EVOLUTION STRATEGIES
REPRESENTATIONS FOR PARTICULAR PROBLEMS
DISTRIBUTED METHODS

Luca Manzoni

EVOLUTION STRATEGIES

EVOLUTION STRATEGIES: IDEAS

- Invented in the ’60
- In some sense similar to GA:
  - There is a population of solutions
  - There are offsprings derived from mutation
  - There is a selection process

EVOLUTION STRATEGIES: IDEAS

- However, they have some key differences:
  - There is (usually) no crossover
  - The most used selection is truncated selection
  - Usually the individuals represent floating points values (which is also possible with GA)

ES PARAMETERS

λ

Number of selected individuals

μ

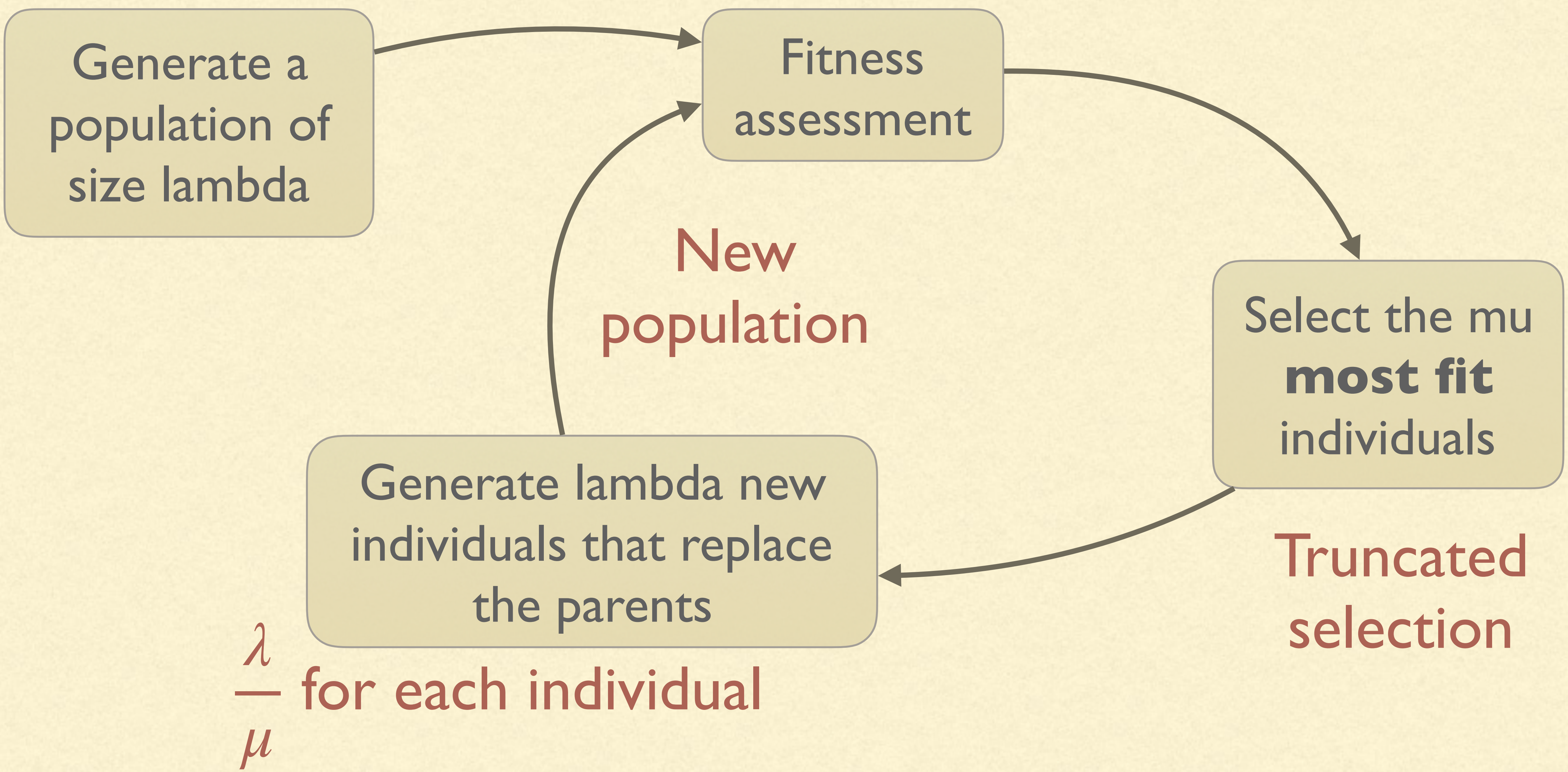
Number of children that are generated

Two different kinds of ES:

