ECM3412/ECMM409 Nature Inspired Computation

Lecture 7 Multi-Objective Genetic Algorithms

Introduction

- Every application we have looked at so far has a fitness function returning a single value to the EA:

e.g. fitness = f(s) = antenna efficiency, jet efficiency etc..

- However, many problems have more than one objectives which are often conflicting:

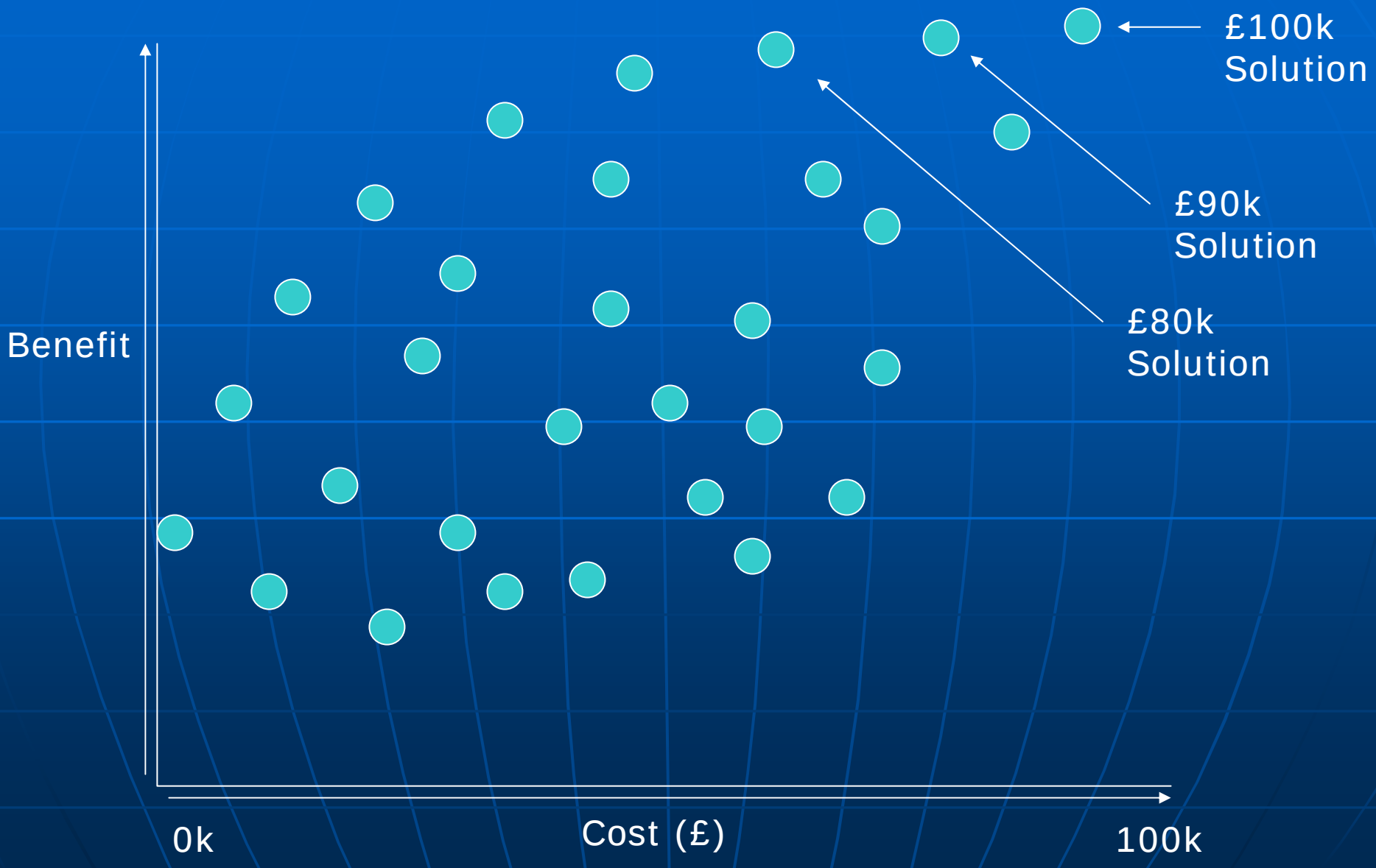
e.g.

- Design an aeroplane – strength vs weight
- Design an antenna – efficiency vs size
- Design a water network – cost vs pressure constraints

Introduction II

- Many real-world problems exist with multiple objectives
- Engineering problems are often ‘multi-objective’
- They are especially useful in industry where the company wants to see the ‘trade-off’ between the costs and benefits of solutions.
- SOGA:“We want to spend £100k how best can we spend it?”
- MOGA: “What happens if we reduce this to £90k? £80k?”
- Could run multiple GA runs…

Introduction III

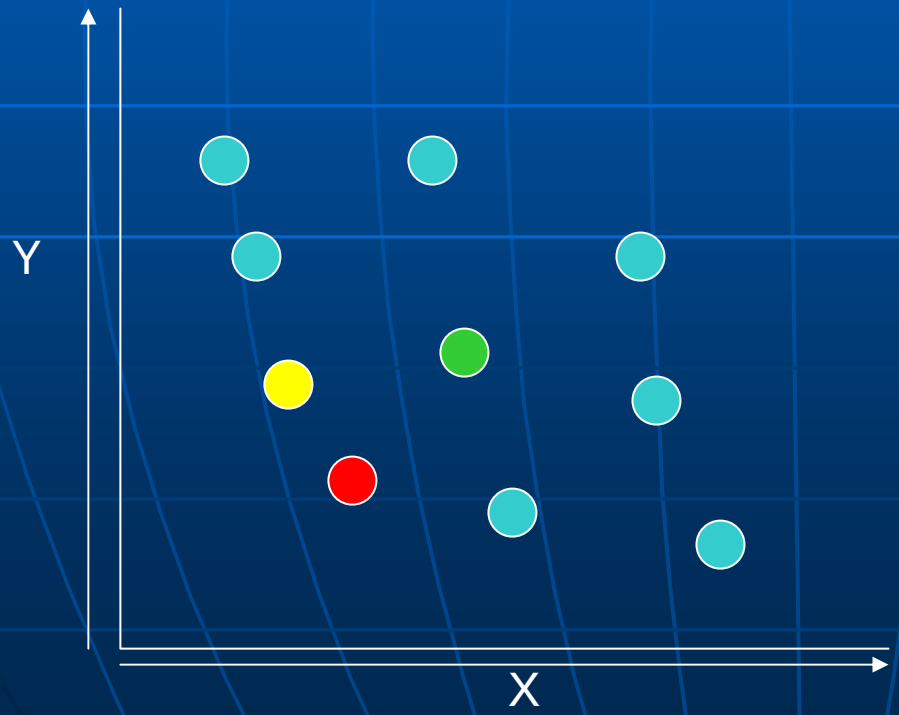


MOEAs

- How can we get EAs to perform Multi-Objective computation?
- Generational GA
- Revised Objective Function
  - Multiple Objectives
- Revised Selection Criteria
  - Domination Criterion
- Revised Visualisation
  - Pareto-Optimal Curves

