DISTRIBUTED METHODS COEVOLUTION

Luca Manzoni

PARALLEL AND DISTRIBUTED METHODS

Parallel and distributed methods refer to the use of multiple processors or computer systems working together to solve a problem or complete a task more efficiently.

Parallel methods involve breaking down a problem into smaller sub-tasks that can be solved simultaneously by different processors. This can greatly reduce the time needed to complete the task, as multiple processors are working in parallel to achieve the same goal.

Distributed methods involve distributing the workload across multiple computer systems or nodes, often connected through a network. Each node works independently on a portion of the task and communicates with other nodes to share data and collaborate on the overall solution.

WHY?

YOU HAVE MULTIPLE SOLUTION AND THEY MIGHT BE EXPENSIVE TO SEARCH

- Running evolutionary algorithms can be expensive:
  we can use multiple cores/multiple computers/special devices
  (e.g., GPU)

- Some distributed models can actually improve the quality of the evolution by preserving more diversity inside the population

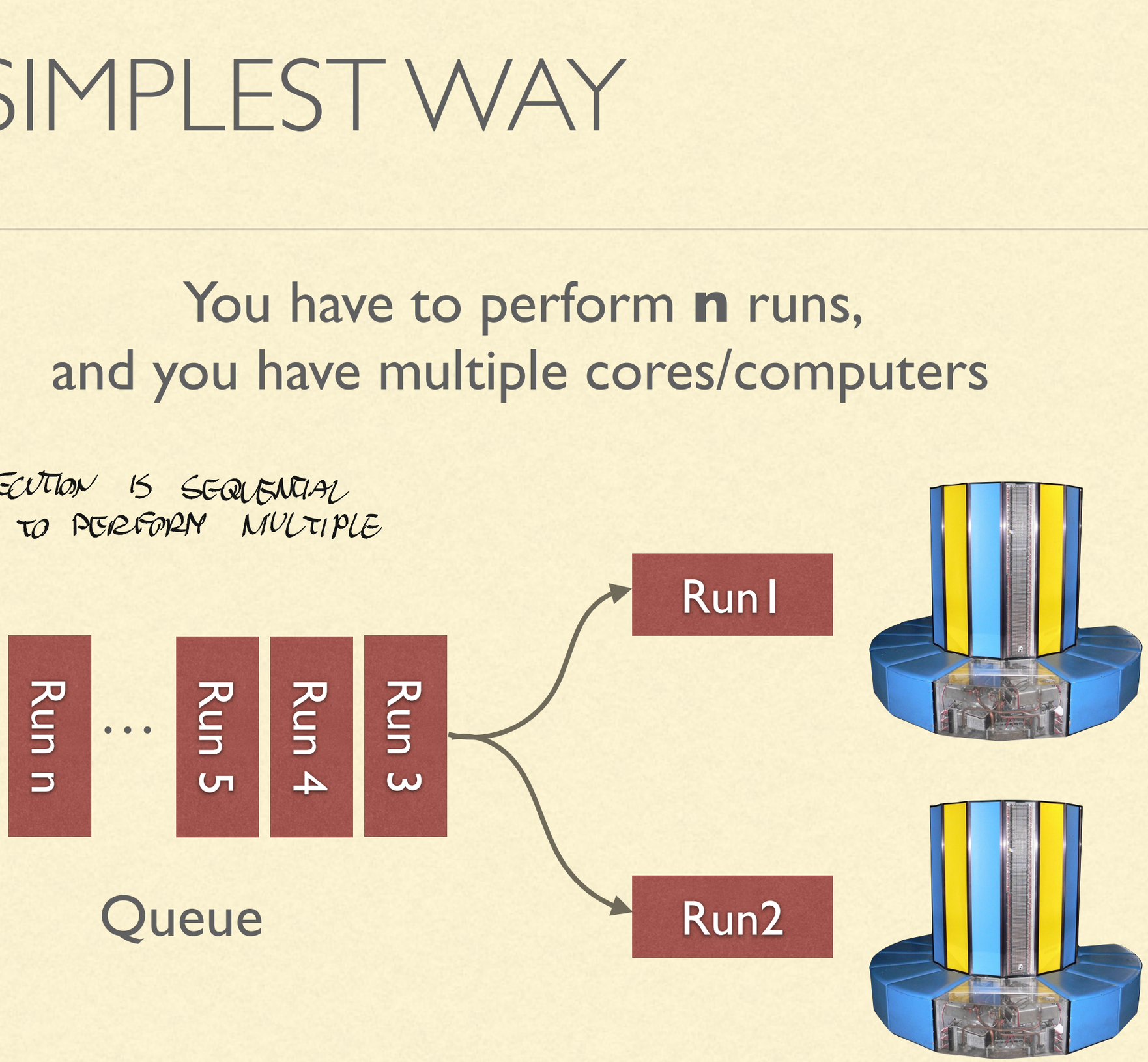
  DIFFERENT POPULATION EXPLORE DIFFERENT PART OF THE SPACE

- Population based evolutionary algorithms are easier to parallelise than many other methods

THE SIMPLEST WAY

You have to perform **n** runs,
and you have multiple cores/computers

THE SINGLE EXECUTION IS SEQUENCIAL
BUT YOU HAVE TO PERFORM MULTIPLE
TIMES

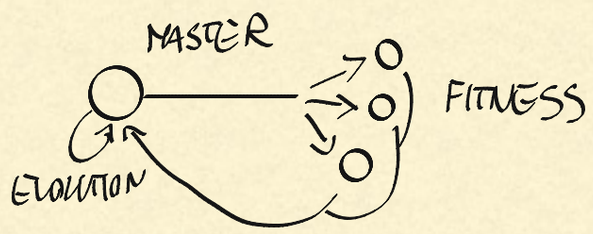


DISTRIBUTED FITNESS ASSESSMENT

YOU CAN EASLY DISTRIBUTE THE COMPUTATION OF THE FITNESS

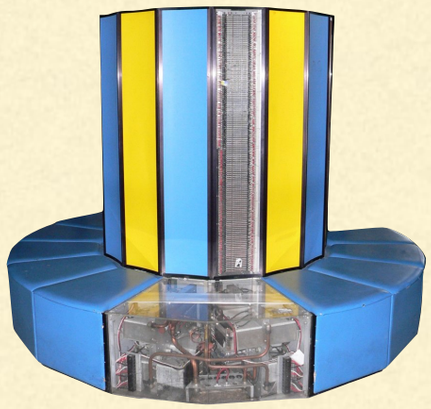
- Also known as client-slave or master-server
- Idea fitness evaluation can be (by far) the most expensive operation

  IT CAN BE A SIMULATION OF SOME CHEMICAL COMPOSTE AND SO ON...
- Keep the evolution process inside a single node (the master)…
- …but move the fitness evaluation among a set of nodes



DISTRIBUTED FITNESS ASSESSMENT

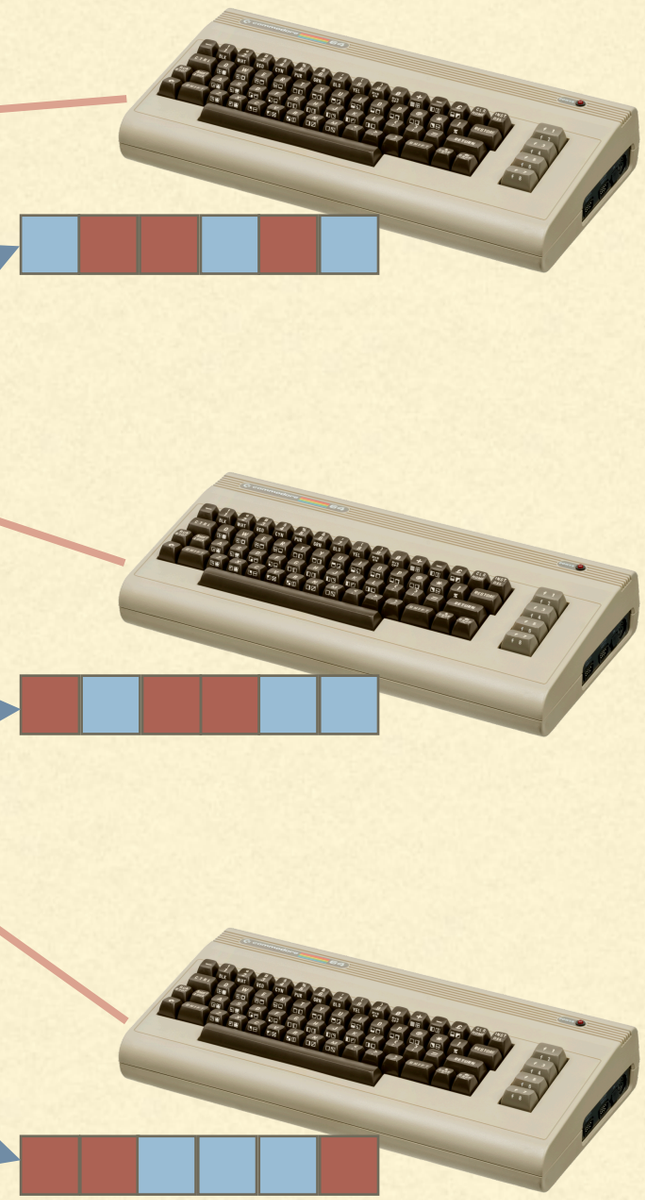
JUST SPREAD THE POPULATION AMONGST ALL CORE



2.4

3.2

8.7



ADVANTAGES AND DISADVANTAGES

- If the fitness assessment is long then this method scales well
- But if the time to transmit the individual to the nodes is in the same order of magnitude of the fitness evaluation then there is no advantage

  IT COULD BE SOMETHING MORE EFFICENT
- You only need to transmit the individuals (possibly compressed) and to receive the fitness value

  EX. DELTA BETWEEN ORIGIN INDIVIDUAL AND CURRENT, AND SO ON
- Resistant to failures of the slave nodes

