UiO : University of Oslo

institutt for informatikk

IN3050/IN4050, Lecture 4
Evolutionary algorithms 2

1: Introduction and repetition

2: Selection

3: Diversity preservation

4: Hybridization

5: Multi-objective optimization

UiO : University of Oslo

institutt for informatikk

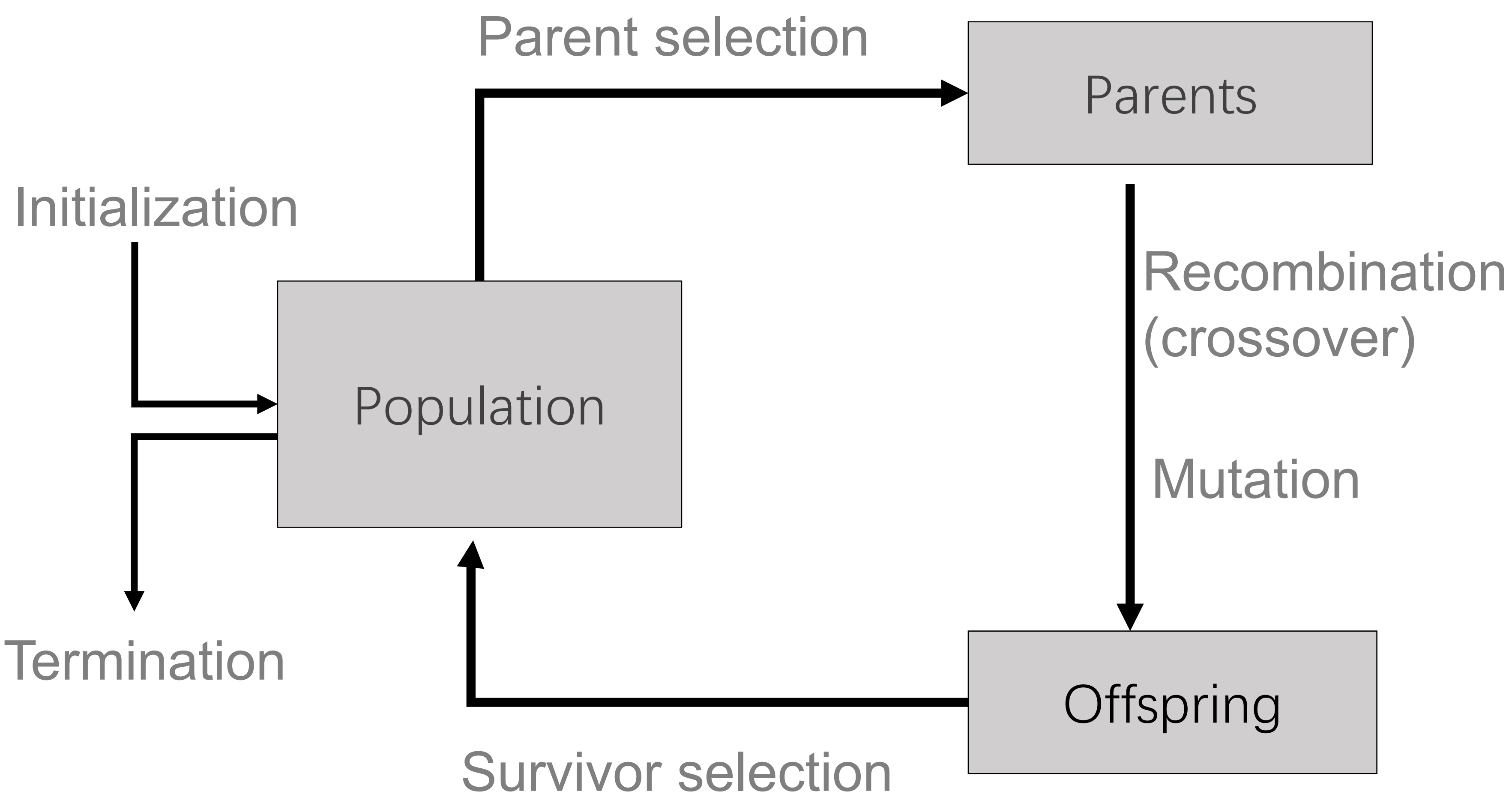
IN3050/IN4050, Lecture 4
Evolutionary algorithms 2

1: Introduction and repetition

Kai Olav Ellefsen

Next video: Selection

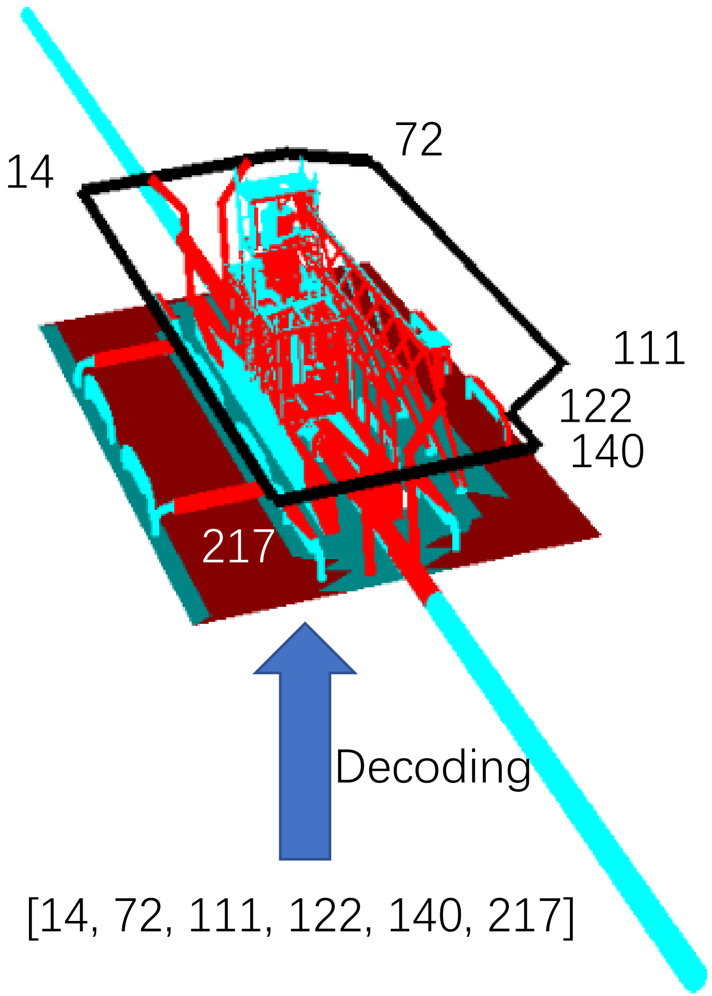
Repetition: General scheme of EAs



Repetition:
Genotype & Phenotype

Phenotype:
A solution representation
we can **evaluate**

Genotype:
A solution representation
applicable to **variation**



UiO : University of Oslo

institutt for informatikk

IN3050/IN4050, Lecture 4
Evolutionary algorithms 2

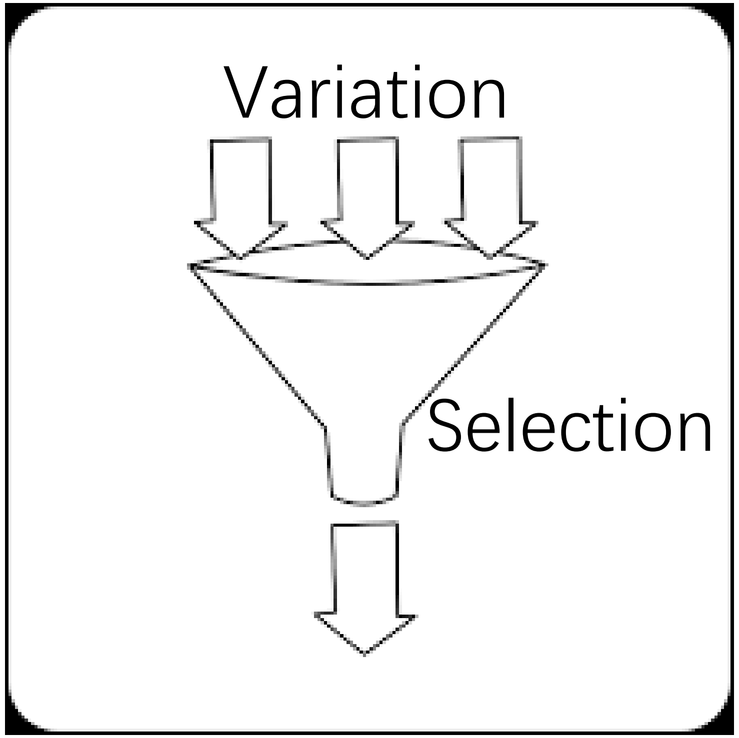
2: Selection

Kai Olav Ellefsen

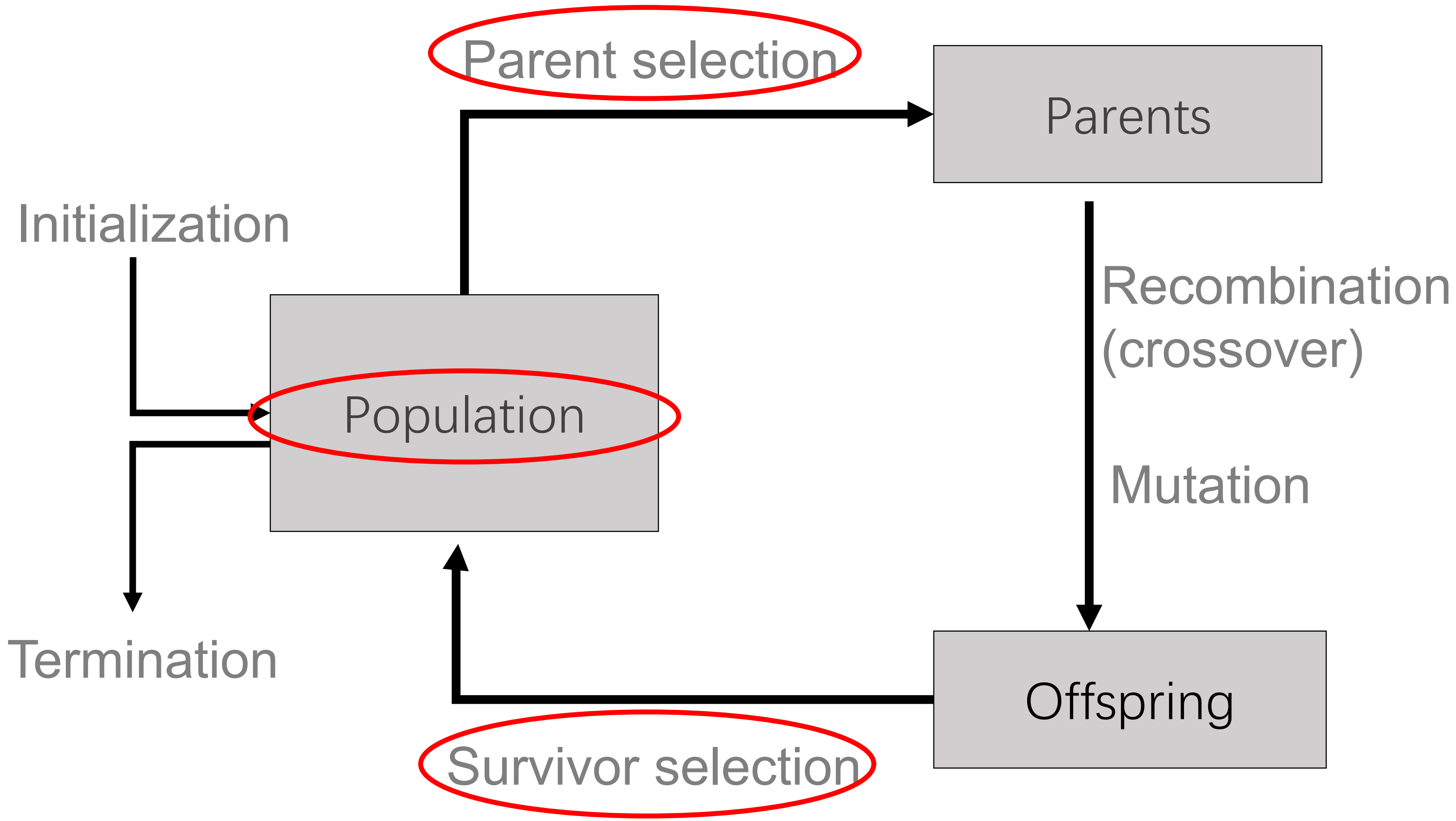
Next video: Diversity preservation

Chapter 5: **Fitness, Selection and Population Management**

- **Selection** is second fundamental force for evolutionary systems
- Topics include:
  - Selection operators
  - Preserving diversity



Scheme of an EA:
General scheme of EAs



Selection

$[0.8 \quad 0.2 \quad 0.3]$

$[1 \quad 2 \quad 3 \quad 5 \quad 4]$

- Selection can occur in two places:
  - **Parent selection** (selects mating pairs)
  - **Survivor selection** (replaces population)
- Selection works on the population

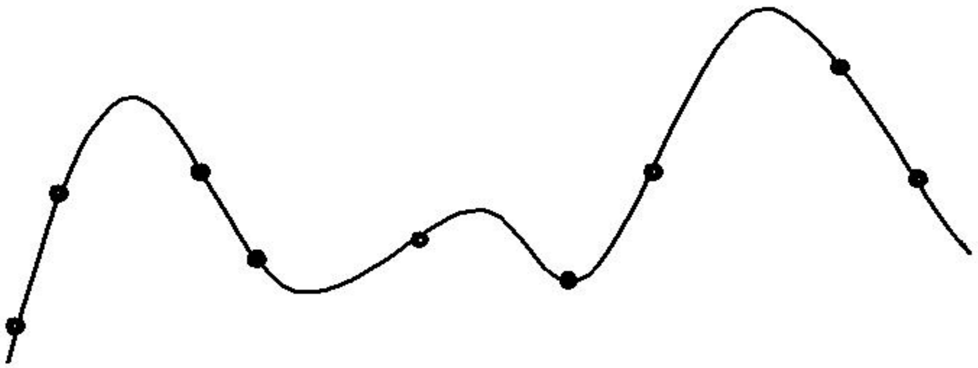
  -> selection operators are **representation-independent** because they work on the fitness value
- **Selection pressure**: As selection pressure increases, fitter solutions are more likely to survive, or be chosen as parents

Effect of Selection Pressure

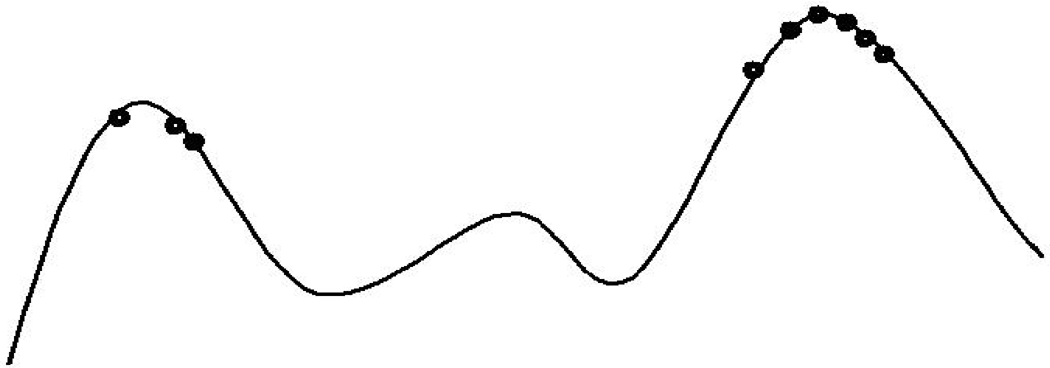
- Low Pressure
- High Pressure


Why Not Always High Selection Pressure?

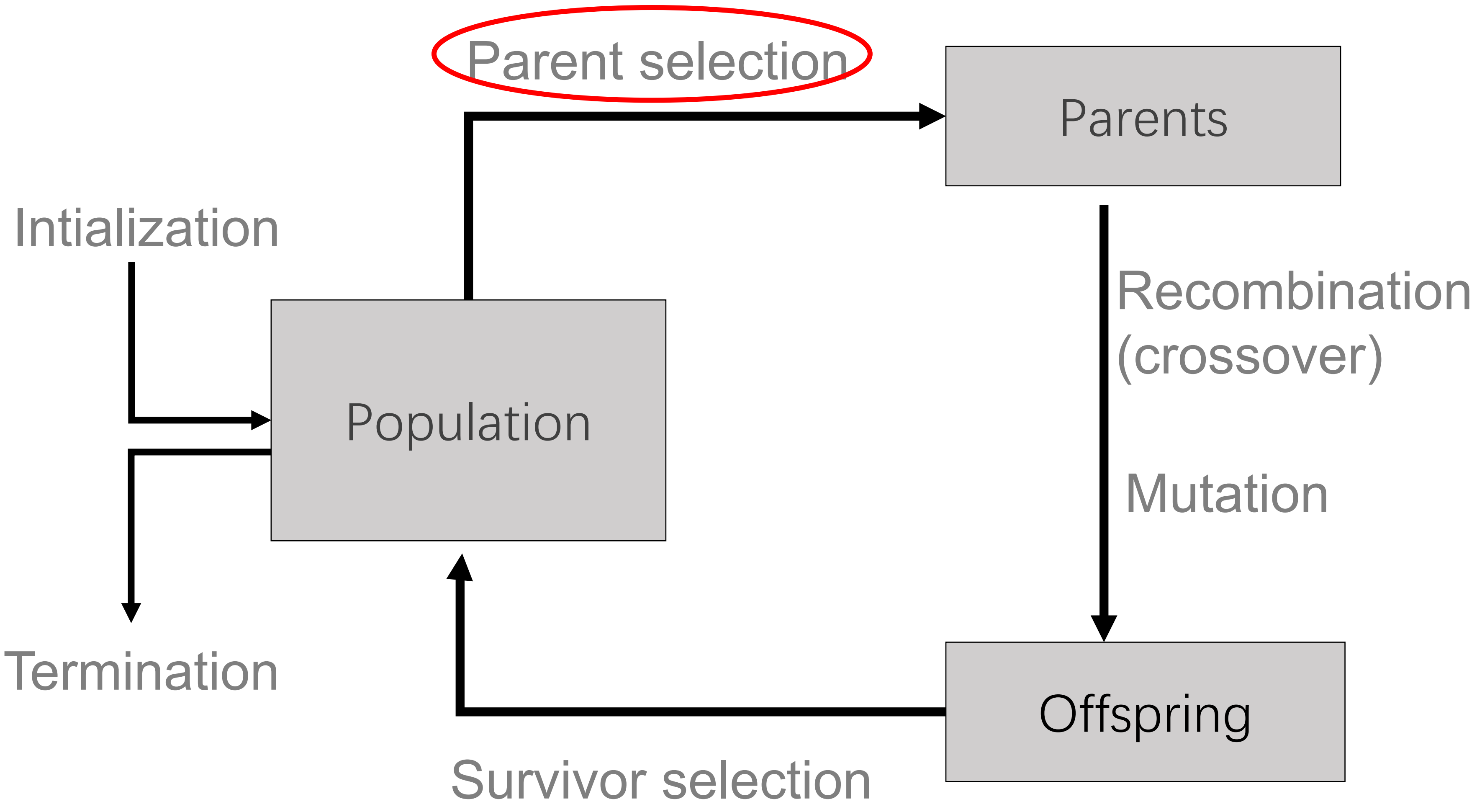
Exploration



Exploitation



Scheme of an EA:
General scheme of EAs



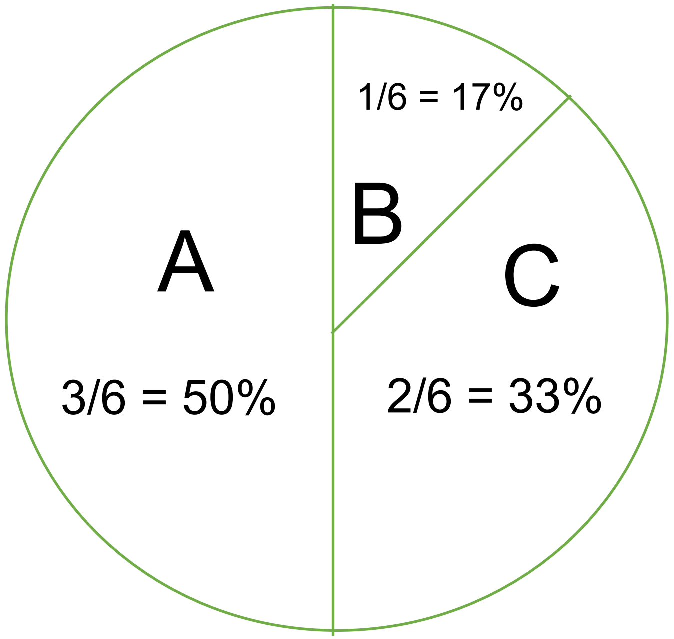
Parent Selection: Fitness-Proportionate Selection

