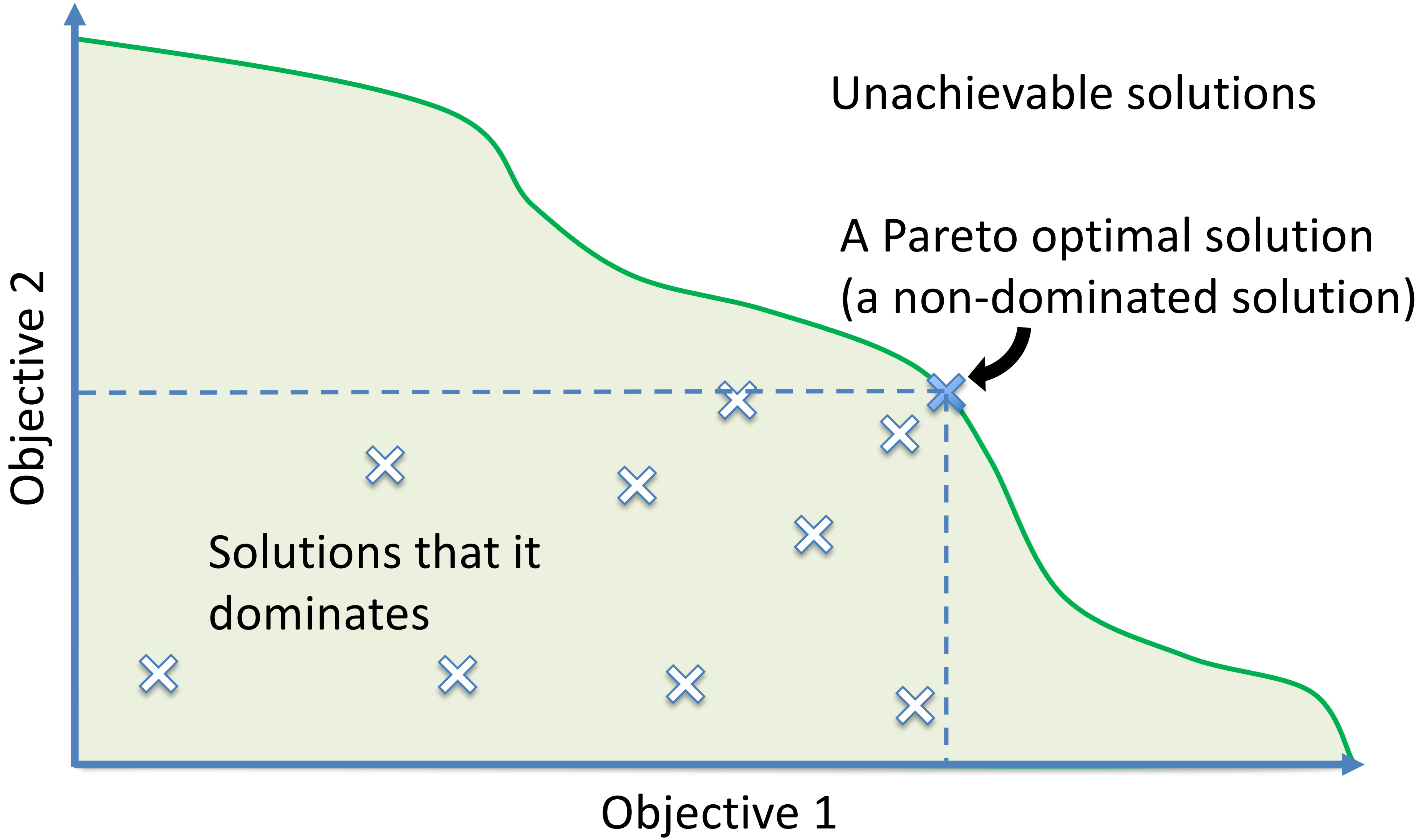


Instead, the optima lie on some curve through the objective space. This is called the **Pareto optimal front**, and it contains all the non-dominated solutions to the problem

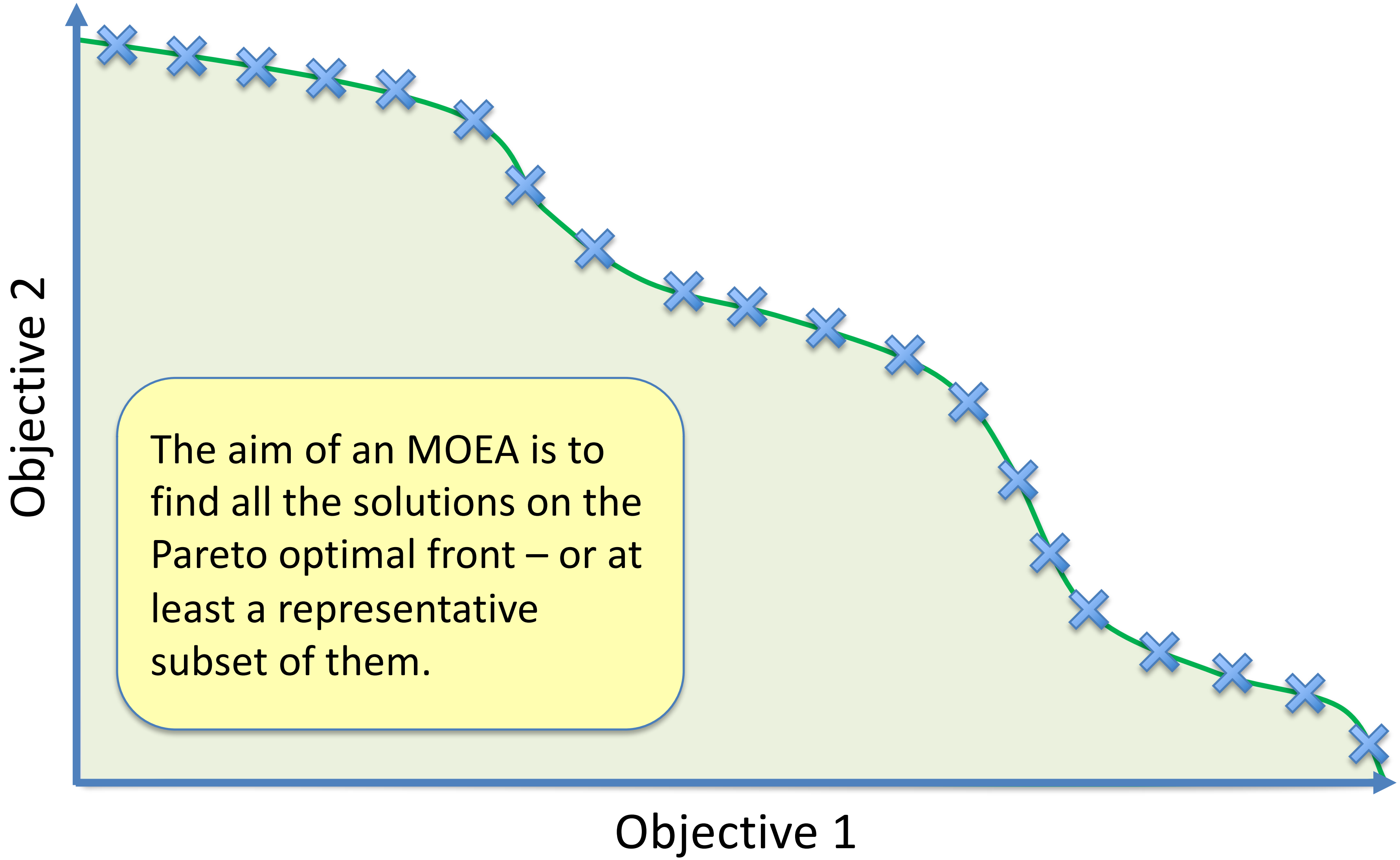
Objective Space



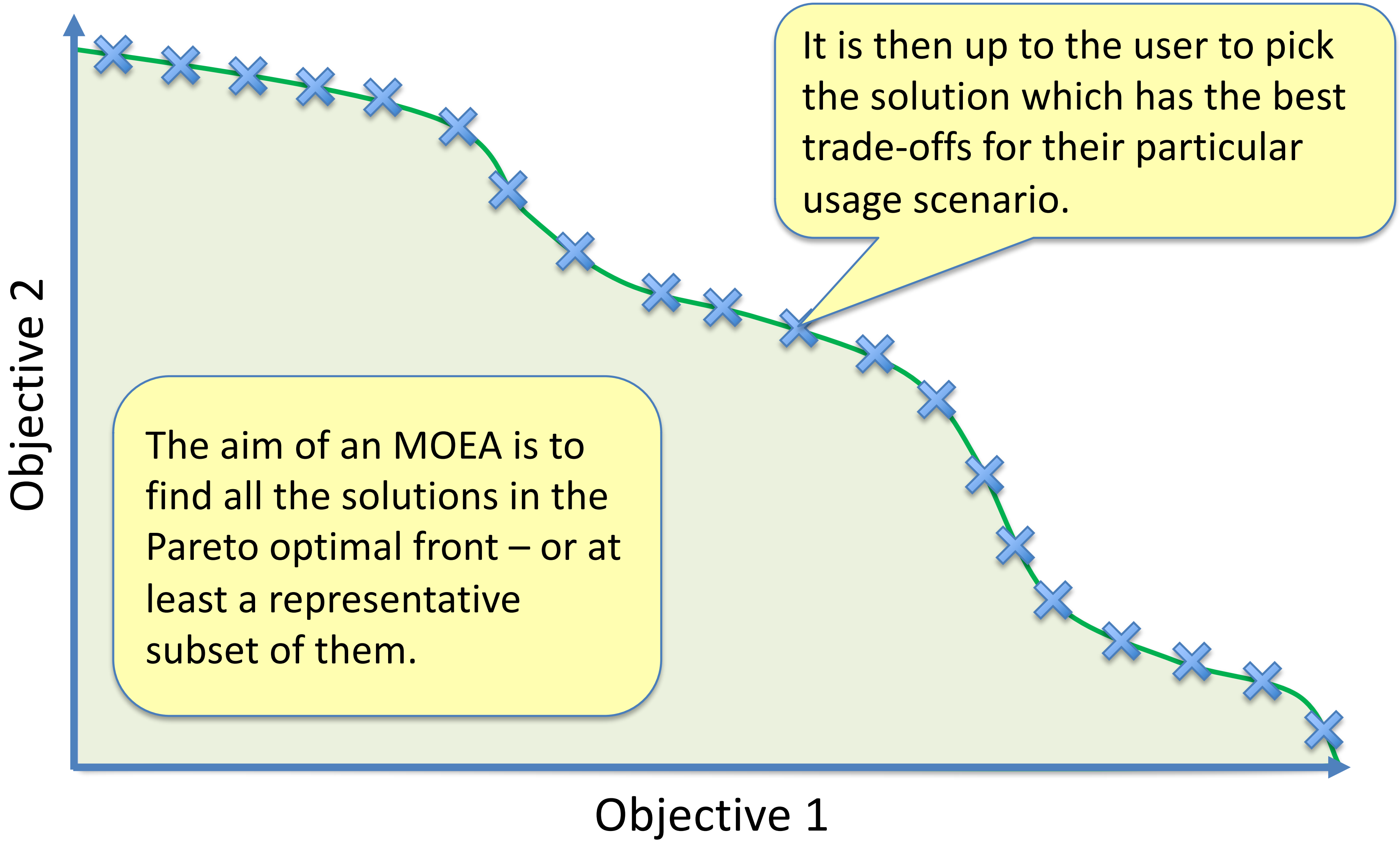
Objective Space



Objective Space

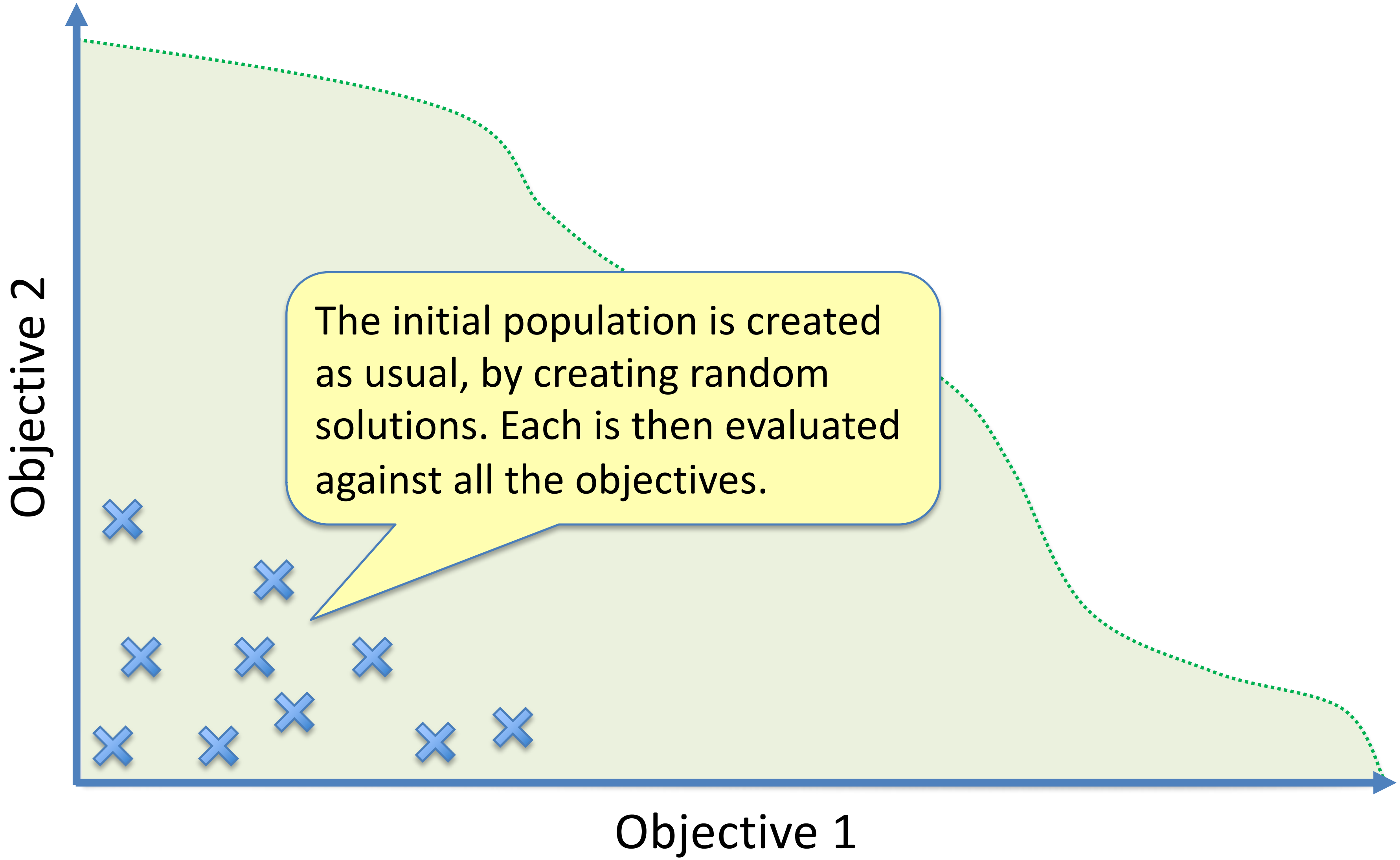


Multiobjective EAs (MOEAs)