  CAN SET A TIME LIMIT, EASLY SCALABLE
- Possible to add/remove nodes when needed

ISLAND MODEL

COMBINED PARALLELIZATION

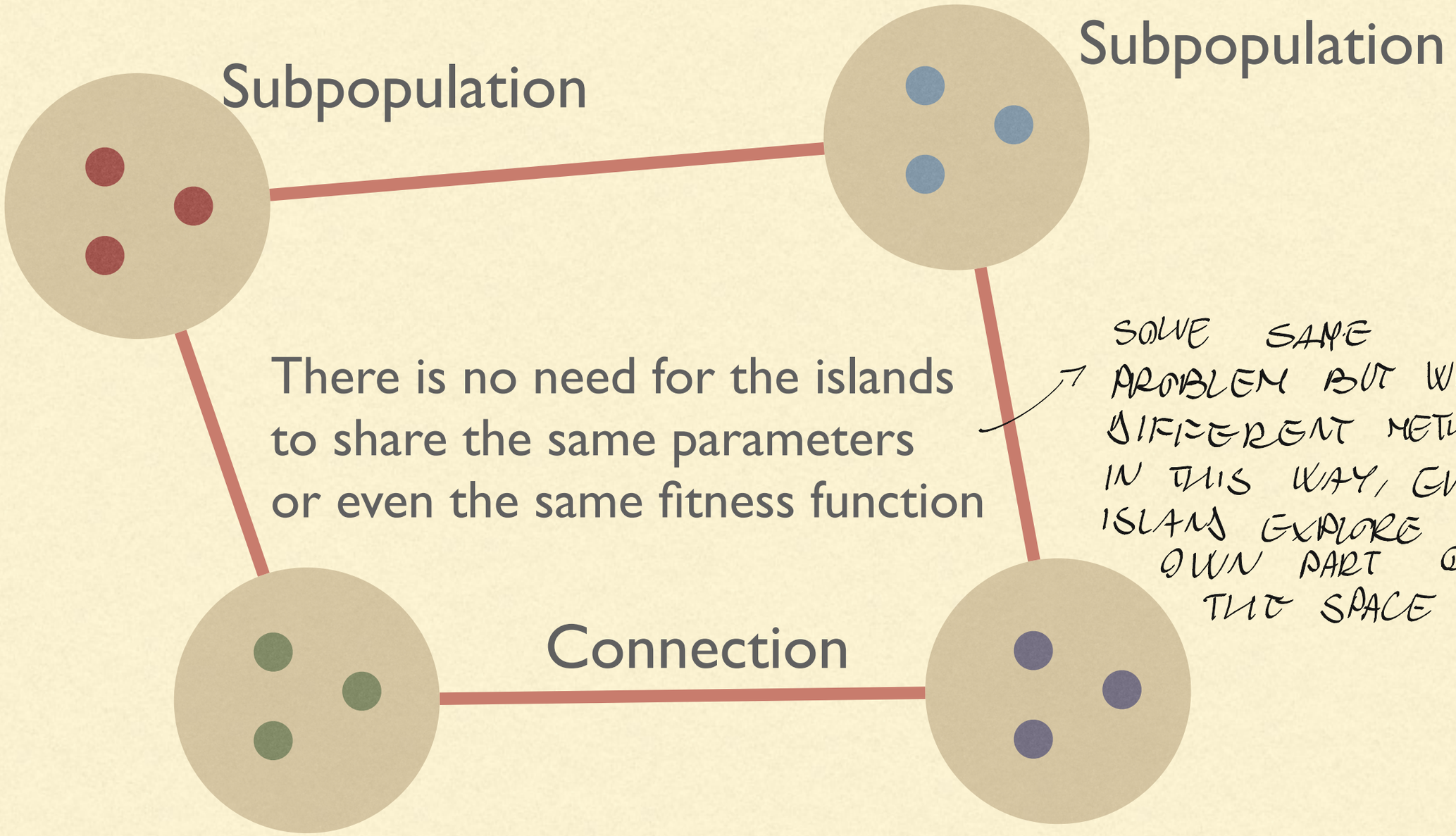
- Distributed evolution: like in real-world islands, populations evolve independently with the occasional exchange of individuals

I CAN HAVE SINGLE INDIVIDUALS OR MULTIPLE GROUP OF SAMPLES

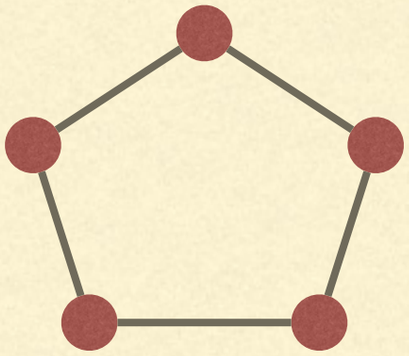
↳ INTRODUCE NEW GENETIC MATERIAL

- New additional parameters:
  - The number of the islands
  - The size of the populations on the islands
  - How the islands can talk (the topology)
  - Which individuals they exchange and how frequently.

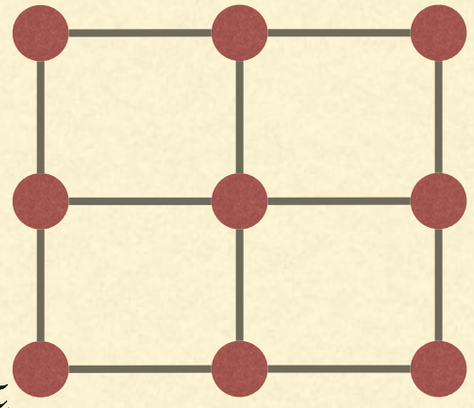
ISLAND MODEL



ISLAND MODEL: TOPOLOGIES



Ring

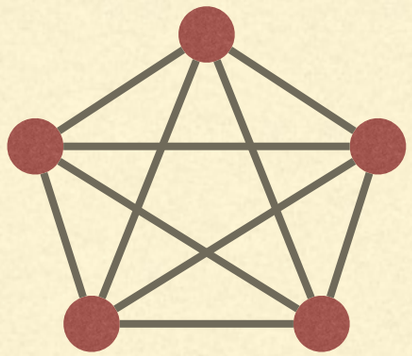


Grid

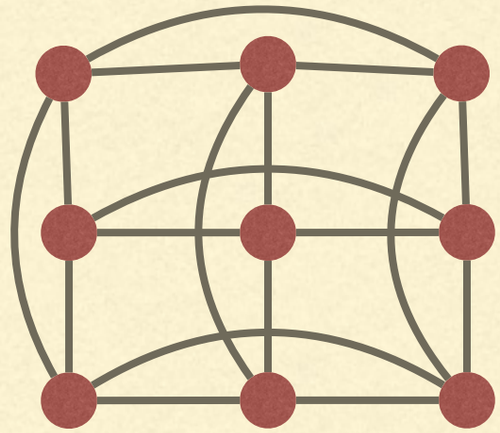
WHAT KIND OF COMMUNICATION YOU WANT? FULLY CONNECTED MUCH FASTER THAN RING WITH ADVANTAGE AND DISADVANTAGE

LITLE EXCHANGE

THE TOPOLOGI CAN BE CHANGE DURING EXECUTION



MOST COMMON



Fully connected

Toroid

IT'S POSSIBLE TO CHANGE IT WHILE RUNNING

EXCHANGE ALL GENETIC MATERIAL

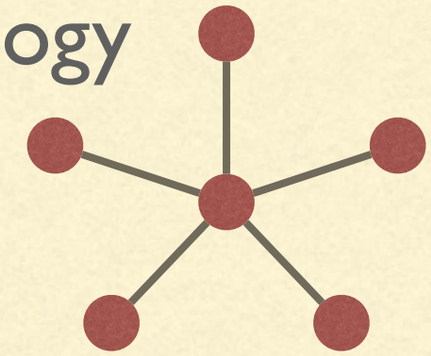
A POSSIBLE ALGORITHM

- Each population evolves independently
- Every K generations the top 5% of the population is copied…
- …and sent to one of the neighbours
- Notice that the exchange can be synchronous or asynchronous: we can wait for each island to be ready or we can do the exchange asynchronously

POTENTIALLY A DISADVANTAGE, YOU SEND SOME INDIVIDUALS AND AFTER 1 GENERATION OR 2 YOUR INDIVIDUALS, WICH ARE THE BEST, ARE SENT BACK

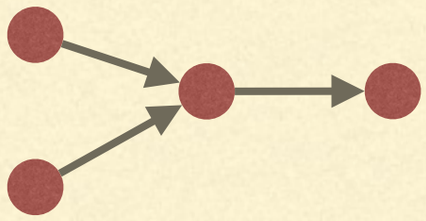
ONE CENTRAL POPULATION

Star topology



- Each population in the “arms” of the star evolves independently
- Each K generation the top X% of each population is sent to the central island
- The central island is, in some sense, the collector of individuals from all the other islands

COLLECTOR TOPOLOGY



A directed acyclic graph

THE HIGHER YOU MOVE IN THE GRAPH, THE HIGHER IS THE PRESSURE DUE TO THE BEST SAMPLE

- Individuals only move in one direction
- This can be useful to optimize only part of the fitness in each "layer" of island
- For example, to make a robot learn to run, we might reward the ability to stand up in the first island, then the ability to walk in the second, and so on.