Domination Criterion

- A solution *a* is said to 'dominate' another *b* in the population if it is at least as good as *b* in every dimension and better than *b* in at least one dimension (objective)



If we are trying to Minimise X and Y

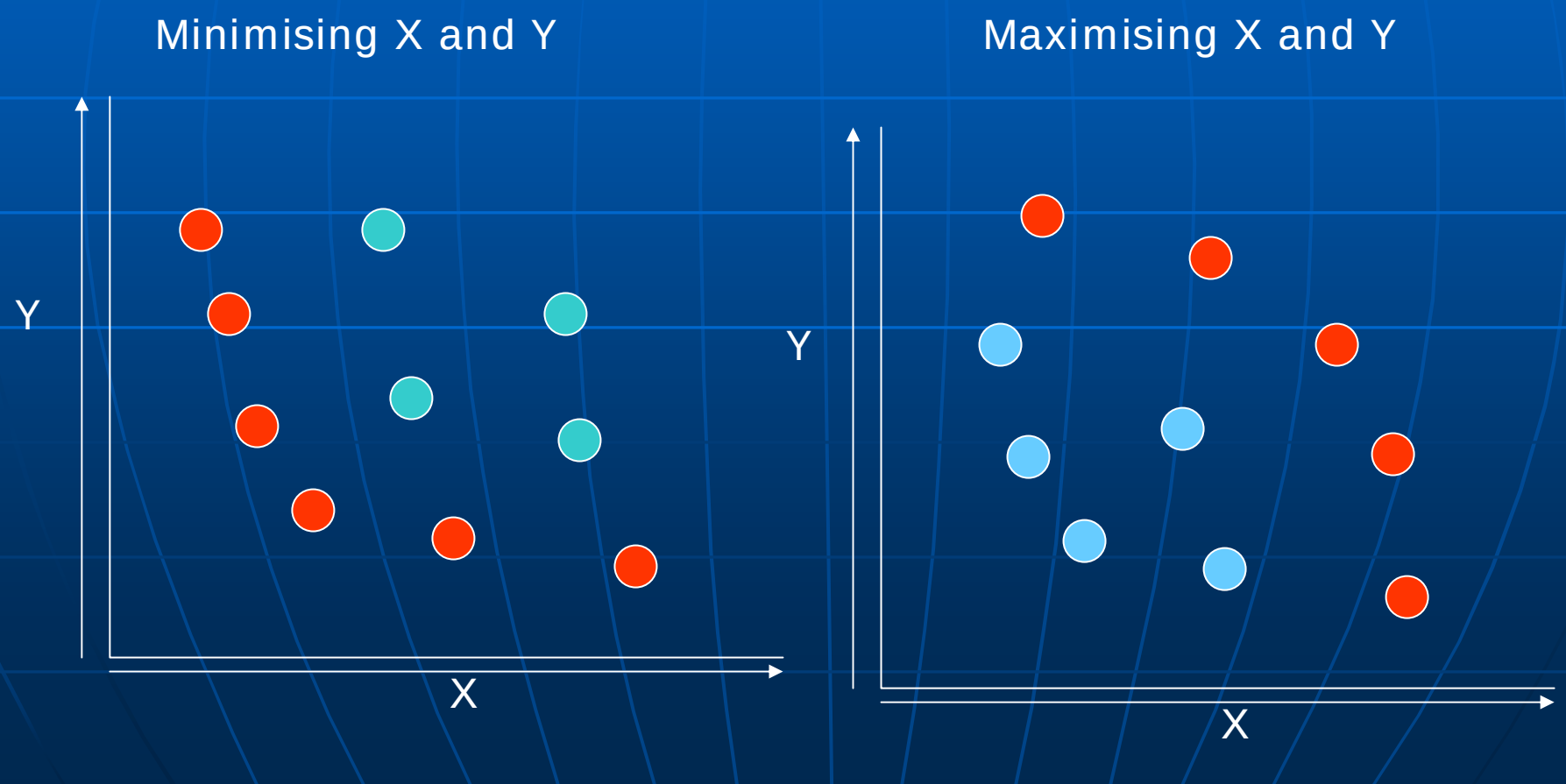
- Red dominates Green because X has lower values than Y in both dimensions
- Yellow also dominates Green

BUT!

- Yellow and Red do not dominate each other – Yellow is better in X and Red is better in Y
- Red and Yellow are said to be non-dominated