$$(\mu, \lambda) - ES \qquad (\mu + \lambda) - ES$$

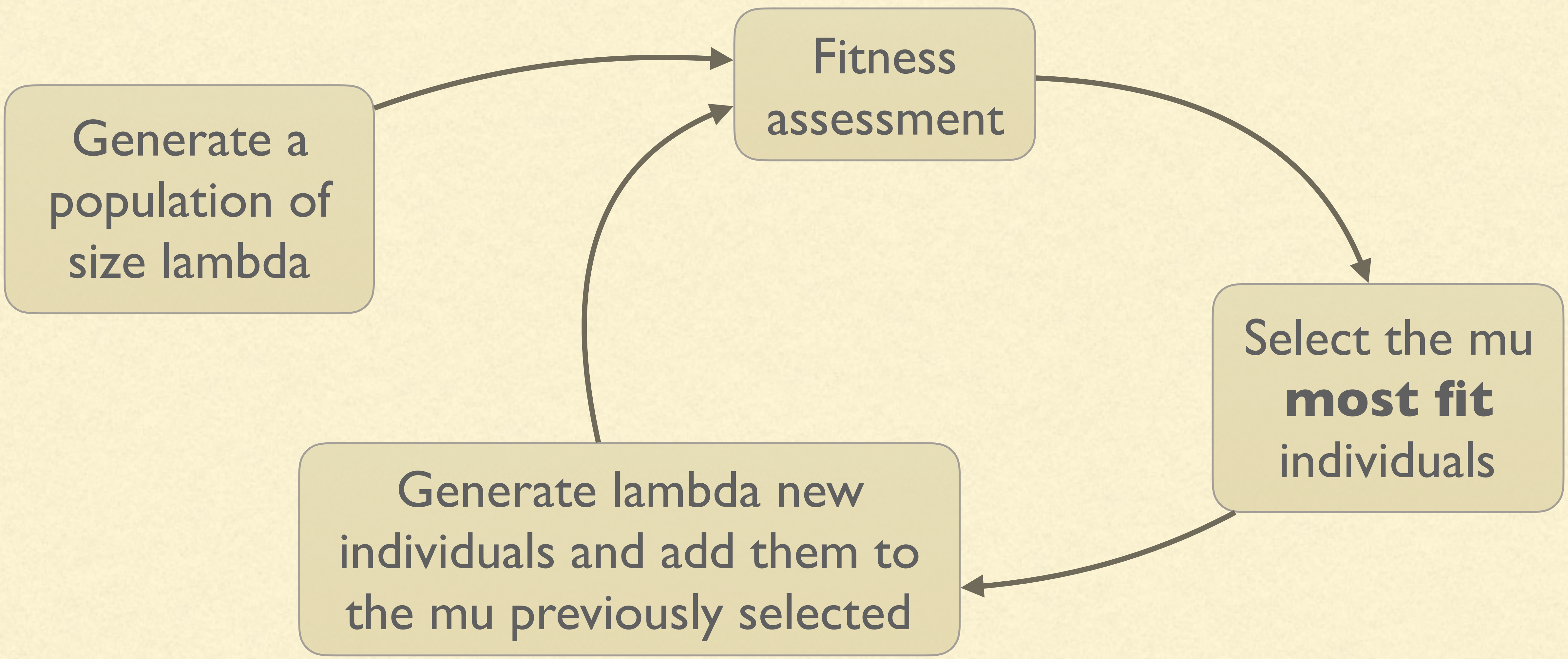
THE **ES** CYCLE

$$(\mu, \lambda) - ES$$



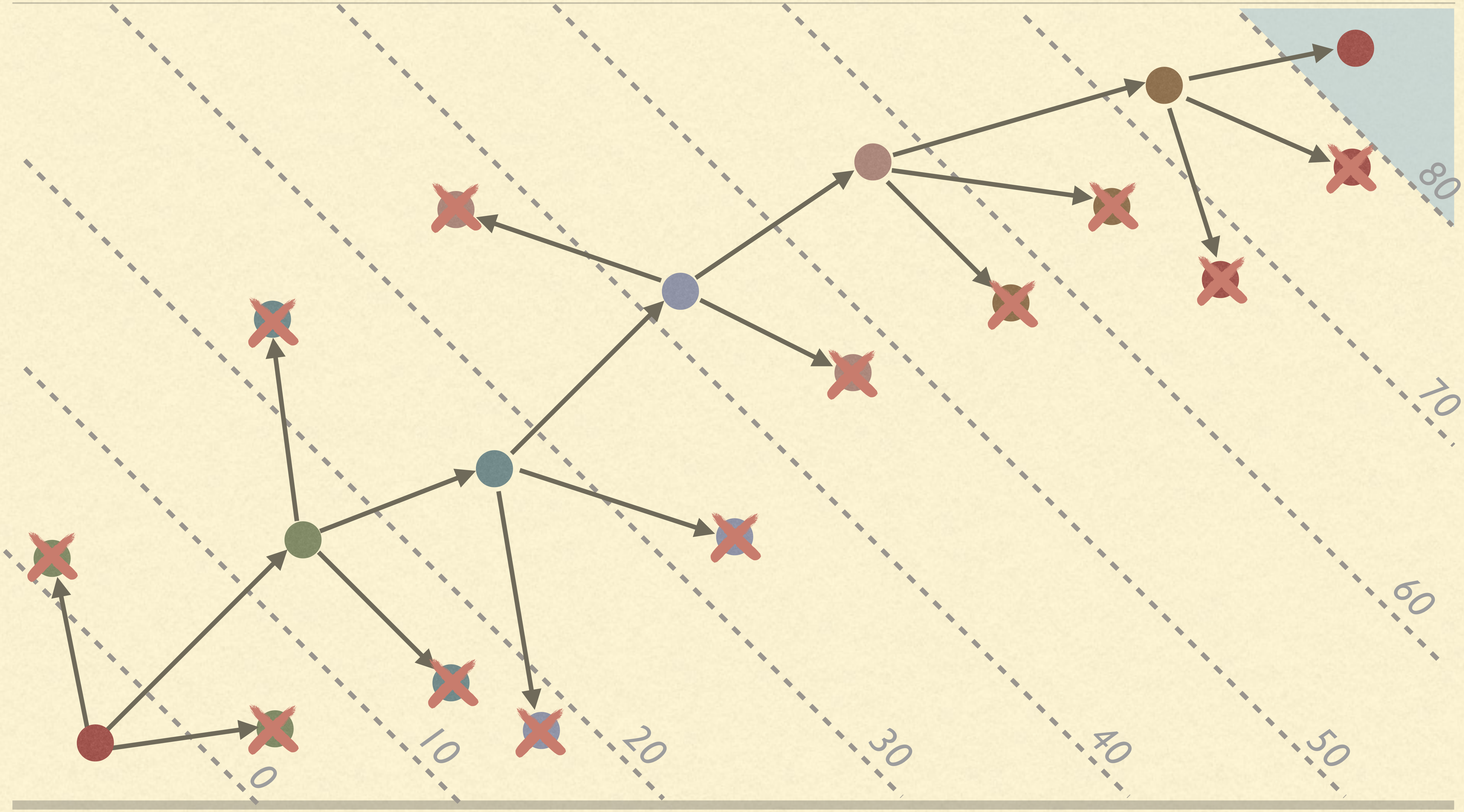
THE **ES** CYCLE

$$(\mu + \lambda) - ES$$



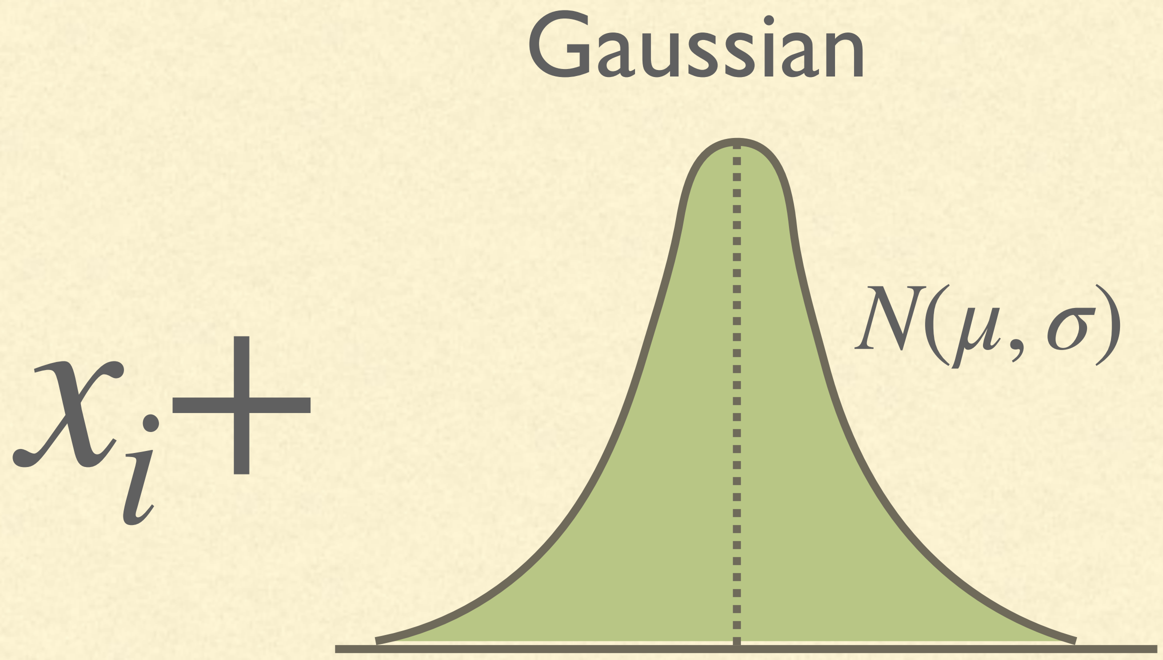
The parents are added together
with the children to the new population

EXAMPLE OF (1,3) ES



MUTATION

In the case of real values the mutation is usually performed by adding a gaussian noise to the coordinates



But how to select the variance/standard deviation?

ONE-FIFTH RULE

- An empirical rule for self-adaptation of the variance of the mutation operator
- If less than 1/5 of the children are fitter than their parents then decrease the variance
- If more than 1/5 of the children are fitter than their parents then increase the variance

REPRESENTATION FOR PARTICULAR PROBLEMS

REAL-VALUED GA

- Until now we have seen binary valued (or integer valued) GA
- We can represent each floating point numbers as 32/64 binary genes…
- …but this means that different bits have different impact on the encoded number
- If each gene is a floating point value then mutation and crossover should be adapted