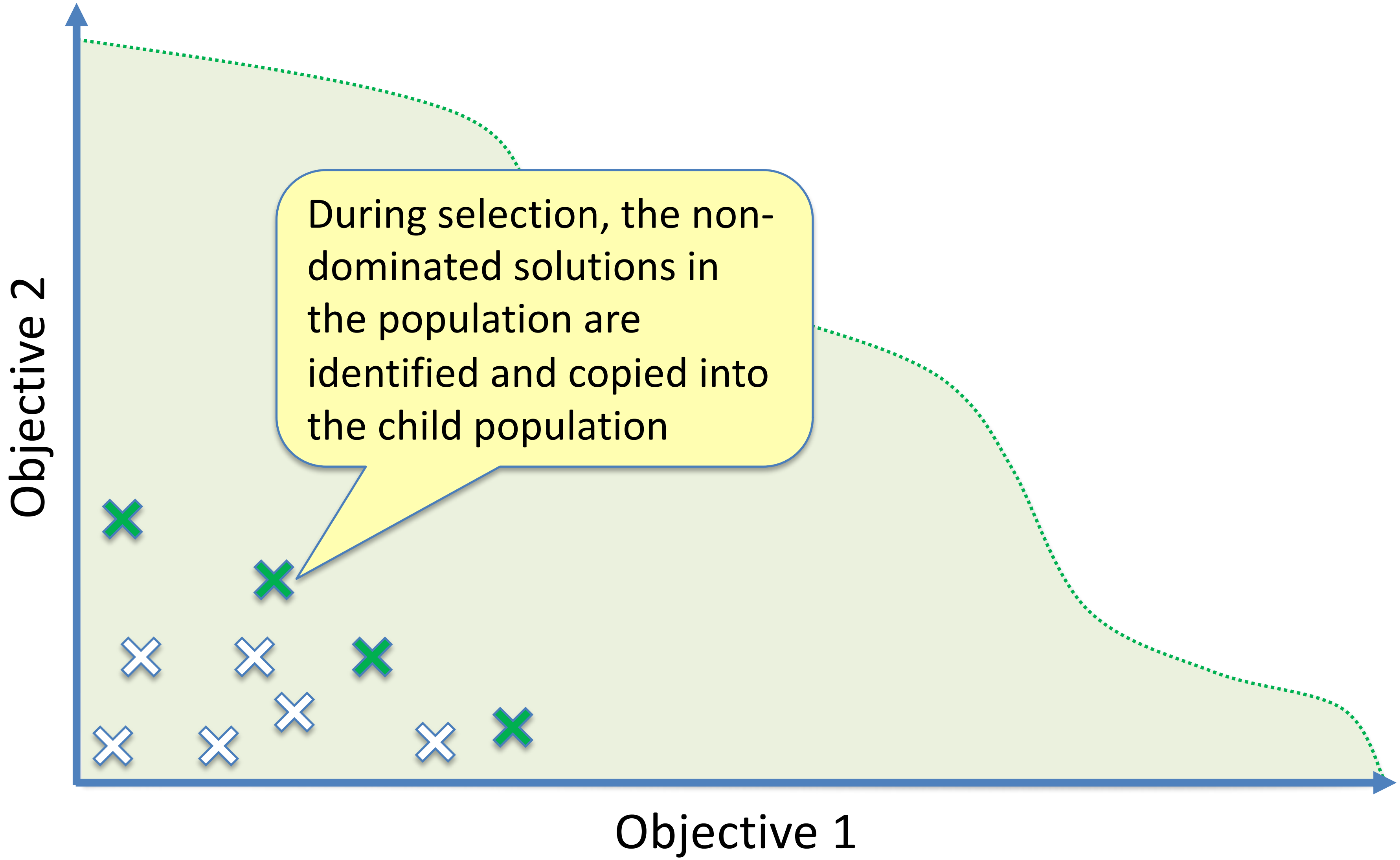


Any Questions?

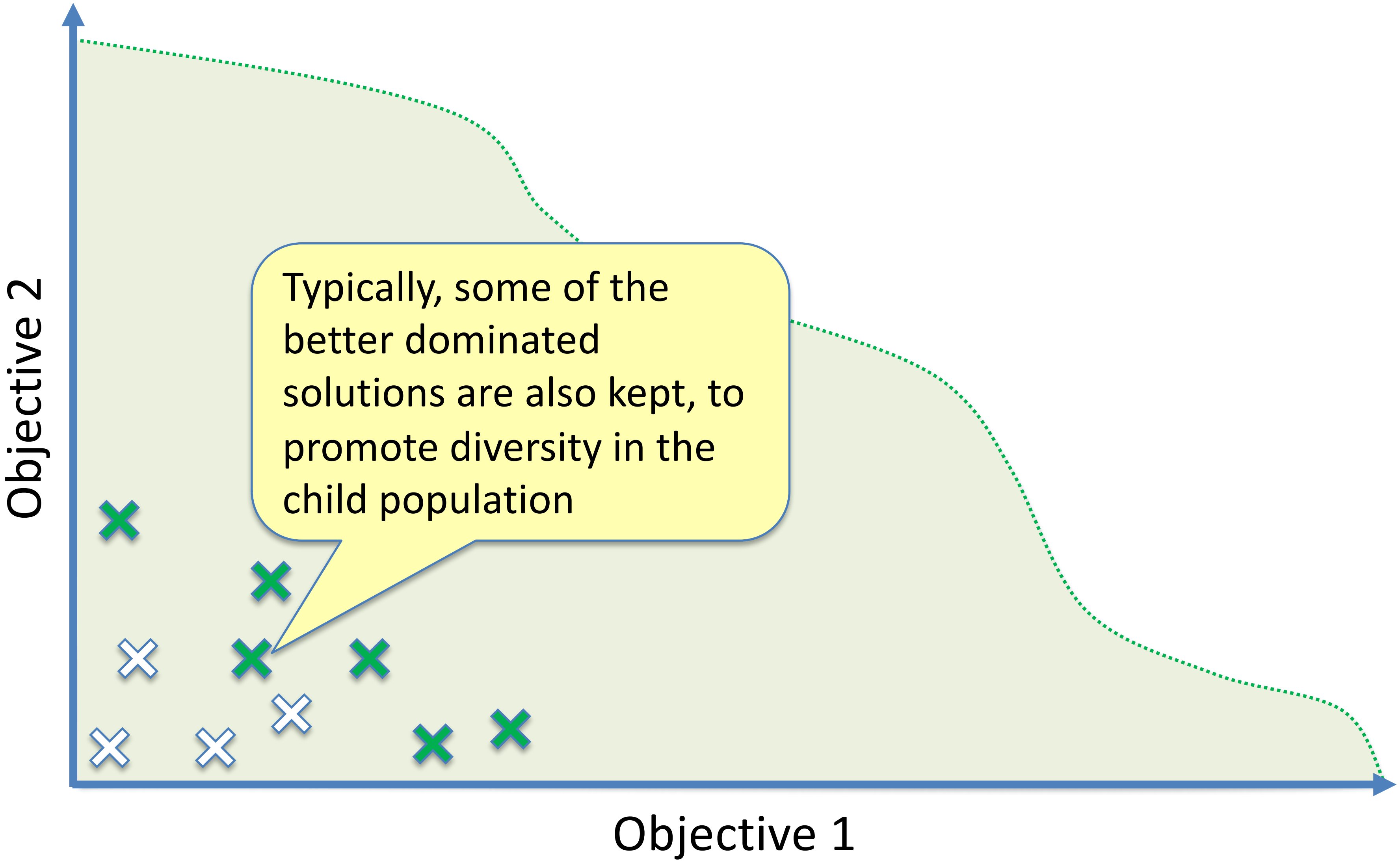
Multiobjective EAs (MOEAs)



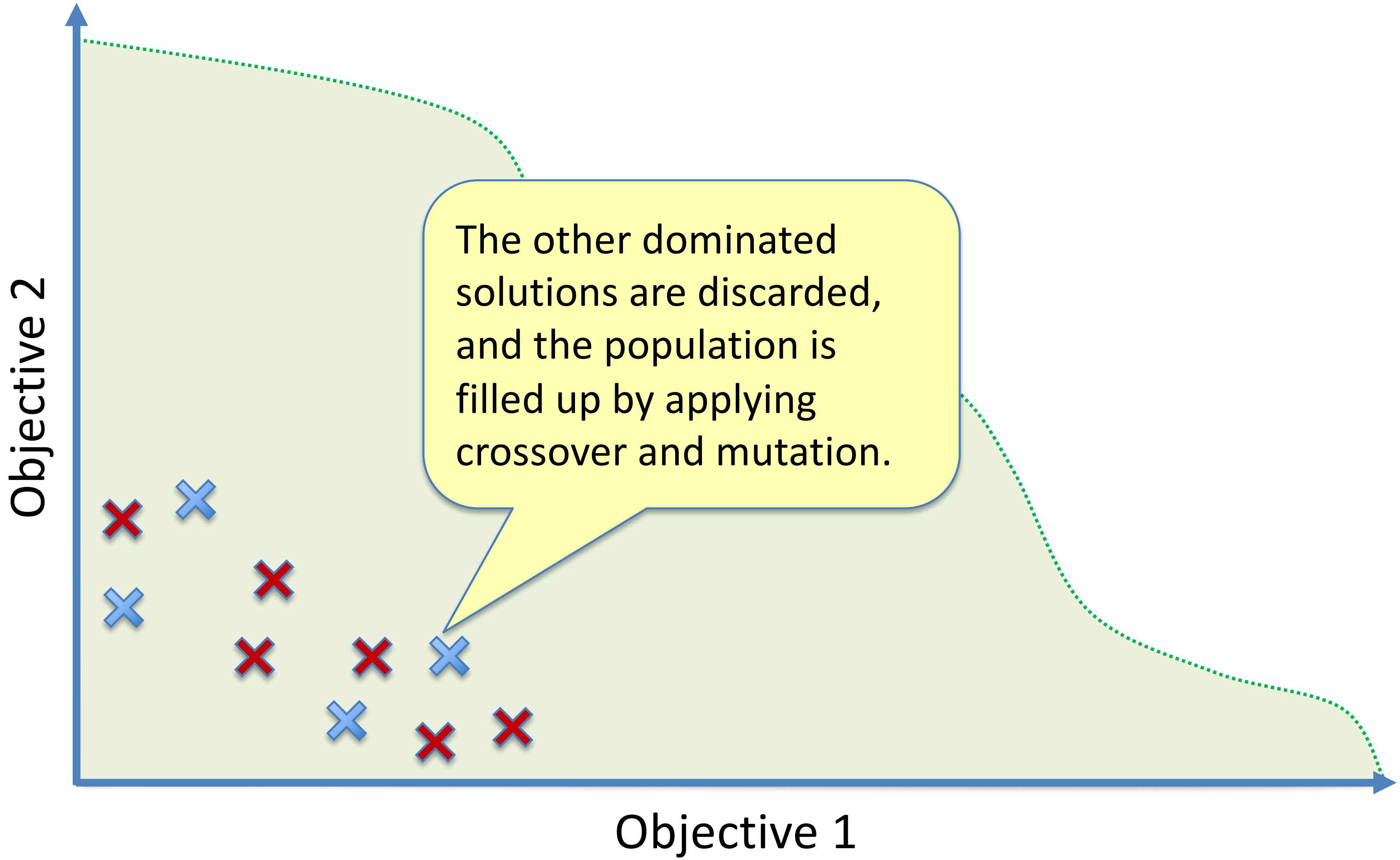
Multiobjective EAs (MOEAs)



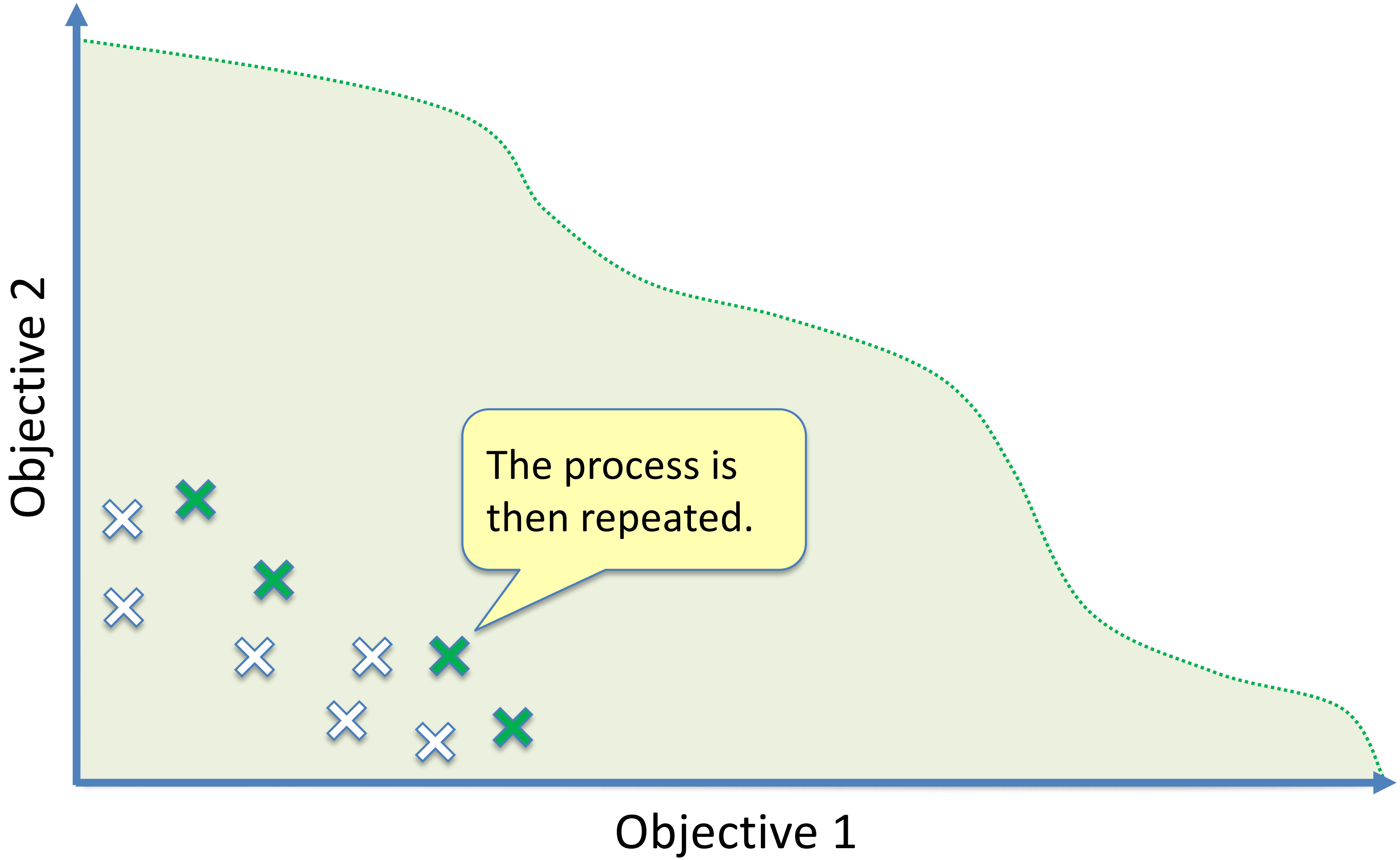
Multiobjective EAs (MOEAs)



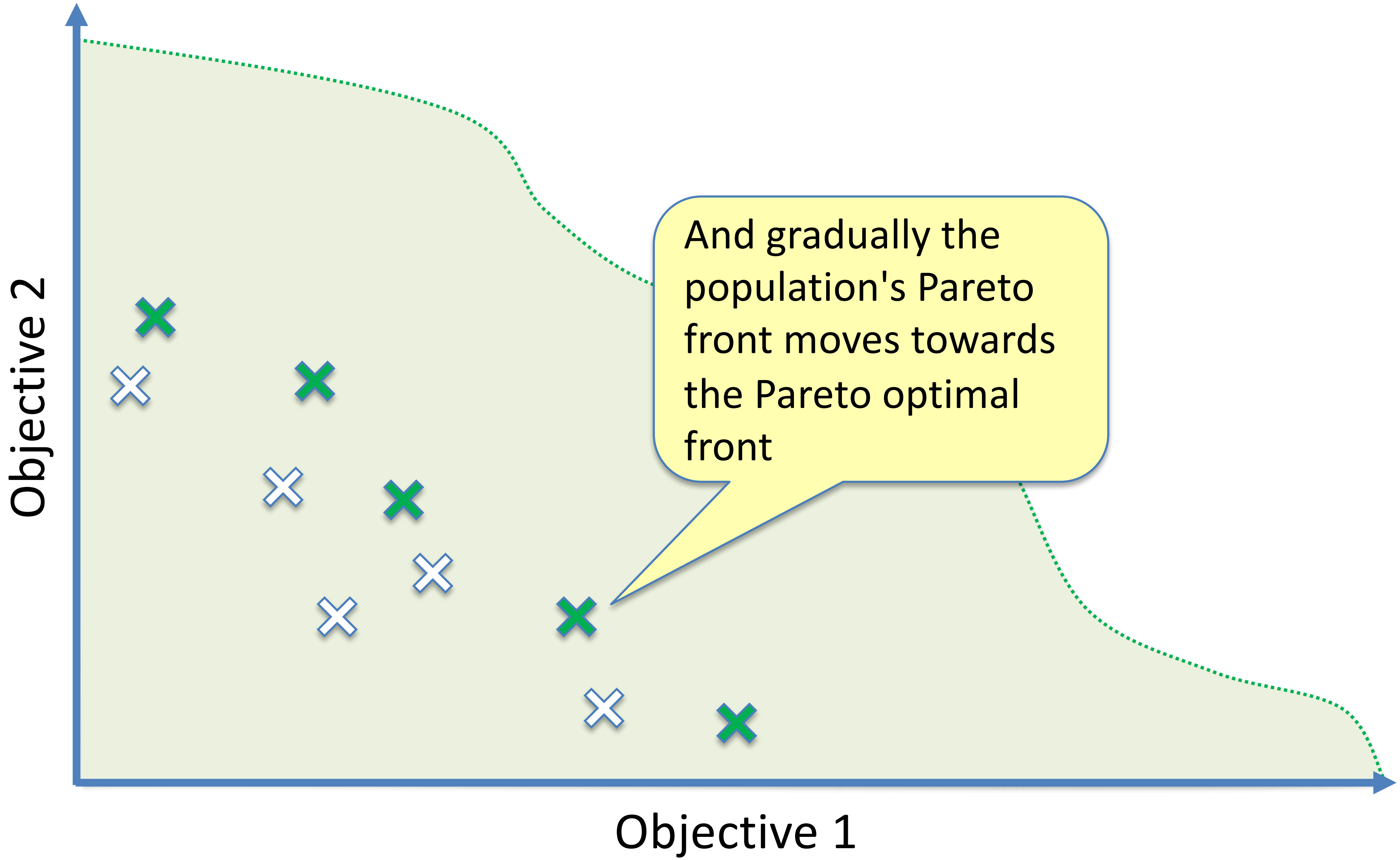
Multiobjective EAs (MOEAs)