Pareto-Front

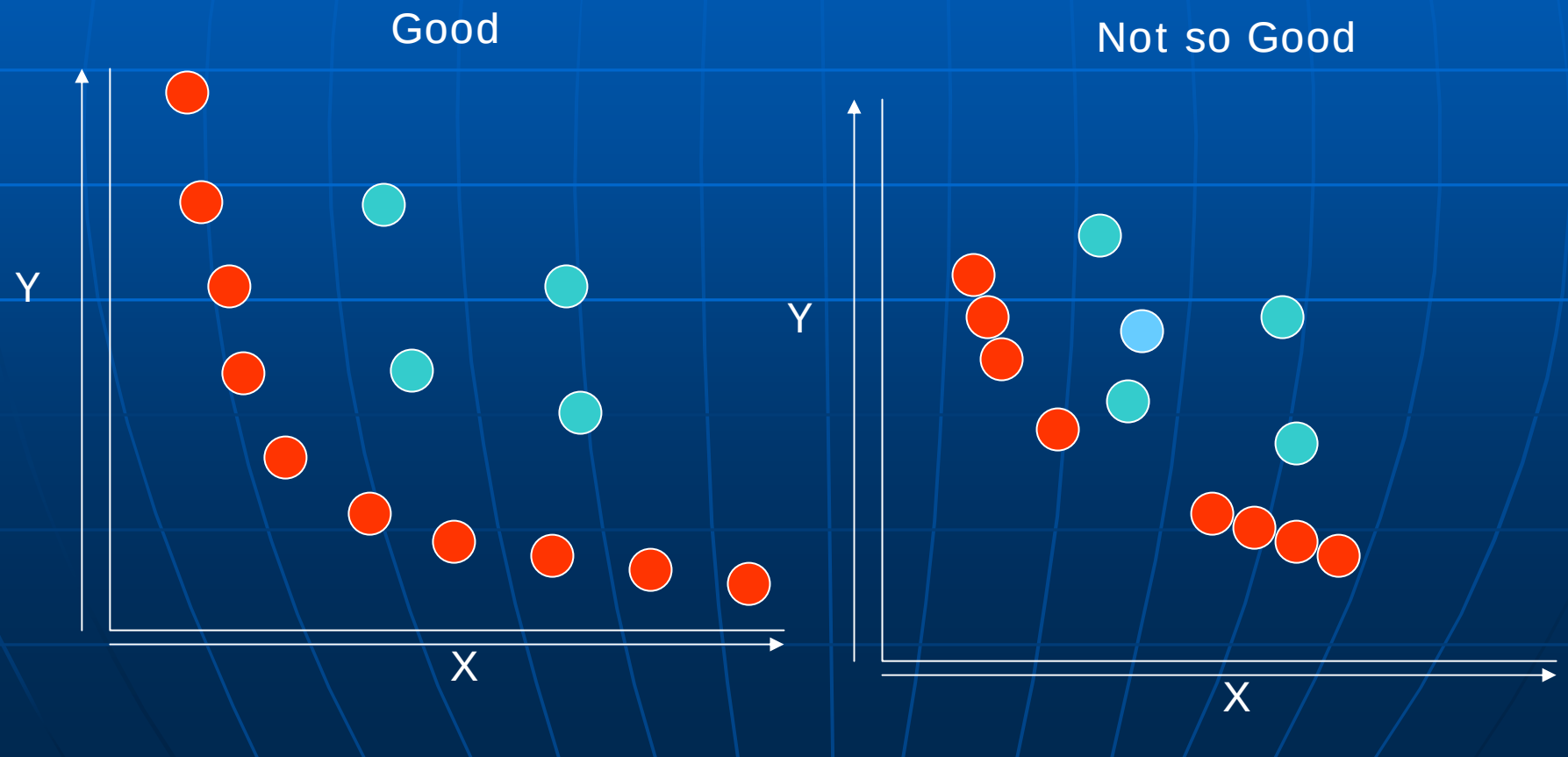
- Given the domination criterion, we know that the best solutions will lie along a curve consisting of non-dominated points. This is known as the ‘pareto-optimal front’ or ‘pareto-front’



Desirable Characteristics of the Pareto-Front

- Evenly-spaced solutions
- Covering the largest possible area of the front

e.g.



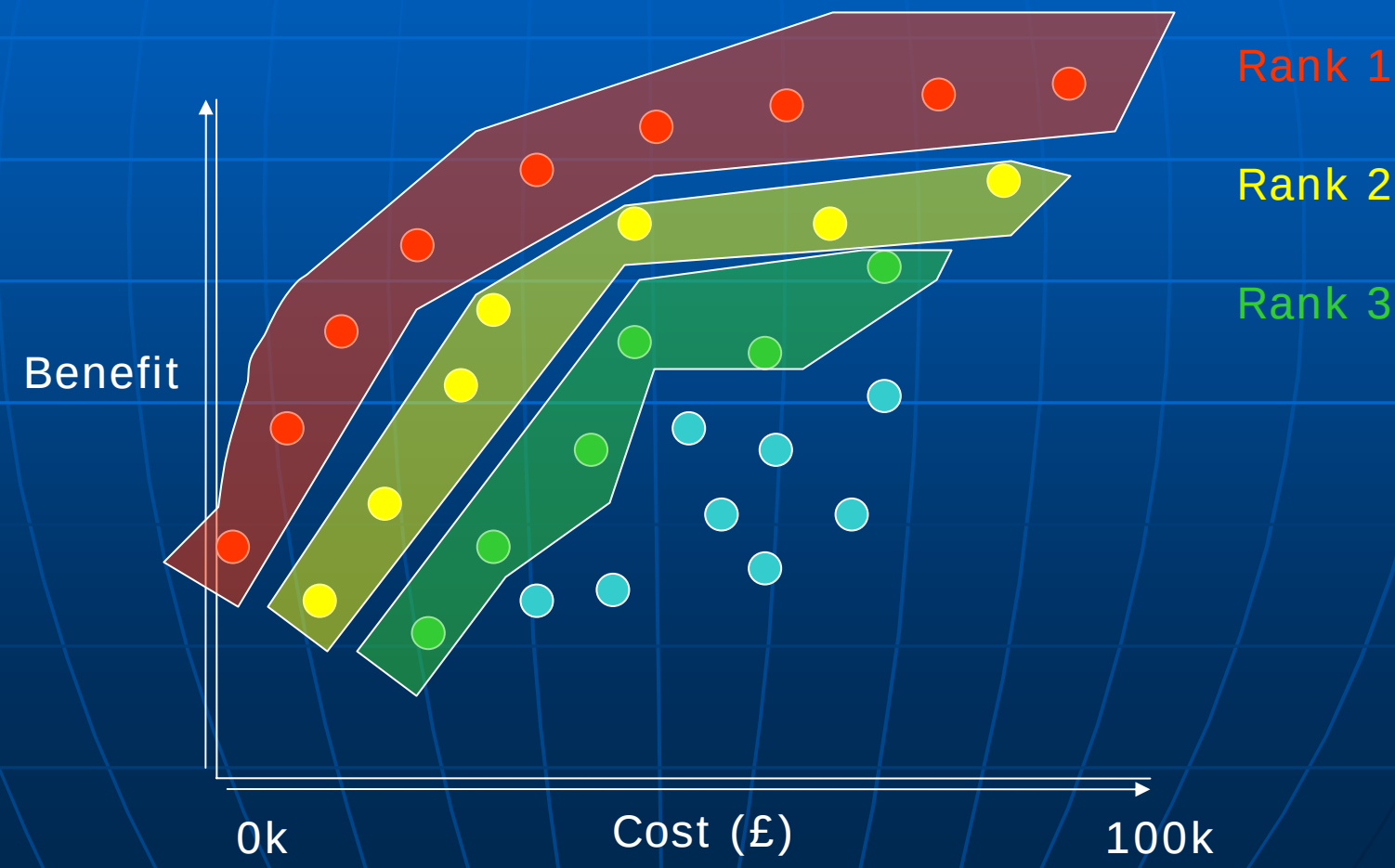
Operator Modifications

- Crossover and mutation can remain broadly the same
- With ranking, we can tell which solutions are best (non-dominated) and which are worst.
- But how can we tell the difference between two solutions which are non-dominated?
- We need a new selection operator