Example: roulette wheel selection

fitness(A) = 3

fitness(B) = 1

fitness(C) = 2



Parent Selection: Fitness-Proportionate Selection (FPS)

- Probability for individual i to be selected for mating in a population size μ with FPS is

$$P_{FPS}(i) = f_i \Big/ \sum_{j=1}^{\mu} f_j$$

- Problems include
  - One highly fit member can rapidly take over if rest of population is much less fit: **Premature Convergence**
  - At end of runs when finesses are similar, **loss of selection pressure**

I_1	I_2	I_3
495	500	505

$P(i) = 0.327,\ 0.333,\ 0.337$

Rank:

0 1 2

I_1	I_2	I_3
0	5	10
0	1/3	2/3

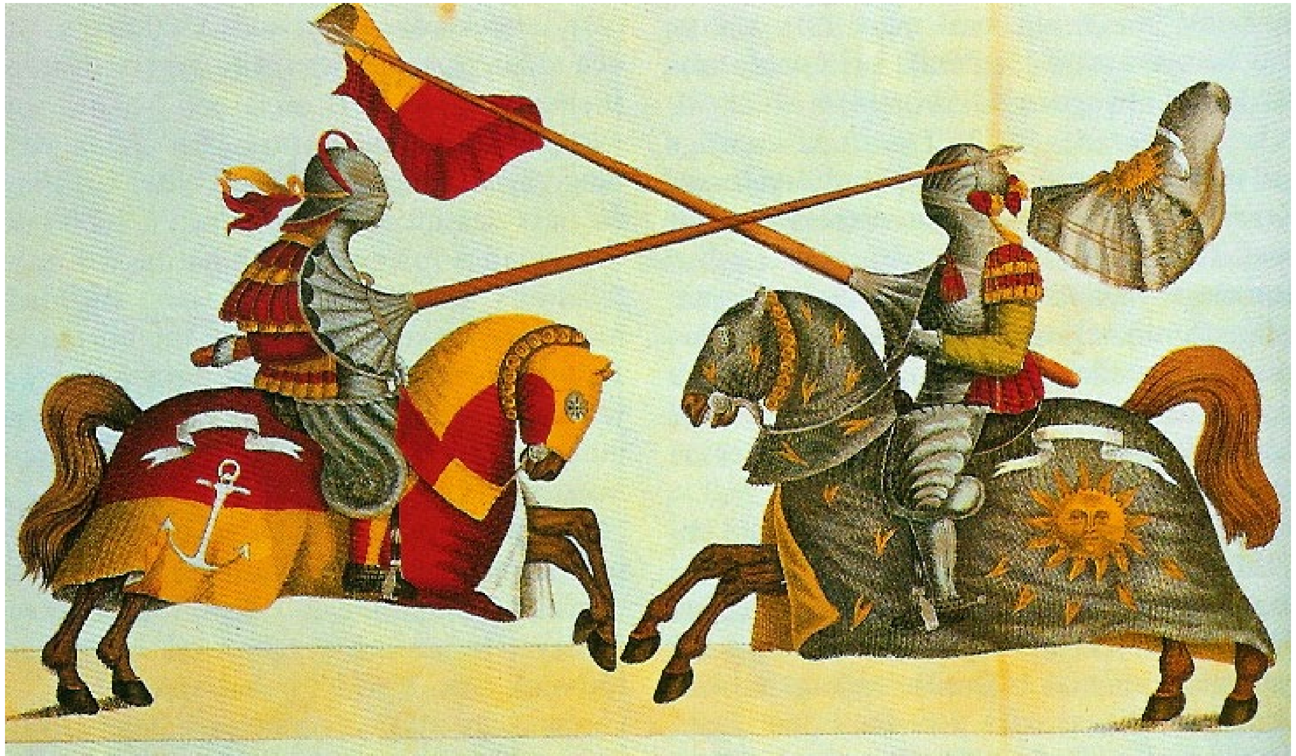
Parent Selection: Tournament Selection (1/3)

- The methods above rely on **global population statistics**
  - Could be a **bottleneck especially on parallel machines**, very large population
  - Relies on presence of external fitness function which might not exist: e.g. evolving game players

Parent Selection: Tournament Selection (2/3)

Idea for a procedure using only local fitness information:

- Pick *k* **members at random** then select the best of these
- **Repeat to select more** individuals

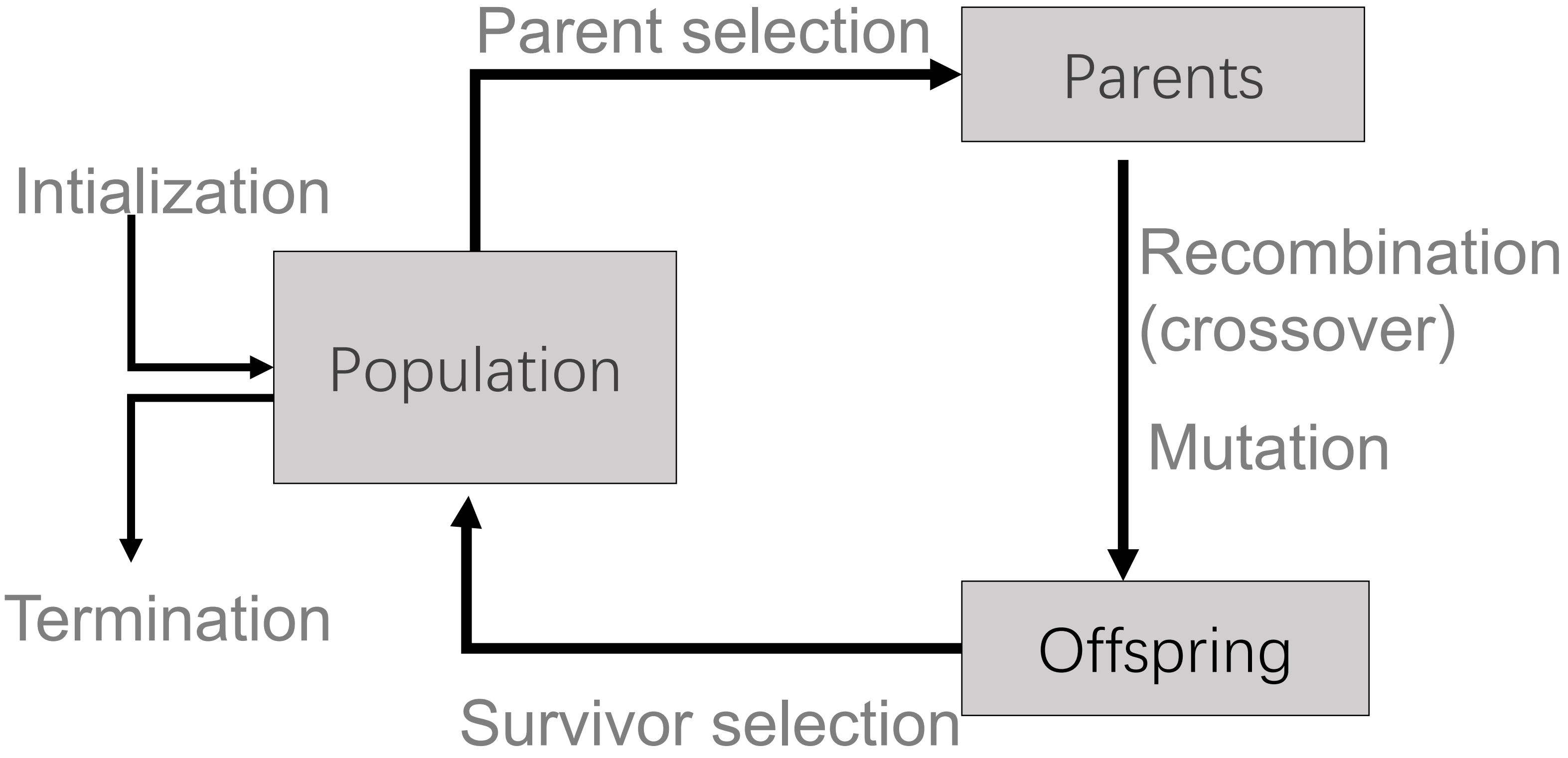


Parent Selection: Tournament Selection (3/3)

- Probability of selecting *i* will depend on:
  - Rank of *i*
  - Size of sample *k*
    - higher *k* increases selection pressure
  - Whether contestants are picked with replacement
    - Picking without replacement increases selection pressure
  - Whether fittest contestant always wins (deterministic) or this happens with probability *p*

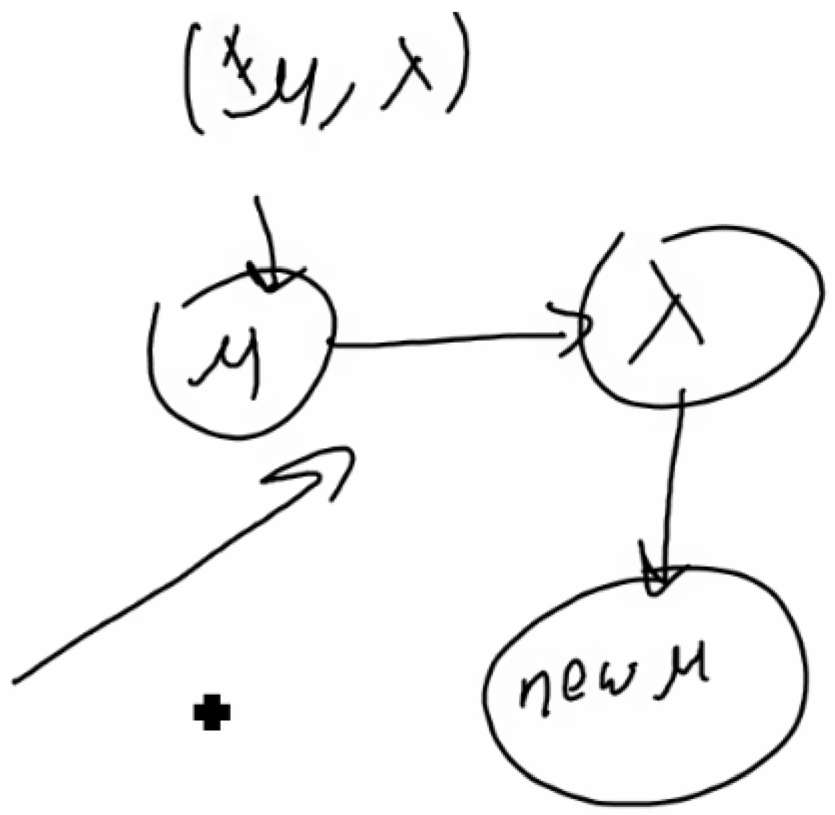
Survivor Selection (Replacement)

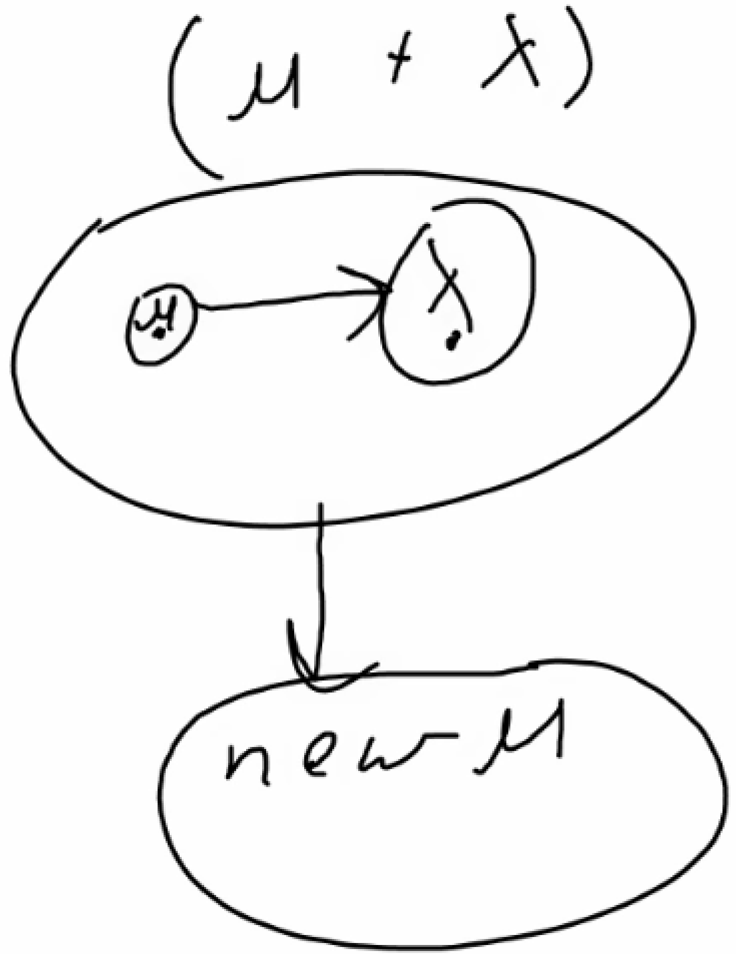
- From a set of μ old solutions and λ offspring: Select a set of μ individuals **forming the next generation**



Fitness-based replacement – examples

- Elitism
  - Always **keep** at least one copy of **the N fittest solution(s)** so far
  - Widely used in most EA-variants
- **(μ,λ)-selection** (best candidates can be lost)
  - based on the set of **children only** ($\lambda > \mu$)
  - choose the **best** μ offspring for next generation
- **(μ+λ)-selection** (elitist strategy)
  - based on the set of **parents and children**
  - choose the **best** μ individuals for next generation
- (μ,λ)-selection may loose the best solution, but is better at leaving local optima


(μ, λ)
μ
λ
new μ


(μ + λ)
μ
λ
new μ

UiO : University of Oslo

institutt for informatikk

IN3050/IN4050, Lecture 4
Evolutionary algorithms 2

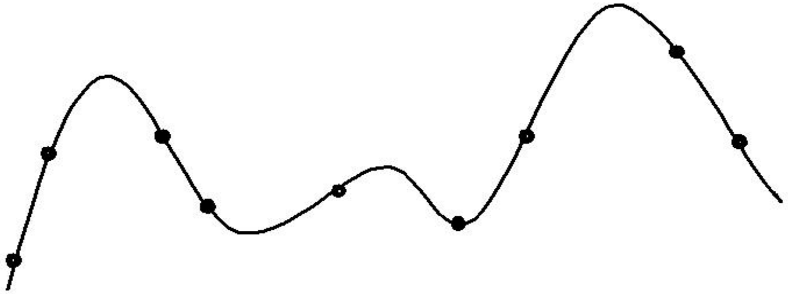
3: Diversity preservation

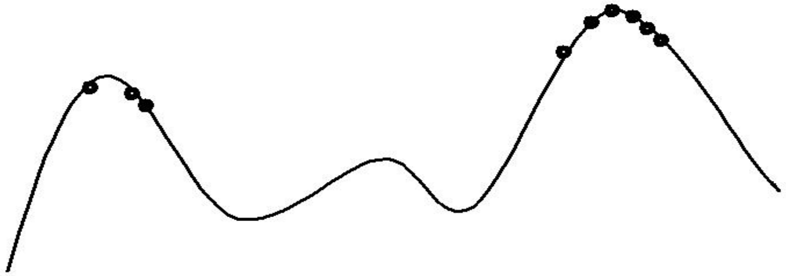
Kai Olav Ellefsen

Next video: Hybridization

Multimodality

- Often might want to identify several possible peaks
- Different peaks may be different good ways to solve the problem.
- We therefore need methods to **preserve diversity** (instead of converging to one peak)





Approaches for Preserving Diversity: Introduction

- Explicit vs implicit
- **Implicit** approaches:
  - Impose an equivalent of **geographical separation**
  - Impose an equivalent of **speciation**
- **Explicit** approaches
  - Make **similar individuals compete** for resources (**fitness**)
  - Make **similar individuals compete** with each other for **survival**

Explicit Approaches for Preserving Diversity: Fitness Sharing (1/2)

- Restricts the number of individuals within a given niche by "sharing" their fitness
- Need to set the size of the niche σ_{share} in either genotype or phenotype space
- run EA as normal but after each generation set