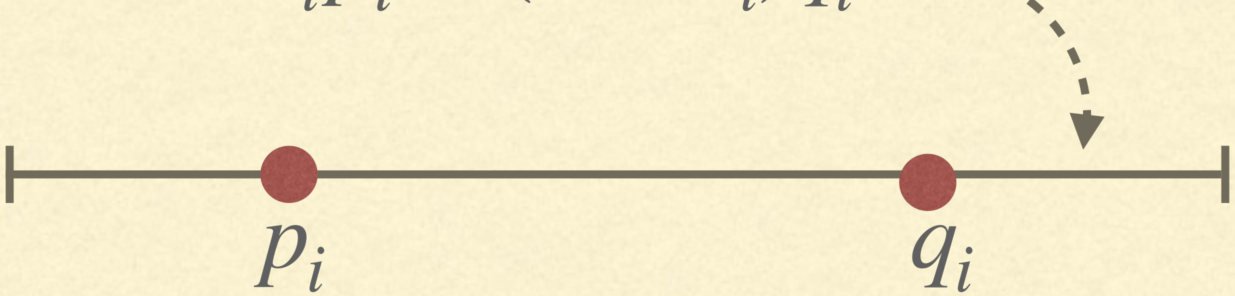
CROSSOVER

$\alpha_i p_i + (1-\alpha_i) q_i$

Intermediate recombination

$$\alpha_i \leftarrow \text{random}(0,1)$$

$\alpha_i p_i + (1-\alpha_i) q_i$

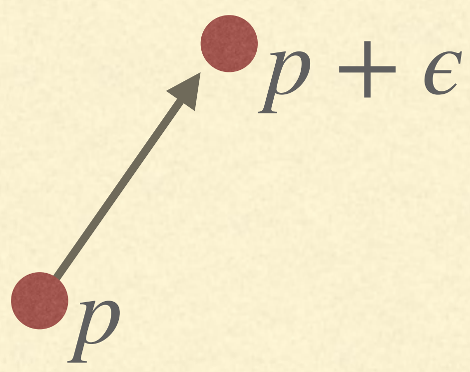


Line recombination

$$\alpha_i \leftarrow \text{random}(-k, 1+k)$$

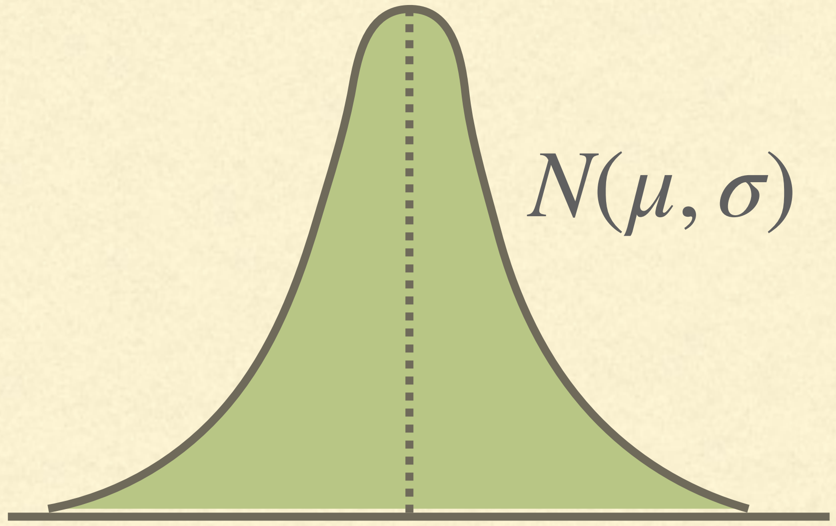
MUTATION

Add a small value to each coordinate of the individual



Uniform between two values