Ranking I

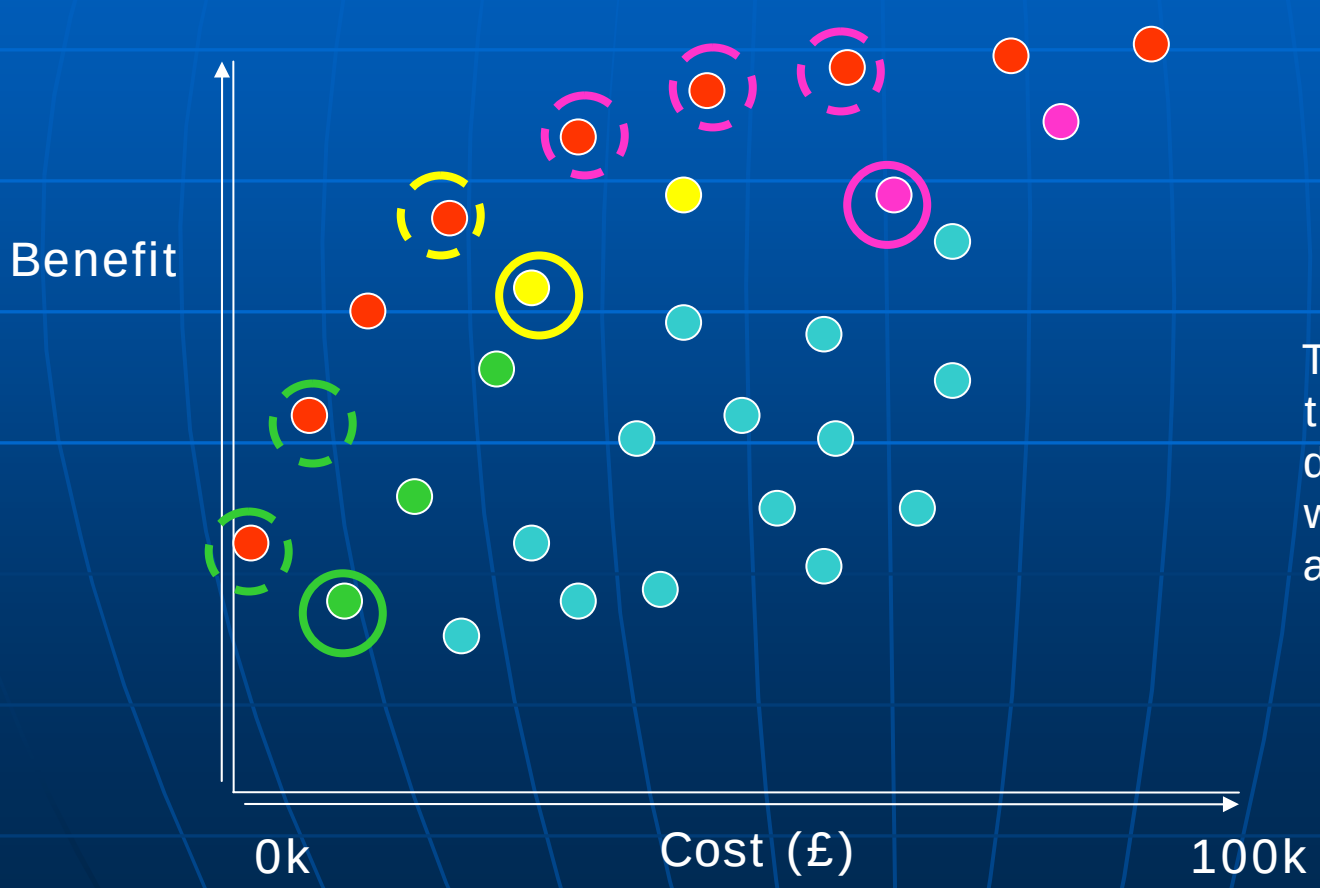
Many algorithms use a rank-based approach to selection

One method is to determine the pareto-front assign it Rank 1, remove it from the population and then determine the next pareto front, assign it Rank 2, and so on….



Ranking II

...Another method is to assign a Rank based on the number of solutions that dominate the solution – e.g. Rank 0 = non-dominated solution, Rank 1 = one solution dominates...etc...



Rank 0

Rank 1

Rank 2

Rank 3

The dotted circles show those solutions which dominate the solution with the solid circle around it.

Pareto Domination Tournament

- Pareto Domination Tournament:

"Select two random individuals from the population, if one dominates the other then select it"

- However, selection pressure is not sufficient, so a randomly selected comparison set is introduced.
- This changes the selection to:

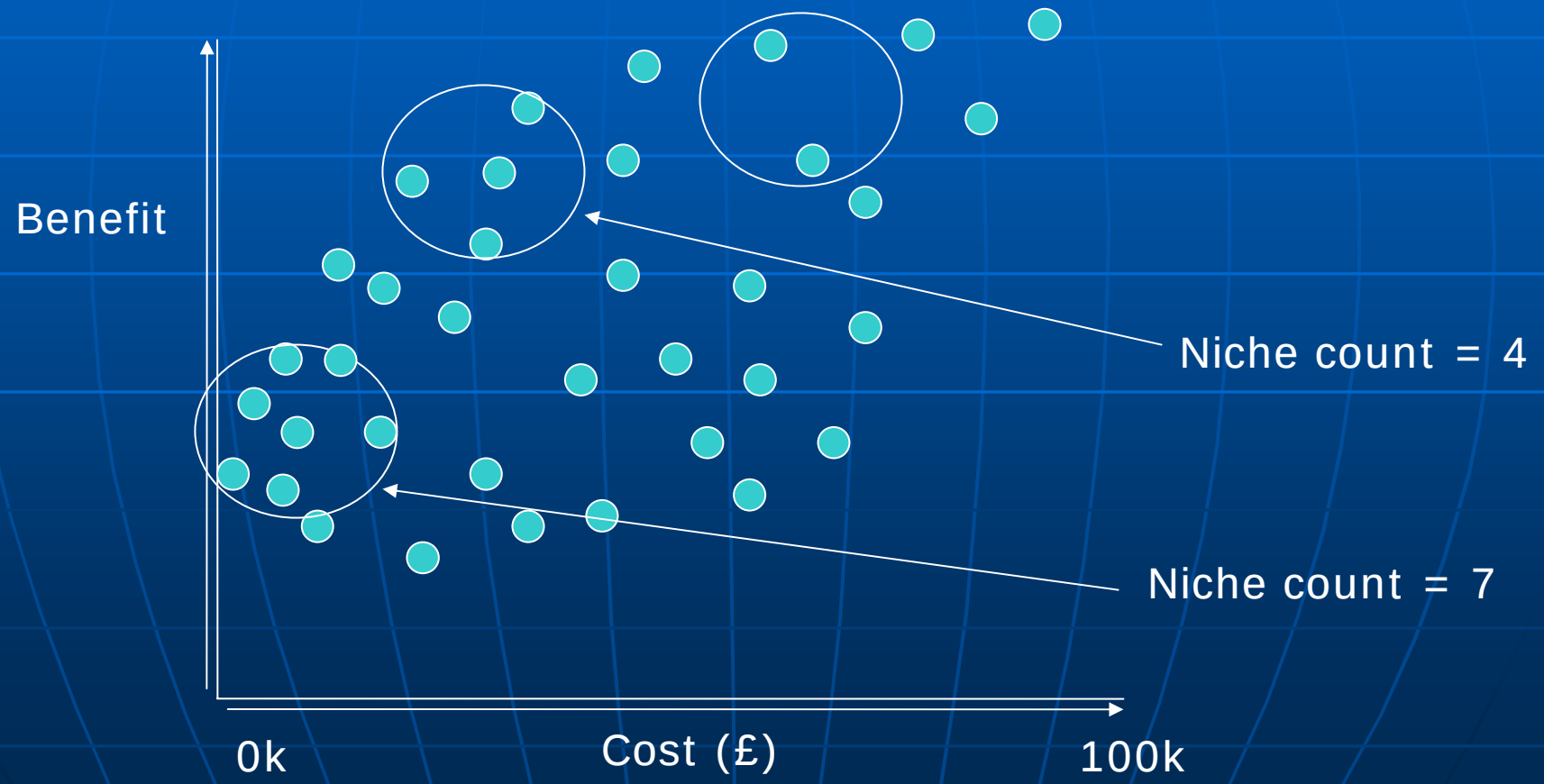
"Select two random individuals *a* & *b* and a separate comparison set *c* from the population. If *a or b* is non-dominated with respect to *c*, then select.

If *a* and *b* have the same domination – tiebreak"

- The size of c can be used as a parameter to change the selection pressure

Tiebreak 1 – Fitness Sharing

- Separate the fitness landscape or genotype into ‘Niches’
- Individuals in a niche ‘share’ the niche fitness
- New parameter – niche radius

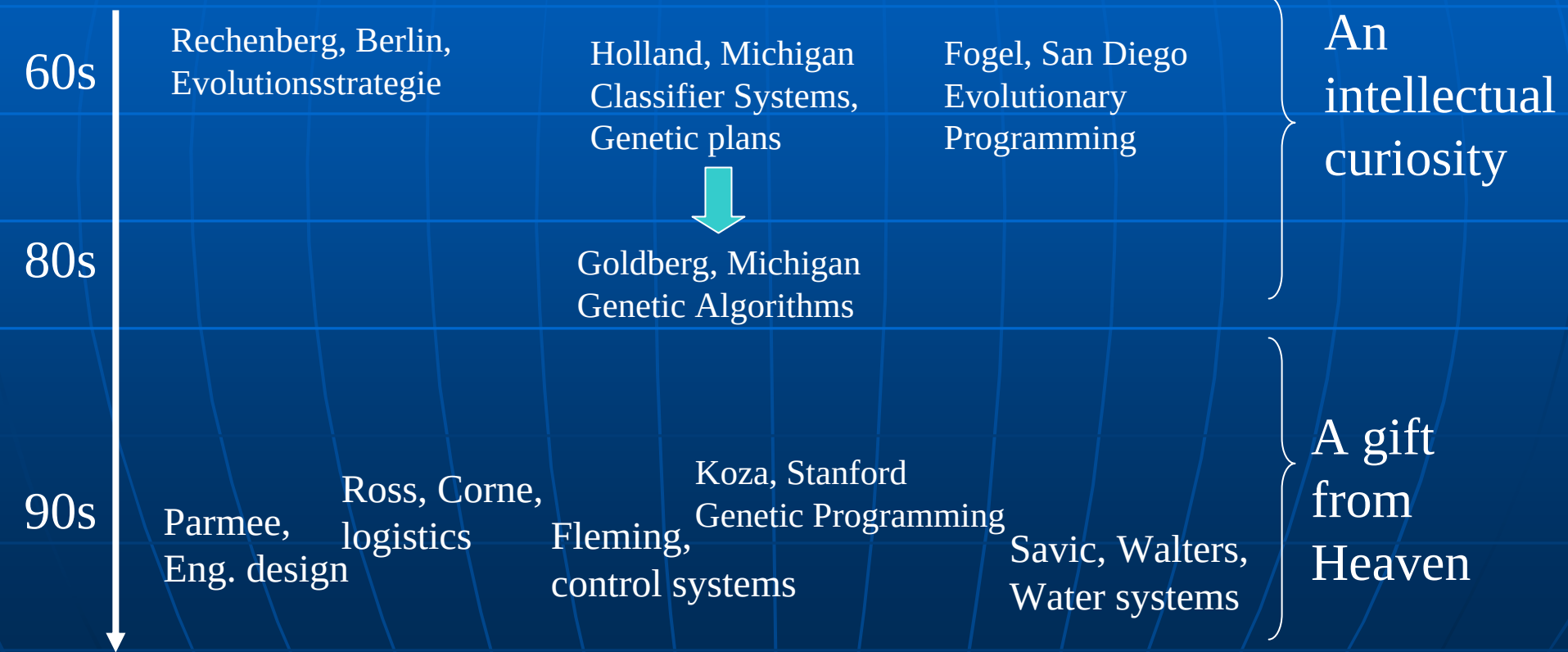


NPGA – Niched Pareto Genetic Algorithm (Horn & Nafpliotis - 1993)