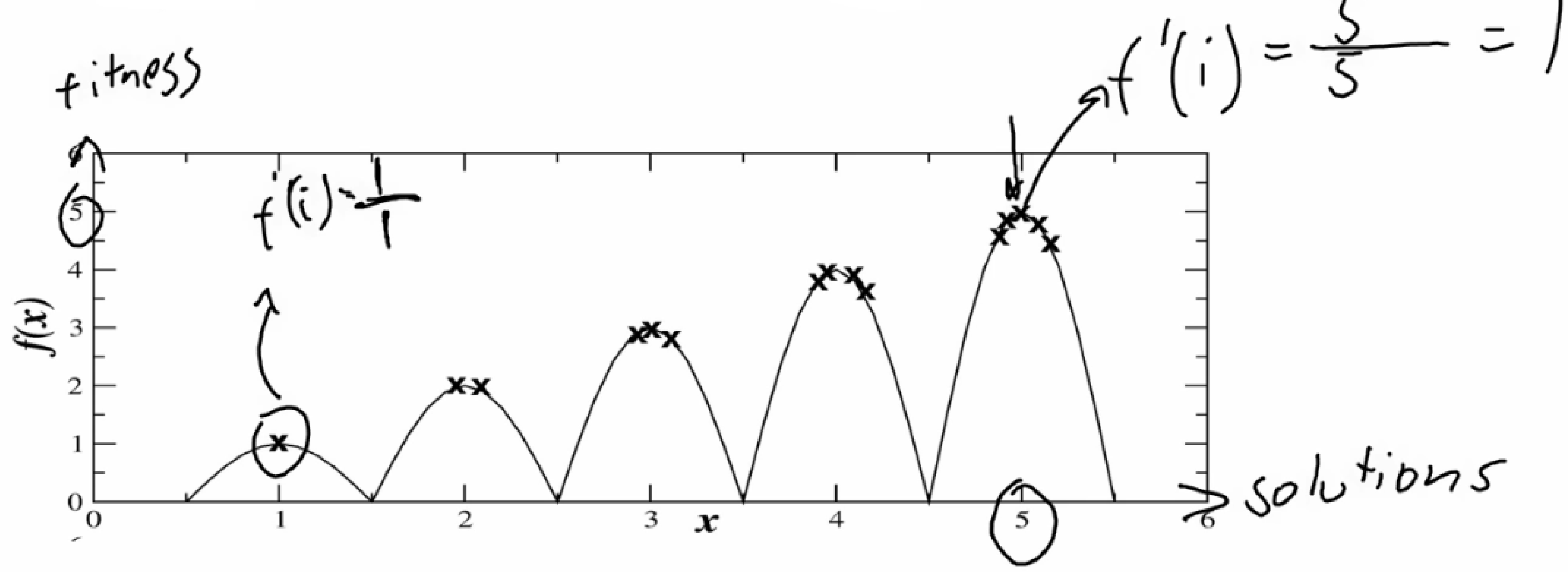
$$f'(i) = \frac{f(i)}{\sum_{j=1}^{\mu} sh(d(i,j))}$$

$$sh(d) = \begin{cases} 1 - d/\sigma & d \le \sigma \\ 0 & otherwise \end{cases}$$

Explicit Approaches for Preserving Diversity: Fitness Sharing (2/2)

$$f'(i) = \frac{f(i)}{\sum_{j=1}^{\mu} sh(d(i,j))}$$

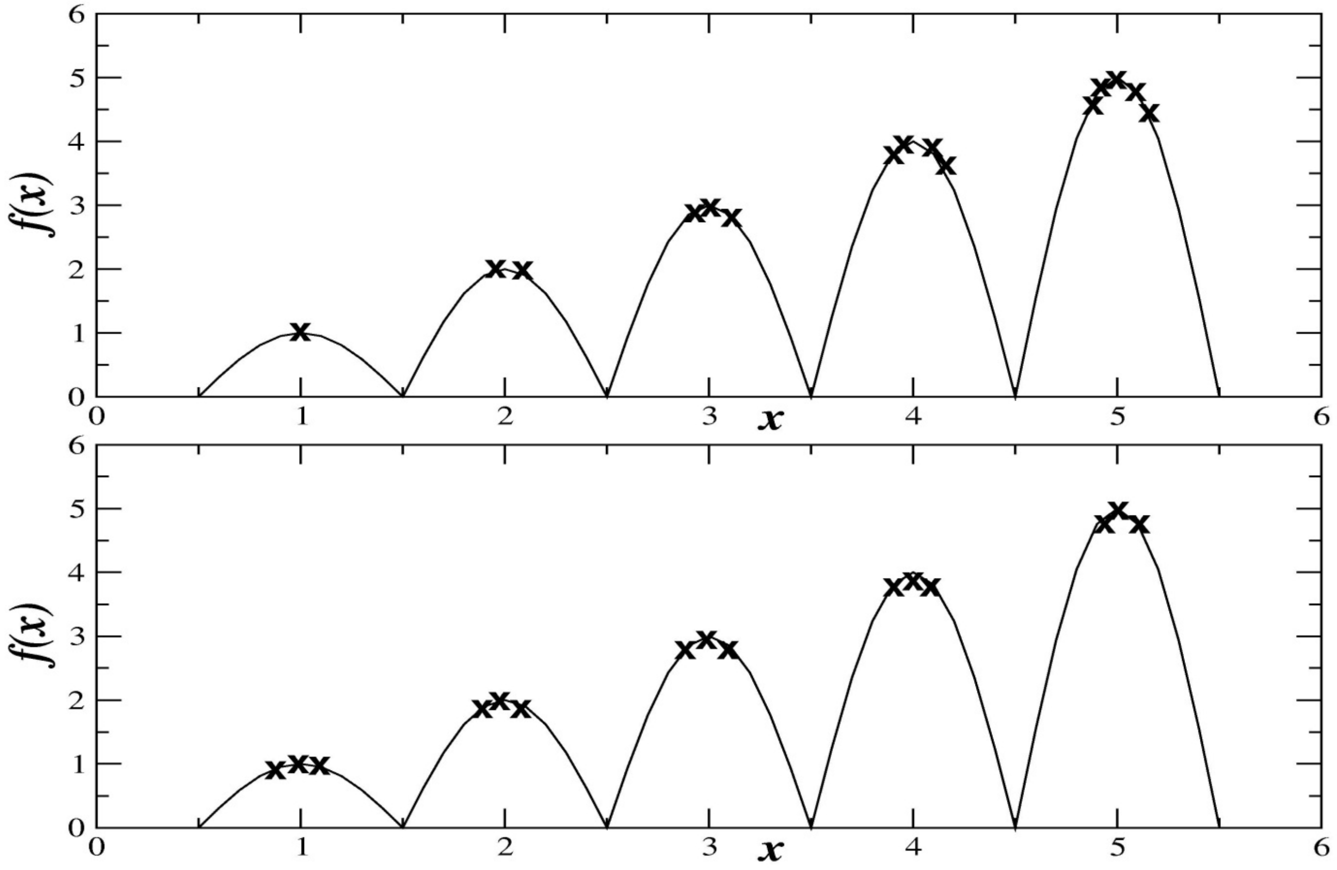
$$sh(d) = \begin{cases} 1 - d/\sigma & d \leq \sigma \\ 0 & otherwise \end{cases}$$



Explicit Approaches for Preserving Diversity: Crowding

- Idea: New individuals replace *similar* individuals
- Randomly shuffle and pair parents, produce 2 offspring
- Each offspring competes with their **nearest** parent for survival (using a distance measure)
- Result: Even distribution among niches.

Explicit Approaches for Preserving Diversity: Crowding vs Fitness sharing



Fitness Sharing

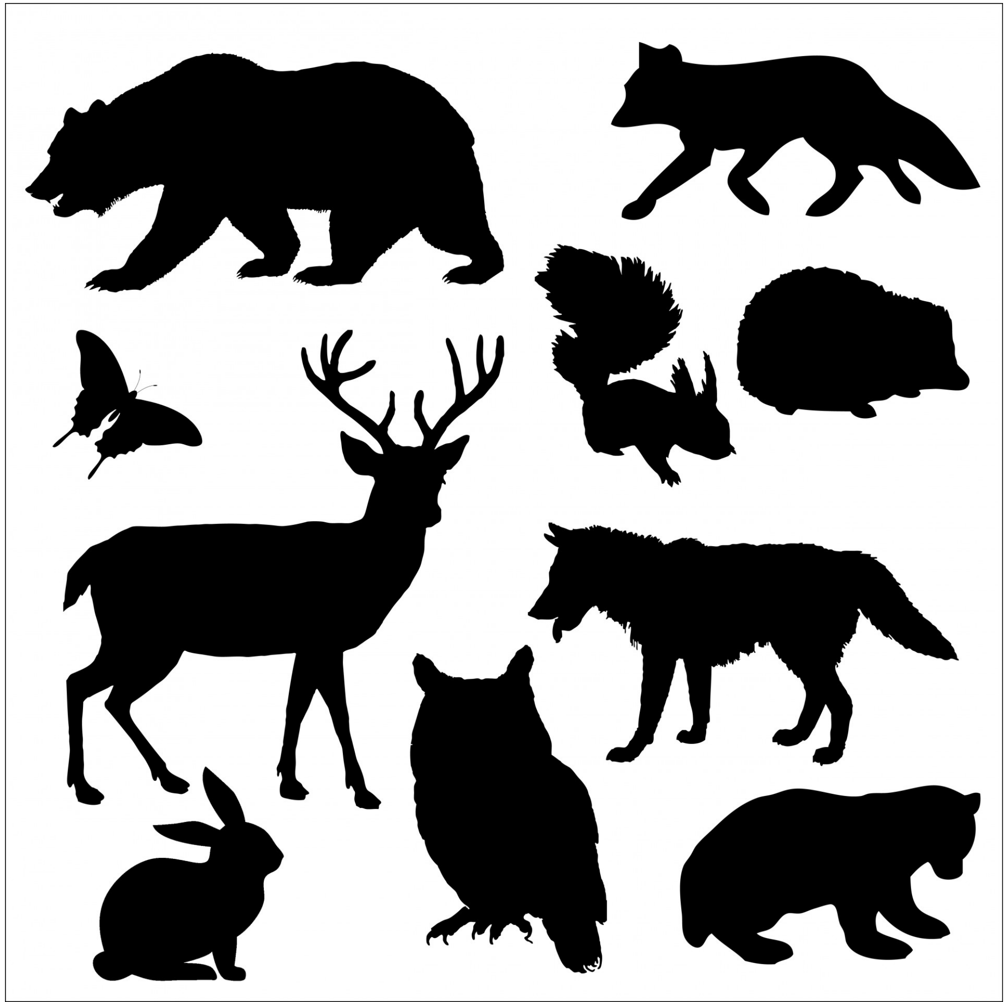
Crowding

Observe the number of individuals per niche

Implicit Approaches for Preserving Diversity: Automatic Speciation

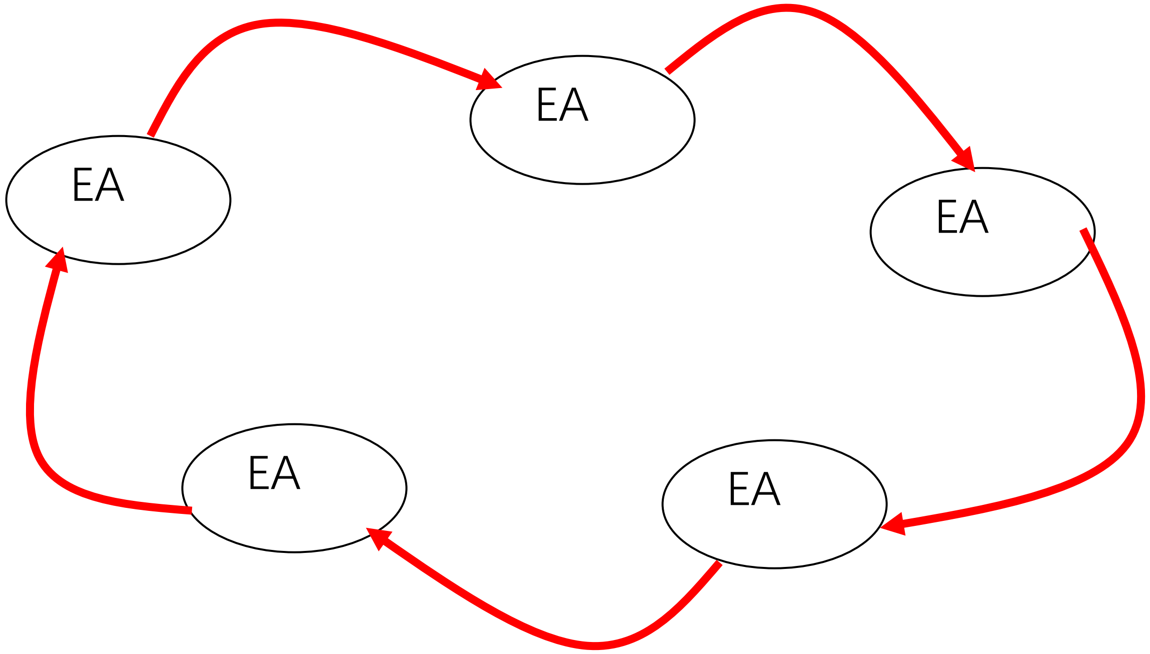
011001 1 2 4 5 3

- Either only mate with genotypically / phenotypically similar members or
- Add species-tags to genotype
  - initially randomly set
  - when selecting partner for recombination, only pick members with a good match



Implicit Approaches for Preserving Diversity: Geographical Separation

- "Island" Model Parallel EA
- Periodic migration of individual solutions between populations



Implicit Approaches for Preserving Diversity: "Island" Model Parallel EAs

- Run multiple populations in parallel
- After a (usually fixed) number of generations (an *Epoch*), exchange individuals with neighbours
- Repeat until ending criteria met
- Partially inspired by parallel/clustered systems

UiO : University of Oslo

institutt for informatikk

IN3050/IN4050, Lecture 4
Evolutionary algorithms 2

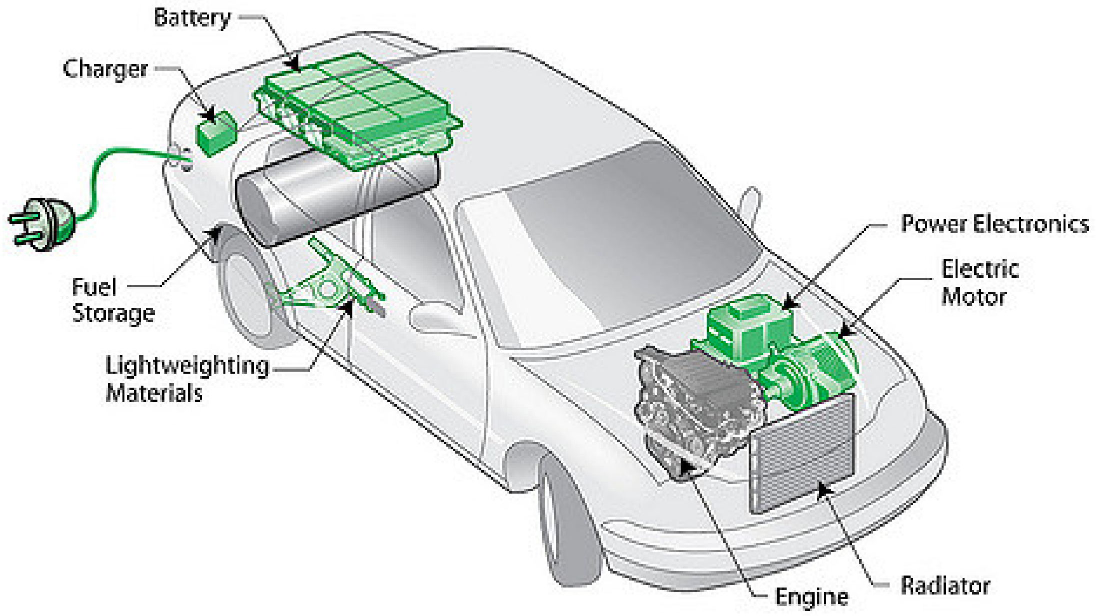
4: Hybridization

Kai Olav Ellefsen

Next video: Multi-objective optimization

Chapter 10:
Hybridisation with Other Techniques: Memetic Algorithms

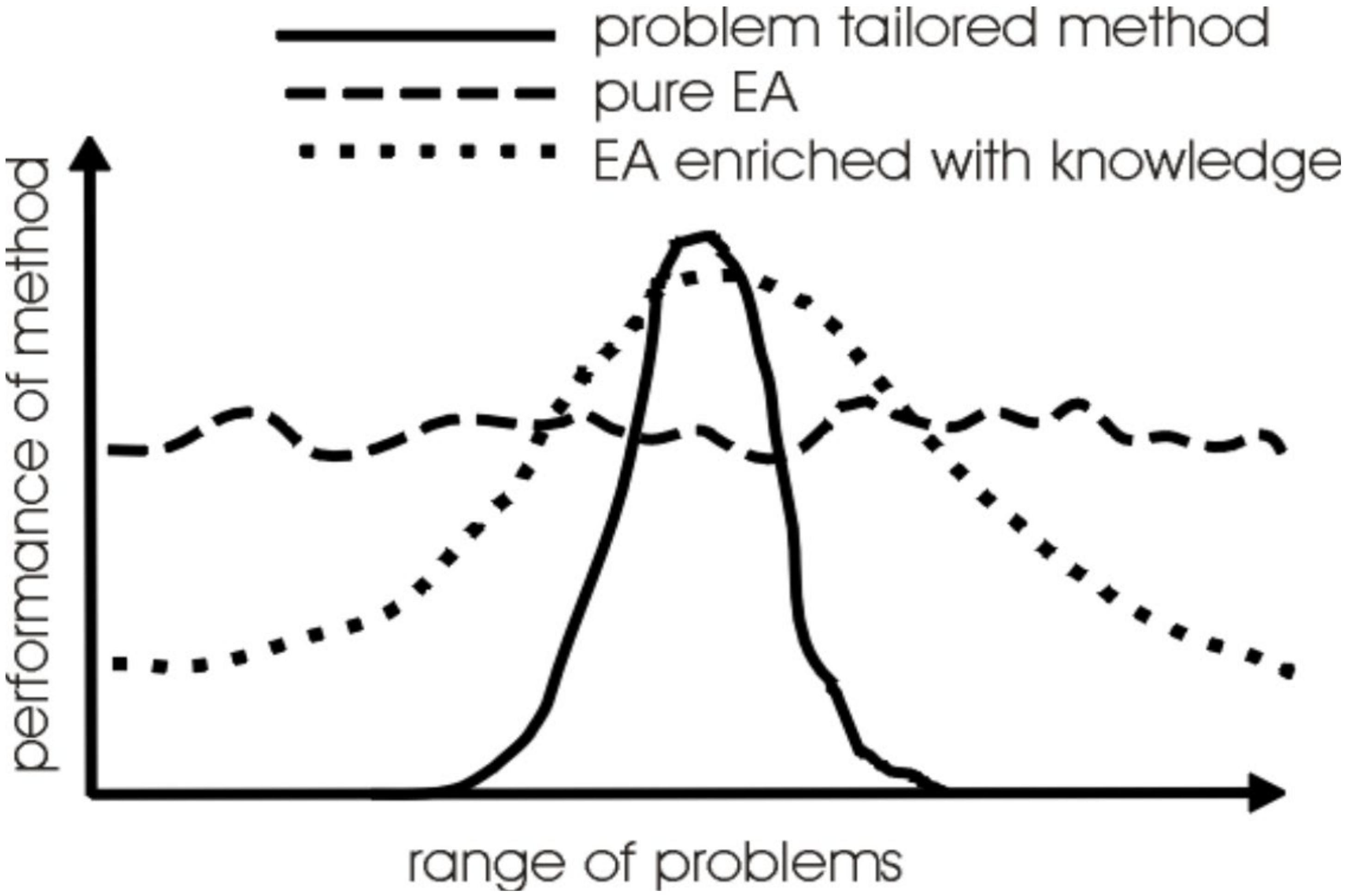
1. Why Hybridise?
2. What is a Memetic Algorithm?
3. Local Search
   - Lamarckian vs. Baldwinian adaptation
4. Where to hybridise



1. Why Hybridise

- Might be looking at **improving on existing techniques** (non-EA)
- Might be looking at **improving EA search** for good solutions