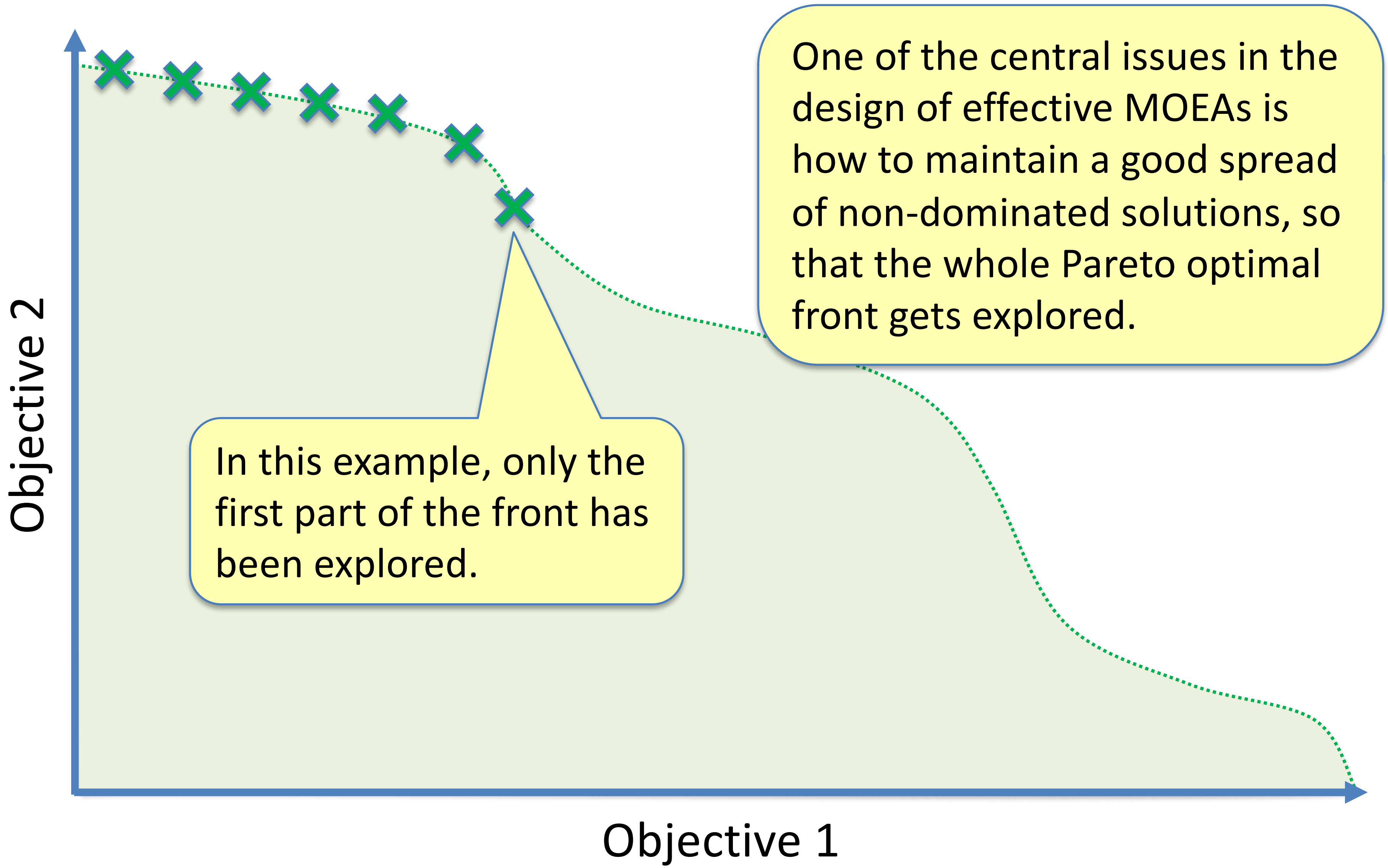
Multiobjective EAs (MOEAs)



Multiobjective EAs (MOEAs)



Multiobjective EAs (MOEAs)

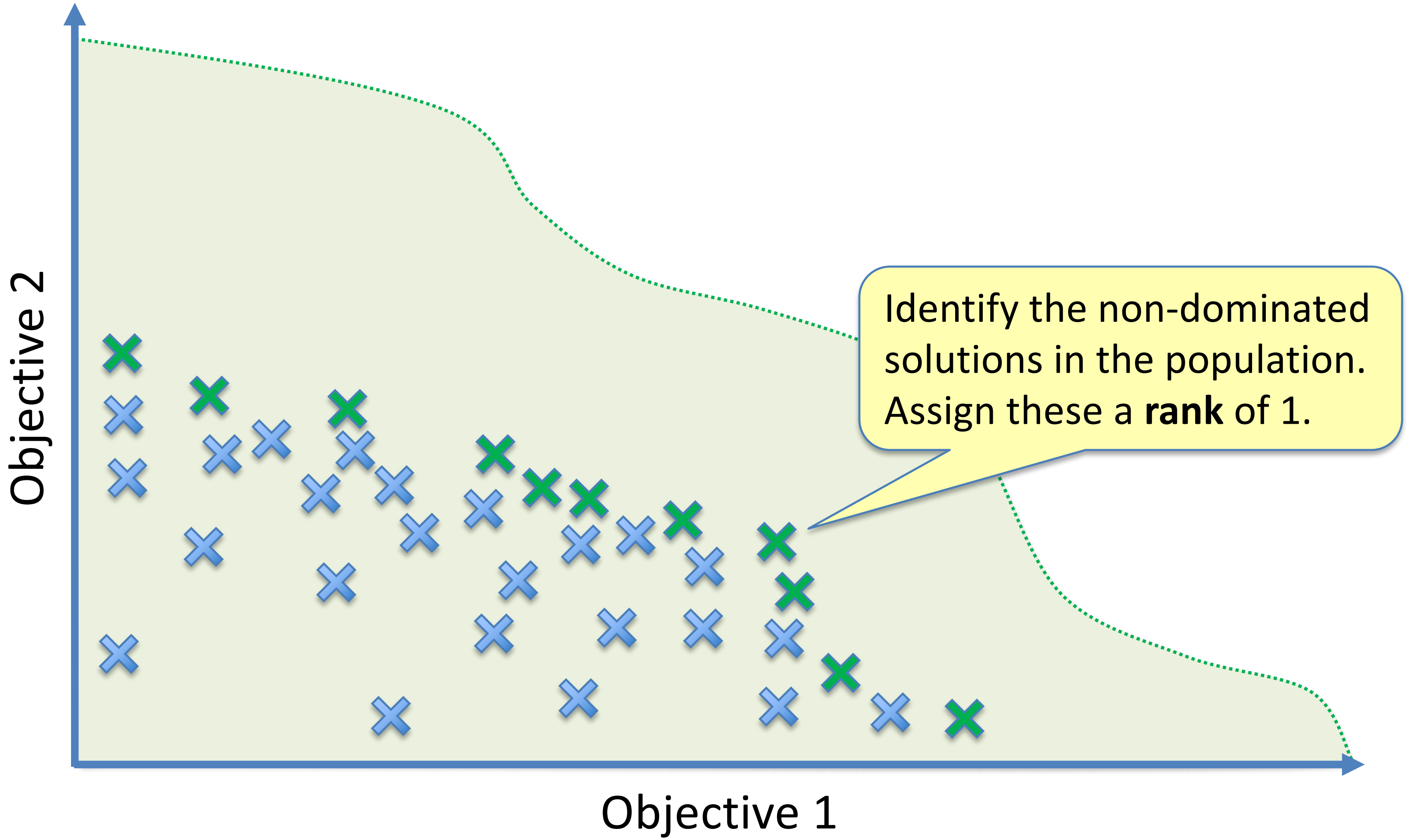


Any Questions?

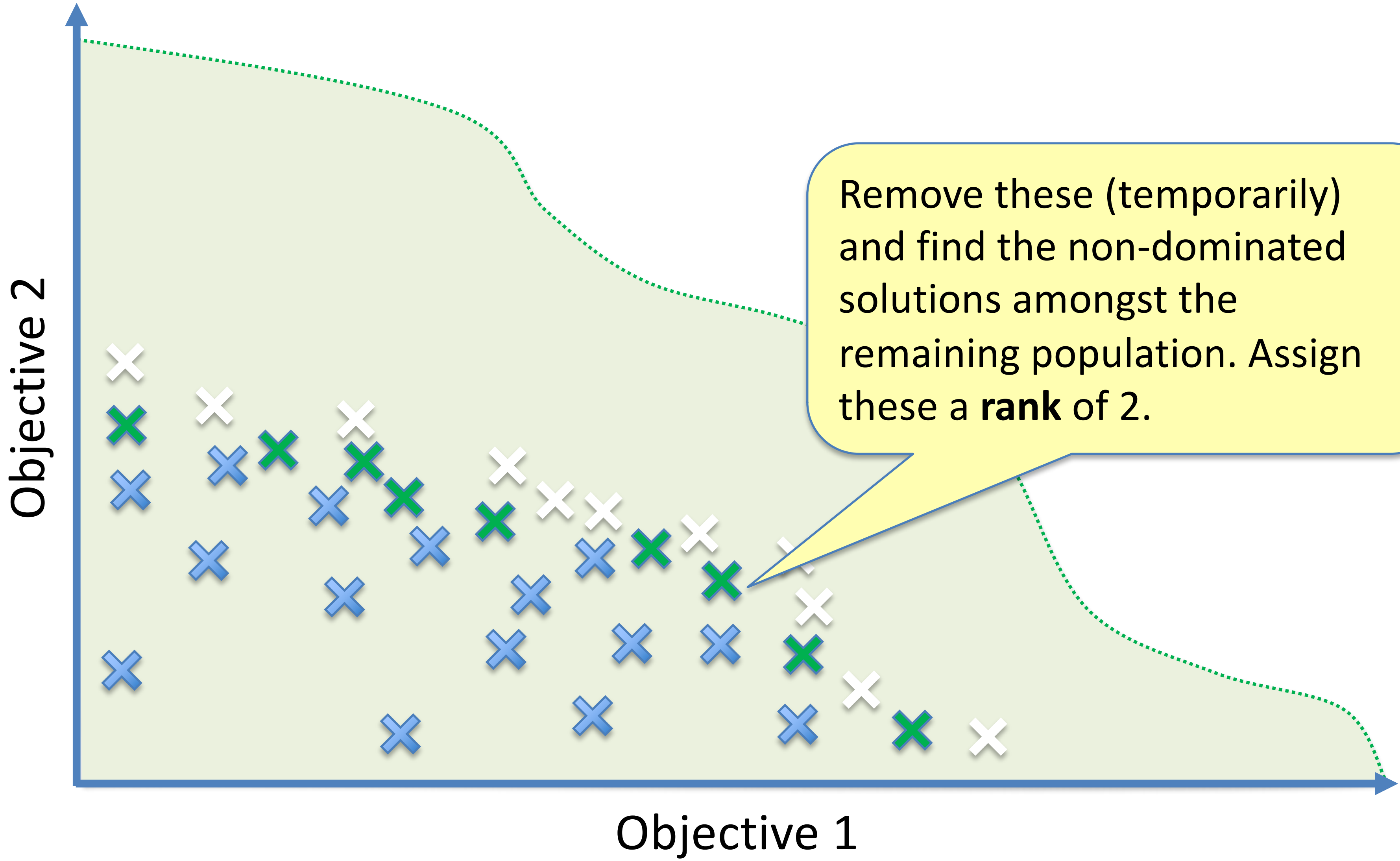
NSGA-II

- There are many MOEAs available, but the best known and most widely used is NSGA-II
  - Which stands for Non-dominated Sorting Genetic Algorithm version 2
  - Published in 2002 by Kalyanmoy Deb et al.
  - ~25,000 citations in Google Scholar (which is a lot)