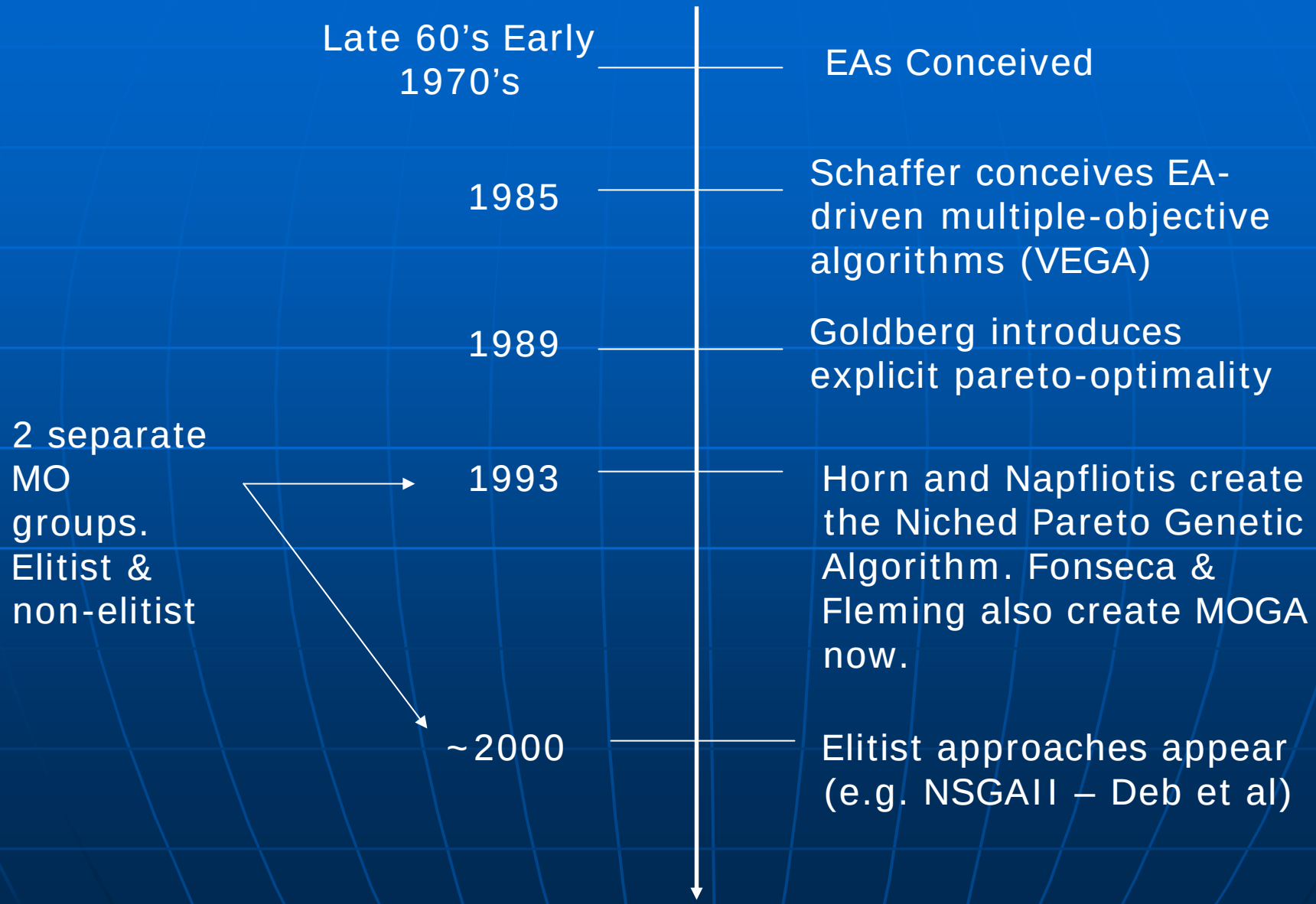
1. Generate initial population of solutions
2. Select individuals:
   1. Run pareto-tournament domination selection. If one solution is non-dominated and another not, go to 3, else go to 2.
   2. Compute niched fitness to separate solutions
   3. Repeat for n selected individuals
3. Crossover and mutate to generate new population

The Evolutionary Computation Fossil Record

The first published ideas using evolution in optimisation came in the 50s. But the lineage of current algorithms is like this:



MOGA Fossil Record

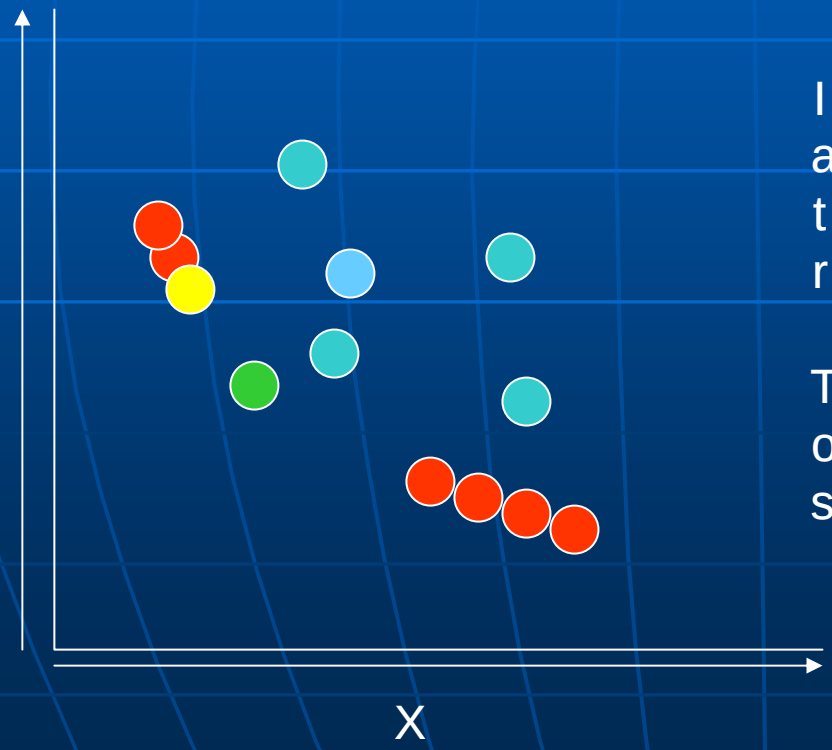


Non-Dominated Sorting Genetic Algorithm (Deb, 2001)