SPATIALLY-EMBEDDED EA

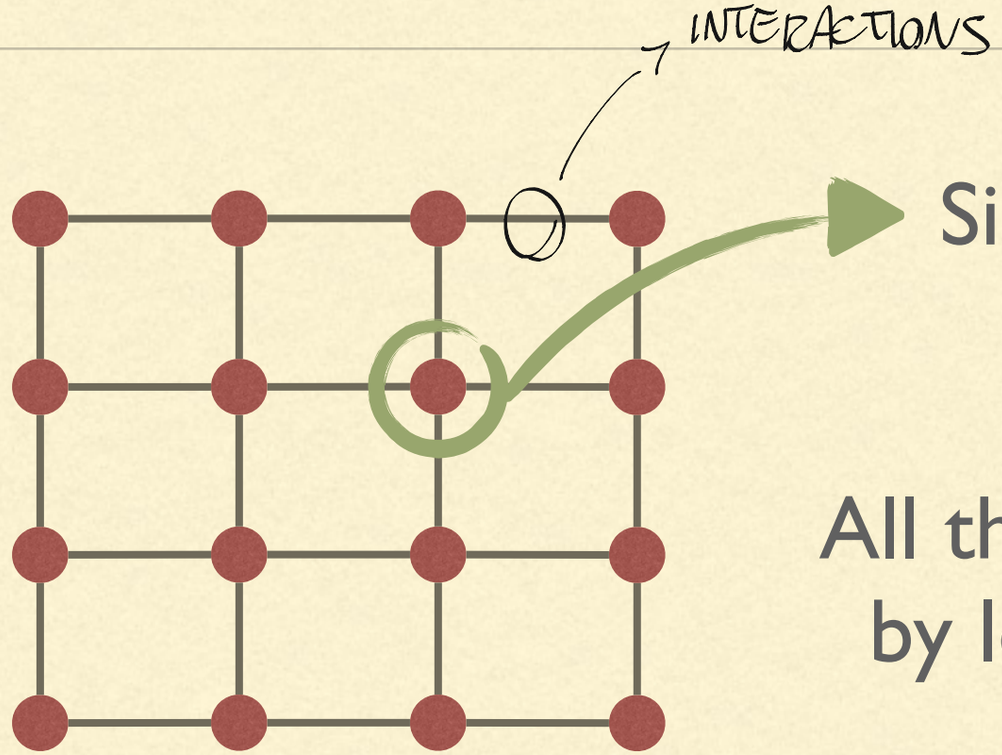
- We have seen some *coarse grained* parallelism. The unit was an entire population

- We can also have more fine grained parallelism (for GPUs for example)

- The idea is that the "element" of the parallelism is a single individual

- A spatial location is added to each individual in a population

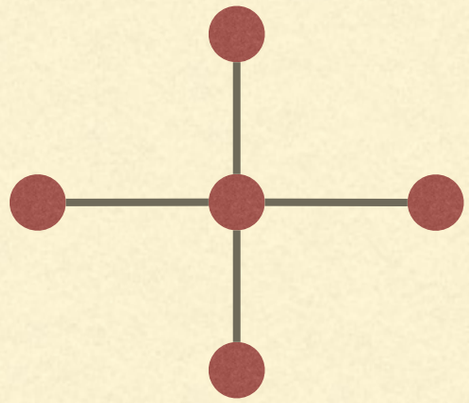
SPATIALLY-EMBEDDED EA



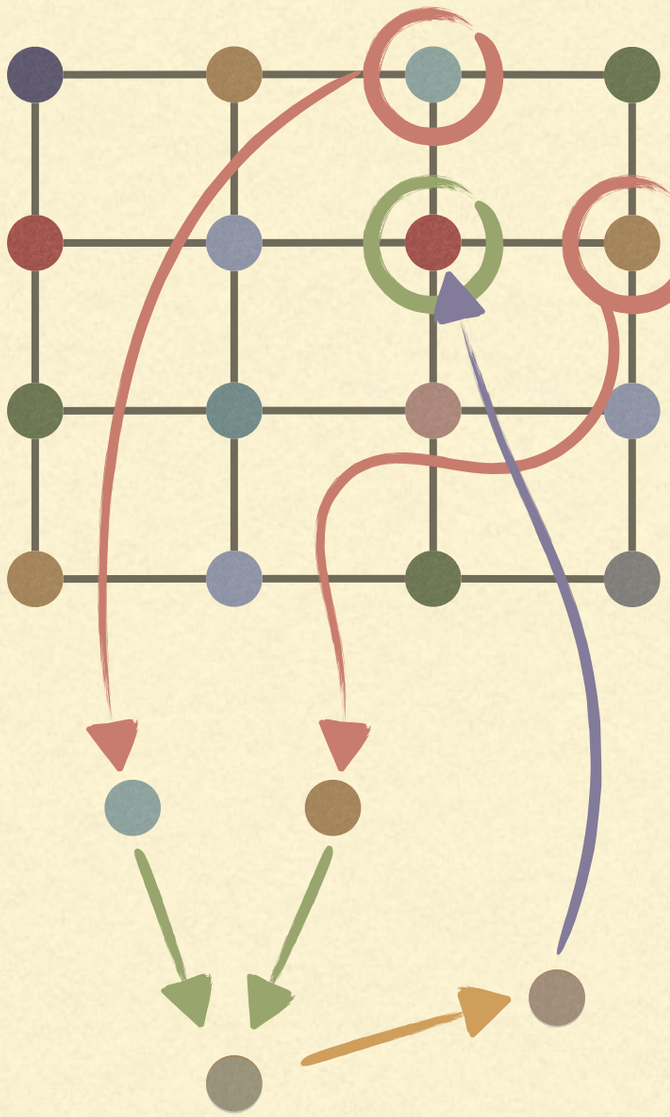
Single individual

All the operations must be performed by looking only at a neighbourhood

E.g.:



SPATIALLY-EMBEDDED EA



- Select two parents from the neighborhood

  I CAN BE DONE BY WHATEVER ALGORITHM, BUT IT MUST BE IN THE NEIGHBORHOOD

- Perform the crossover between them
- Mutate the resulting individual
- Replace the individual in this node with the new one

  THIS IS LIMITED BY THE SIZE OF THE TOPOLOGY MAP

COEVOLUTION

Coevolution is a process by which two or more species influence each other's evolution. This can occur in a variety of ways, such as through predator-prey interactions, mutualistic relationships, or competition for resources. In coevolution, changes in one species can drive evolutionary changes in another species, leading to a dynamic and interconnected evolutionary relationship between the two. This can result in the development of specialized traits or behaviors that help both species adapt to their changing environments. Overall, coevolution plays a crucial role in shaping the diversity and complexity of ecosystems.

COEVOLUTION: IDEAS

- The fitness of a single individual is now influenced by “external factors”:
  - Performance against the other individual of the population
  - “Collective performance”: the entire population counts
  - By the similarity to other individuals: too many similar individuals are “bad”.

THE ROLE OF FITNESS

Optimising absolute fitness involves directly maximizing or minimizing a specific measure of success, such as profit, speed, accuracy, or some other performance metric, without considering the impact of other individuals or factors. This is typically the ultimate goal in a given scenario.
On the other hand, optimising relative fitness involves considering the fitness of an individual in relation to the fitness of other individuals in the population or system. In this case, the algorithm is focused on improving the individual's fitness compared to its peers, rather than achieving a specific absolute fitness value.

- Usually we are interested in maximizing/minimizing a certain fitness value…

- …but now the fitness is influenced by the other individuals

- **Absolute fitness**: the fitness that we actually want to optimise

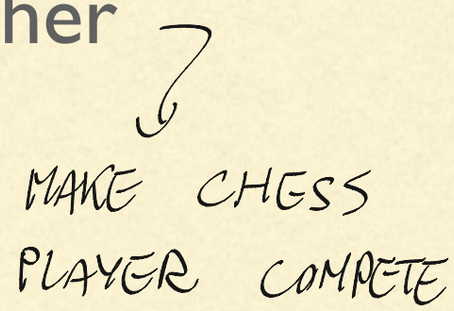
EX: IF WE WANT TO EVOLVE A CHESS PLAYER, WHAT WE WANT TO OPTIMIZE IS THE ABILITY TO PLAY CHESS

- **Relative fitness**: the fitness that the algorithm is optimising, hopefully also improving the absolute fitness

Ultimately, the goal of optimising relative fitness is to improve the overall performance or success of the population as a whole, by encouraging competition among individuals and driving them to continuously improve their own fitness compared to others. This can result in a more dynamic and adaptive system, where individuals are constantly striving to outperform each other and collectively drive the population towards greater success.

ONE-POPULATION COMPETITIVE COEVOLUTION

- Usually employed when producing individuals that play games
- Works when evolving agents where the quality of the solution can be assessed by making them play one against the other
- Here evaluating the "real fitness" might be difficult