1. Why Hybridise

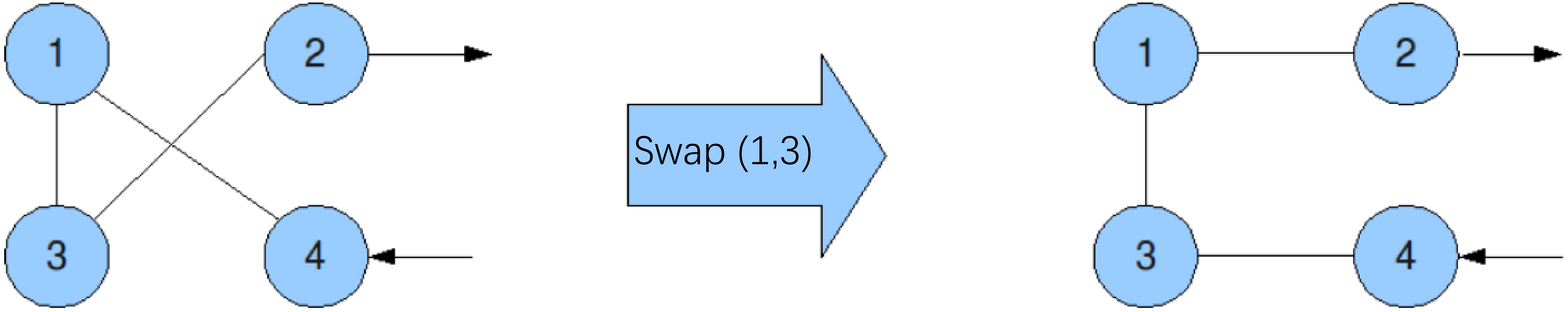


2. What is a Memetic Algorithm?

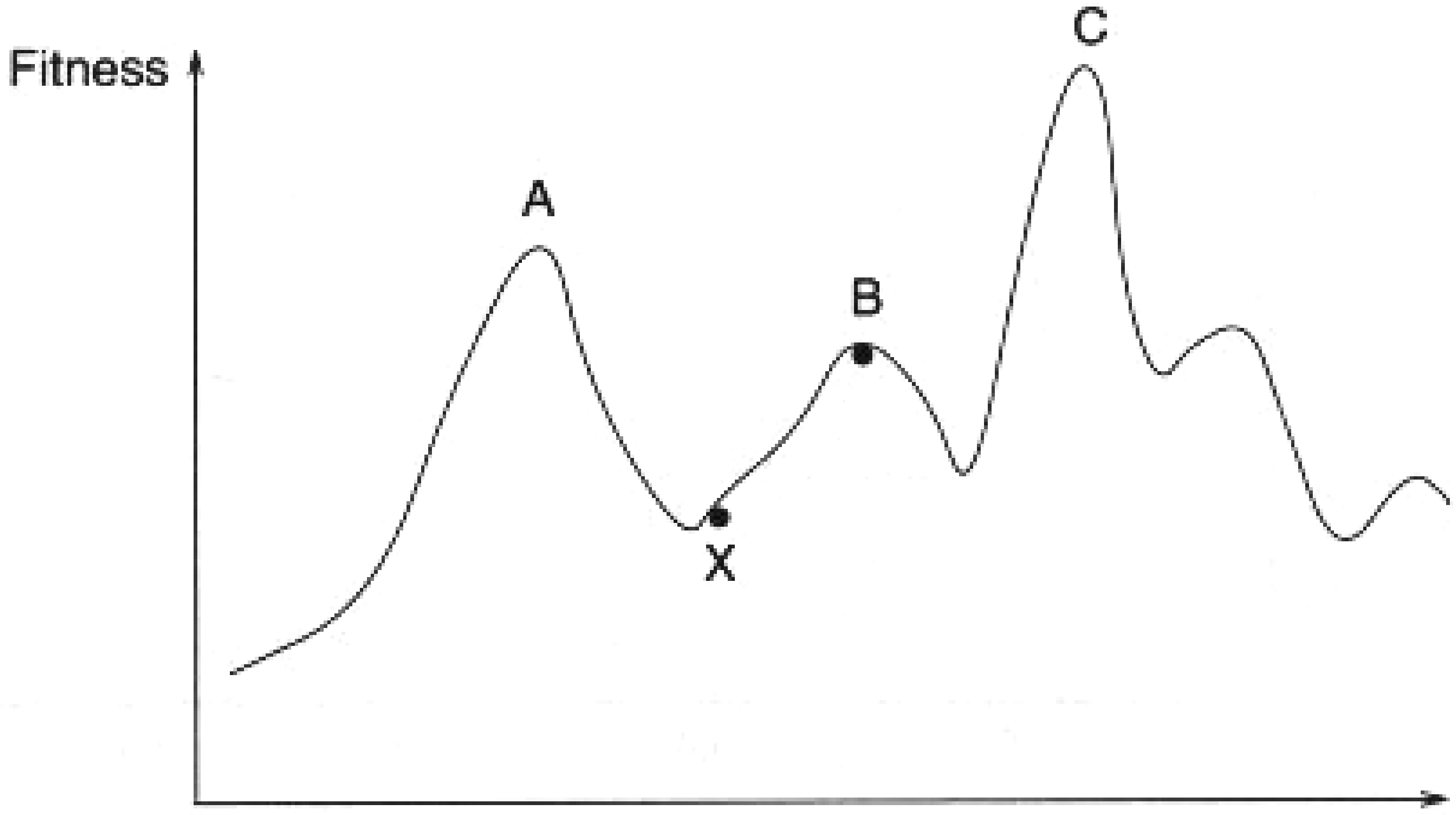
- The combination of Evolutionary Algorithms with **Local Search Operators** that work within the EA loop has been termed "**Memetic Algorithms**"
- Term also applies to EAs that use **instance-specific knowledge**
- Memetic Algorithms have been shown to be orders of magnitude **faster and more accurate** than EAs on some problems, and are the "state of the art" on many problems

3. Local Search: Main Idea

- Make a small, but intelligent (problem-specific), change to an existing solution
- If the change improves it, keep the improved version
- Otherwise, keep trying small, smart changes until it improves, or until we have tried all possible small changes

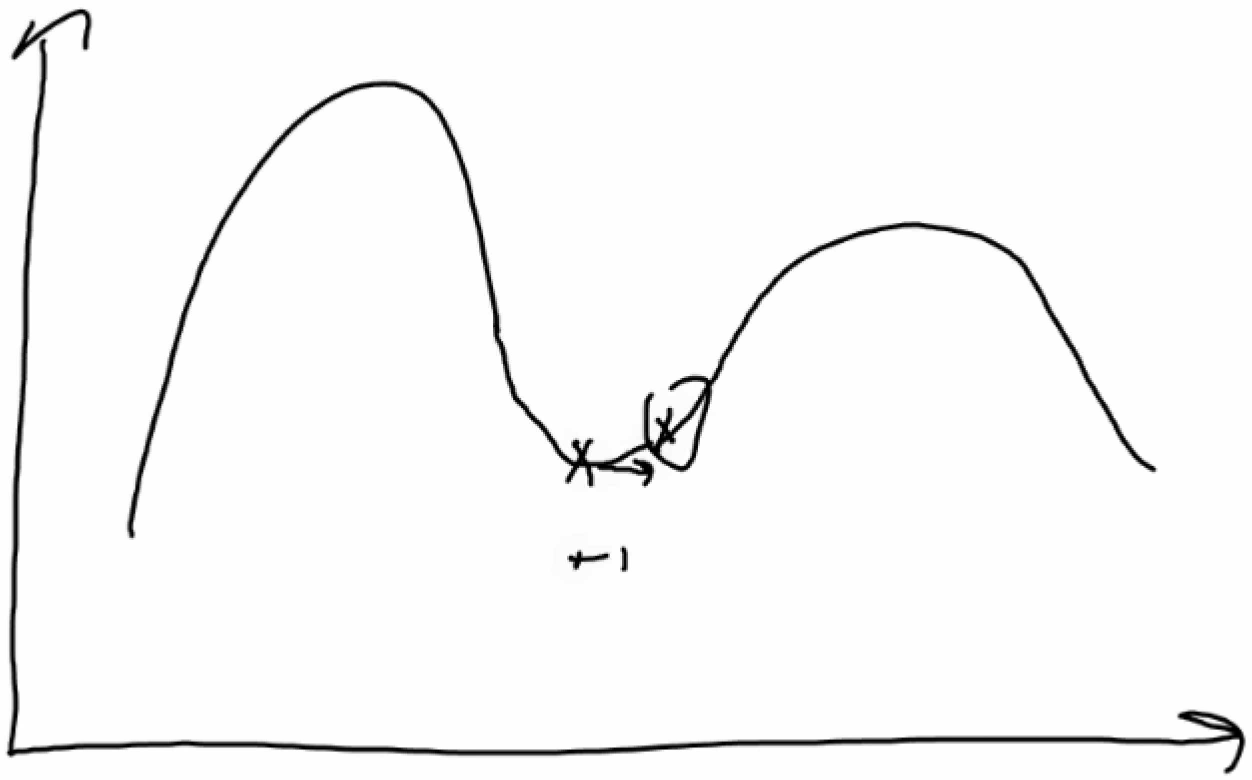


3. Local Search: Motivation



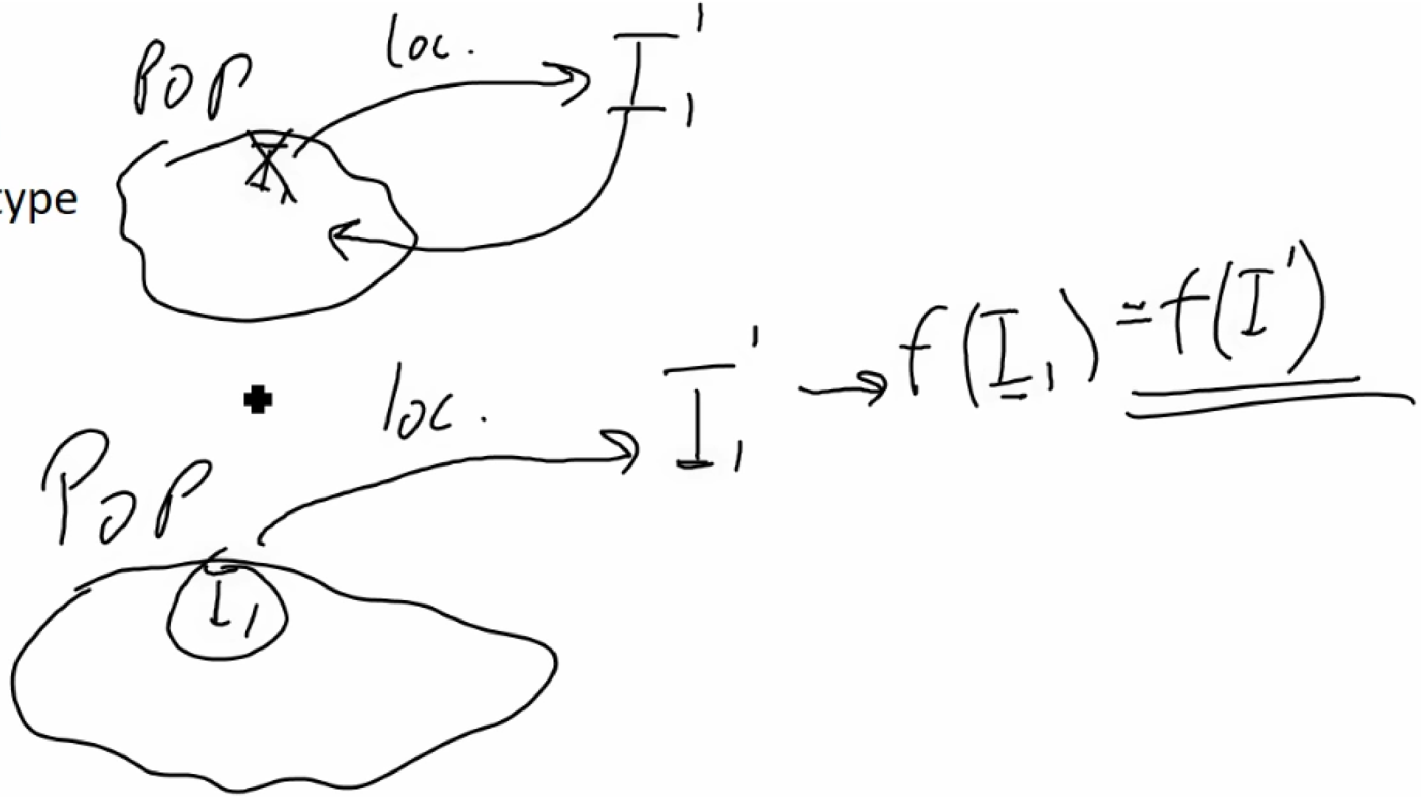
3. Local Search: Pivot Rules

- Is the neighbourhood searched randomly, systematically or exhaustively ?
- does the search stop as soon as a fitter neighbour is found (*Greedy Ascent*)
- or is the whole set of neighbours examined and the best chosen (*Steepest Ascent*)
- of course there is no one best answer, but some are quicker than others to run ……..



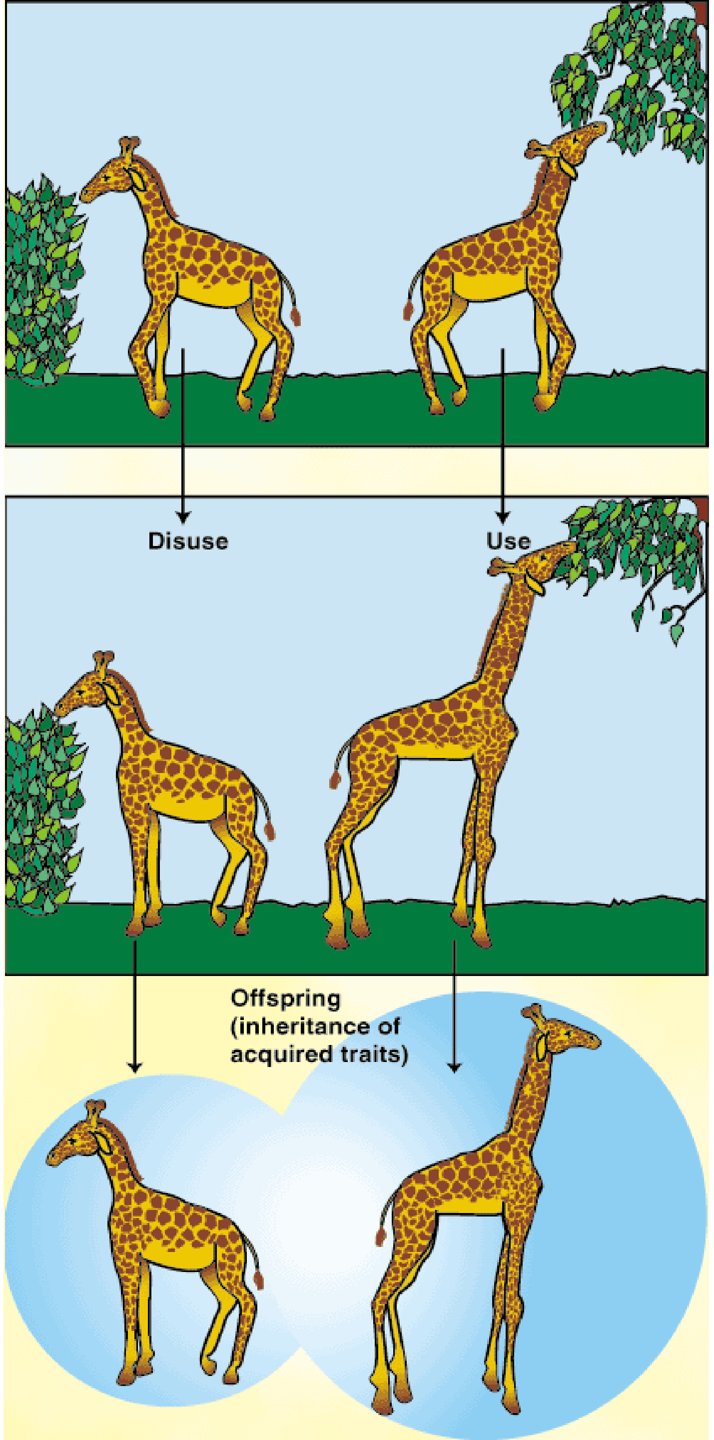
4. Local Search and Evolution

- Do offspring inherit what their parents have "learnt" in life?
  - Yes - Lamarckian evolution
    - Improved fitness and genotype
  - No - Baldwinian evolution
    - Improved fitness only



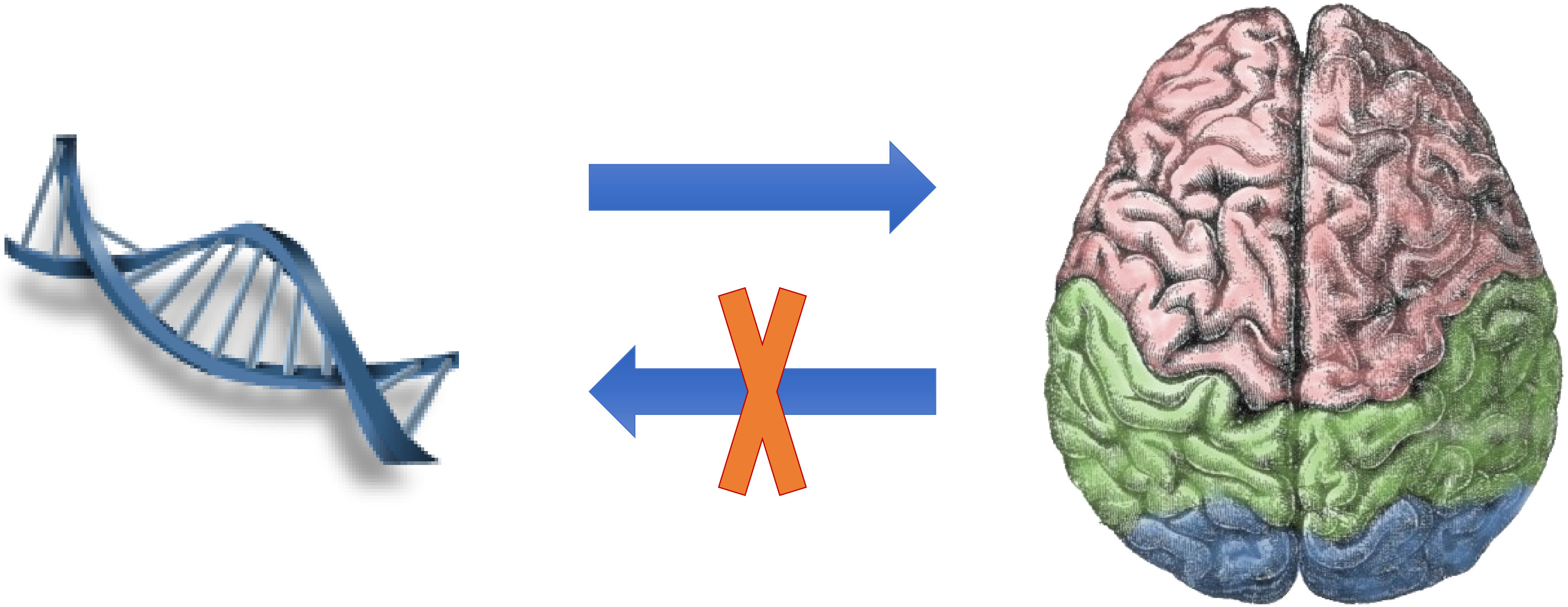
4. Lamarckian Evolution

- Lamarck, 1809: Traits acquired in parents' lifetimes can be inherited by offspring
- This type of direct inheritance of acquired traits is not possible, according to modern evolutionary theory