- Elitist MOGA
- Generates new solutions using selection, crossover & mutation
- Uses a fast non-dominated sort (hence the name) to rank solutions
- Uses *crowding distance* as a tie-breaker

Tiebreak Resolution II – Crowding Distance

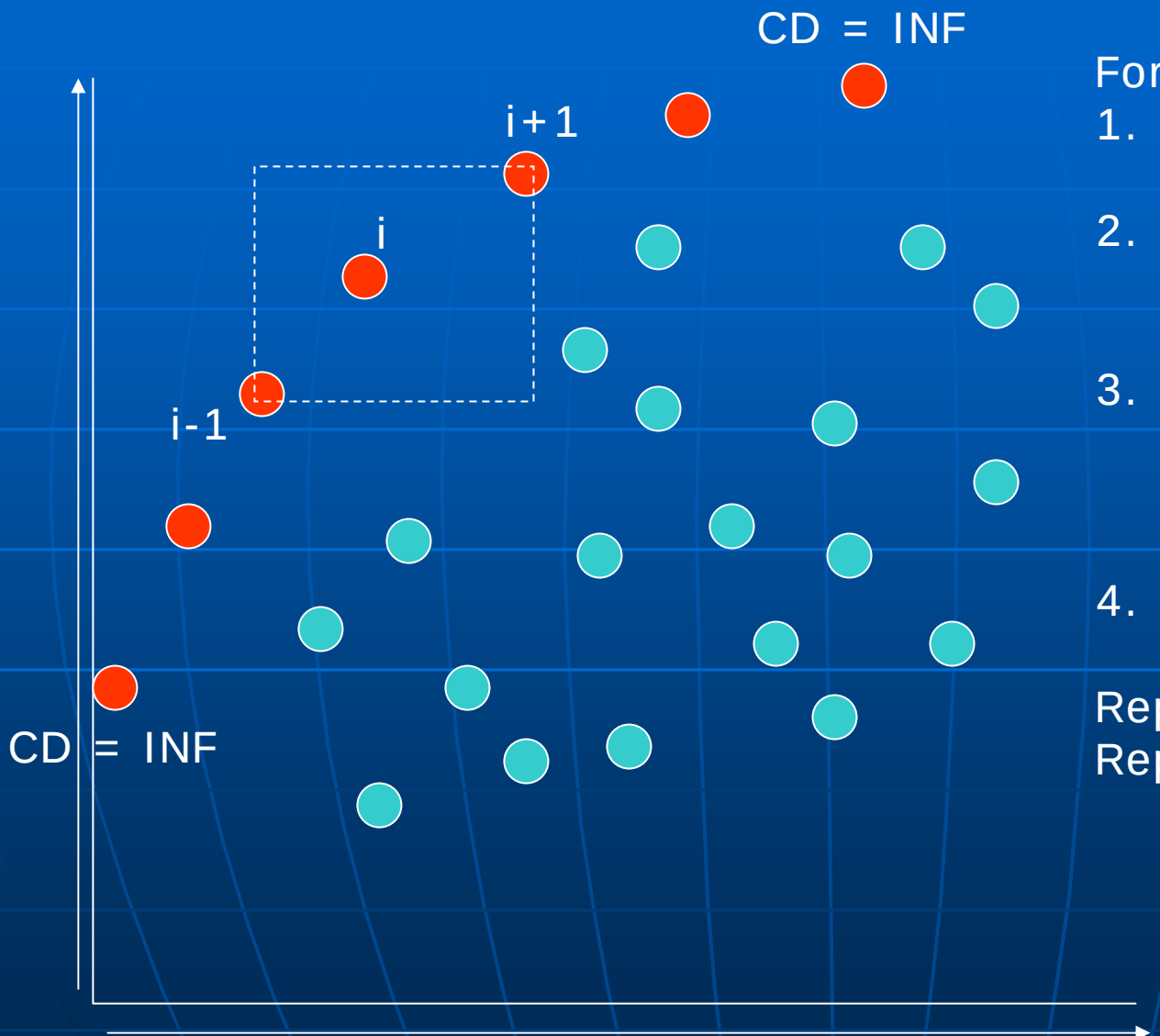
- A problem with Pareto Domination Tournament Selection, is that two individuals can be non-dominated/dominated.
- Crowding Distance is a method of tie-break between 2 such individuals and ensures a good spread of solutions:



If the yellow and green solutions are selected by the tournament, then both will be non-dominated, regardless of the comparison set.

The green solution is selected as it occupies a less-densely populated space.