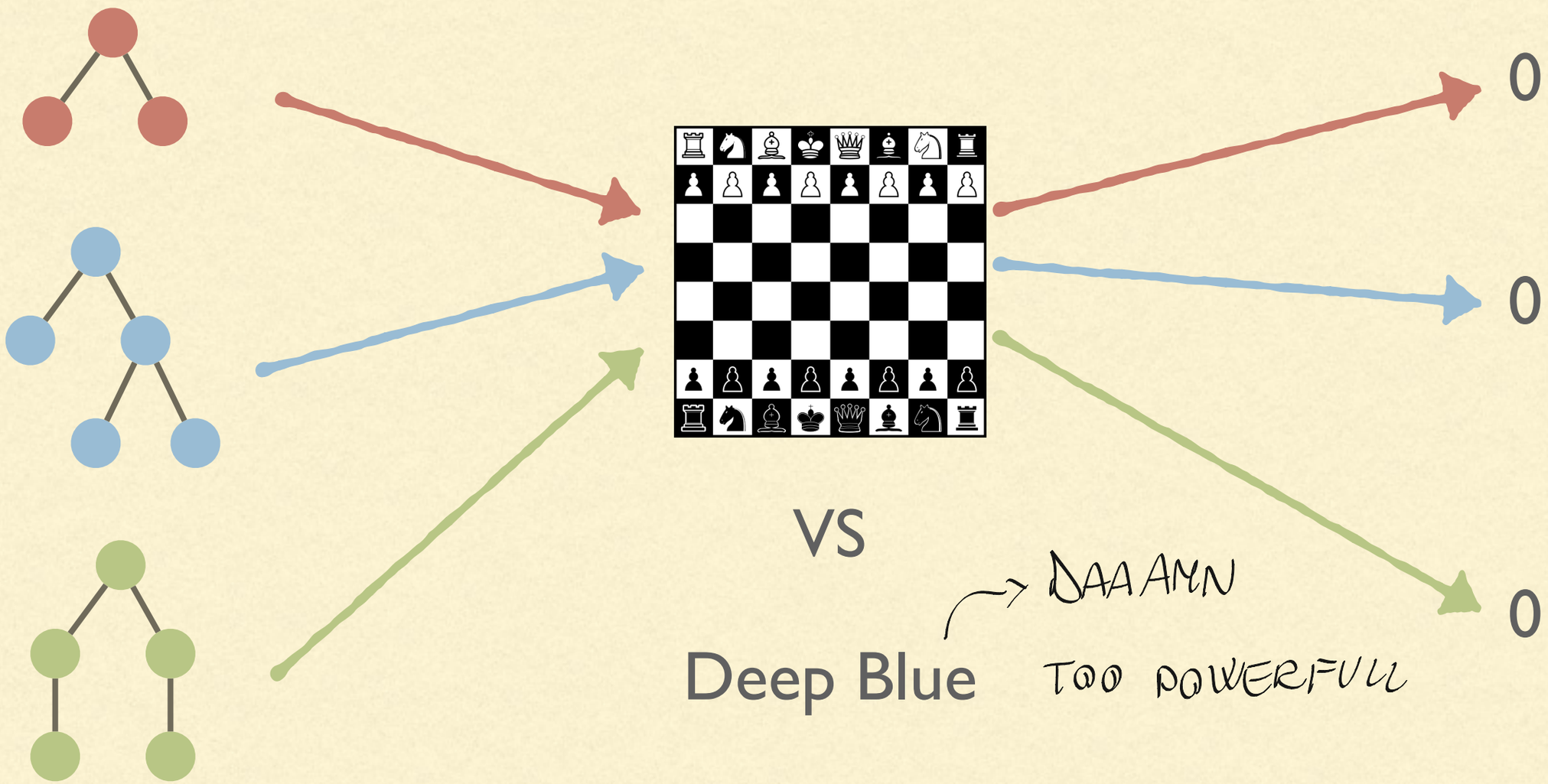


WHY?

Fitness evaluation
with a “competent adversary”

Population at generation 0

Fitness
Number of victories

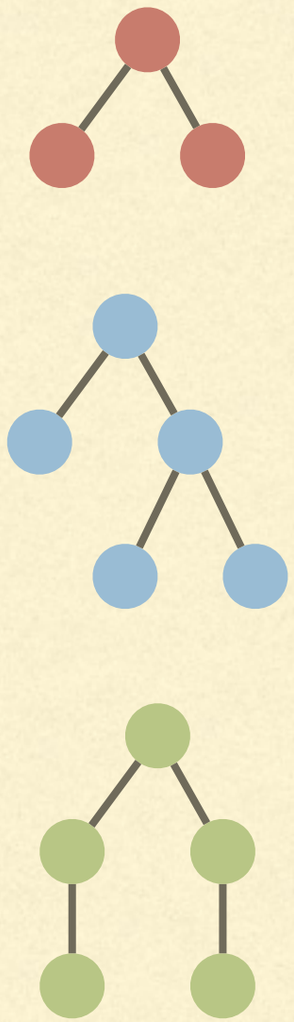


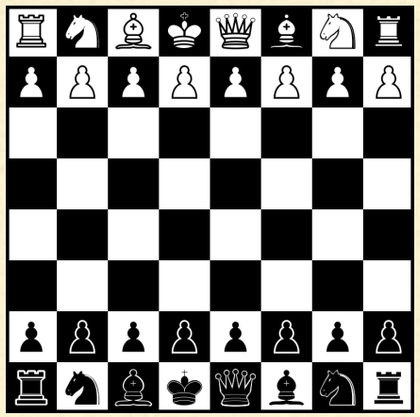
WHY?

Fitness evaluation
with a “not-too-competent adversary”

Population at generation 100

Fitness
Number of victories





100

100

VS

100

1K ZX Chess (1982)

NOT GOOD ENOUGH

PROBLEM OF USING A FIXED OPPONENT

- Evaluating the fitness of the individuals high be difficult if the individuals are too weak or too strong w.r.t. the opponent
- This might be solved by using a collection of fixed opponents…
- …or by using directly other individuals as opponents

INTERNAL VS EXTERNAL FITNESS

Individuals might get better at winning against other individuals…

Internal fitness

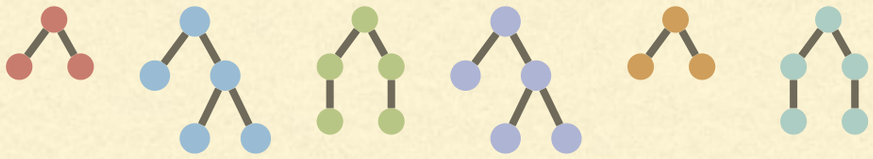
FOR EXAMPLE IF SOMETHING NEW HAPPENS

External fitness

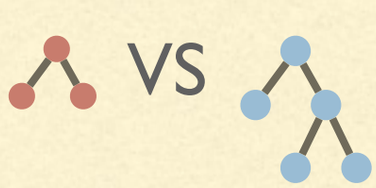
…but not agains real opponents

Progress should be evaluated with both fitnesses

EVALUATION: PAIRWISE



Only make individual n compete with individual n+1

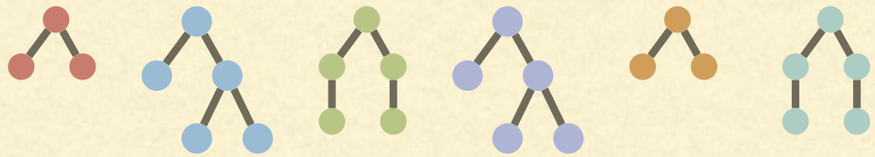


+only O(n) tests

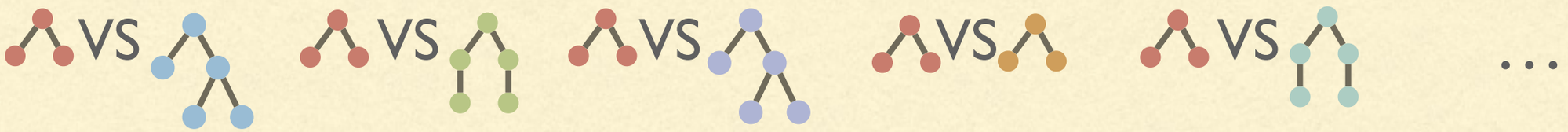
-very dependent on the adversary

EX. 1° AND 2° ARE THE BEST, 5° AND 6° ARE THE WORST. BUT WHEN WINNING ONE OF THE BEST IS MERGE WITH ONE OF THE WORST

EVALUATION: COMPLETE



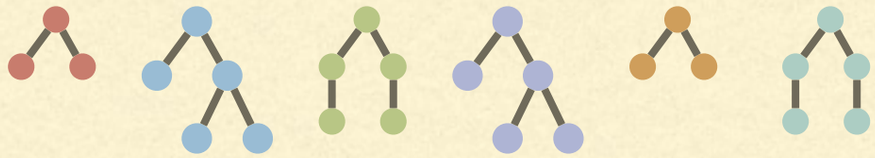
Every individual competes with every other individual



+precise assessment of the fitness

-with $O(n^2)$ tests

EVALUATION: K-FOLD

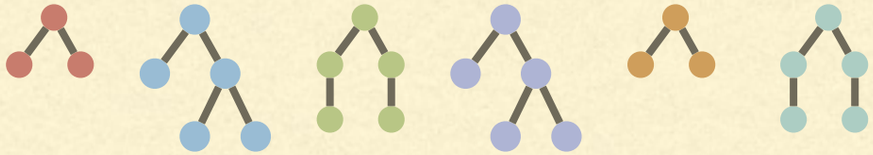


Every individual competes with k randomly selected individual

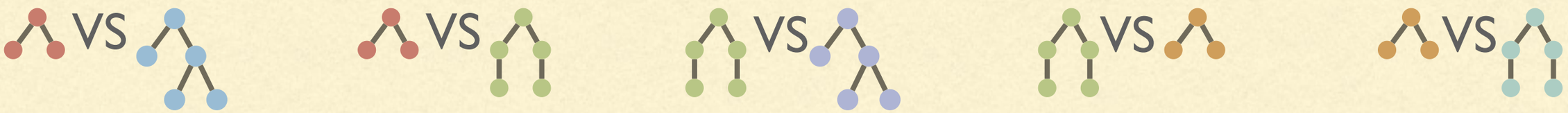
+tunable number of tests

-some individuals might be tested more than others

EVALUATION: SINGLE-ELIMINATION TOURNAMENT



Every individual proceeds with the competitions until it is beaten

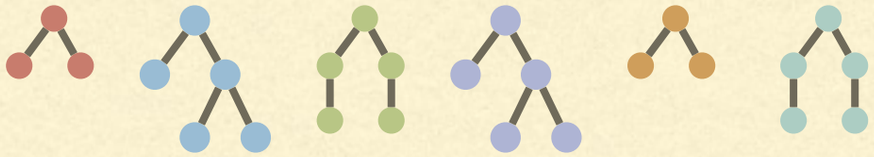


+Only O(n) tests

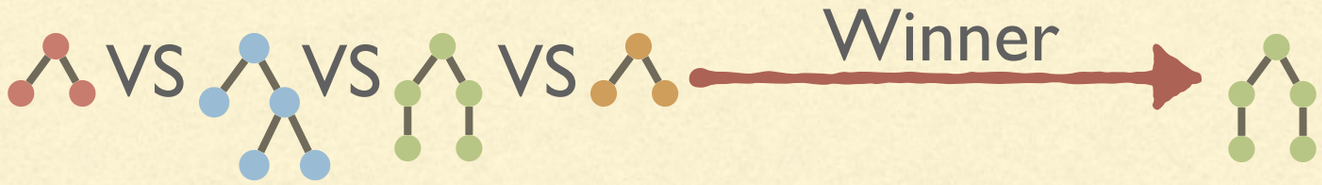
+Better individuals are tested more times