(Image from sparknotes.com)

4. Inheriting Learned Traits?

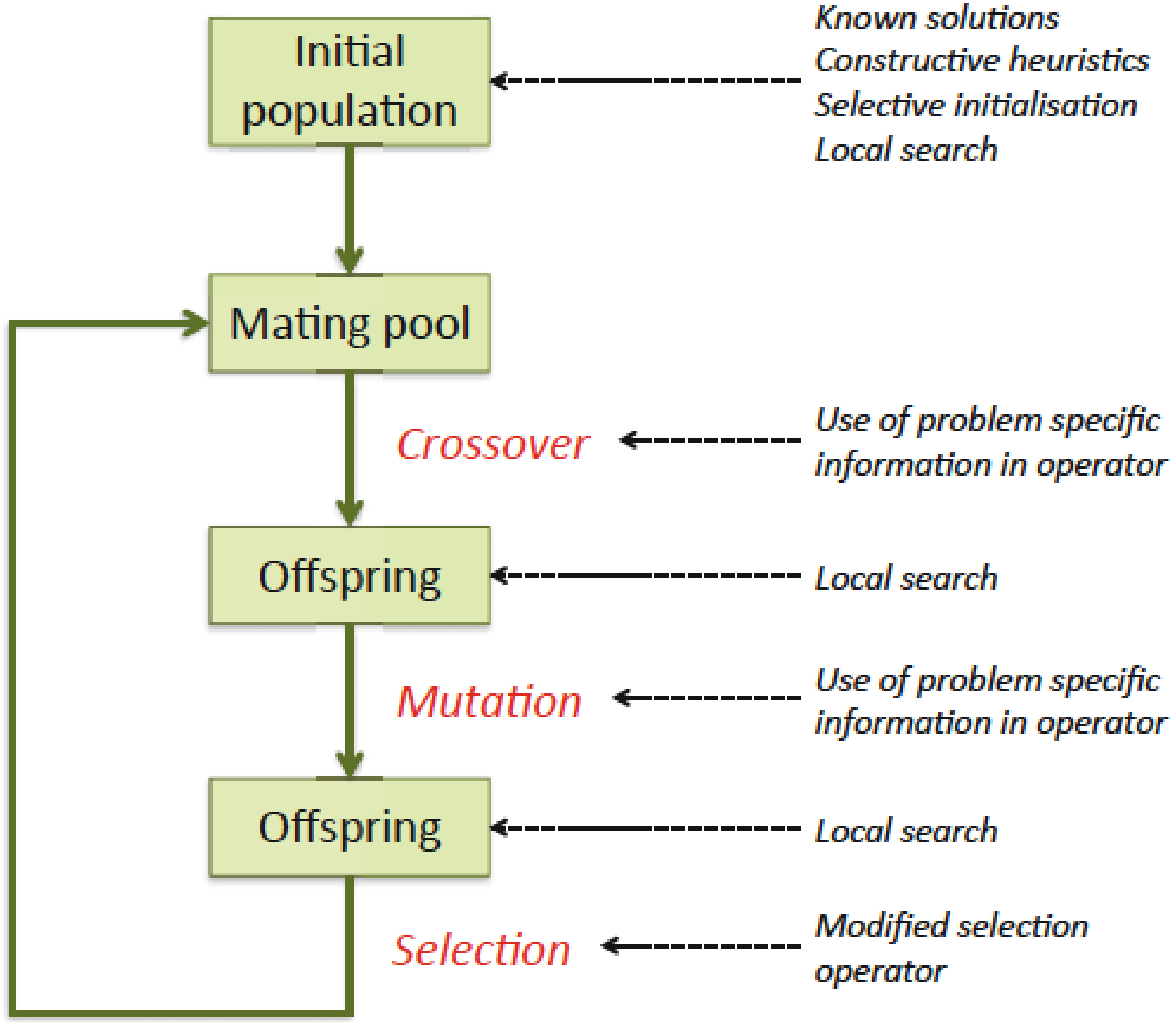


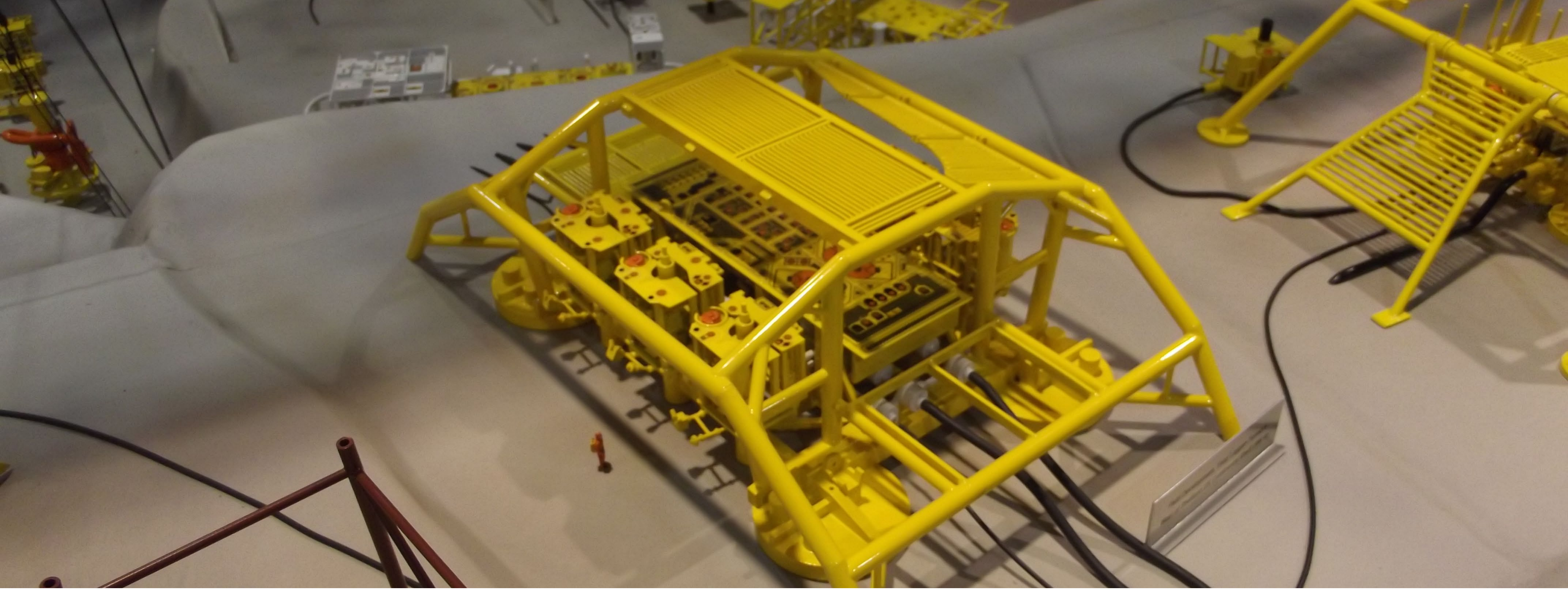
(Brain from Wikimedia Commons)

4. Local Search and Evolution

- In practice, most recent Memetic Algorithms use:
  - Pure Lamarckian evolution, or
  - A stochastic mix of Lamarckian and Baldwinian evolution

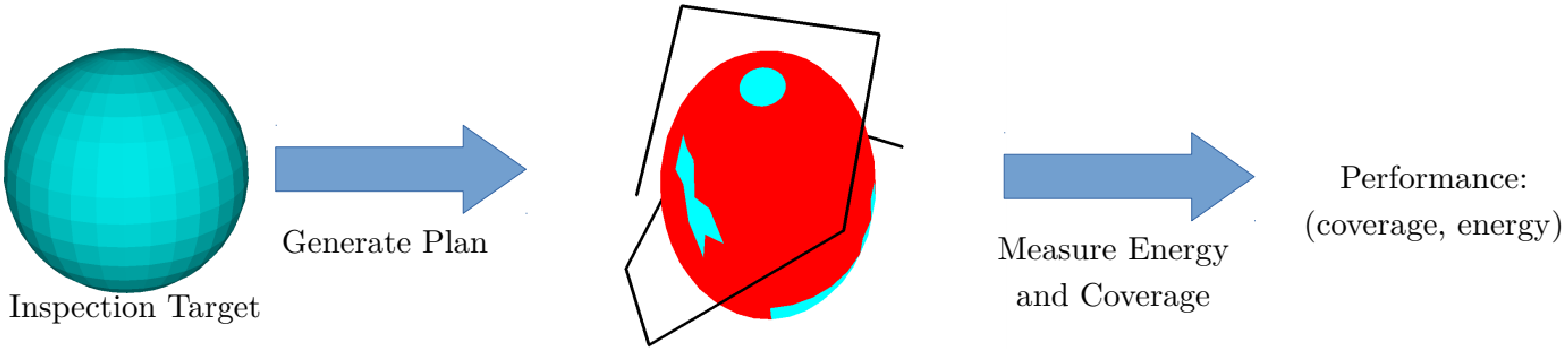
5. Where to Hybridise:





Hybridization - Example

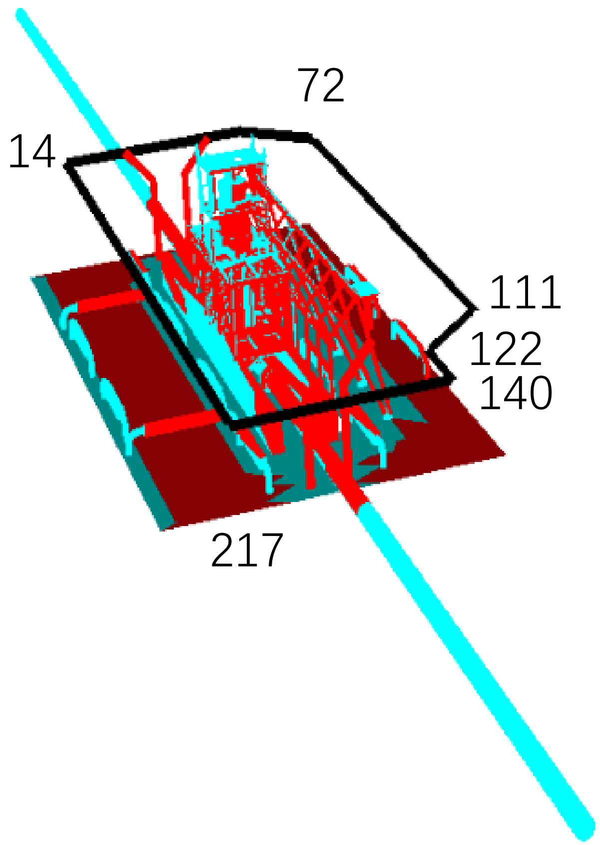
Reminder: Optimizing Inspection Plans with an EA



Each Plan is a Sequence of Waypoint IDs

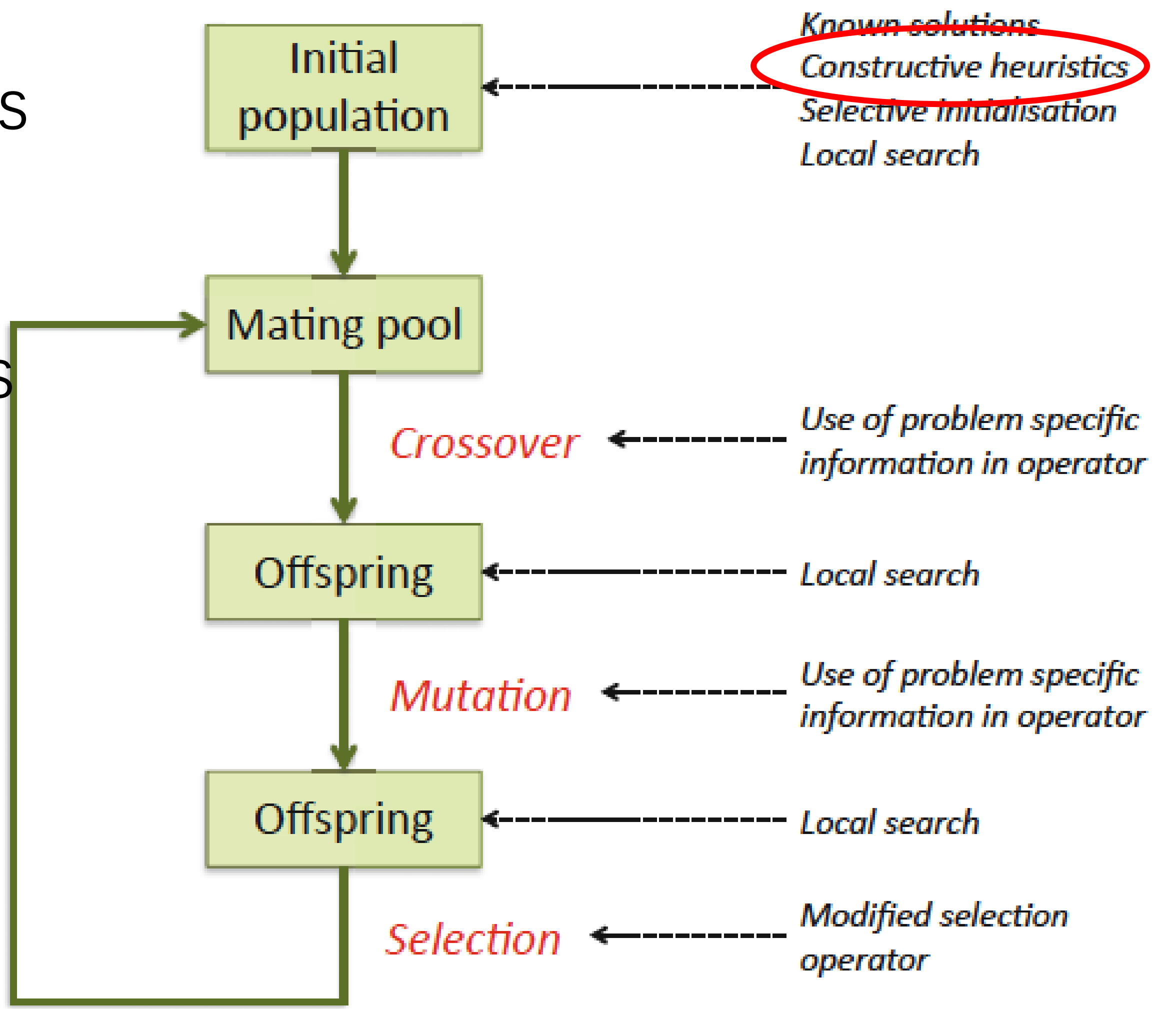
- E.g. the plan [14, 72, 111, 122, 140, 217]
- Would you hybridize the search for plans? How could we insert some knowledge in the search?

decode

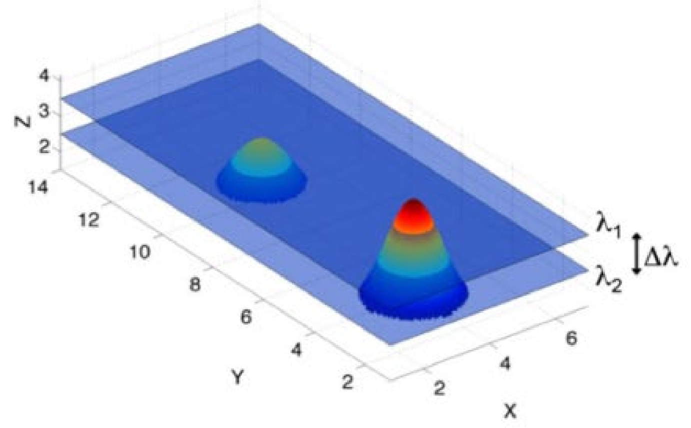


Constructive Heuristics

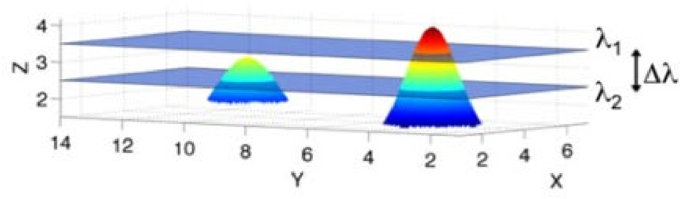
- Idea: Find examples of OK plans to **seed** the initial population
- The plans don't have to be great, just better starting points than random plans
- What could be a simple, OK plan for inspections?



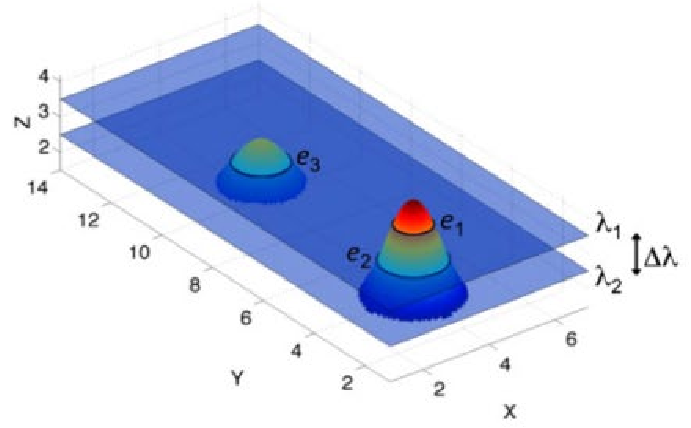
My idea: Make simple, circling solutions



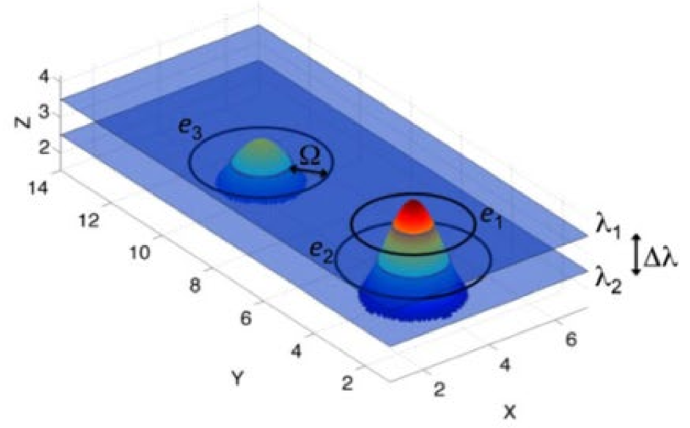
(a) Slice planes (slanted view).



(b) Slice planes (side view).