Crowding Distance Computation

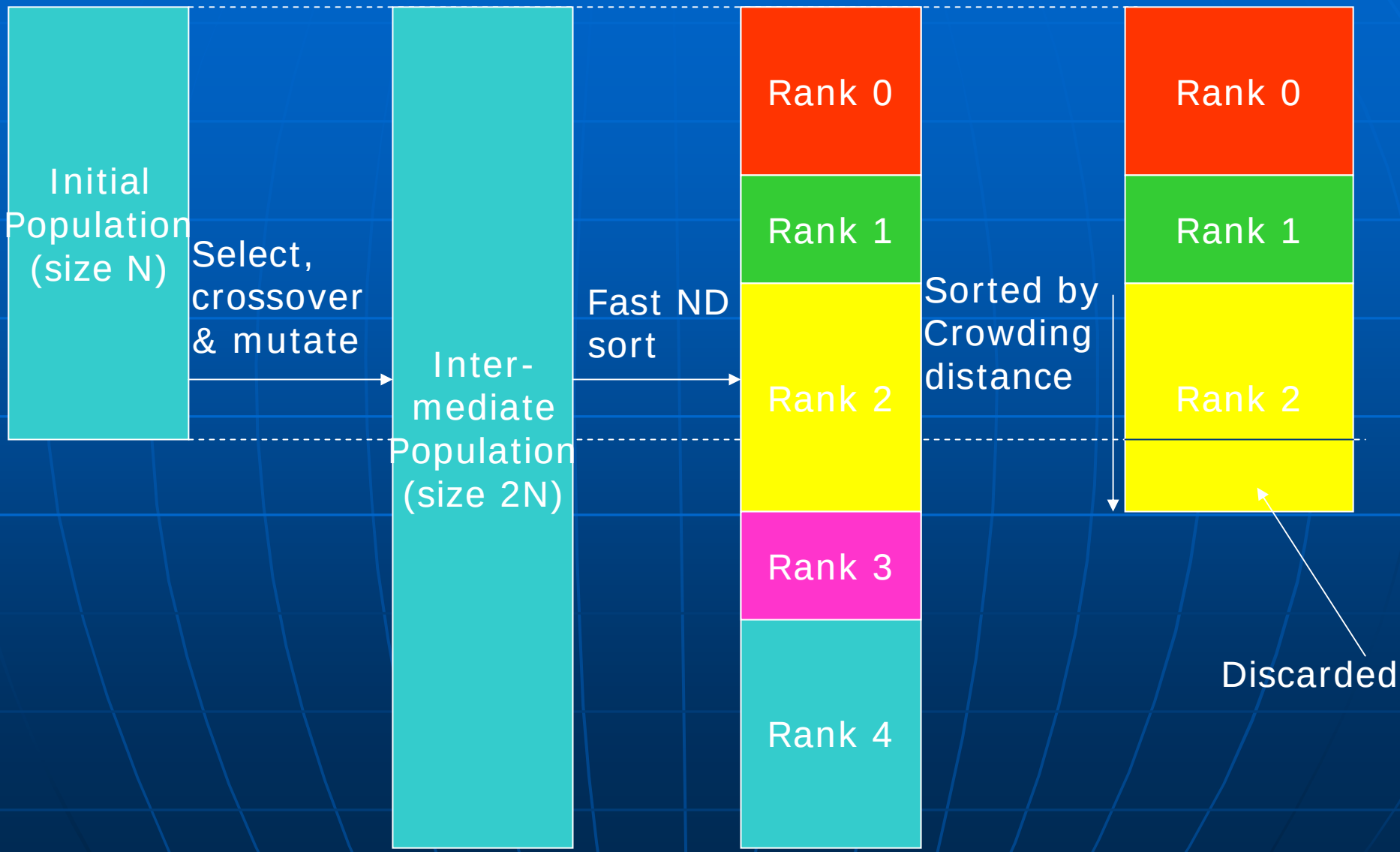


For each objective *m*:

1. Sort population by value of *m*.
2. For end solutions crowding distance = INF
3. For other solutions (i), the crowding distance for objective *m* = (i+1).m – (i-1.m)
4. Add this distance to the sum for i.

Repeat for all i.
Repeat for all m.

NSGAII Execution

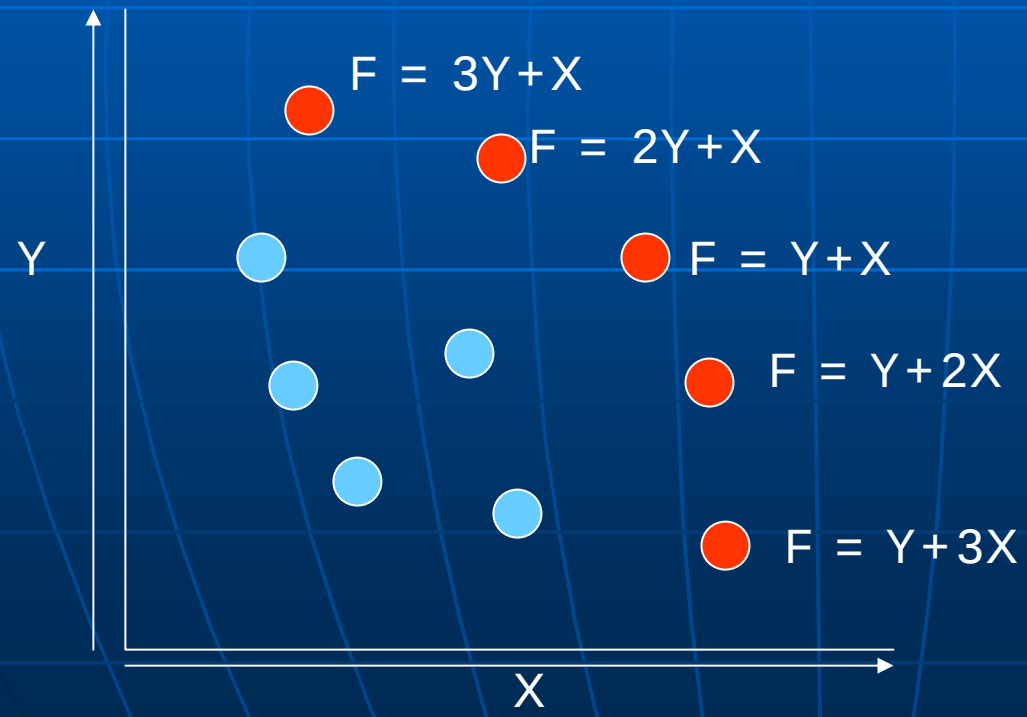


Elitist vs Non-Elitist

- Elitist MOGAs (PAES, SPEA, NSGAII) have somewhat superseded non-elitist versions.
- Why?
- NSGAII requires no extra parameters (e.g. niche sizes)
- Executes & converges faster
- Some evidence shows that on certain problems, it can prematurely converge.

Alternative Approaches

- The population-based approach of GAs is ideal for MO optimisation
- Could we use single objective GAs?
- Yes. By weighting each objective, we can optimise for different points on the pareto-curve:



BUT!

- You have to do a separate GA run for each point
- Final shape of the curve not defined for MOGAs

Archiving

- With single-objective EAs, keeping the best solution is easy, but what about MOEAs?
- What if the possible pareto front > population size?
- We can maintain a global 'best pareto-optimal population'. (e.g. if a new solution *a* is non-dominated w.r.t the rest of the 'best population' then it is added to it). BUT!
  - The best pop can be of infinite size
  - Will take longer and longer to search as size increases
  - Will have a huge size for problems with >2 objectives
- Archiving is a current research topic.
- Most archiving algorithms focus on keeping those solutions with greatest optimality & spread.

Example MOGA Run

- Water Distribution Networks