-Bad pairing might penalise some individuals

FITNESSLESS SELECTION



We never compute the fitness:
in tournament selection the selected individual is the winner of the tournament



YOU DON'T HAVE ANY FITNESS BECAUSE MAKE THE INDIVIDUALS COMPETE IS A WAY TO SELECT THE BEST ONE

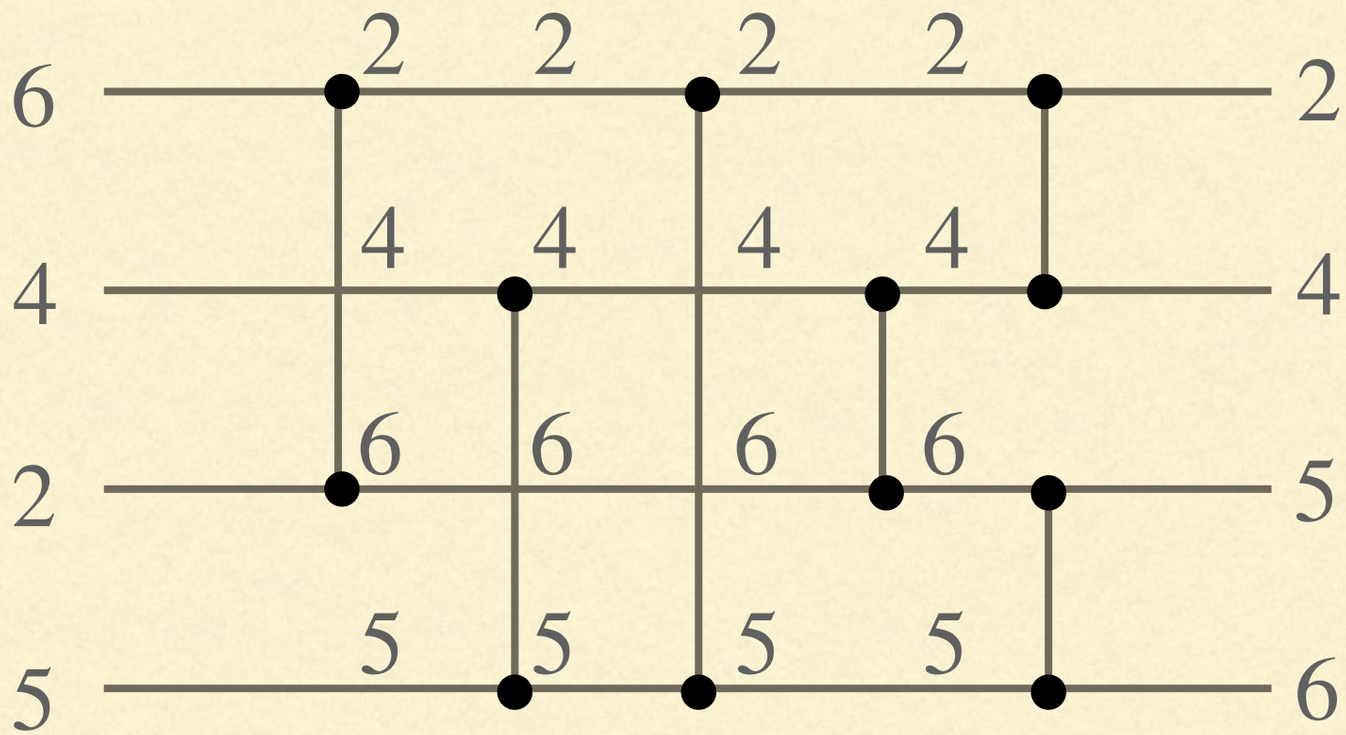
TWO-POPULATIONS COMPETITIVE COEVOLUTION

- Sometimes we want two populations to compete against each other to improve together

- **Primary population**: the individuals we are actually interested in

- **Alternative (foil) population**: individuals that try to beat/ foil the individuals in the primary population

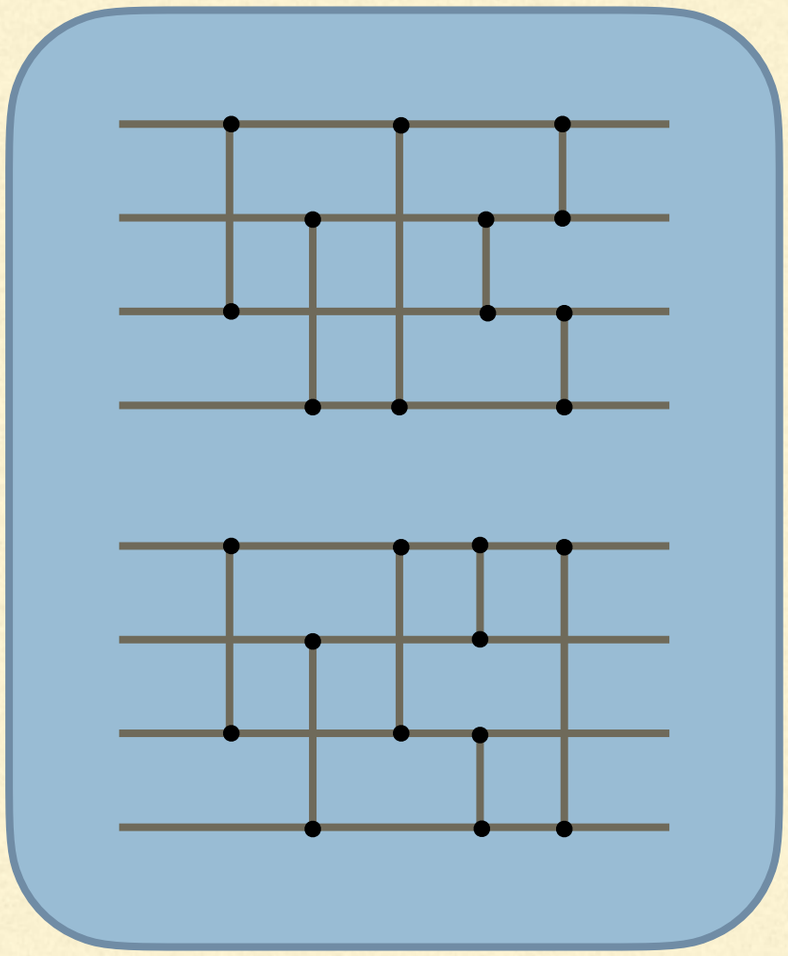
SORTING NETWORKS



A comparator exchanges the two values if the one at the top is bigger than the one at the bottom.

The optimal depth of a sorting network for n elements is known only for small n

EVOLVING OF SORTING NETWORKS



Population
of sorting networks

[1,3,5,4]

[7,5,6,2]

[4,3,2,1]

[1,3,2,4]

Population
of “difficult to sort” arrays

HOW TO DEFINE THE FITNESS

For sorting networks

Number of correctly sorted arrays

↓

Networks that sort well

For arrays

Number of "foiled" networks

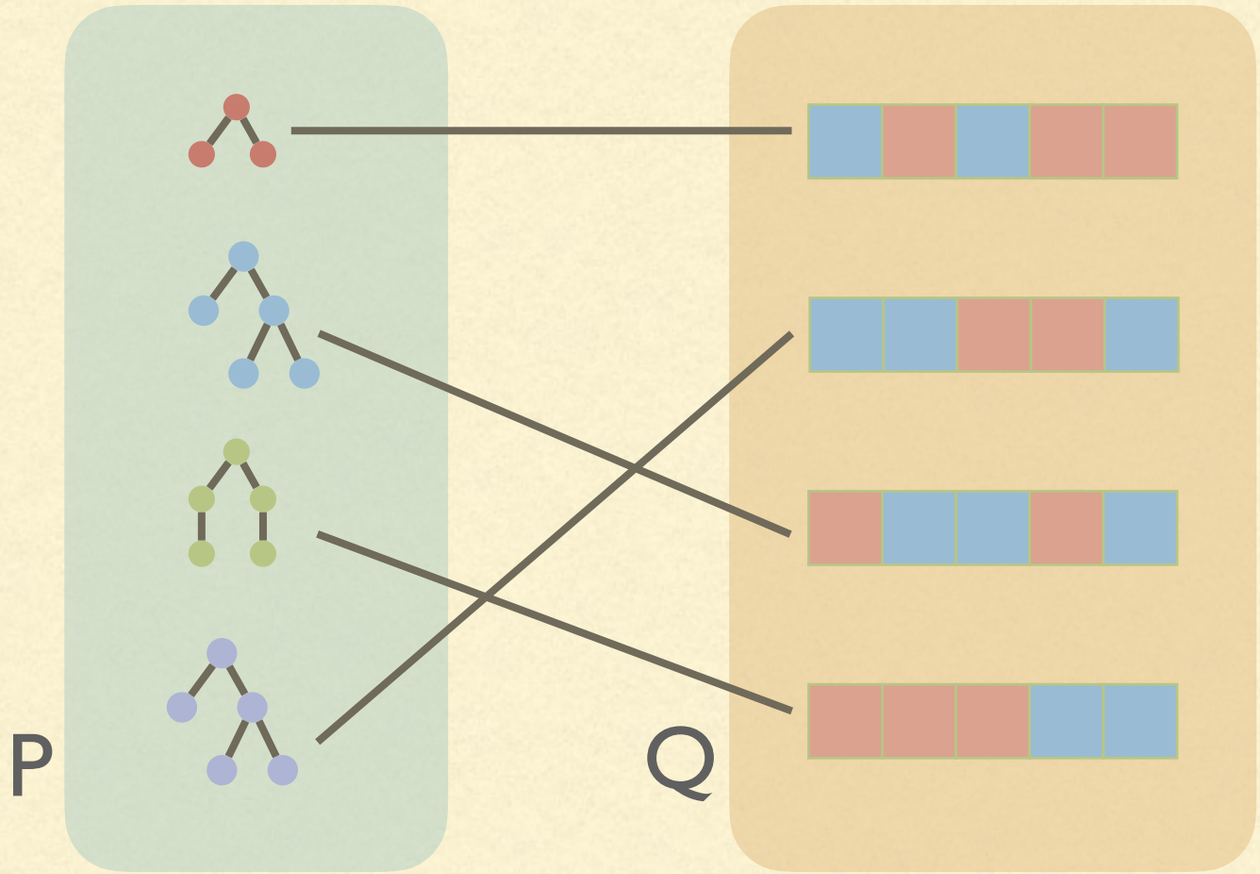
↓

Arrays that represents corner cases for networks

The two fitnesses are "in conflict", thus the coevolution is competitive

HOW THE EVOLUTION IS PERFORMED (1)

P and Q are always changing,
how to assess the fitness of the individuals?