NSGA-II: Ranking

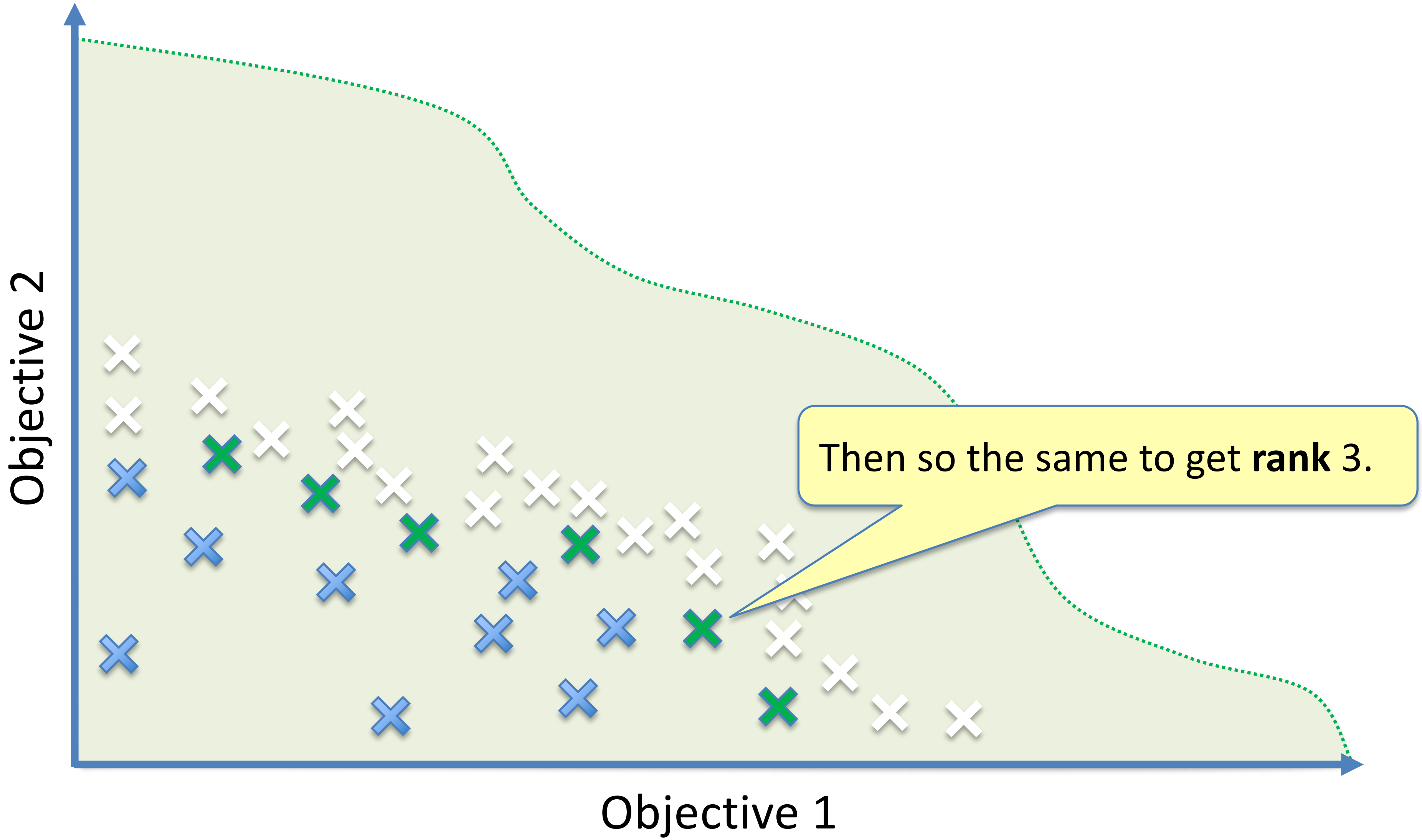


NSGA-II: Ranking

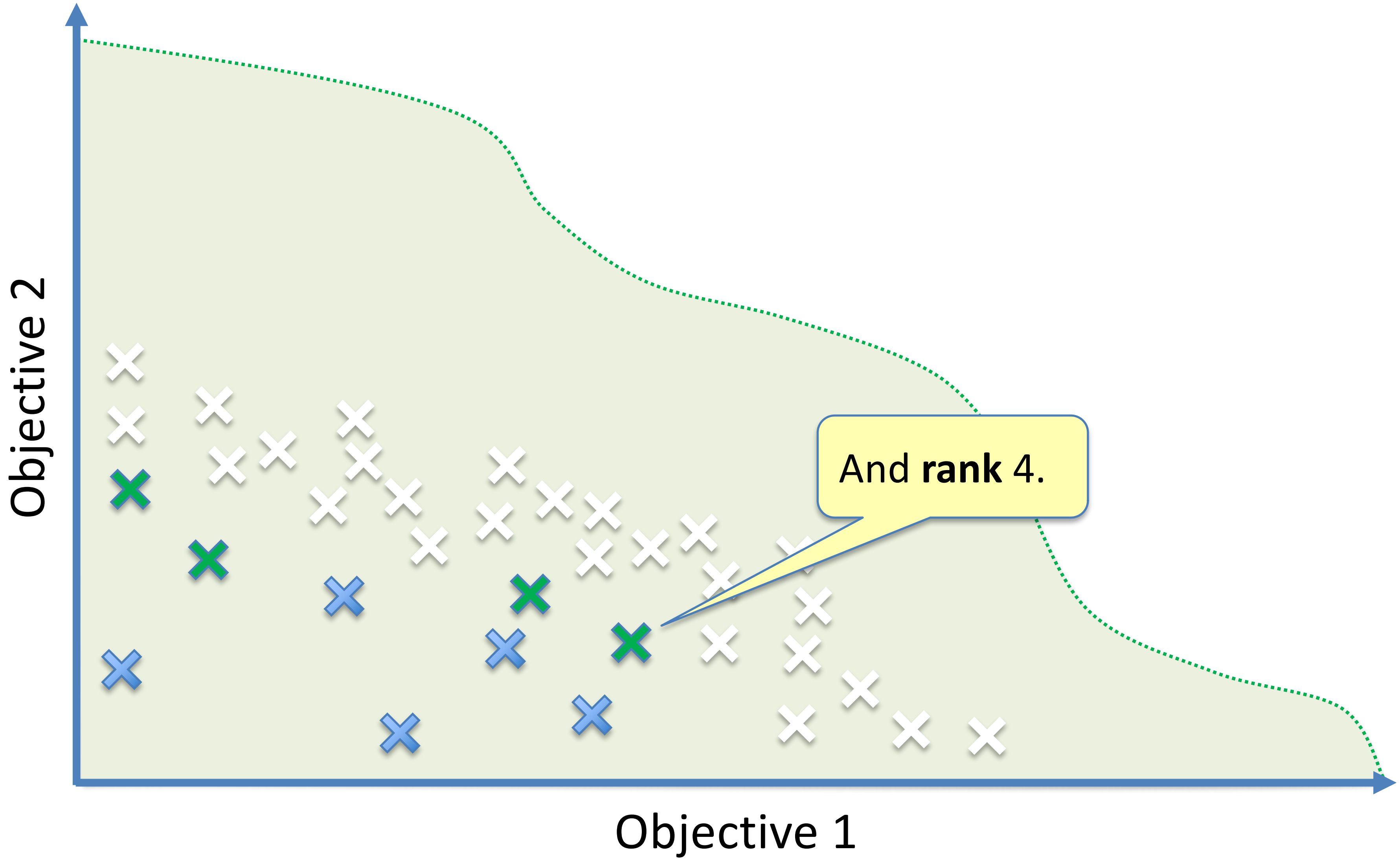


Remove these (temporarily) and find the non-dominated solutions amongst the remaining population. Assign these a **rank** of 2.

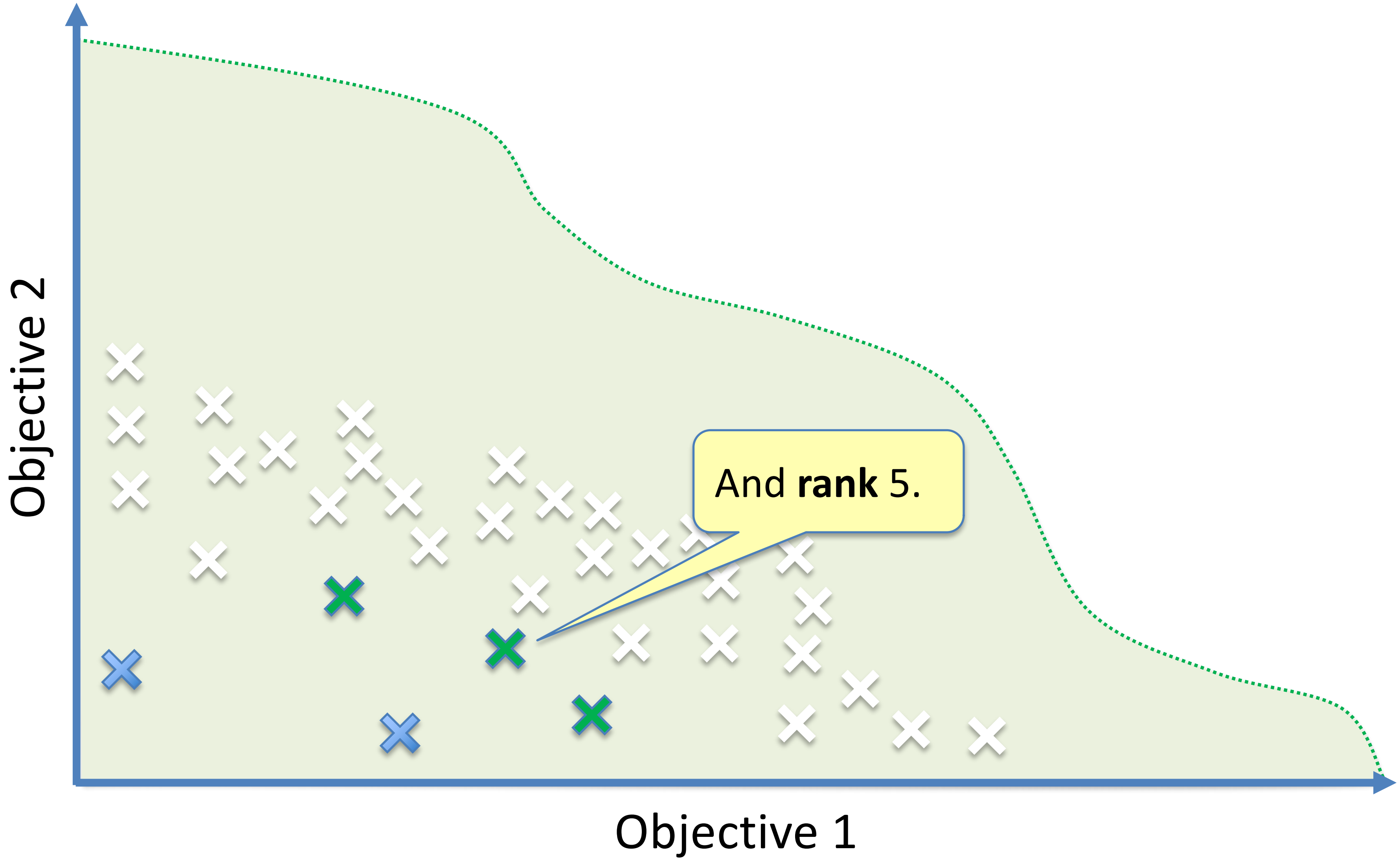
NSGA-II: Ranking



NSGA-II: Ranking

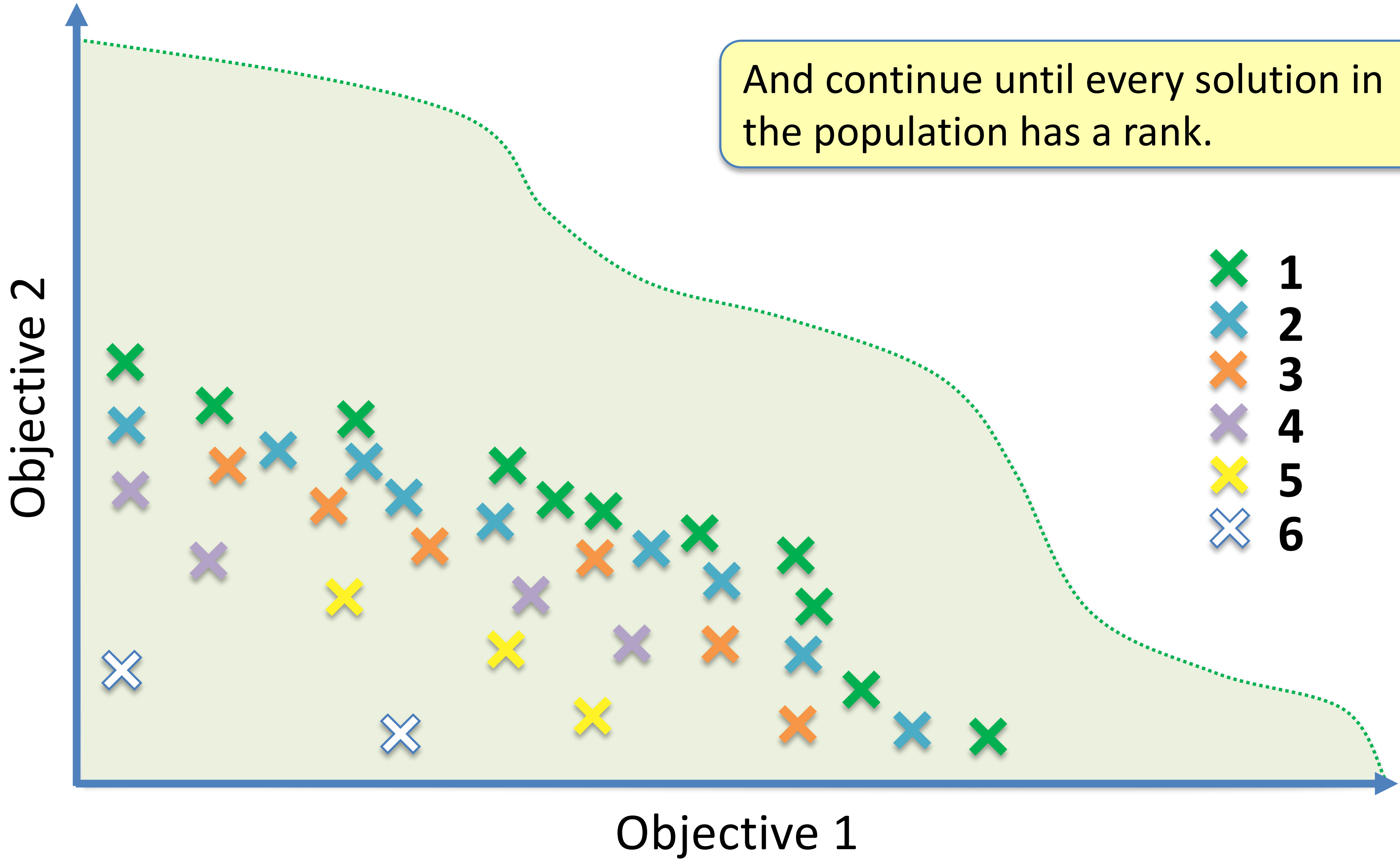


NSGA-II: Ranking



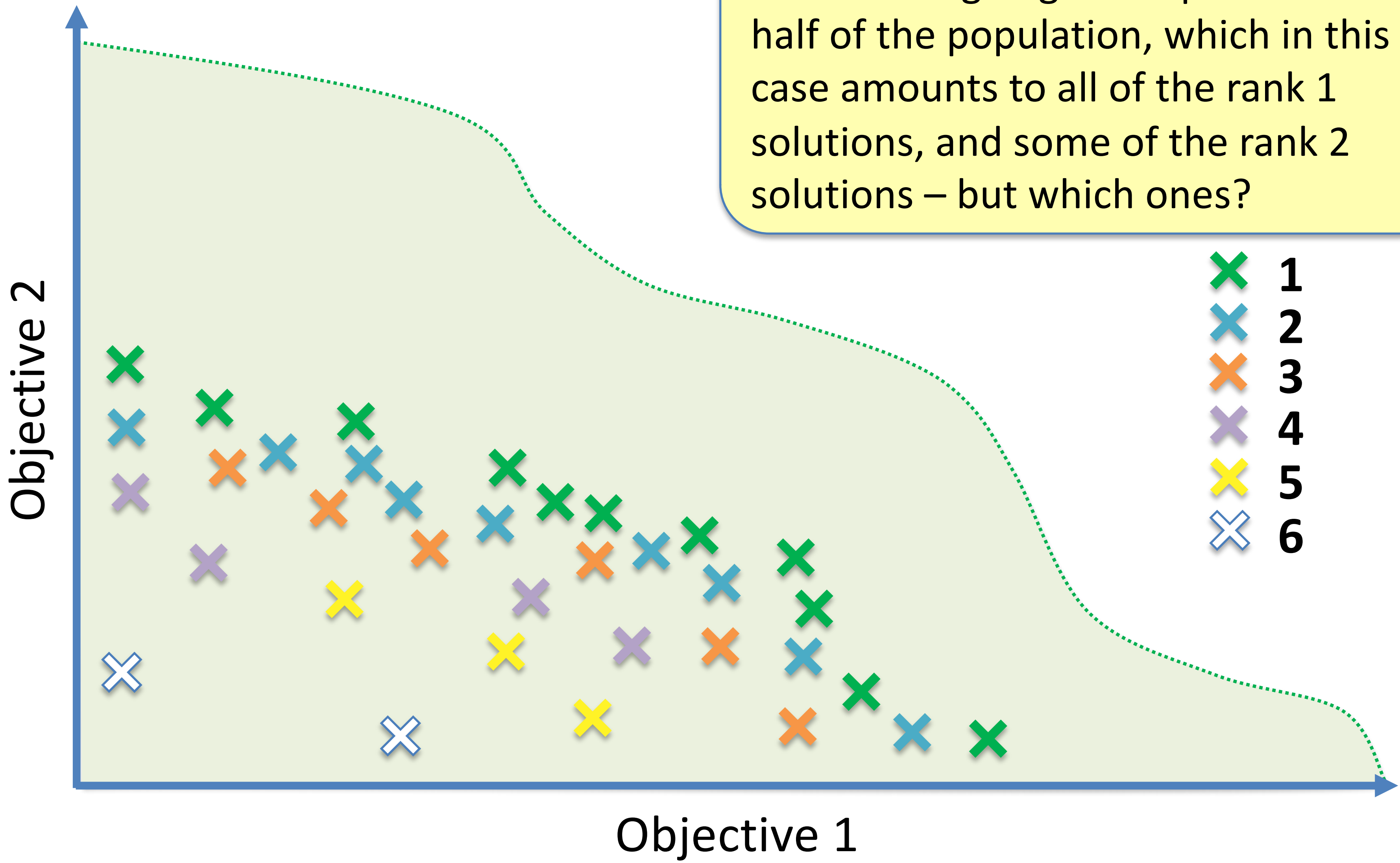
NSGA-II: Ranking

And continue until every solution in the population has a rank.