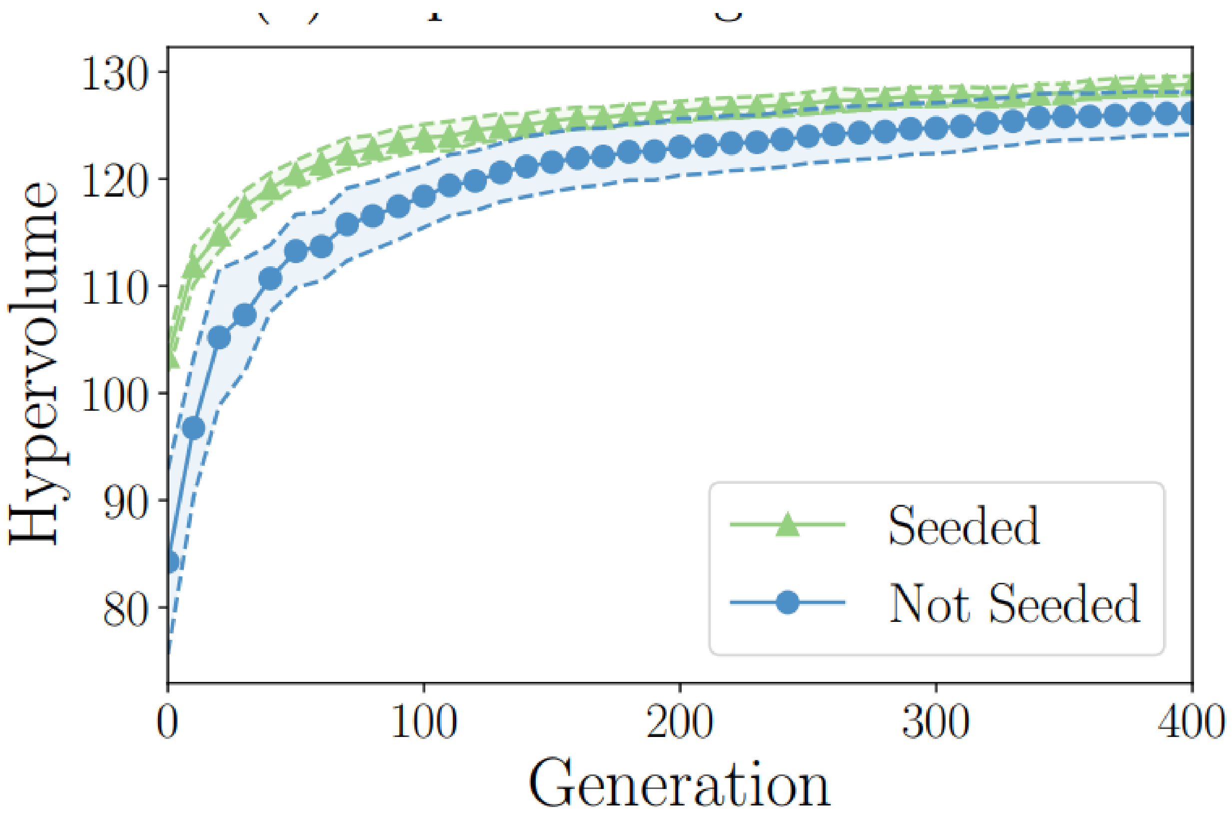
(c) Intersection edges (see Algorithm 1, line 4).



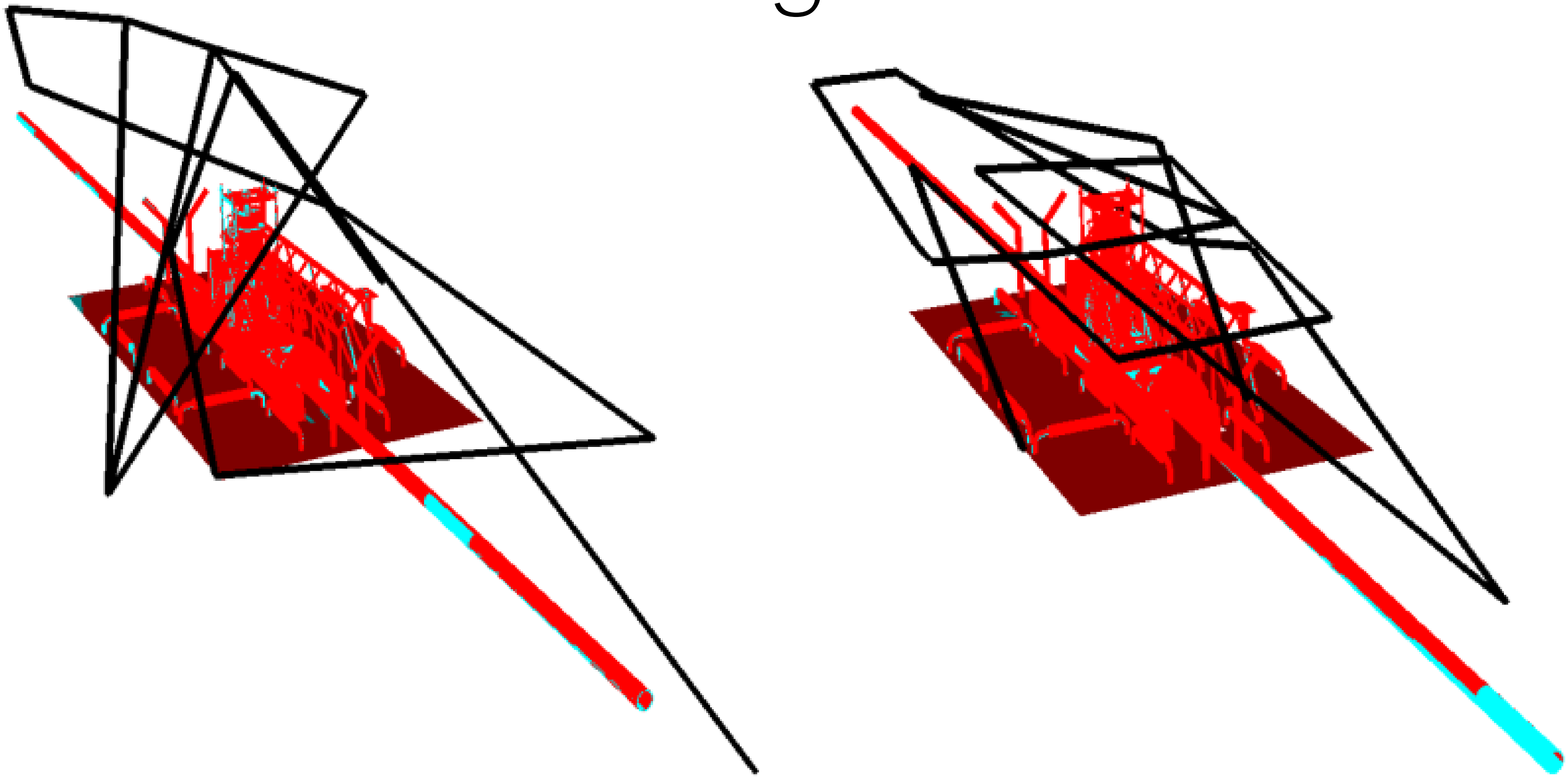
(d) Offset edges (see Algorithm 1, line 6).

E. Galceran, R. Campos, Narcis Palomeras, D. Ribas, M. Carreras, and P. Ridao. Coverage Path Planning with Real-time Replanning and Surface Reconstruction for Inspection of Three-dimensional Underwater Structures using Autonomous Underwater Vehicles. *Journal of Field Robotics*, 32(7):952–983, 2014.

Did this kind of seeding work?



Did this kind of seeding work?



(a) Plan evolved without seeding – scores (0.23, 71.9).

(b) Plan evolved with seeding – scores (0.21, 73.2).

Figure 14: **The results on evolved plan structure of seeding.** When seeding (right), evolved plans retain much of the clean structures of the seed plans, whereas evolutionary runs without seeding (left) have a more irregular structure.

Hybrid Algorithms Summary

- It is **common** practice **to hybridise EA’s** when using them in a real world context.
- This may involve the use of operators from other algorithms which have already been used on the problem, or the incorporation of domain-specific knowledge
- Memetic algorithms have been shown to be orders of magnitude faster and more accurate than EAs on some problems, and are the “state of the art” on many problems

UiO : University of Oslo

institutt for informatikk

IN3050/IN4050, Lecture 4
Evolutionary algorithms 2

5: Multi-objective optimization

Kai Olav Ellefsen

Chapter 12: Multiobjective Evolutionary Algorithms

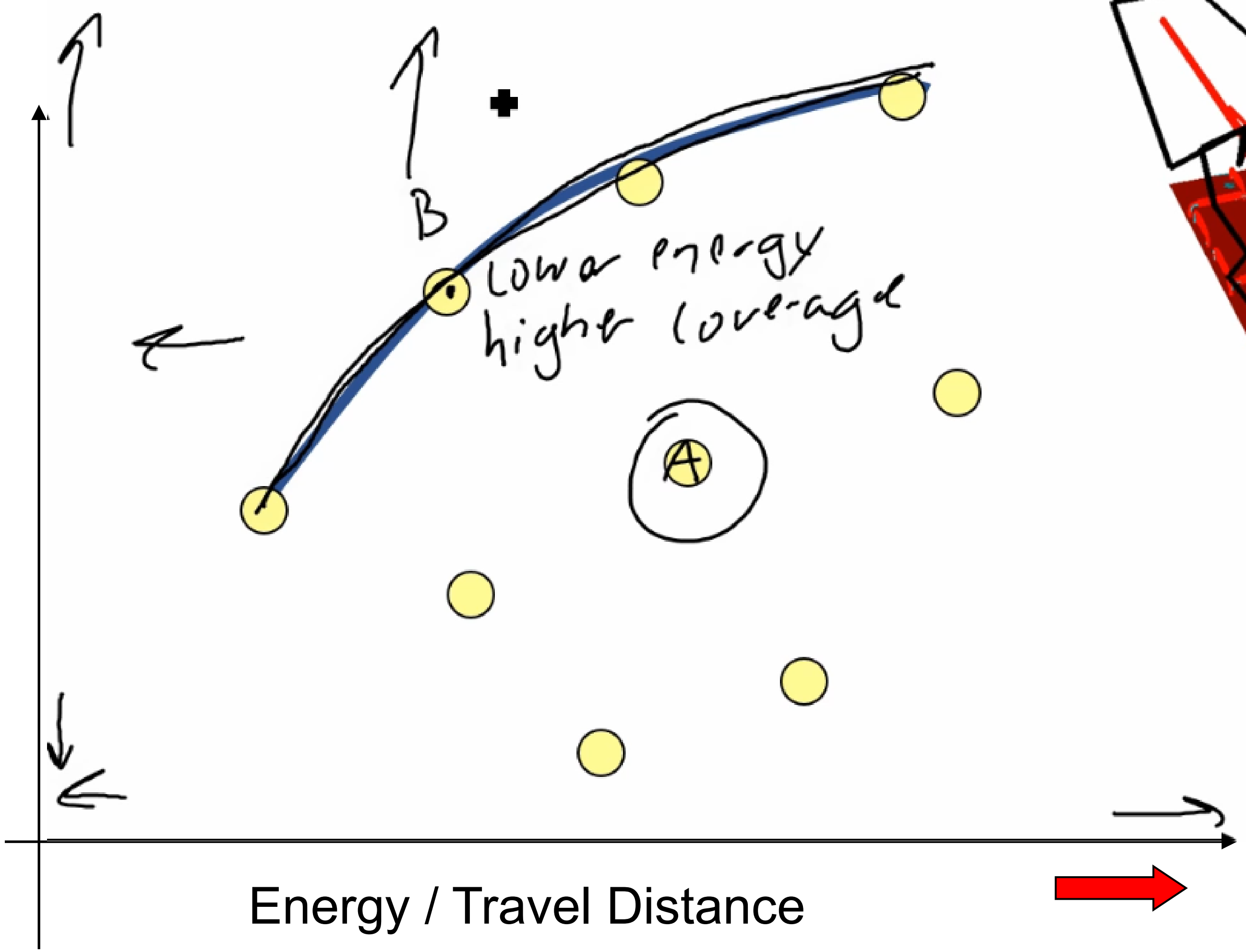
- **Multiobjective optimisation problems** (MOP)
  - Pareto optimality
- EC approaches
  - Selection operators
  - Preserving diversity

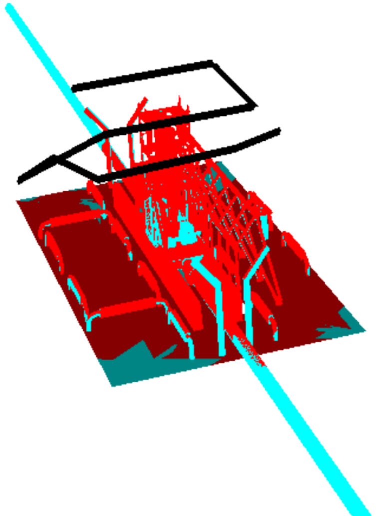
Multi-Objective Problems (MOPs)

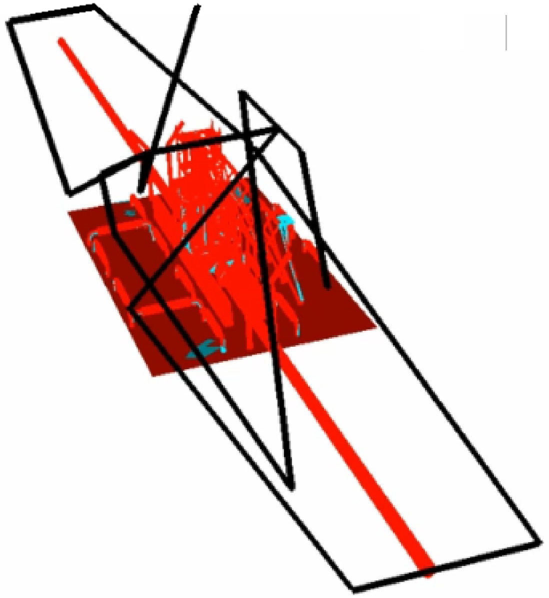
- Wide range of problems can be categorised by the presence of a number of ***n*** **possibly conflicting objectives**:
  - buying a car: speed vs. price vs. reliability
  - engineering design: lightness vs. strength
  - Inspecting infrastructure: Energy usage vs completeness
- Two problems:
  - finding set of good solutions
  - choice of best for the particular application

An example: Inspecting Infrastructure

Completeness







Two approaches to multiobjective optimisation

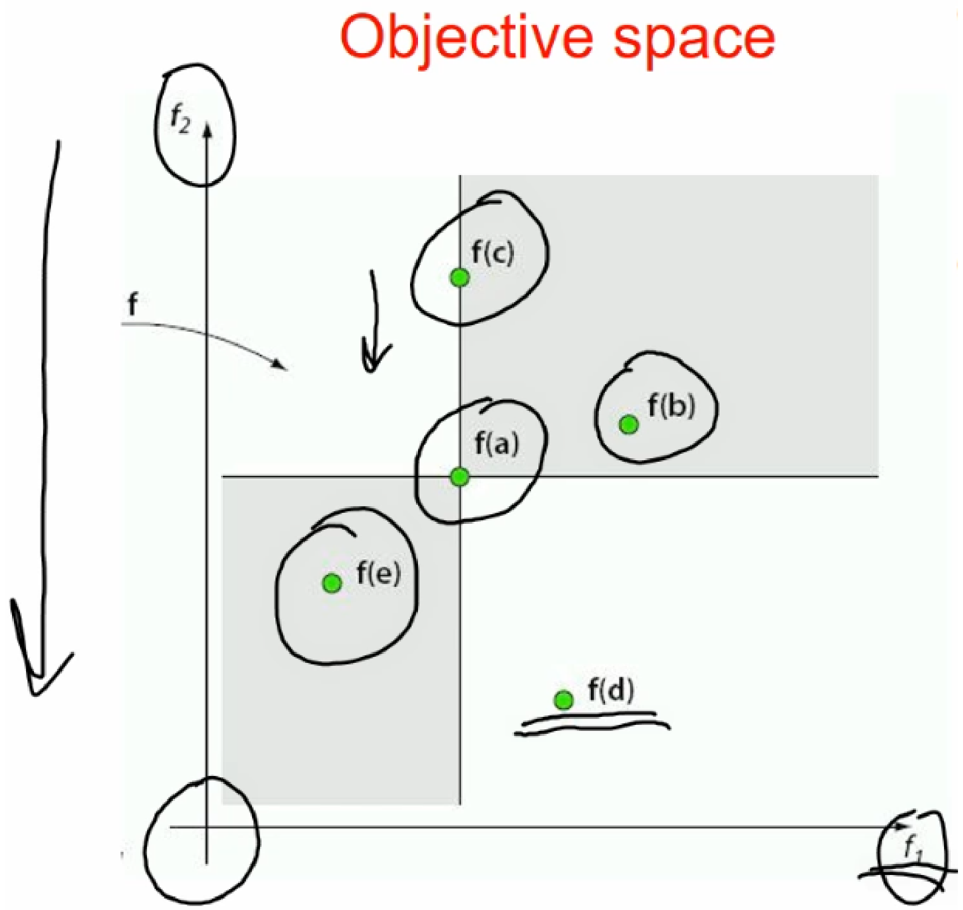
$$\text{fitn} = \boxed{X \cdot \text{price} + Y \cdot \text{size}}$$

70%   30%

- **Weighted sum (scalarisation):**
  - transform into **a single objective** optimisation method
  - compute a weighted sum of the different objectives

- **A set of multi-objective solutions (Pareto front):**
  - The **population-based** nature of EAs used to *simultaneously* search for a set of points approximating Pareto front