Shuffle and pairing

Repeat k times

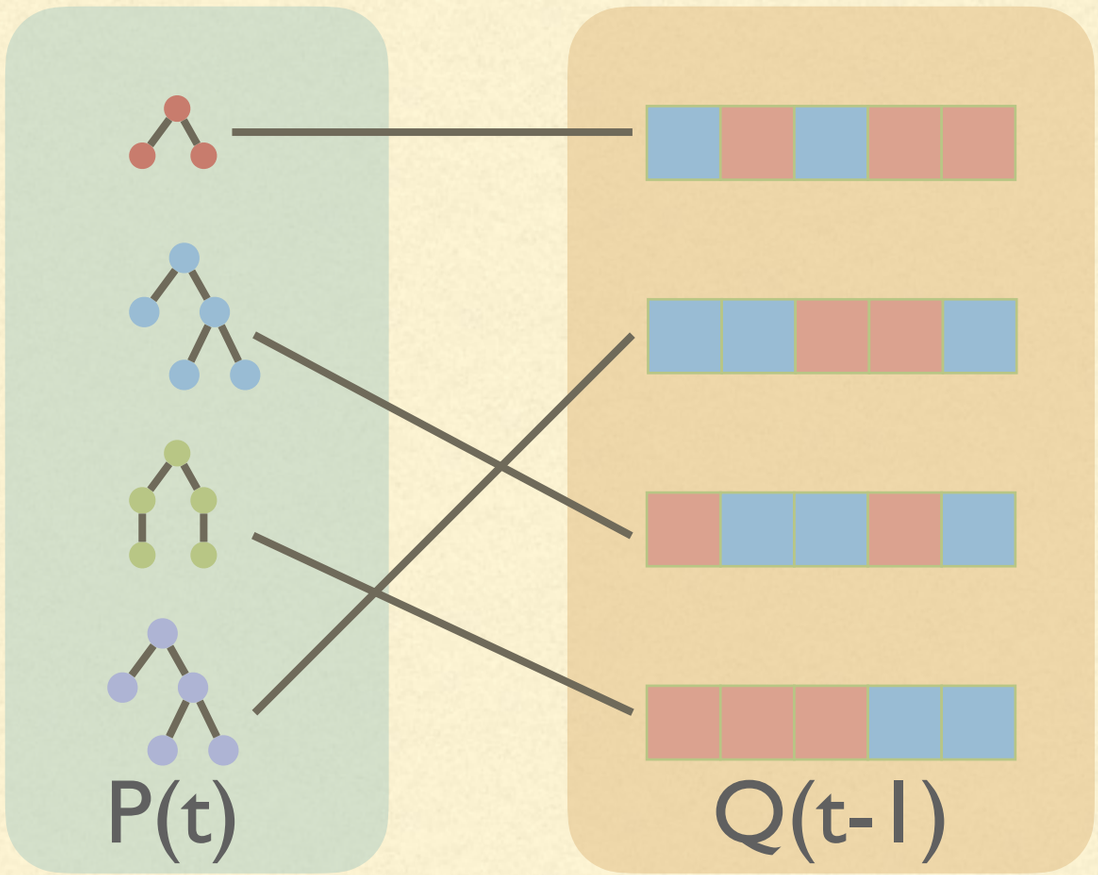
Avoid performing the same evaluation more than once

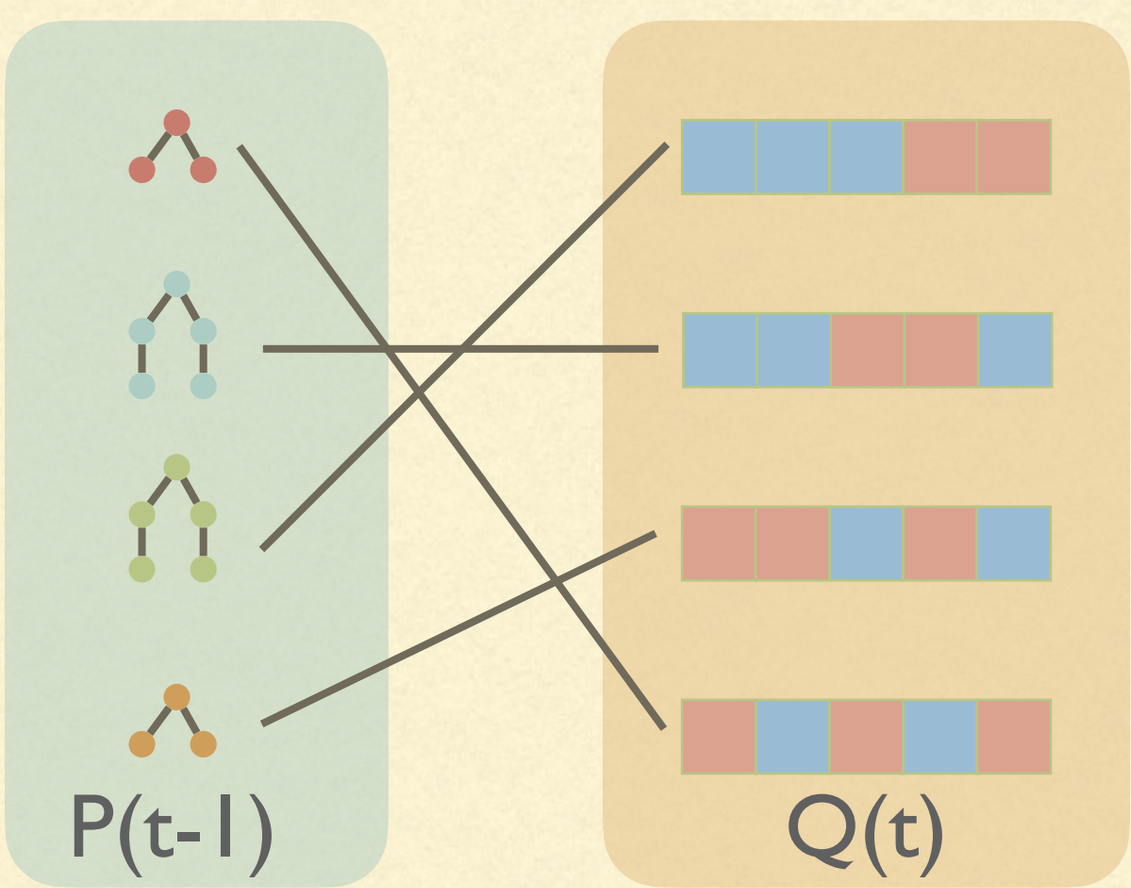
HOW THE EVOLUTION IS PERFORMED (II)

To make evolution easier we can use the same procedure but:

1) Q at time t is compared with P at time t-1
2) P at time t is compared with Q at time t-1

OTHERWISE, ONE WILL BE ALWAYS MORE ADVANCE THAN THE OTHER





HOW THE EVOLUTION IS PERFORMED (III)

- The comparison with the previous generation allows to perform some further tuning:

- We can compare each individual of P (resp., Q) at time t with the k best individuals in population Q (resp., P) at time t-1

- We can combine the two approaches and select k_1 individuals from the best and k_2 randomly

BUT WHY NOT JUST USE TWO POPULATION COMPETITIVE COEVOLUTION IN EVERY SITUATION?

LOSS OF GRADIENT

- Two-populations competitive coevolution is not without problems:

- If one of the two populations is too strong w.r.t. the other then the internal fitness gives no information…

- …thus, the evolution process proceeds without any guidance (i.e., the selection is almost uniform among the individuals)

- It might be useful to stop the evolution of the stronger population and allow the weaker one to "catch up"

N-POPULATION COOPERATIVE COEVOLUTION

- Sometimes a solution can be decomposed into multiple interacting sub-solutions, each one with its own characteristics

- Sone examples: robot soccer, any multiple player game with more than one role

- It is always possible to create an enormous individual, but it might be useful to have multiple interacting populations

HOW TO ASSESS THE FITNESS?