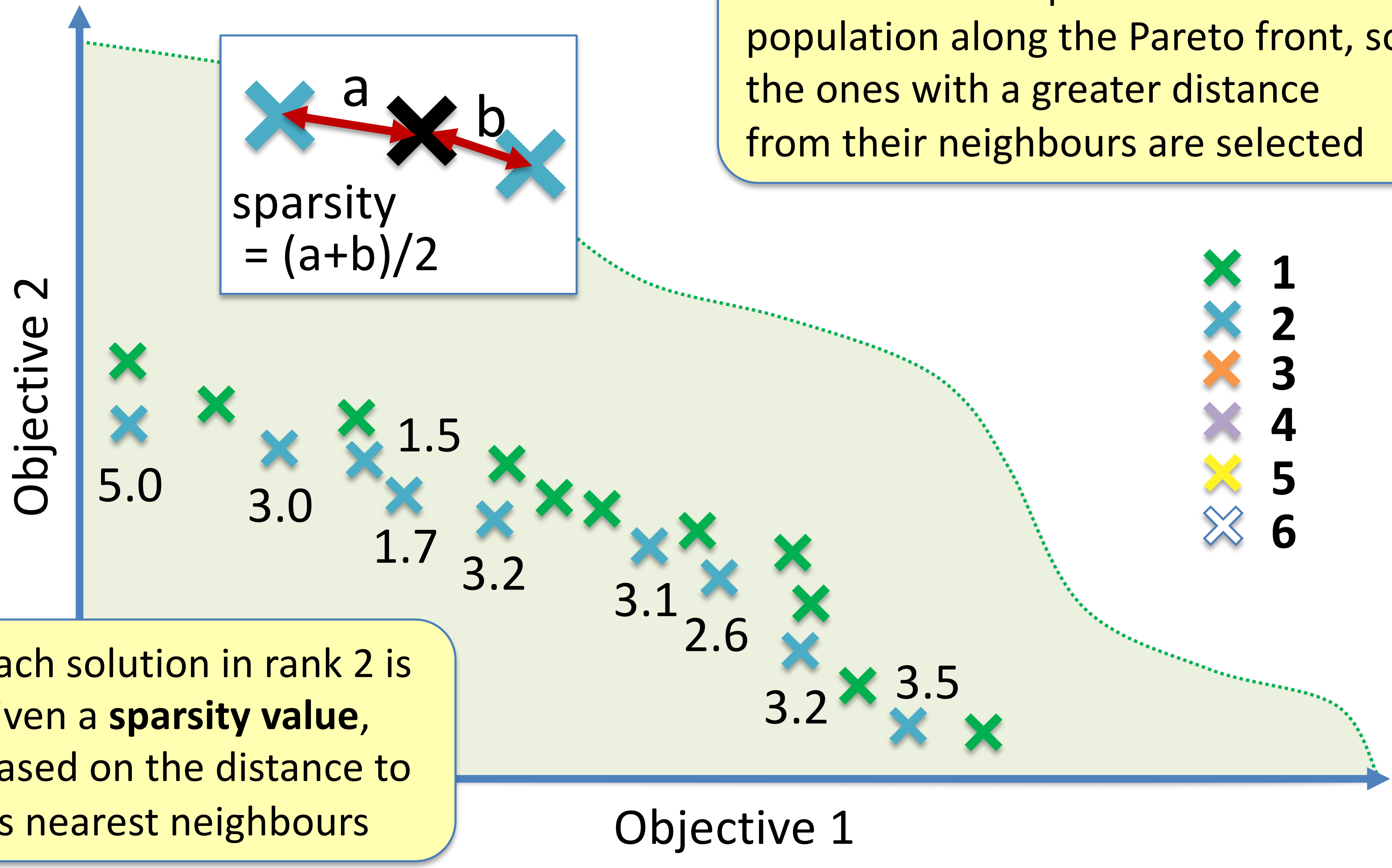
NSGA-II: Selection

We're then going to keep the best half of the population, which in this case amounts to all of the rank 1 solutions, and some of the rank 2 solutions – but which ones?



NSGA-II: Sparsity

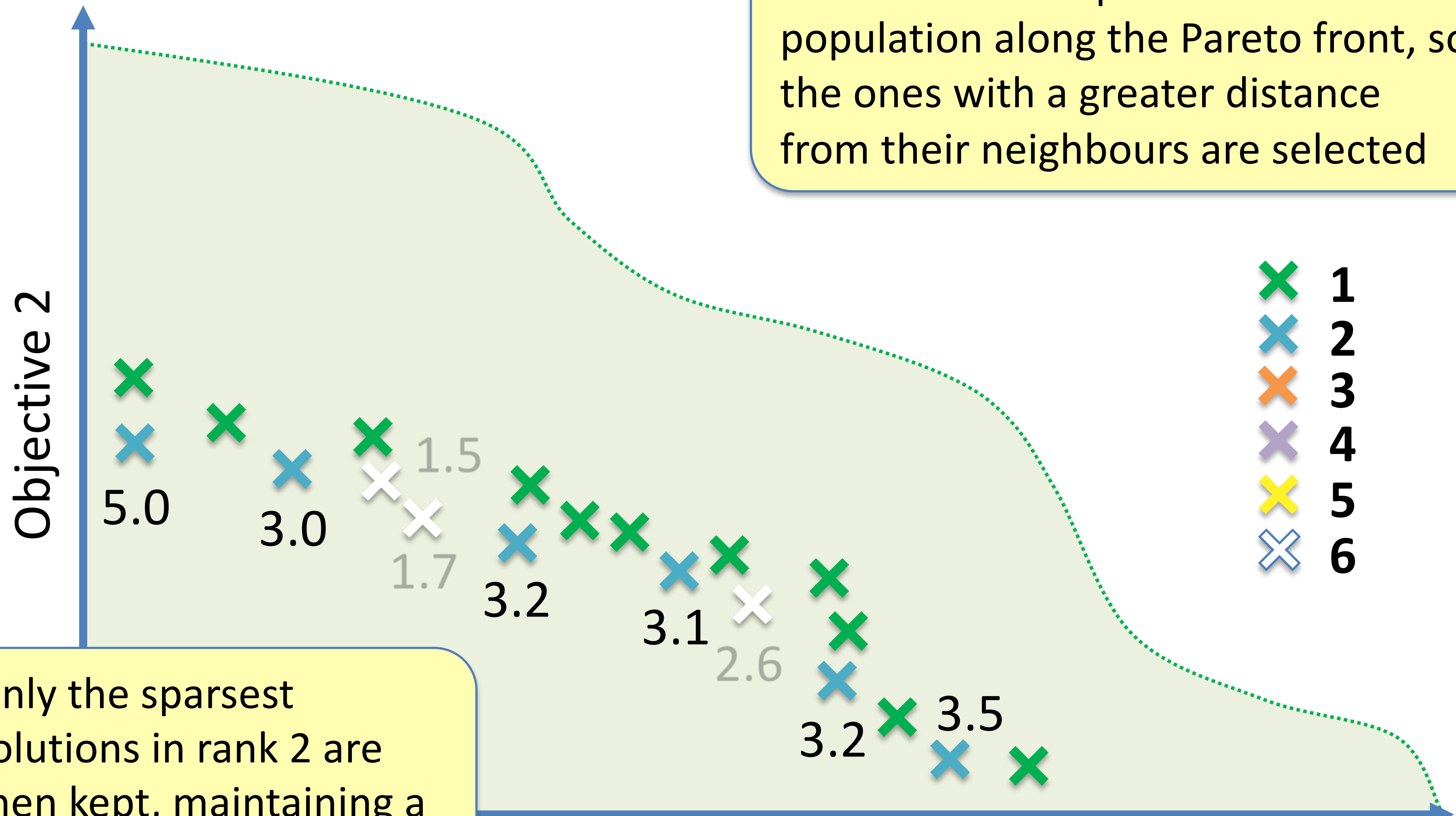
NSGA-II tries to spread the population along the Pareto front, so the ones with a greater distance from their neighbours are selected



Each solution in rank 2 is given a **sparsity value**, based on the distance to its nearest neighbours

NSGA-II: Sparsity

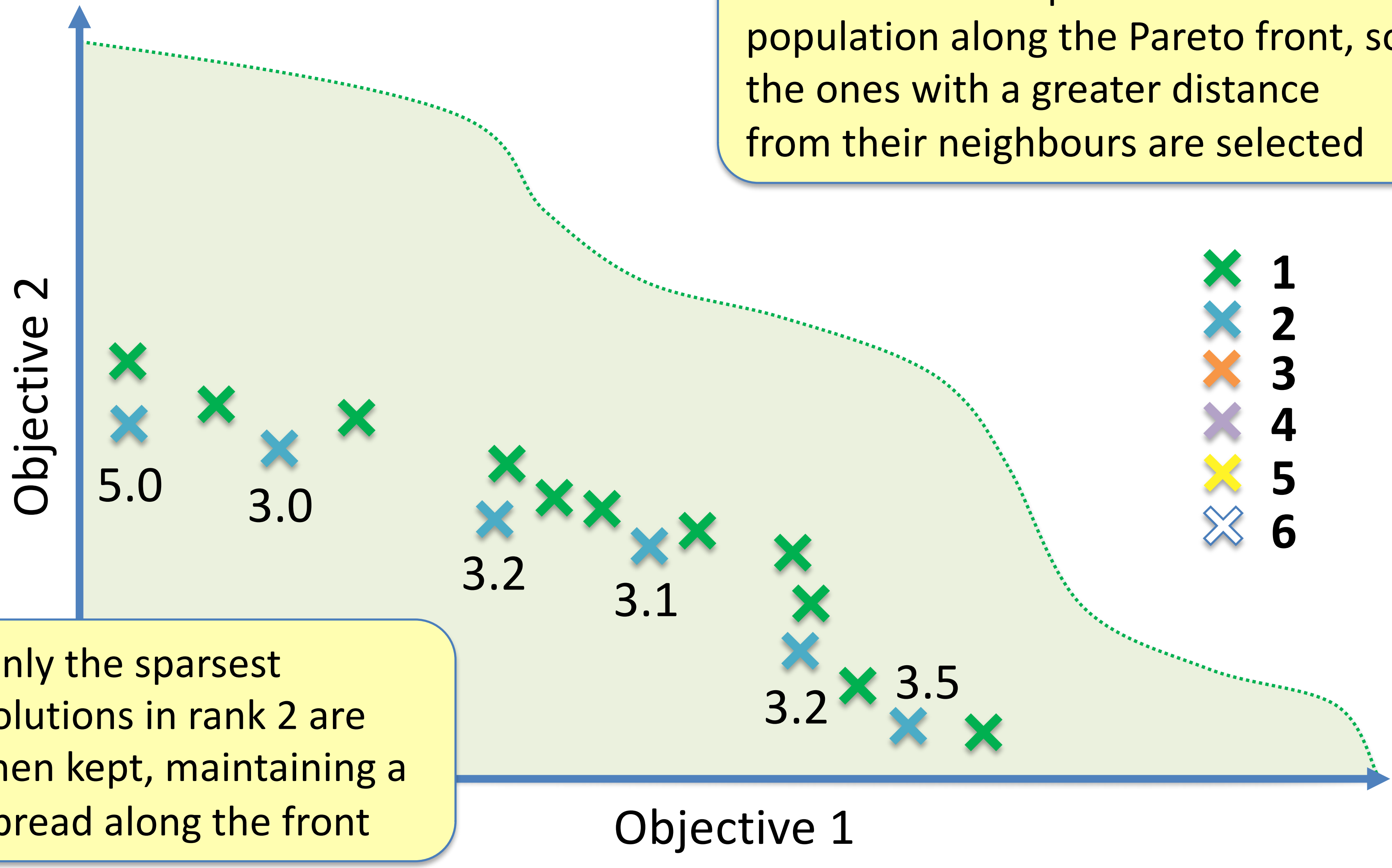
NSGA-II tries to spread the population along the Pareto front, so the ones with a greater distance from their neighbours are selected



Only the sparsest solutions in rank 2 are then kept, maintaining a spread along the front

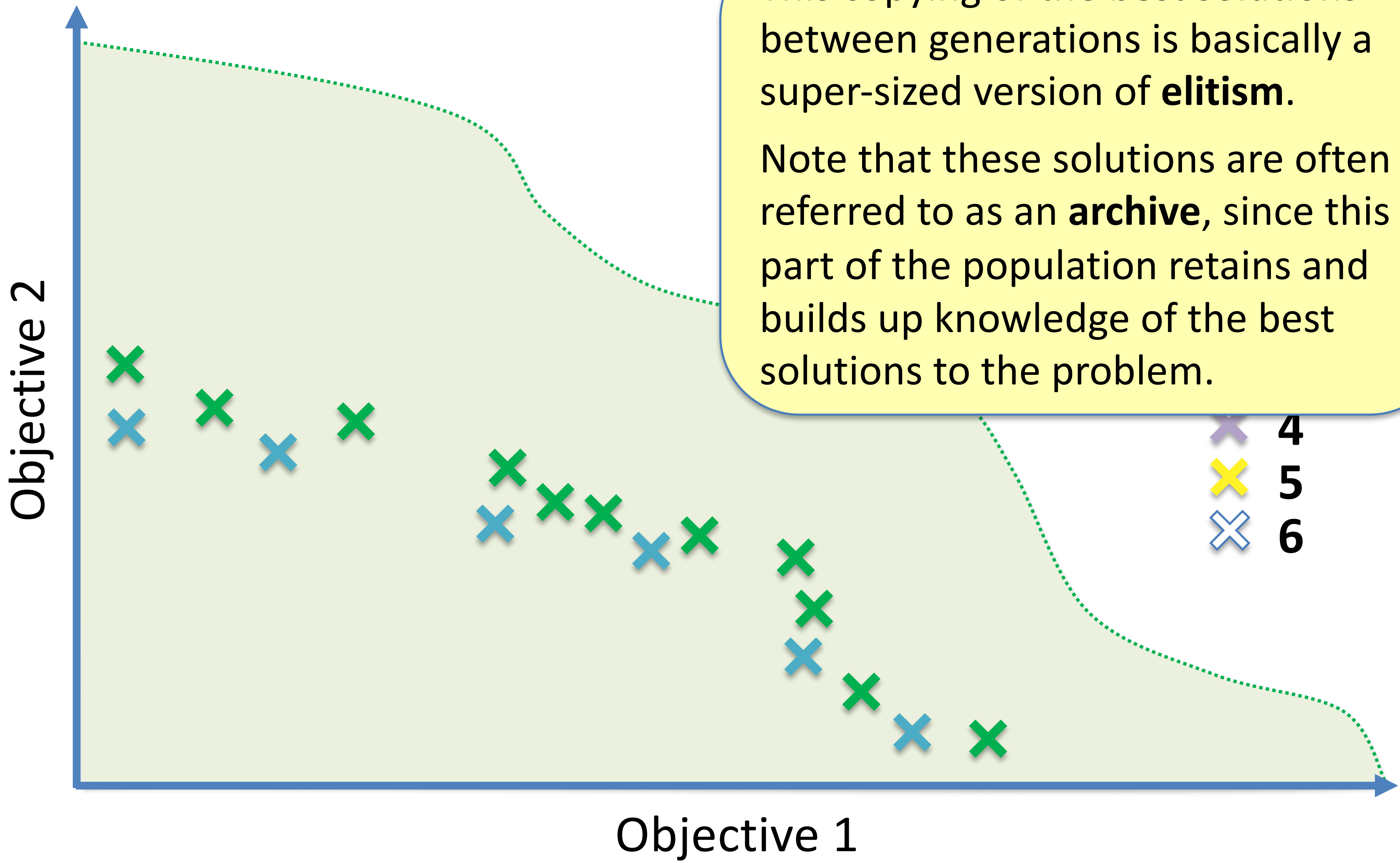
NSGA-II: Sparsity

NSGA-II tries to spread the population along the Pareto front, so the ones with a greater distance from their neighbours are selected



Only the sparsest solutions in rank 2 are then kept, maintaining a spread along the front