Comparing solutions

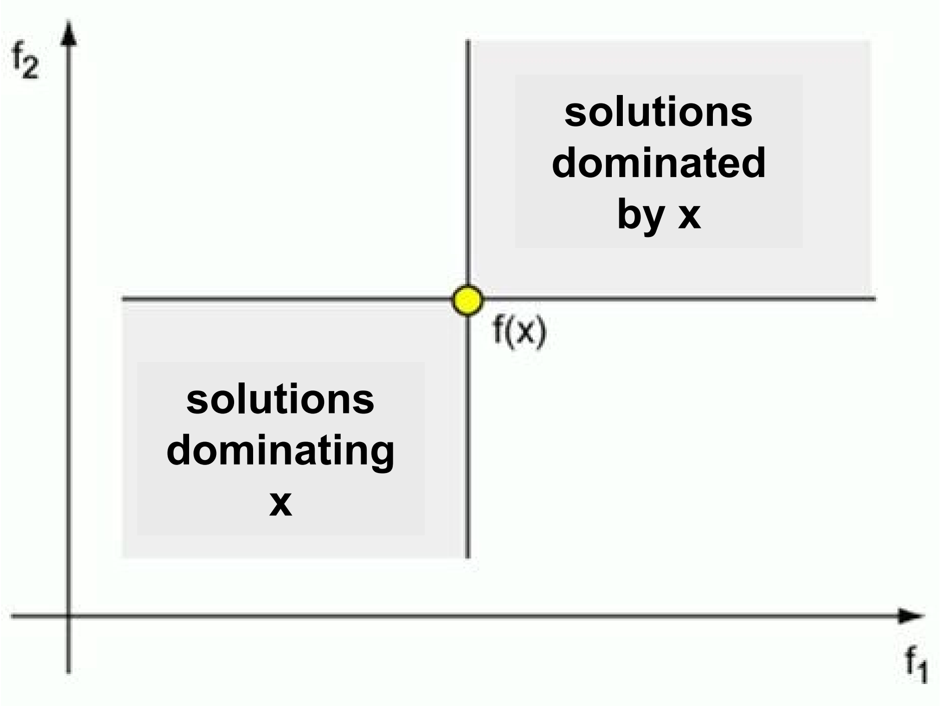


- Optimisation task:
  Minimize both f_1 and f_2

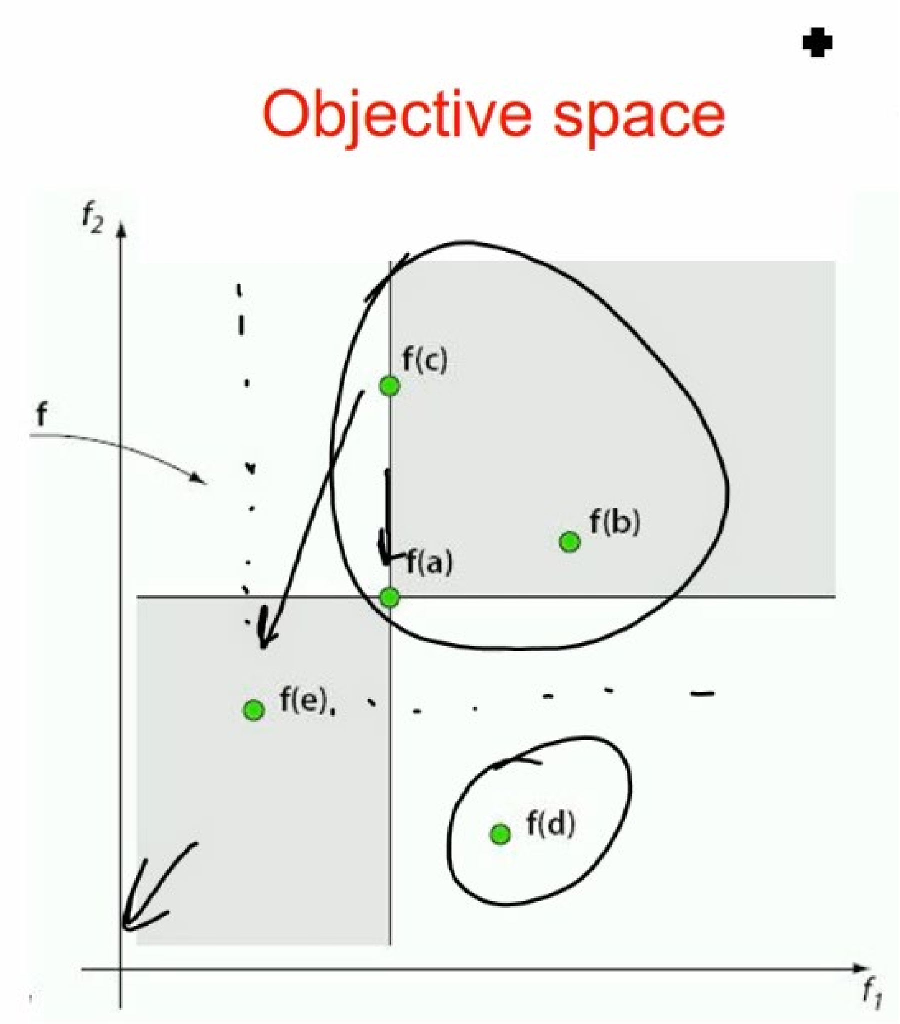
- Then:
  a is better than b
  a is better than c
  a is worse than e
  a and d are incomparable

Dominance relation

- Solution x dominates solution y, ($x \preceq y$), if:
  - x is better than y in at least one objective,
  - x is not worse than y in all other objectives



Dominance relation

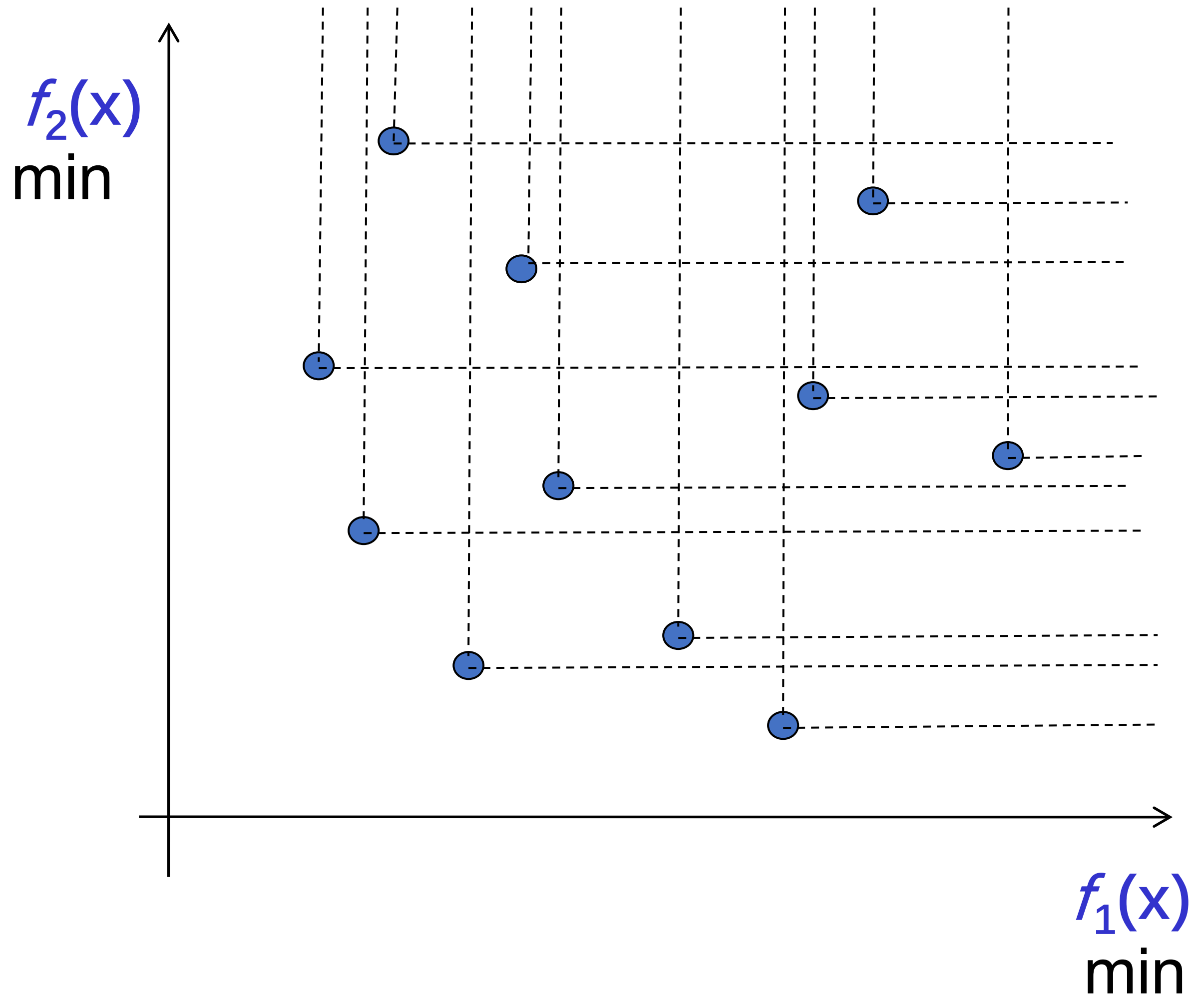


- Who is c dominated by?
- Who does e dominate?

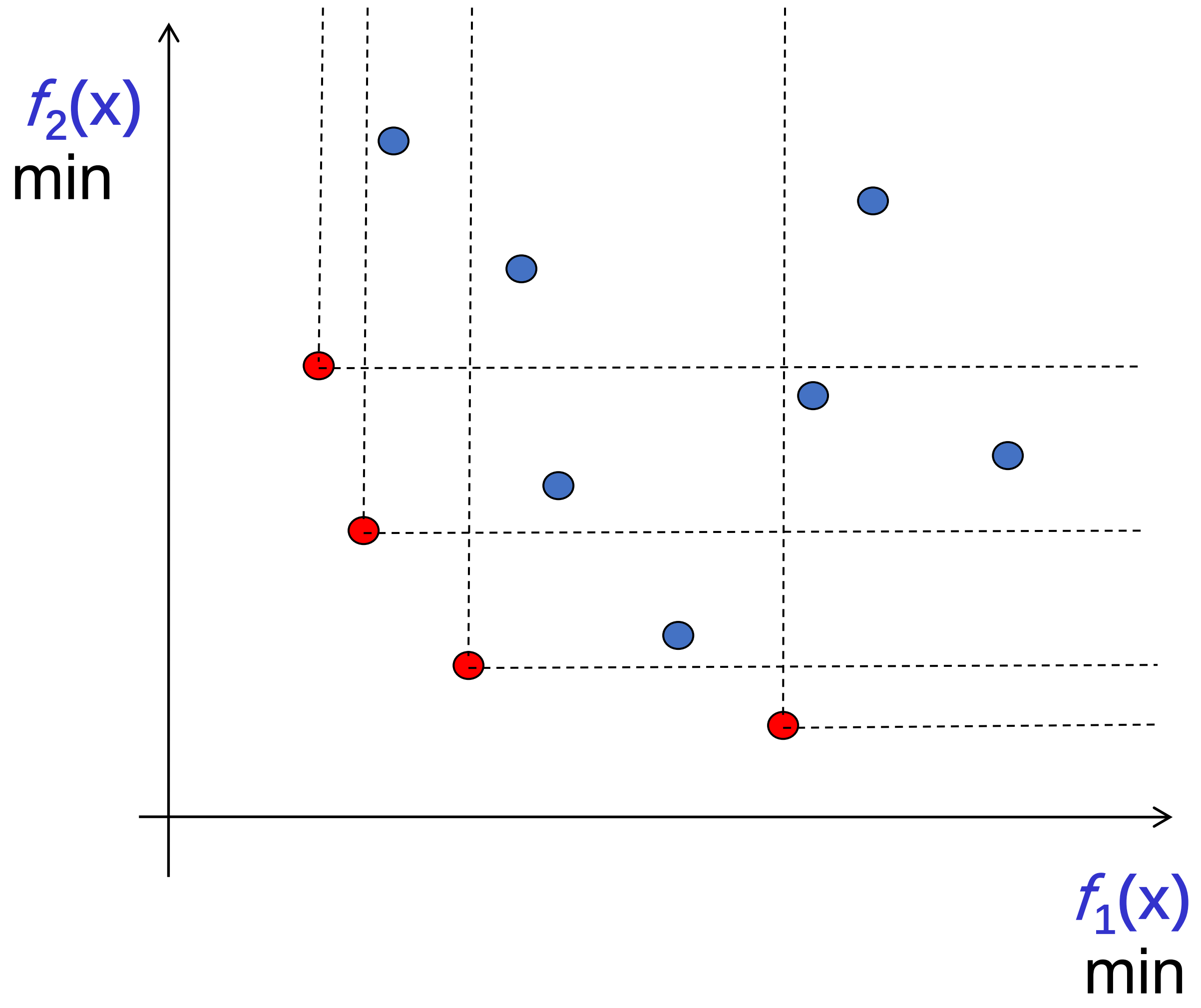
Pareto optimality

- Solution x is **non-dominated** among a set of solutions Q if no solution from Q dominates x
- A set of non-dominated solutions from the entire feasible solution space is the **Pareto set**, or **Pareto front**, its members Pareto-optimal solutions

Which are non-dominated?

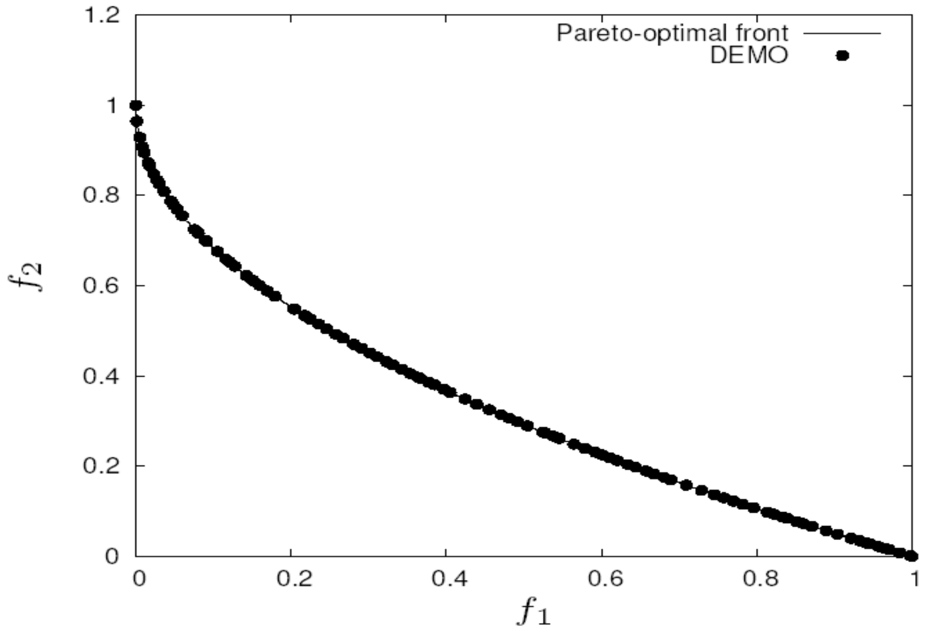


Which are non-dominated?



Goal of multiobjective optimisers

- Find a set of non-dominated solutions (**approximation set**) following the criteria of:
  - **convergence** (as close as possible to the Pareto-optimal front),
  - **diversity** (spread, distribution)



EC approach: Requirements

1. Way of assigning fitness and **selecting individuals**,
   - usually based on dominance

2. Preservation of a **diverse set of points**
   - similarities to multi-modal problems

3. Remembering all the **non-dominated points** you have seen
   - usually using elitism or an archive

EC approach:
1. Selection

- Could use aggregating approach and change weights during evolution
- Different parts of population use different criteria
  - no guarantee of diversity
- Dominance (made a breakthrough for MOEA)
  - ranking or depth based
  - fitness related to whole population

Example: Dominance Ranking in NSGA-II

- new gen.

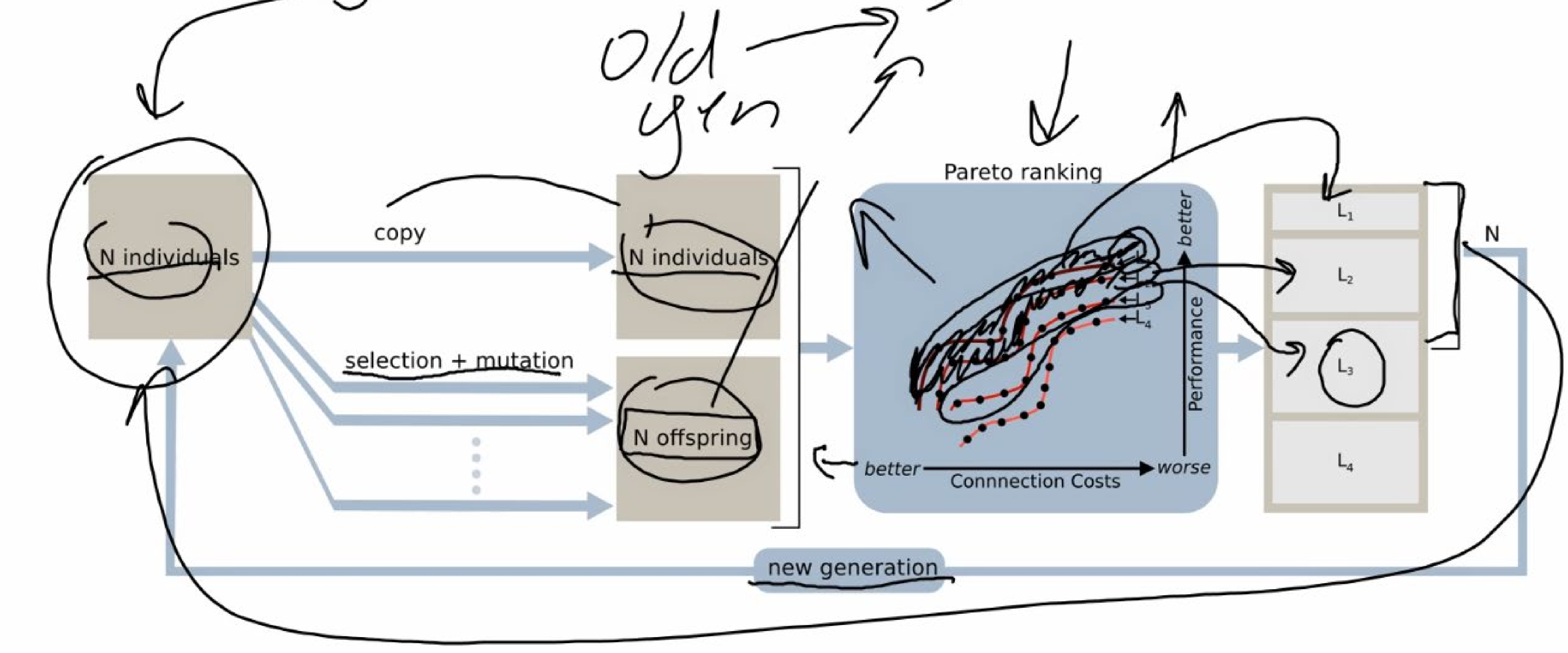
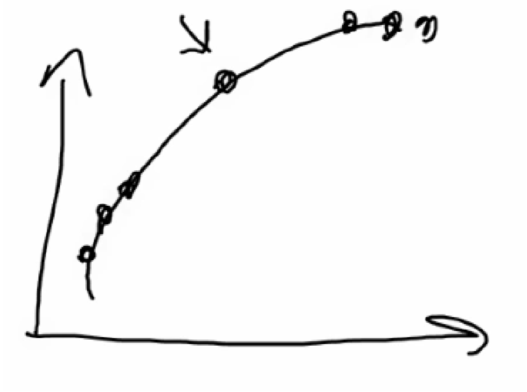


Figure from Clune, Mouret & Lipson (2013): "The evolutionary origins of modularity"

EC approach:
2. Diversity maintenance

- Aim: Evenly distributed population along the Pareto front
- Usually done by niching techniques such as:
  - fitness sharing
  - adding amount to fitness based on inverse distance to nearest neighbour
- All rely on some distance metric in genotype / phenotype / objective space