Parallel methods

The evolution happens for all populations at the same time

The same methods of competitive evolution

As usual, we need to specify how individuals are paired

Sequential methods

Populations are evolved one at a time

This is a case of Alternating Optimization

Not actually suitable for competitive evolution

POSSIBLE DRAWBACKS

Suppose that you want to evolve a soccer team

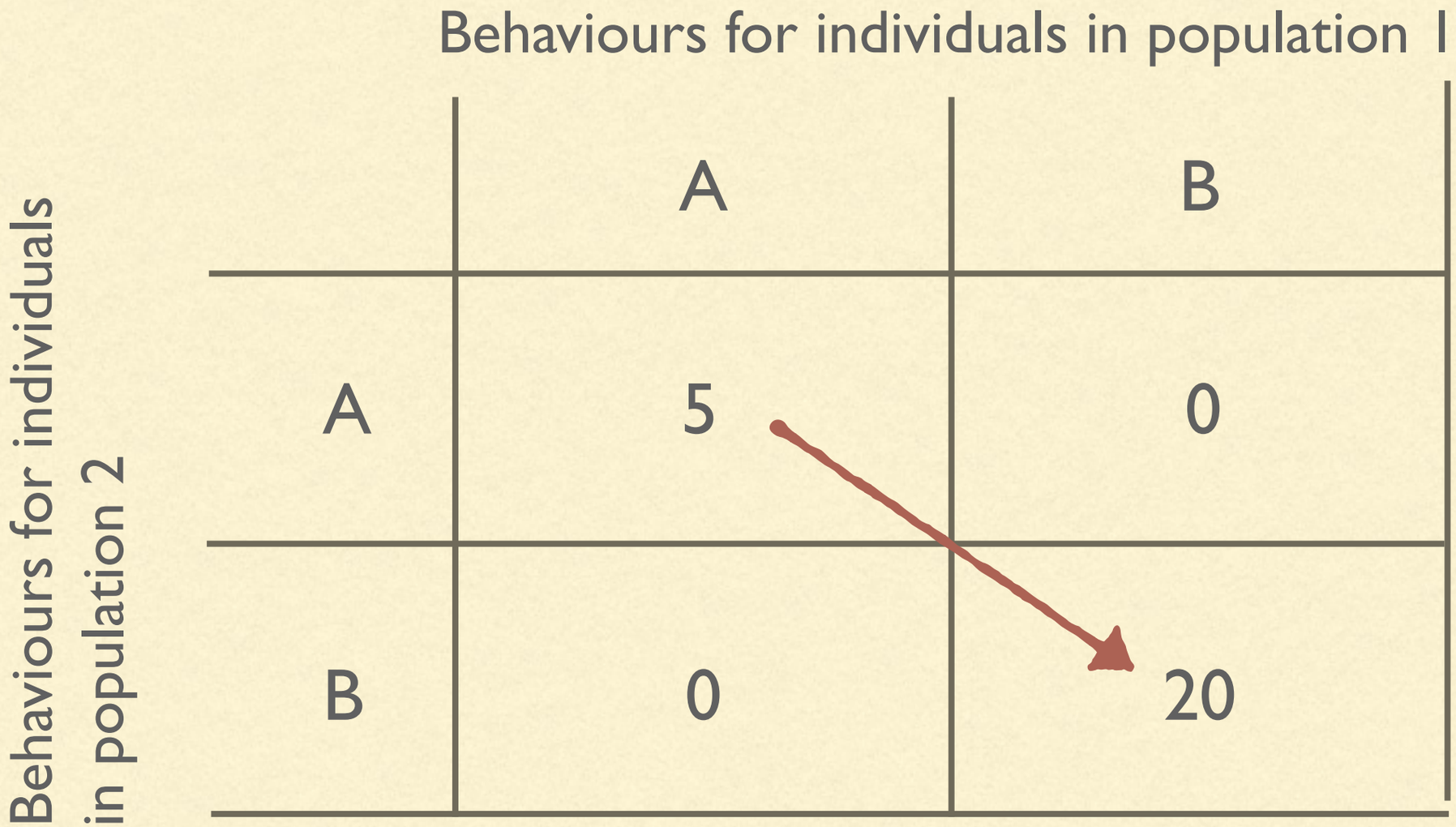
It is not doing anything useful for the global solution

Still has a good fitness because the other individuals are strong

POSSIBLE DRAWBACKS

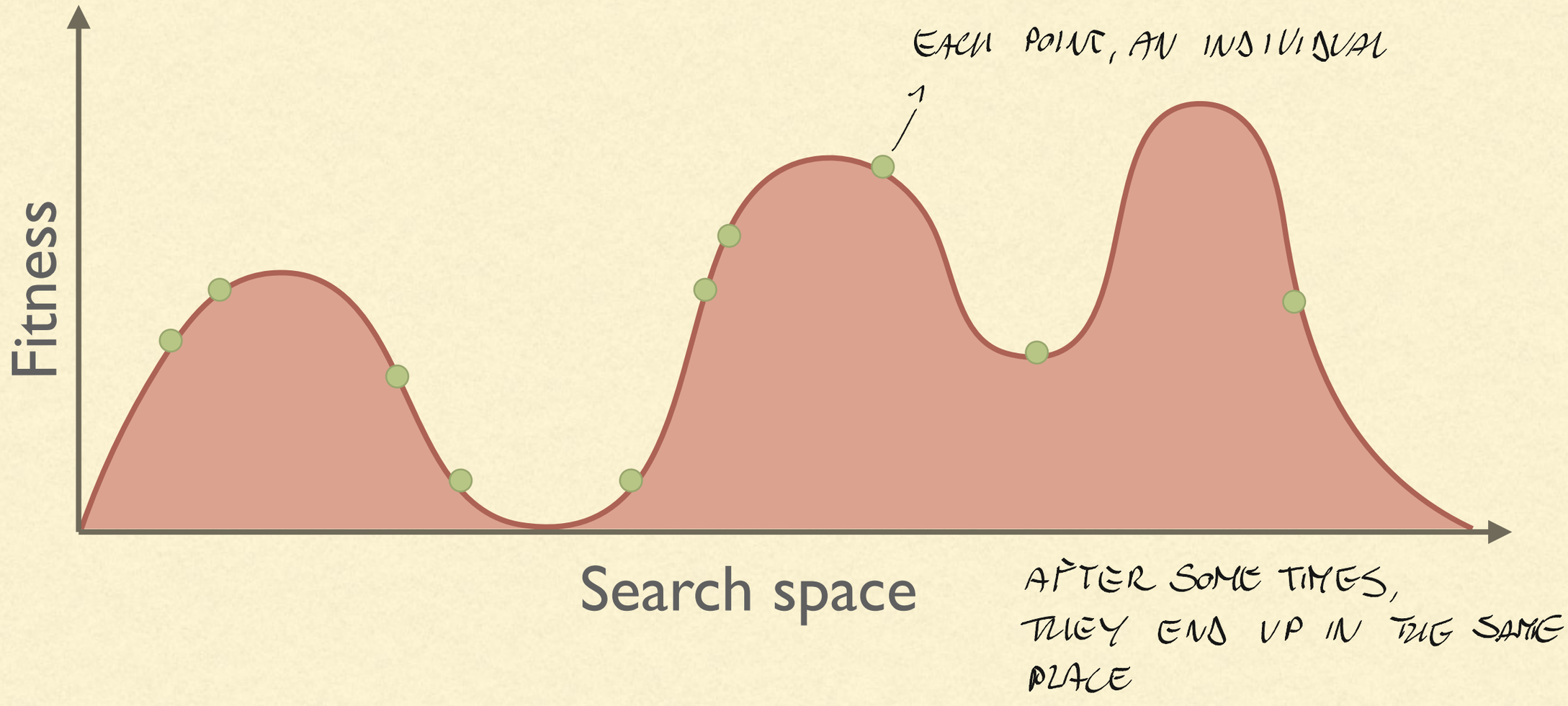
- Overgeneralization: every individual is decent at everything…
- …but good at nothing (but their average performance are good)
- One possible solution is to compute the fitness as the max (resp., min) among all the k tests, instead of taking the average, but it is still not a perfect solution