NSGA-II: Sparsity

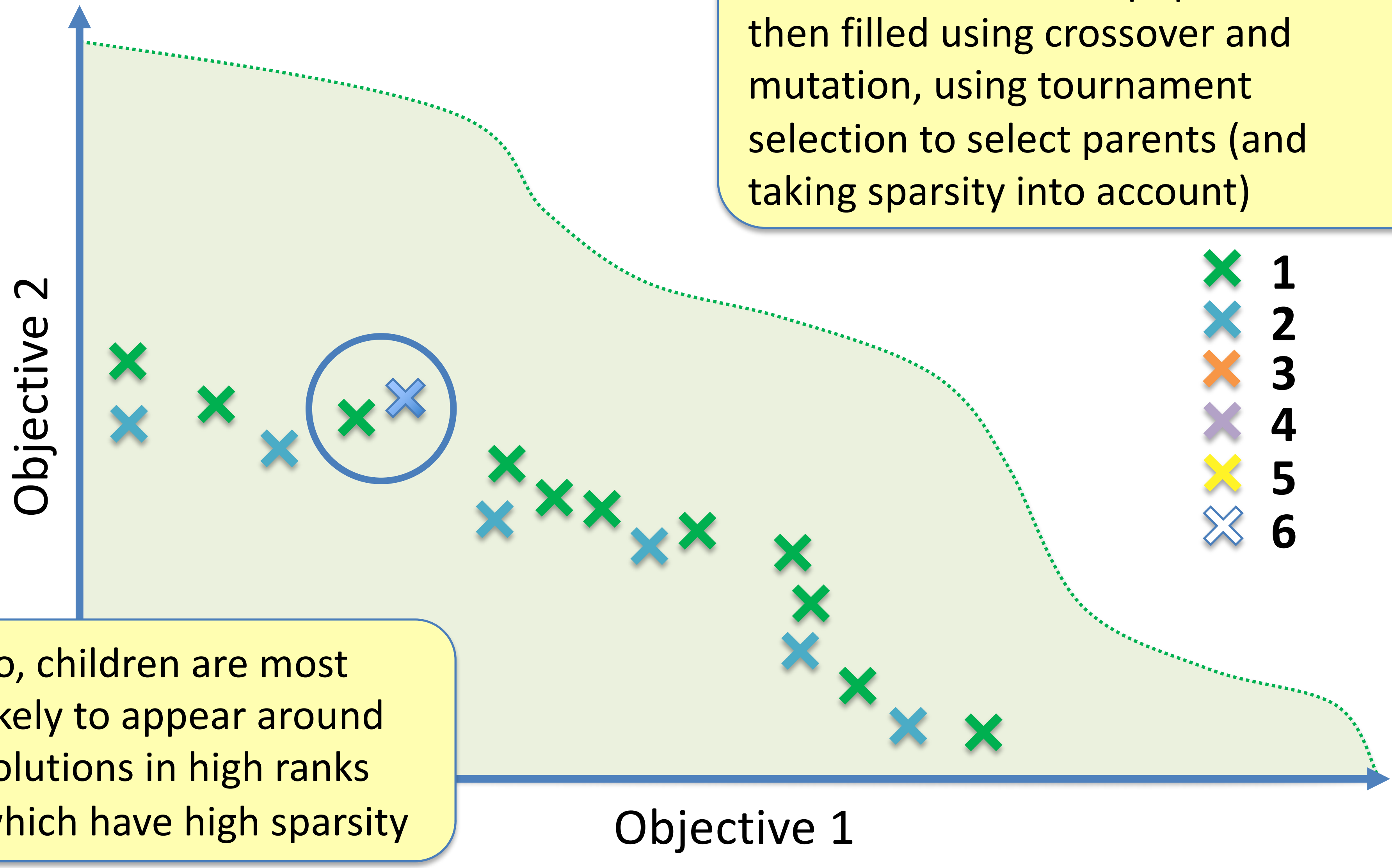


This copying of the best solutions between generations is basically a super-sized version of **elitism**.

Note that these solutions are often referred to as an **archive**, since this part of the population retains and builds up knowledge of the best solutions to the problem.

NSGA-II: Reproduction

The other half of the population is then filled using crossover and mutation, using tournament selection to select parents (and taking sparsity into account)



So, children are most likely to appear around solutions in high ranks which have high sparsity

NSGA-II

- At each generation:
  - Assign every population member a rank
  - Create an empty child population
  - For each rank, from first to last:
    - If adding all solutions in the rank to the child population will result in it being <=50% full
      - then copy all solutions in the rank to the child population
    - Else
      - Sort the rank into a list ordered by decreasing sparsity
      - Copy across the solutions from the top of this list
  - Fill the other half of the child population using crossover and mutation from those already copied across

NSGA-II: Pseudocode

Algorithm 104 *An Abstract Version of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II)*

$m \leftarrow$ desired population size
$a \leftarrow$ desired archive size ▷ Typically $a = m$

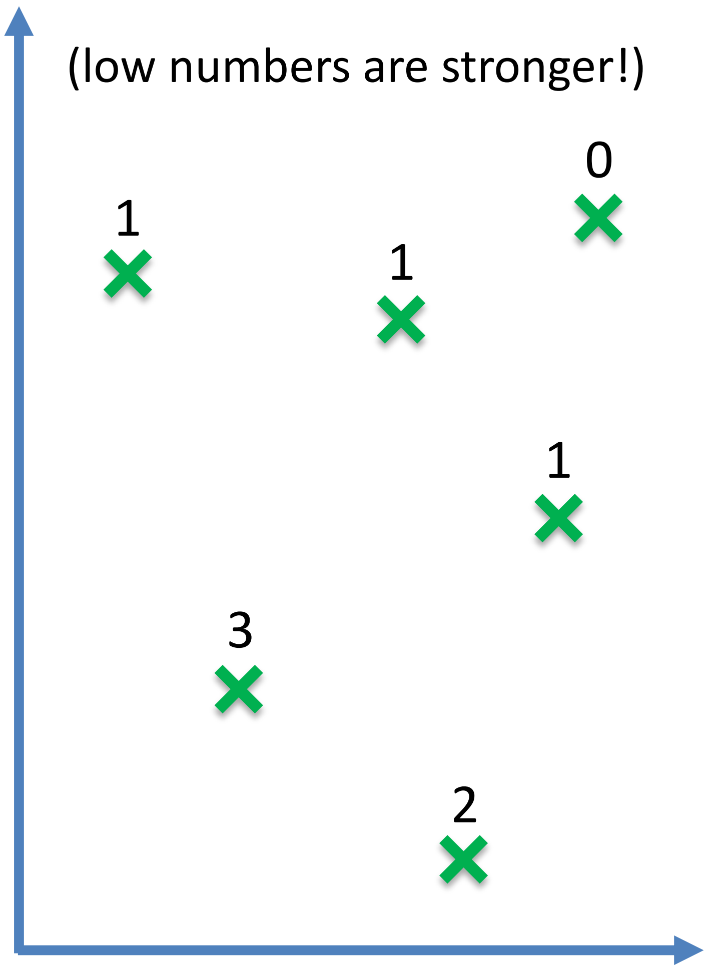
$P \leftarrow \{P_1, ..., P_m\}$ Build Initial Population
$A \leftarrow \{\}$ archive
repeat
    AssessFitness(P) ▷ Compute the objective values for the Pareto front ranks
    $P \leftarrow P \cup A$ ▷ Obviously on the first iteration this has no effect
    $BestFront \leftarrow$ Pareto Front of P
    $R \leftarrow$ Compute Front Ranks of P
    $A \leftarrow \{\}$
    for each Front Rank $R_i \in R$ **do**
        Compute Sparsities of Individuals in R_i ▷ Just for R_i, no need for others
        if $||A|| + ||R_i|| \geq a$ **then** ▷ This will be our last front rank to load into A
            $A \leftarrow A \cup$ the Sparsest $a - ||A||$ individuals in R_i, breaking ties arbitrarily
            break from the **for** loop
        else
            $A \leftarrow A \cup R_i$ ▷ Just dump it in
    $P \leftarrow$ Breed(A), using Algorithm 103 for selection (typically with tournament size of 2)
until $BestFront$ is the ideal Pareto front or we have run out of time
return $BestFront$

See "Essentials of Metaheuristics", p137-141 for a more thorough discussion of NSGA-II and its implementation

Any Questions?

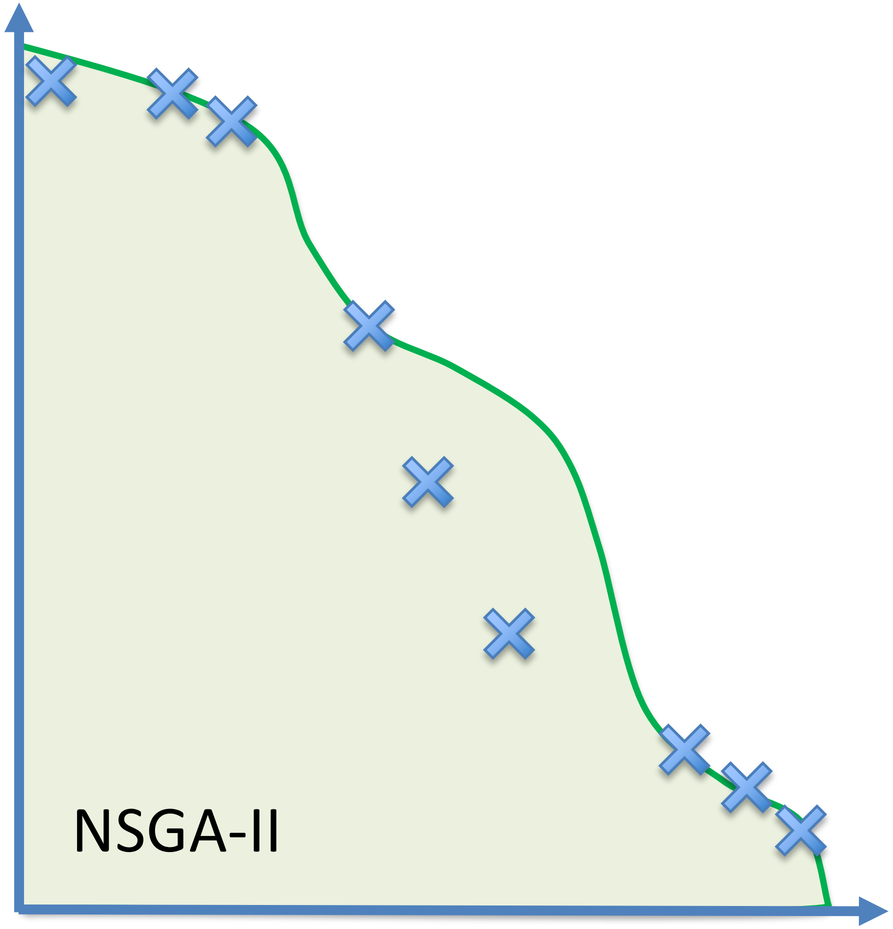
Another MOEA: SPEA2

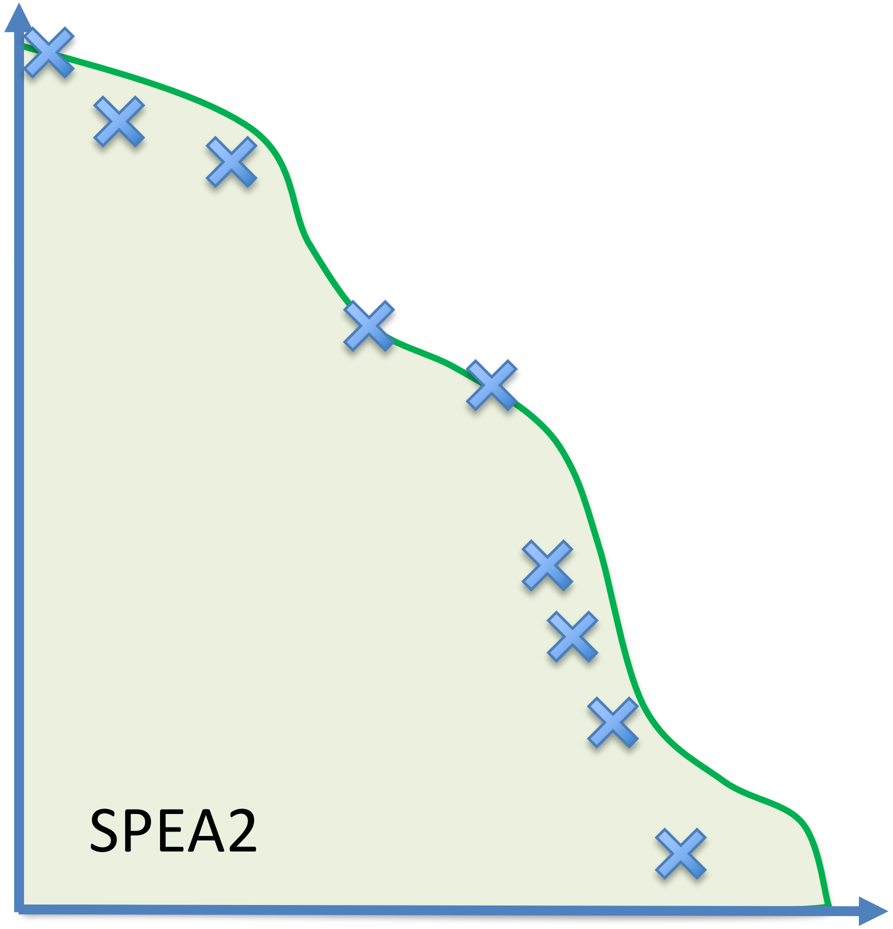
- SPEA2, Strength Pareto Evolutionary Algorithm
  - In the great scheme of things, generally similar to NSGA-II
  - However, solutions are kept based on their **strength**, which measures how many solutions they are dominated by within the population
  - "Essentials of Metaheuristics", p141-146 has a good account of this algorithm



Comparing MOEAs

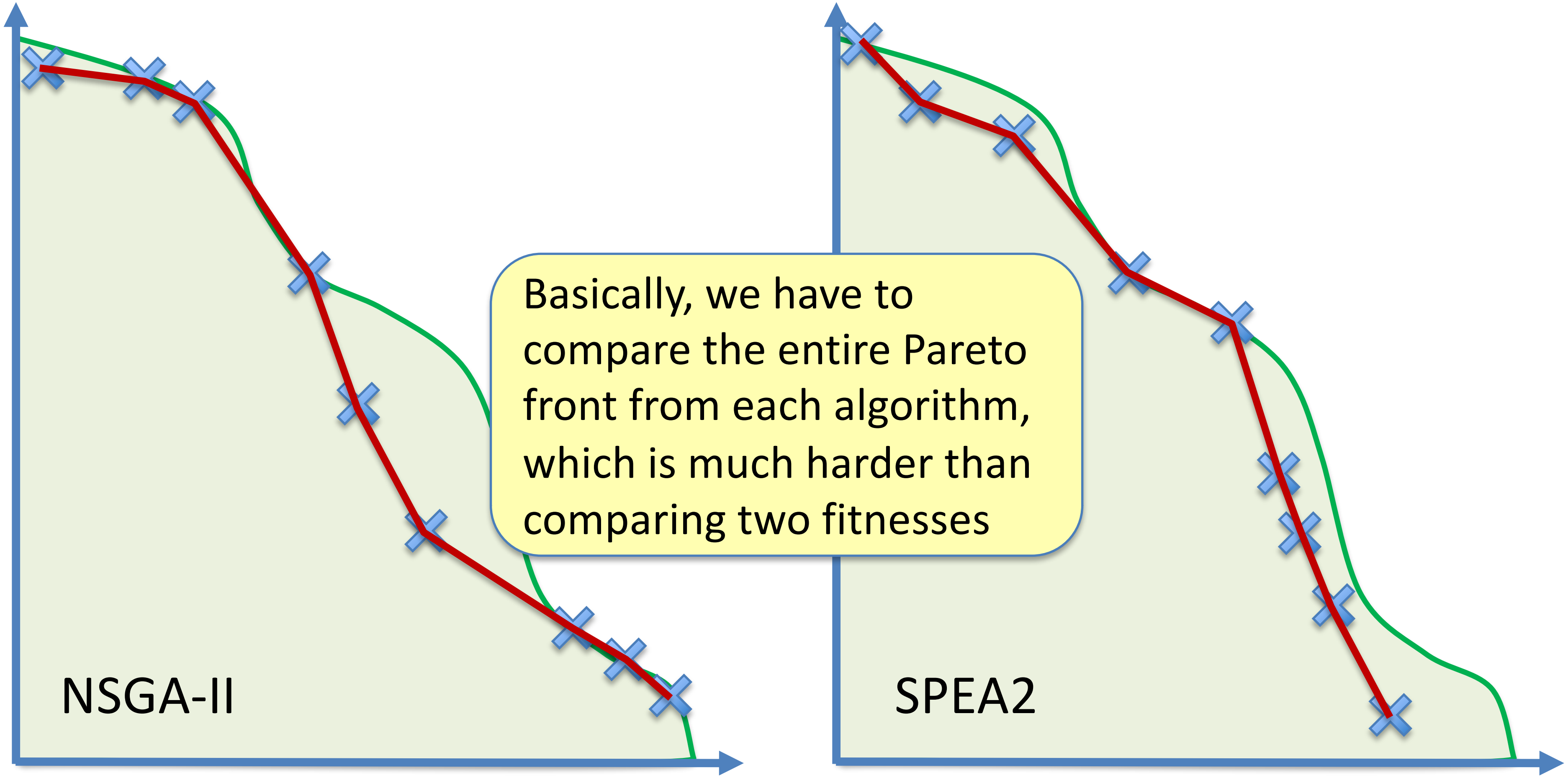
- How do you tell which MOEA performs best on a particular problem?