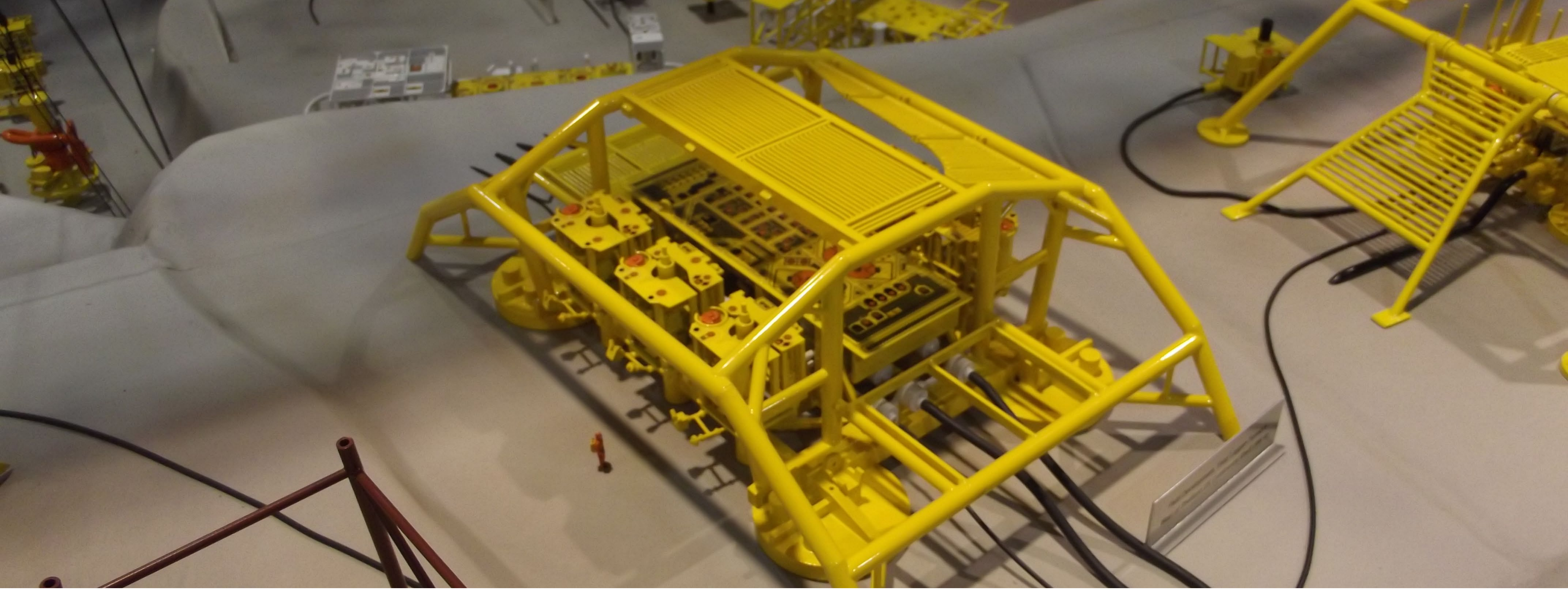


EC approach:
3. Remembering Good Points

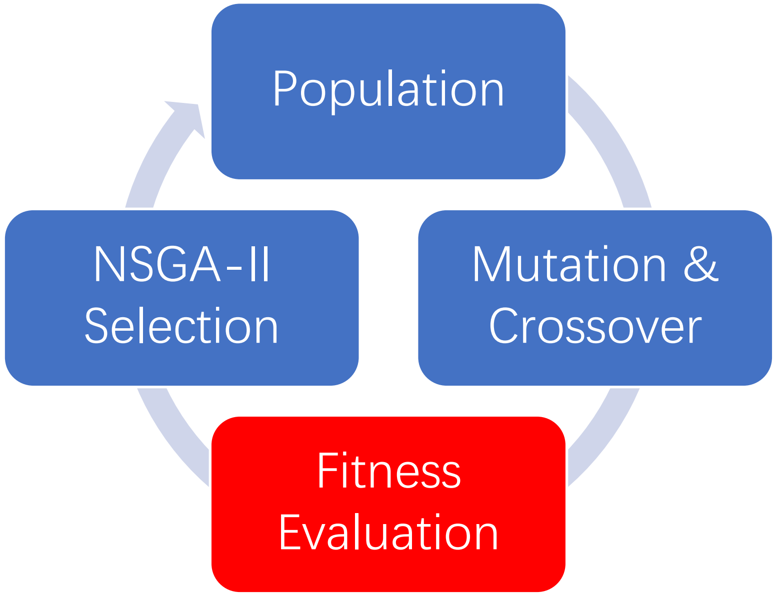
- Could just use an elitist algorithm
- Common to maintain an archive of non-dominated points
  - some algorithms use this as a second population that can be in recombination etc.

Multi objective problems - Summary

- MO problems occur very frequently
- EAs are very good at solving MO problems
- MOEAs are one of the most successful EC subareas

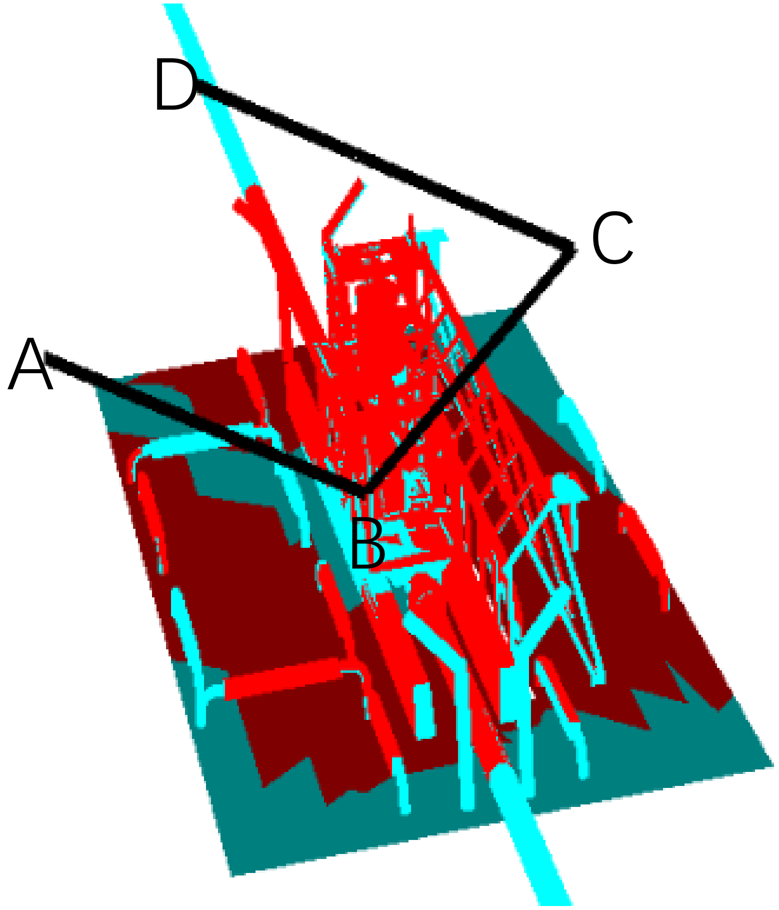


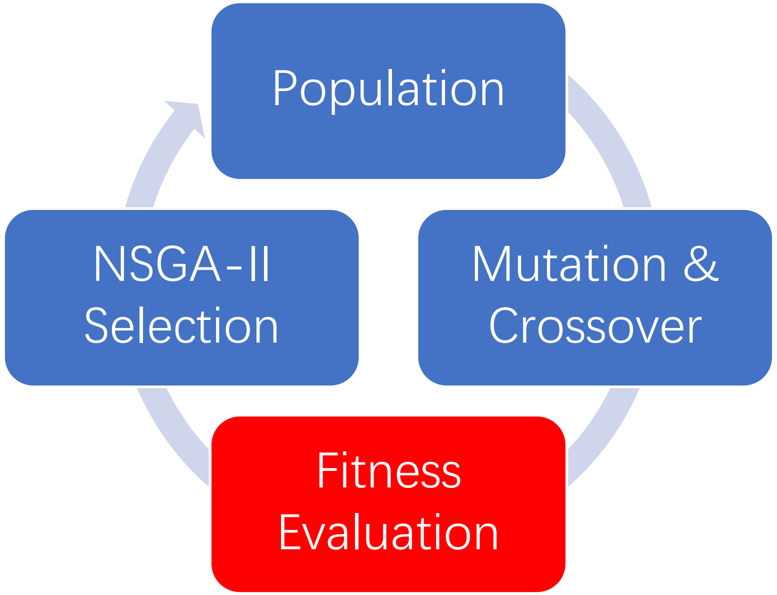
Multiobjective Optimization- Example



1. Decode genotype into plan

[A, B, C, D] ->

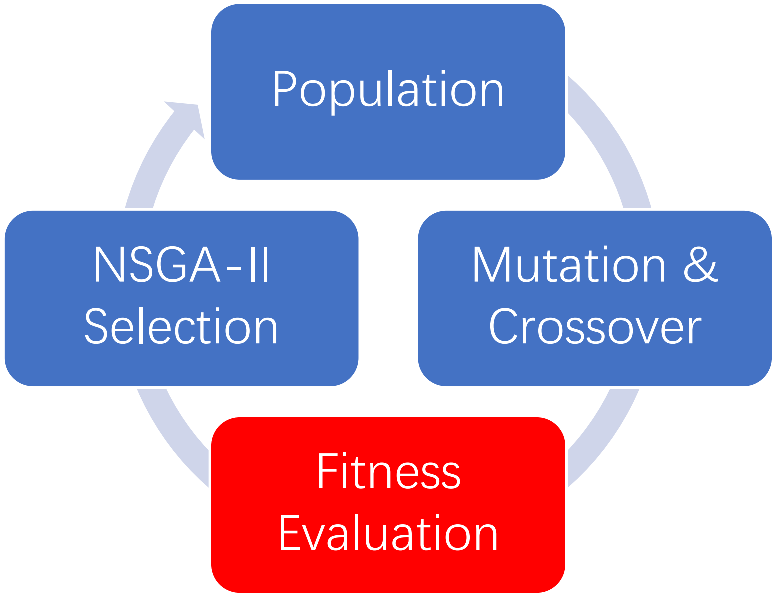




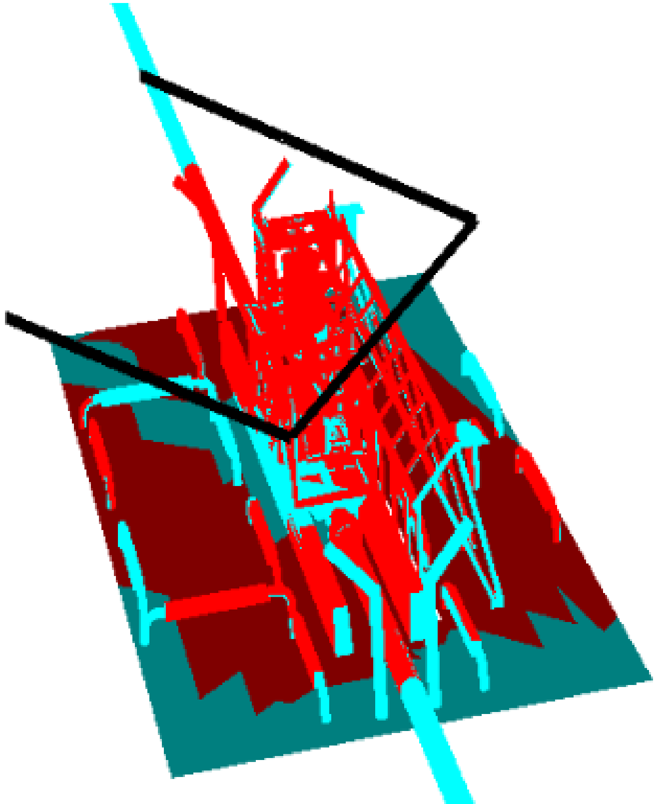
1. Decode genotype into plan
2. Estimate energy usage

$$\sum_{\vec{e} \in Edges(plan)} (w_{trans} \cdot ||\vec{e}||) + (w_{rot} \cdot (1 - cos(\theta_{\vec{e}-1,\vec{e}})))$$

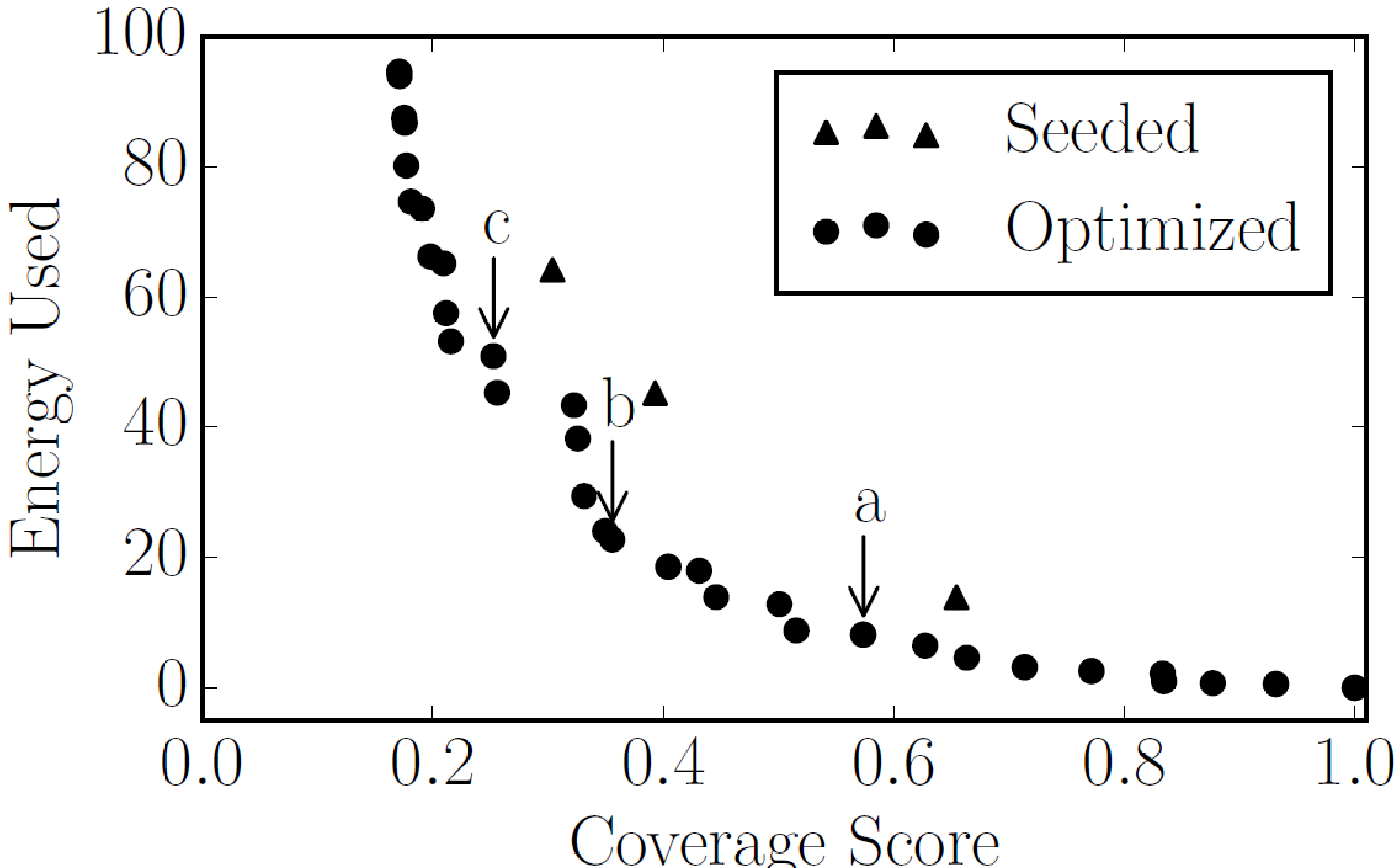
-Any edges introducing a collision are *penalized* by adding a constant term to their energy usage

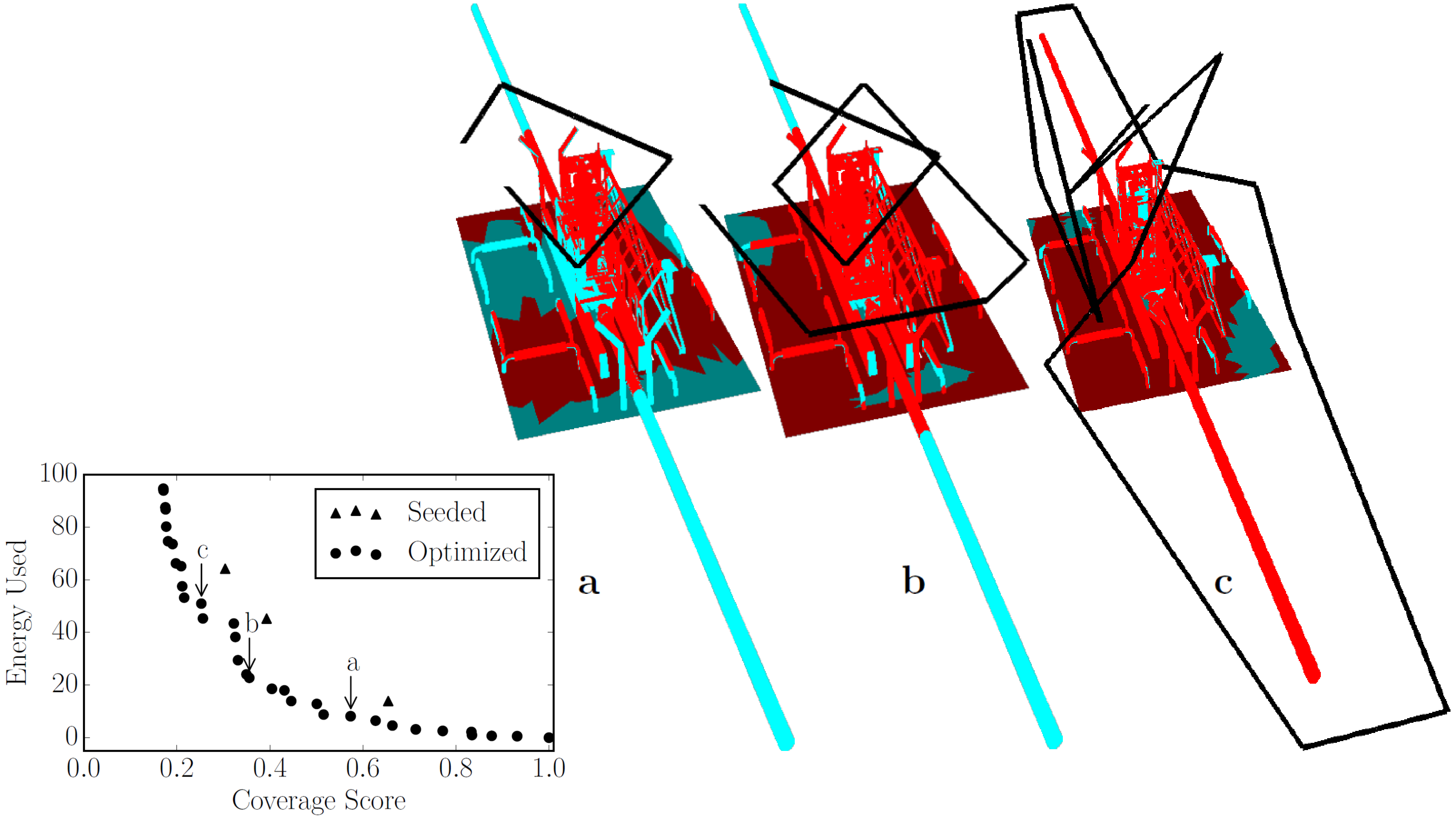


1. Decode genotype into plan
2. Calculate energy usage
3. Estimate coverage:
   - Estimate all geometric primitives covered by each edge in the plan
   - Calculate the total area of covered primitives
   - Coverage score : $1.0 - (covered\_area/total\_area)$



AUV Inspection Plans After 1000 Generations of NSGA-II



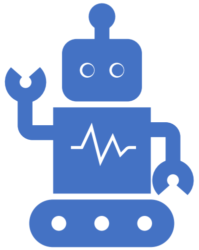


a
b
c
Seeded
Optimized
Energy Used
Coverage Score
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
1.0

UiO : University of Oslo

institutt for informatikk

IN3050/IN4050, Lecture 4
Evolutionary algorithms 2



1: Introduction and repetition

2: Selection

3: Diversity preservation

4: Hybridization

5: Multi-objective optimization