POSSIBLE DRAWBACKS

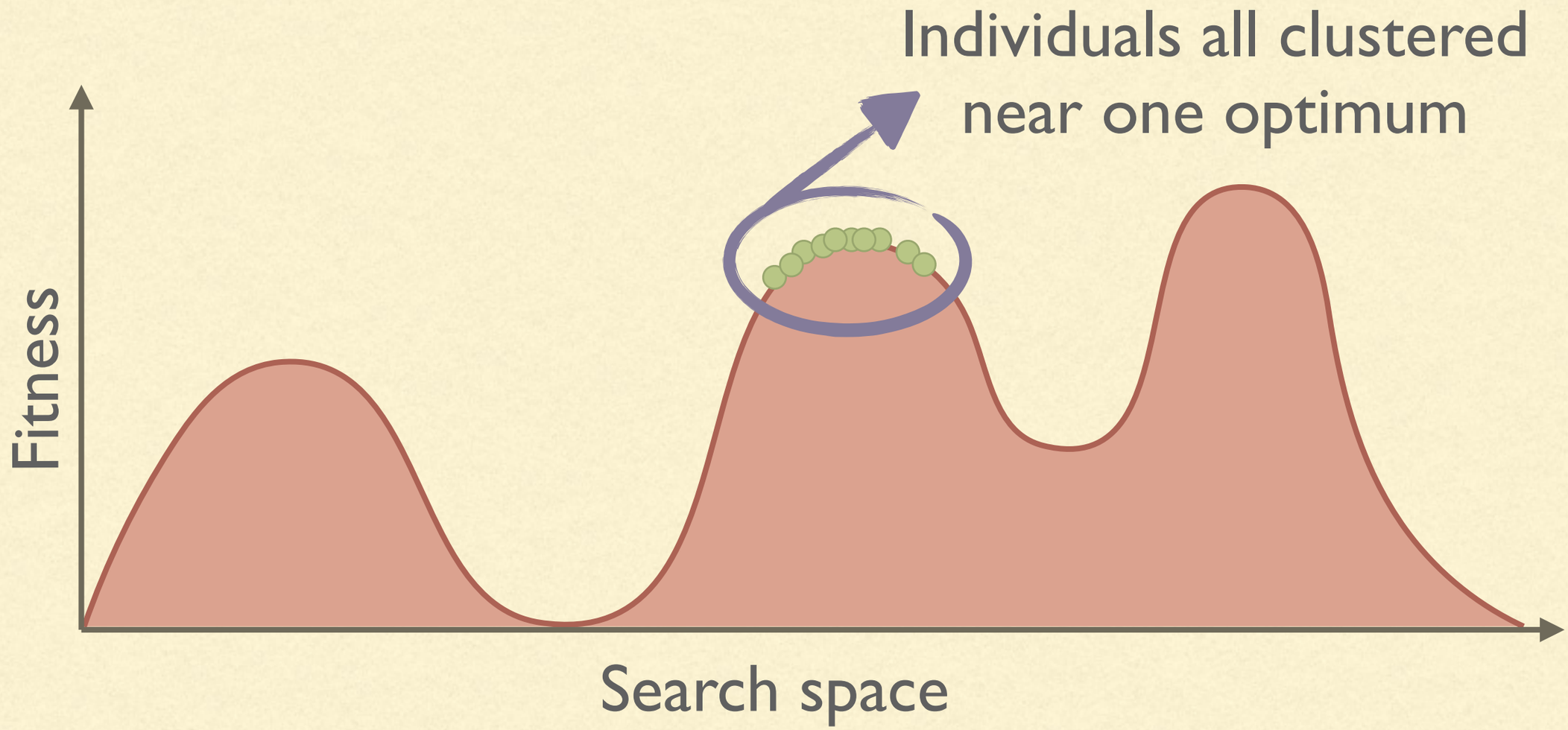


To move from the local optimum 5 the two population must change behavior "at the same time" → LOW PROBABILITY

DIVERSITY MAINTENANCE

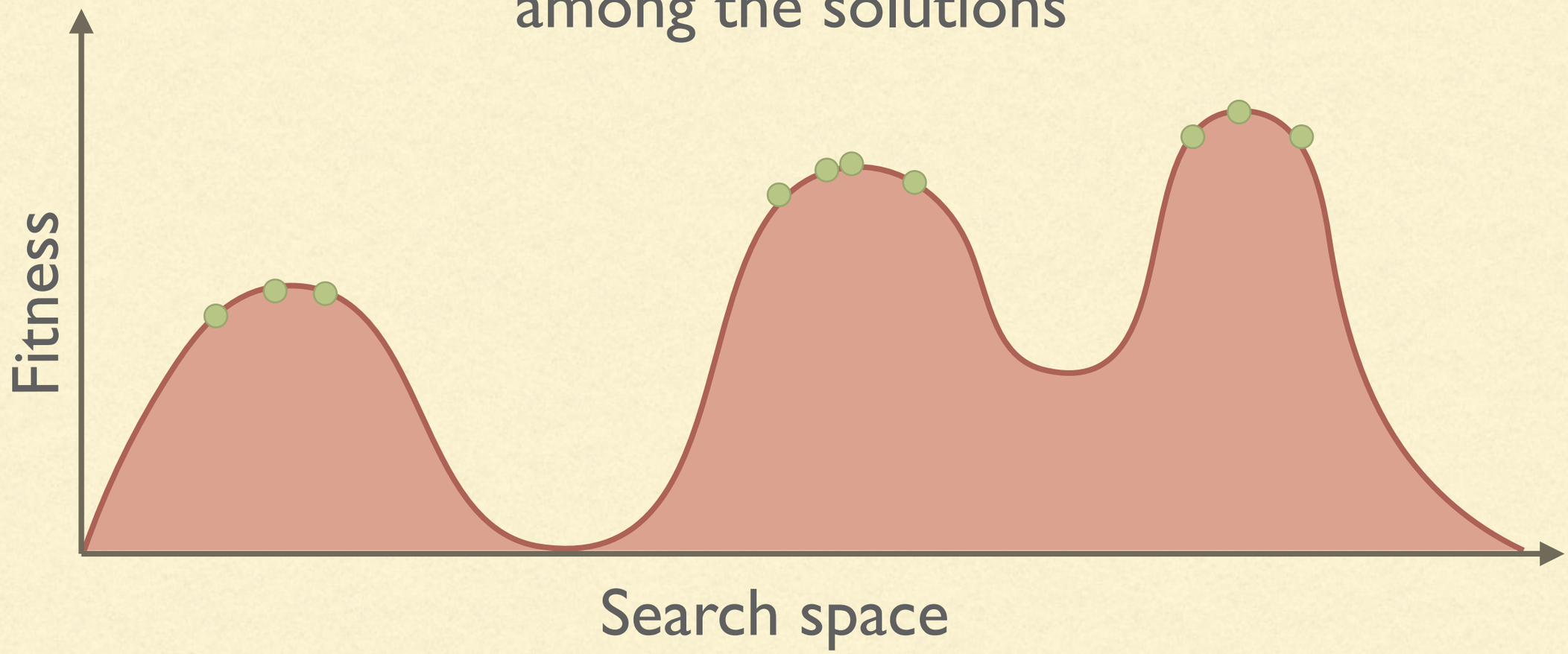


DIVERSITY MAINTENANCE



DIVERSITY MAINTENANCE

But we would like to preserve some diversity among the solutions



DIVERSITY MAINTENANCE

- One of the problems to avoid in optimization is premature convergence to a sub-optimal solution

- One possible way to avoid this is to increase the size of the population or “fiddling” with the parameters of the algorithm

- Otherwise, it is possible to modify the algorithm to enforce some kind of diversity in the population.

MEASURING SIMILARITY

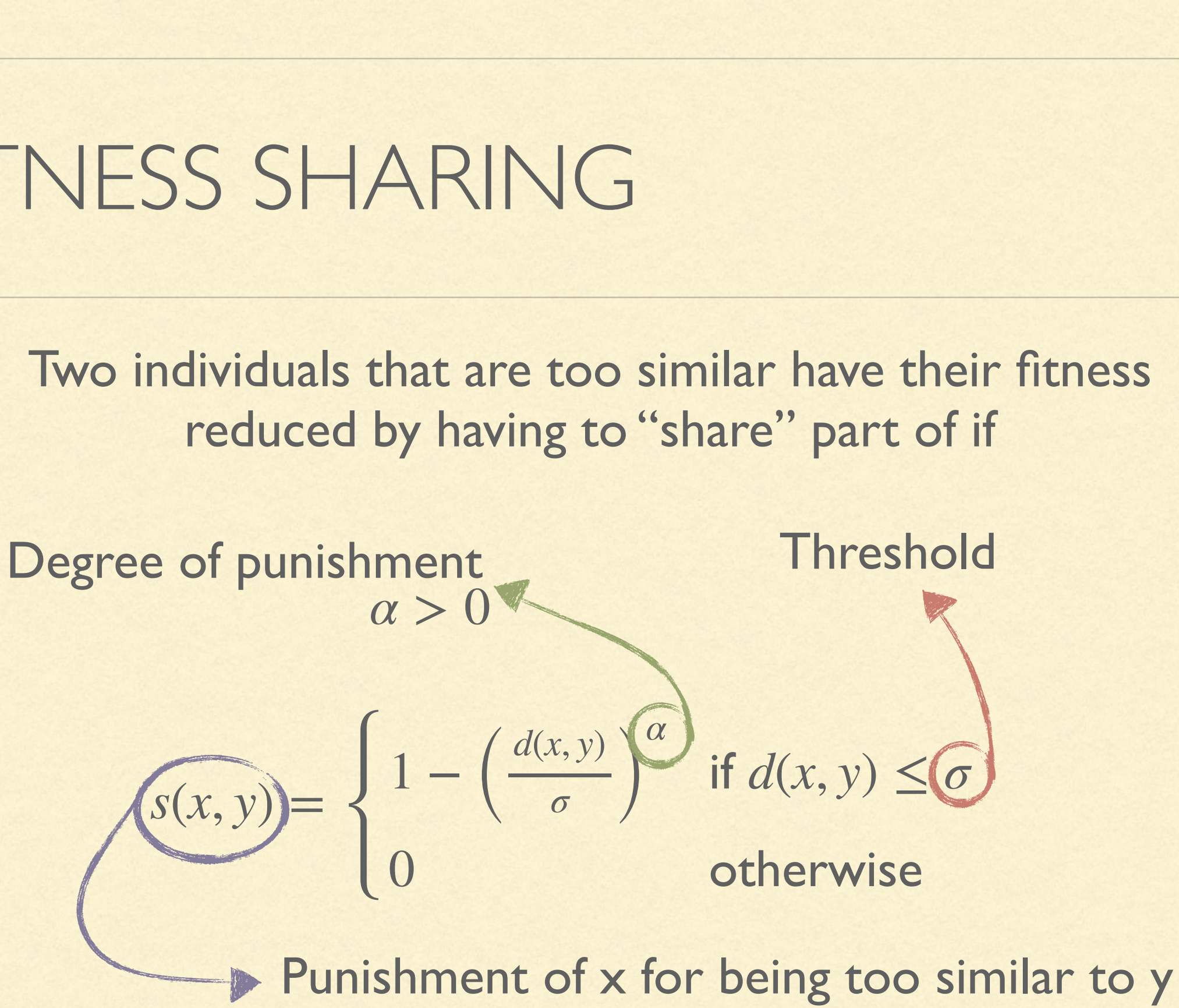
- **Genotype**: similar construction (e.g., small Hamming distance for sequences of bits) ↳ SUBTREES, EUCLIDIAN, ...

- **Phenotype**: similar behavior (the solution encoded is similar). For example (+ x x) and (* x 2) have the same phenotype even if they are represented differently

- **Fitness**: similar fitness value (useful in the case of multi-objective optimisation)

FITNESS SHARING

Two individuals that are too similar have their fitness reduced by having to "share" part of if



FITNESS SHARING

The new value of the fitness is computed using the punishment:

Raw fitness

Scaling factor
$\beta > 1$

$$f(x) = \frac{r(x)^{\beta}}{\sum_y s(x, y)}$$

A total of 3 parameters: α, β, σ