Comparing MOEAs

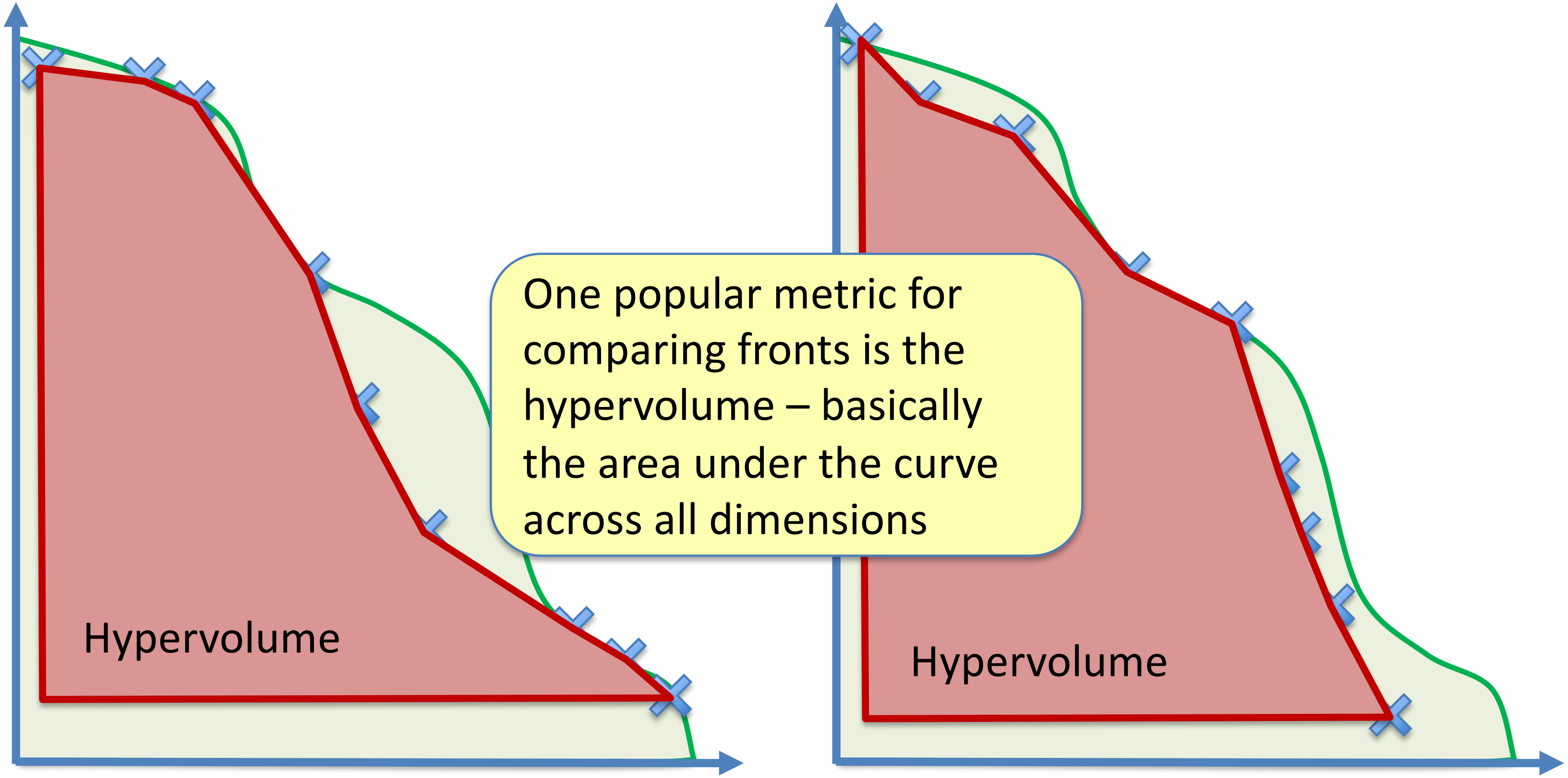
- How do you tell which MOEA performs best on a particular problem?



Basically, we have to compare the entire Pareto front from each algorithm, which is much harder than comparing two fitnesses

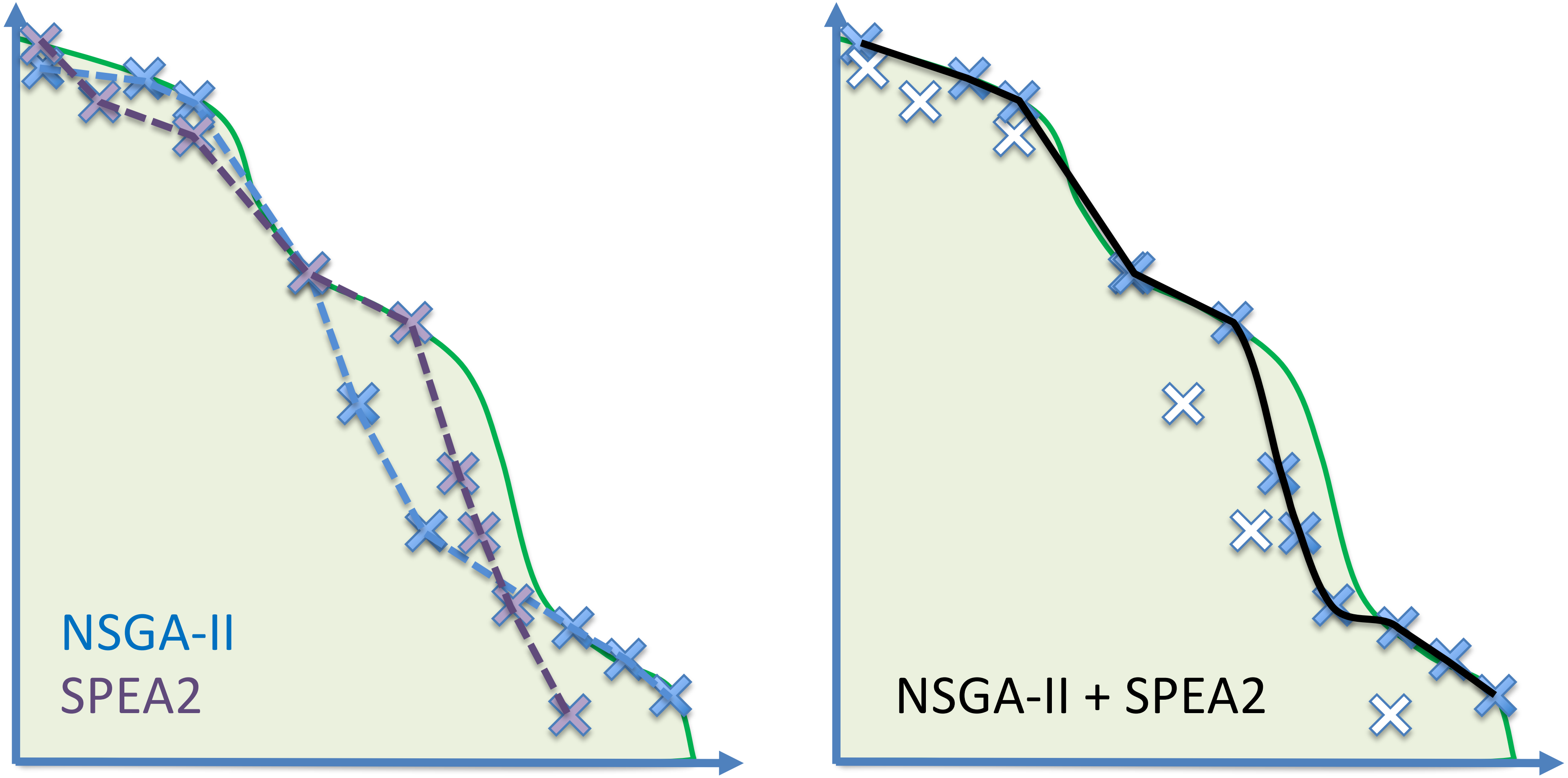
Comparing MOEAs

- How do you tell which MOEA performs best on a particular problem?



Combining MOEAs

- Though probably more useful in practice is to combine fronts from multiple runs or algorithms



Yet Another MOEA: PAES

- This is kind of what **PAES** does
  - Pareto Archived Evolution Strategy
  - Developed by our very own Prof. David Corne
  - It basically carries out multiple runs of an evolution strategy, each run finding a few optimal solutions and avoiding places where previous runs went
  - The fronts from all runs are then assembled together to form a complete Pareto front with a nice spread

Many-Objective Optimisation

- Many real world problems are **many-**objective
  - That is, they have more than 4 objectives
  - As the number of objectives grows, more and more solutions become non-dominated
  - Meaning that dominance-based approaches to selection become ineffective, and the size of the Pareto optimal front becomes enormous
  - So, as the number of objectives increases, traditional MOEAs such as NSGA-II and SPEA2 tend to struggle

Many-Objective Optimisation

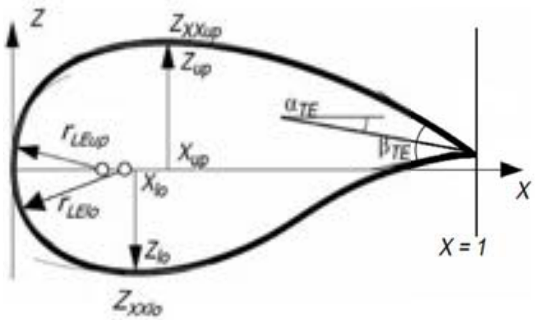
- Many real world problems are many-objective
  - This has promoted a lot of research in the emerging area of **many-objective optimisation**
  - An recent example is NSGA-III, which builds upon NSGA-II and tackles the problem of an expanding objective space by breaking it up into smaller regions
  - I won't go into detail, since these algorithms tend to be quite complicated! For a recent review, see: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.709.4835&rep=rep1&type=pdf

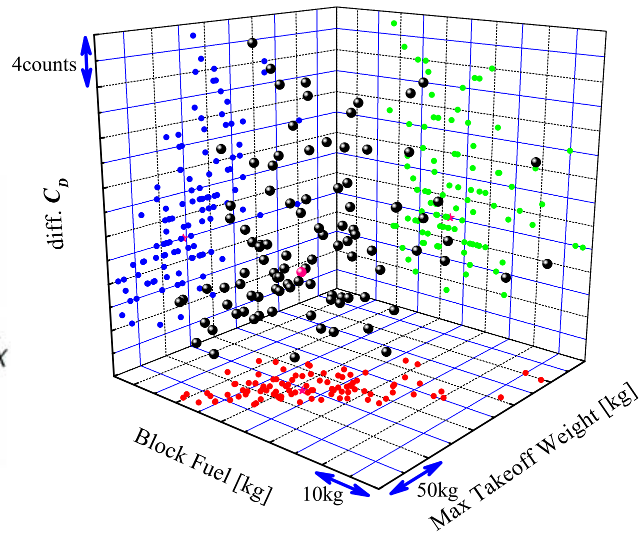
Any Questions?

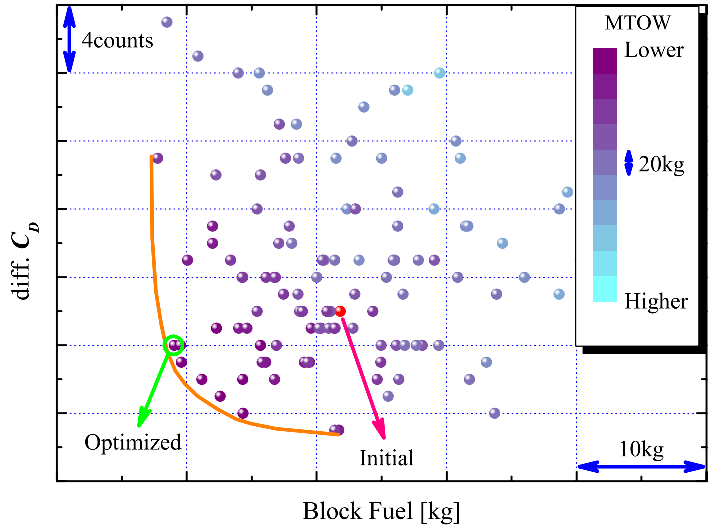
Example: Aircraft Components

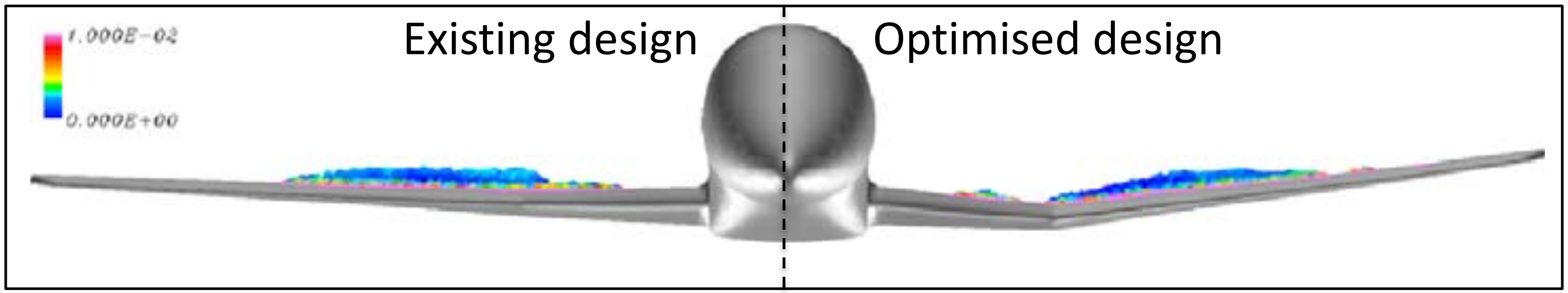
- MOEAs have been widely used to optimise the shapes of components in aircraft, e.g.

Optimising 9 wing design parameters:





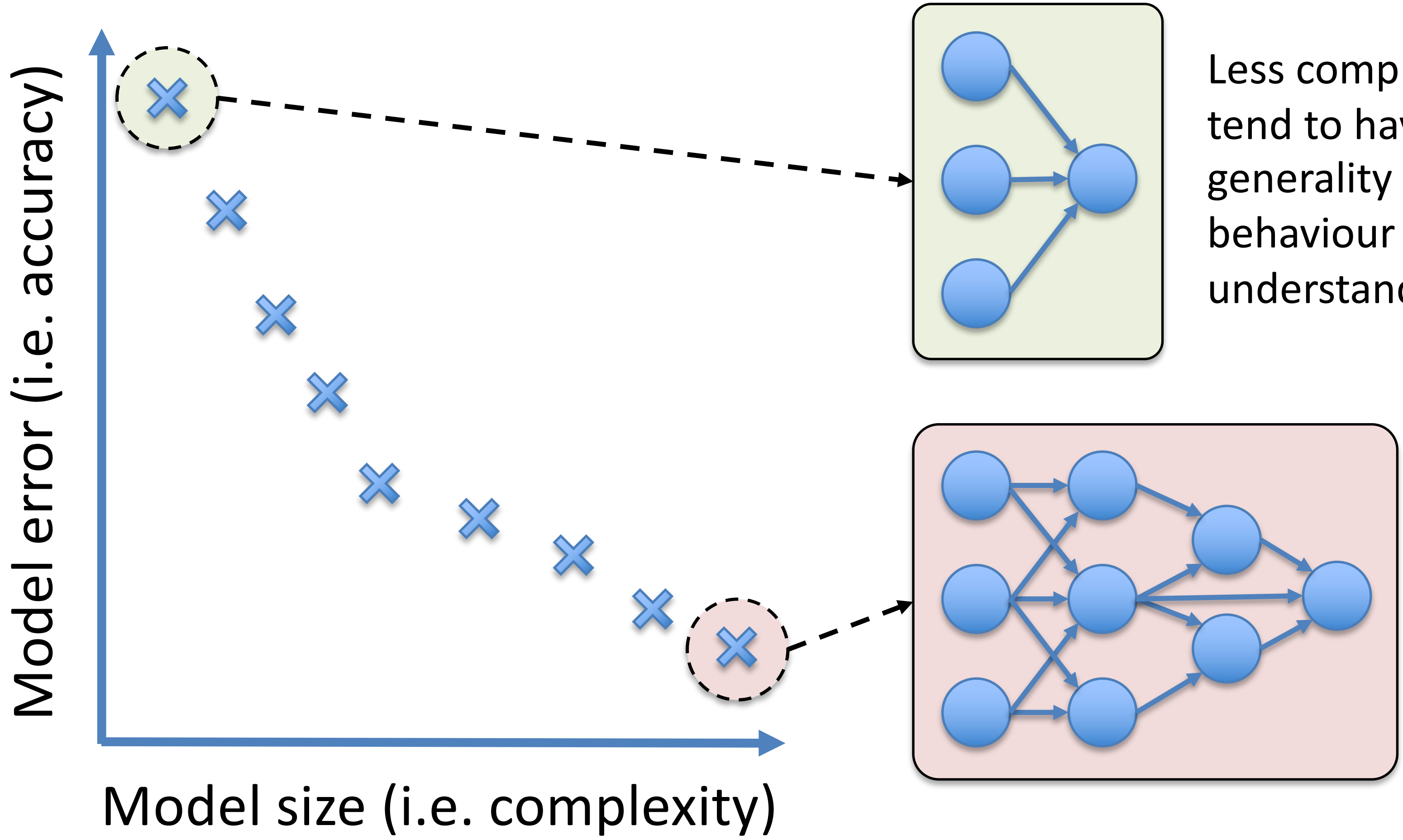




Chiba et al, High-Fidelity Multidisciplinary Design Optimization of Wing Shape for Regional Jet Aircraft, EMO 2005

Example: Neural Networks

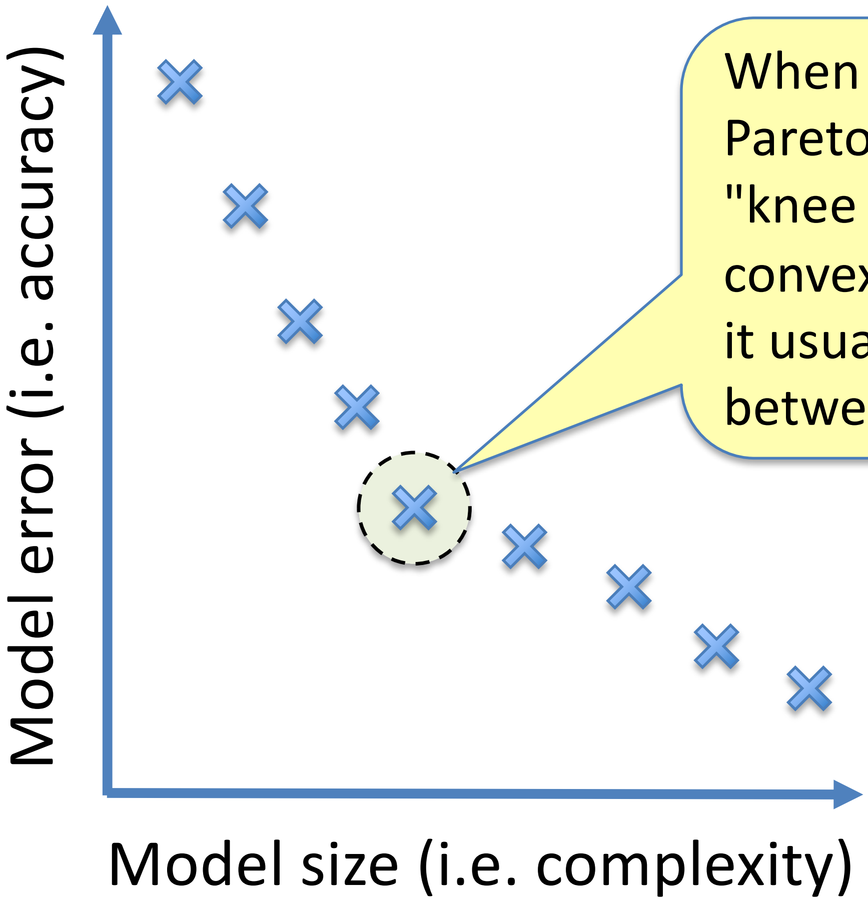
- MOEAs have been used to evolve ANNs, usually to maximise accuracy and minimise complexity:



Less complex models tend to have better generality and their behaviour is easier to understand

Example: Neural Networks

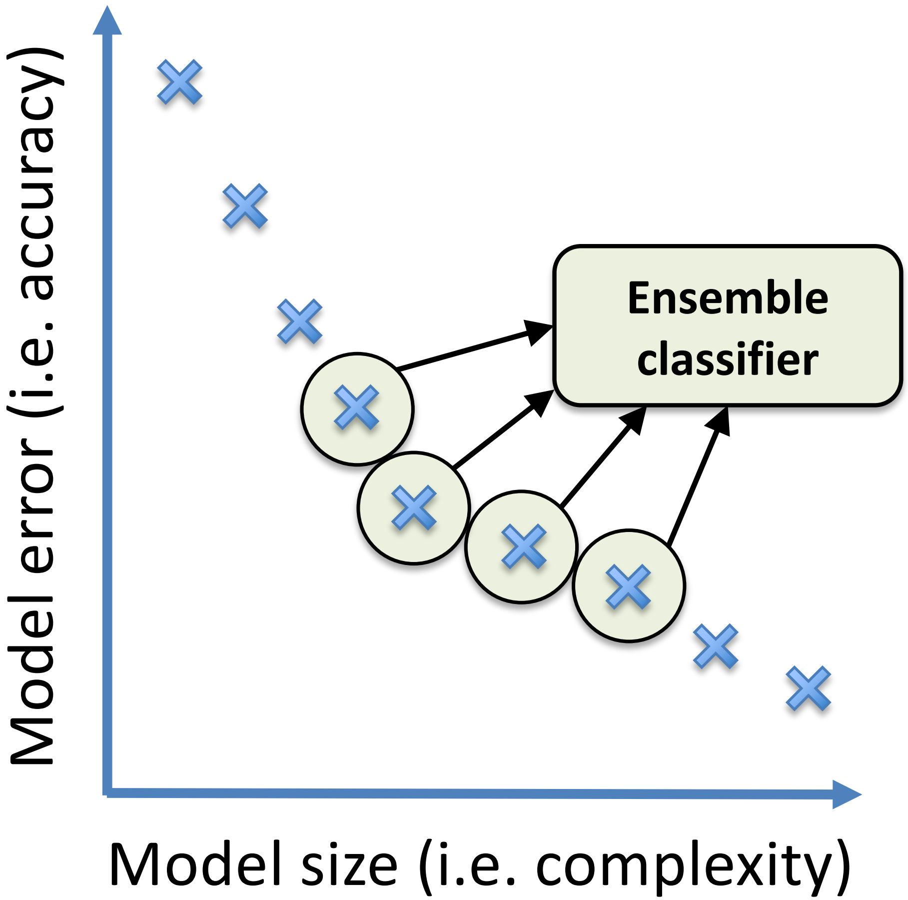
- MOEAs have been used to evolve ANNs, usually to maximise accuracy and minimise complexity:



When selecting a solution from the Pareto front, the solution nearest the "knee point" (the point of maximum convex bulge) is often chosen, since it usually represents a good trade-off between the various objectives

Example: Neural Networks

- MOEAs have been used to evolve ANNs, usually to maximise accuracy and minimise complexity:

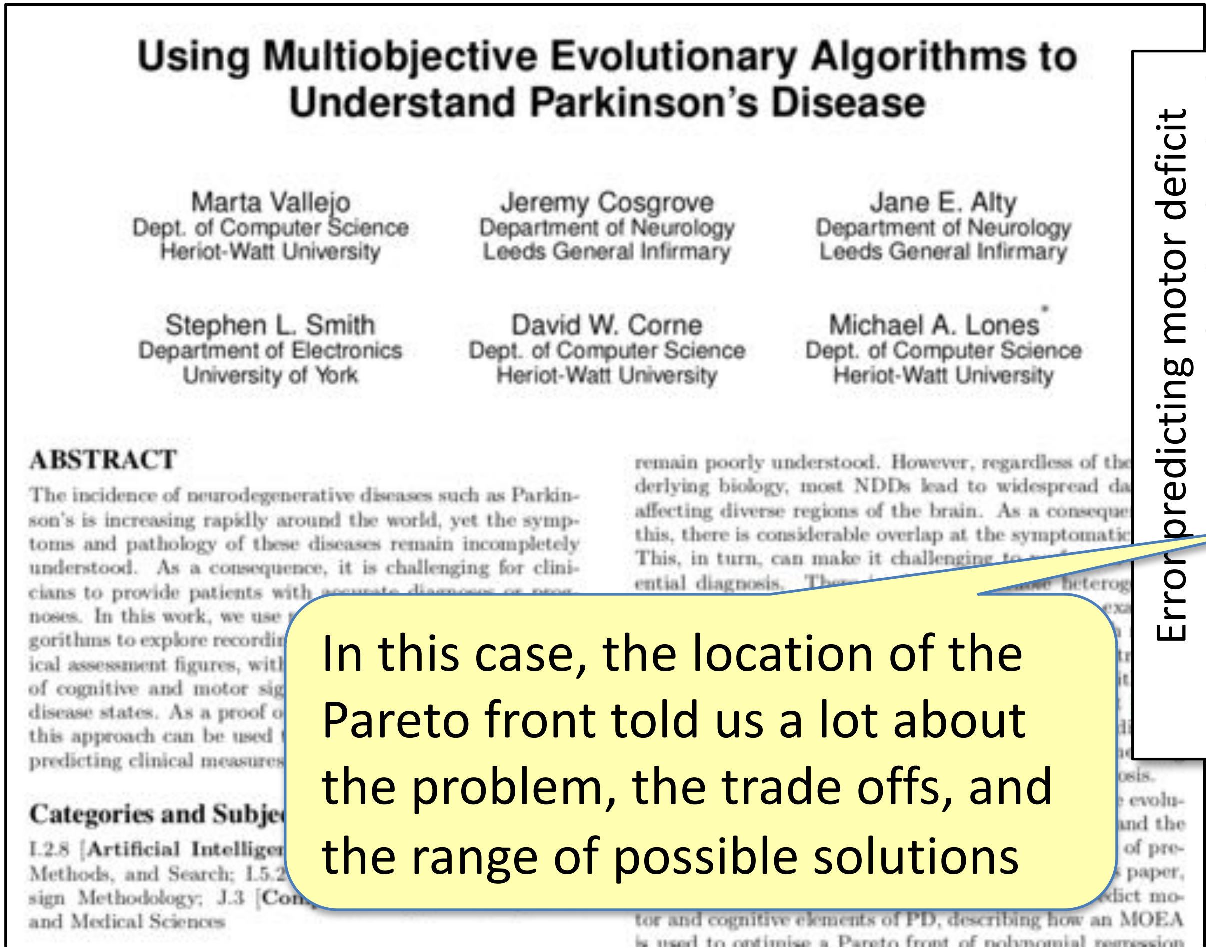


Another common approach when using MOEAs to evolve Pareto fronts of classifiers is to combine either all, or just a subset, of the classifiers in to an ensemble model

Smith and Jin, Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction, Neurocomputing 143(2):302-311, 2014

Example: Disease Diagnosis

- MOEAs can be useful for exploring solution spaces
  - E.g. Diagnostic classifiers for Parkinson's disease



Using Multiobjective Evolutionary Algorithms to Understand Parkinson's Disease

Marta Vallejo
Dept. of Computer Science
Heriot-Watt University

Jeremy Cosgrove
Department of Neurology
Leeds General Infirmary

Jane E. Alty
Department of Neurology
Leeds General Infirmary

Stephen L. Smith
Department of Electronics
University of York

David W. Corne
Dept. of Computer Science
Heriot-Watt University

Michael A. Lones*
Dept. of Computer Science
Heriot-Watt University

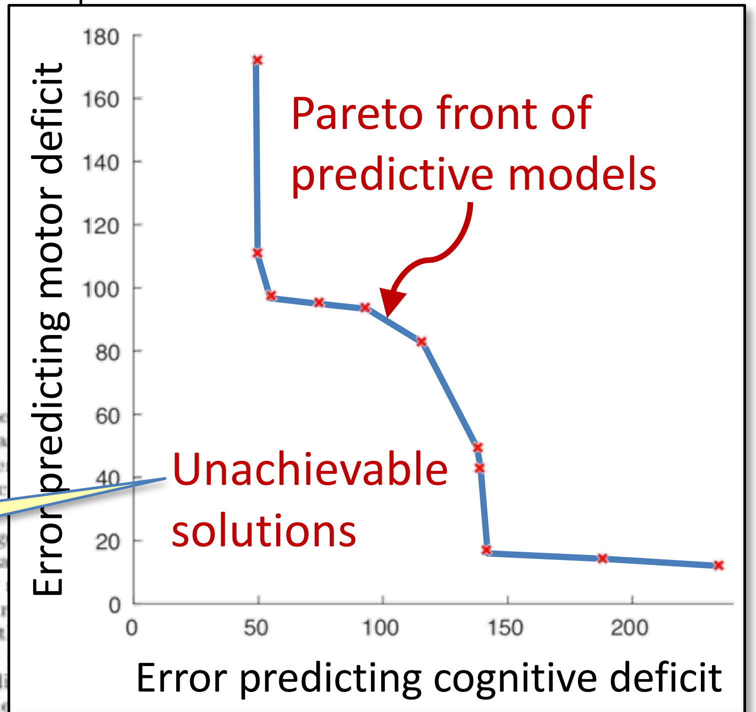
ABSTRACT

The incidence of neurodegenerative diseases such as Parkinson's is increasing rapidly around the world, yet the symptoms and pathology of these diseases remain incompletely understood. As a consequence, it is challenging for clinicians to provide patients with ...

remain poorly understood. However, regardless of the ... derlying biology, most NDDs lead to widespread da... affecting diverse regions of the brain. As a conseque... this, there is considerable overlap at the symptomatic ... This, in turn, can make it challenging ...

Categories and Subje...

In this case, the location of the Pareto front told us a lot about the problem, the trade offs, and the range of possible solutions



Suggested Reading

- Essentials of Metaheuristics
  - Chapter 7: Multiobjective Optimisation
- Lots of introductory papers available:
  - e.g. https://link.springer.com/chapter/10.1007/978-81-322-1958-3_1 (accessible from campus network)
- Mukhopadhyay et al, "A Survey of Multiobjective Evolutionary Algorithms for Data Mining"
  - https://ieeexplore.ieee.org/document/6658835 (accessible from campus network)

MOEA Software

HERIOT WATT UNIVERSITY

- If you want to try out some MOEAs:
  - MOEA Framework (http://moeaframework.org) is a popular Java-based tool with lots of MOEAs
  - jMetal (http://jmetal.sourceforge.net) also has a good selection of MOEAs, again in Java
  - If you're not a Java programmer, then there is plenty of code for other languages, particularly for NSGA-II

What's Next?

- In the next two lectures, I'll be talking about:
  - Cellular Automata
  - Artificial Gene Regulatory Networks
  - Both are models of how biological systems work, and can be used to generate complex computational behaviours from relatively simple systems