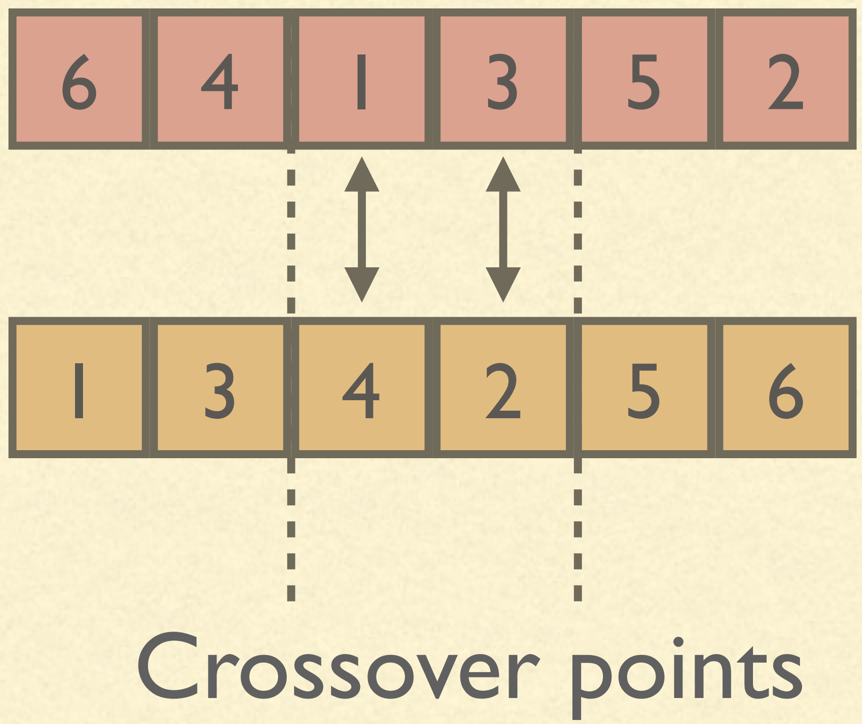
Gaussian

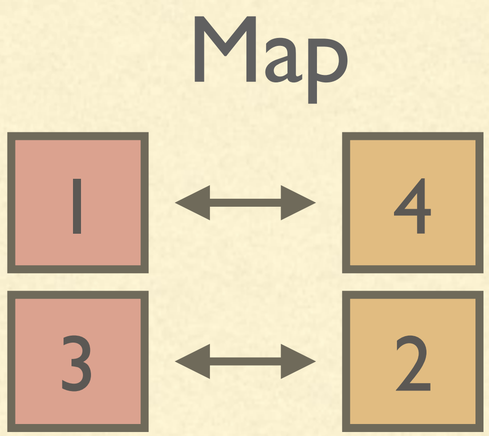


CYCLE AND PMX CROSSOVERS

- Sometimes we need additional constraints in the representation of an individual by GA
- One usual constraint is that each individual must be a permutation of the numbers from 1 to n
- Mutation can be performed by swapping two positions
- Traditional crossover usually do not respect the constraints

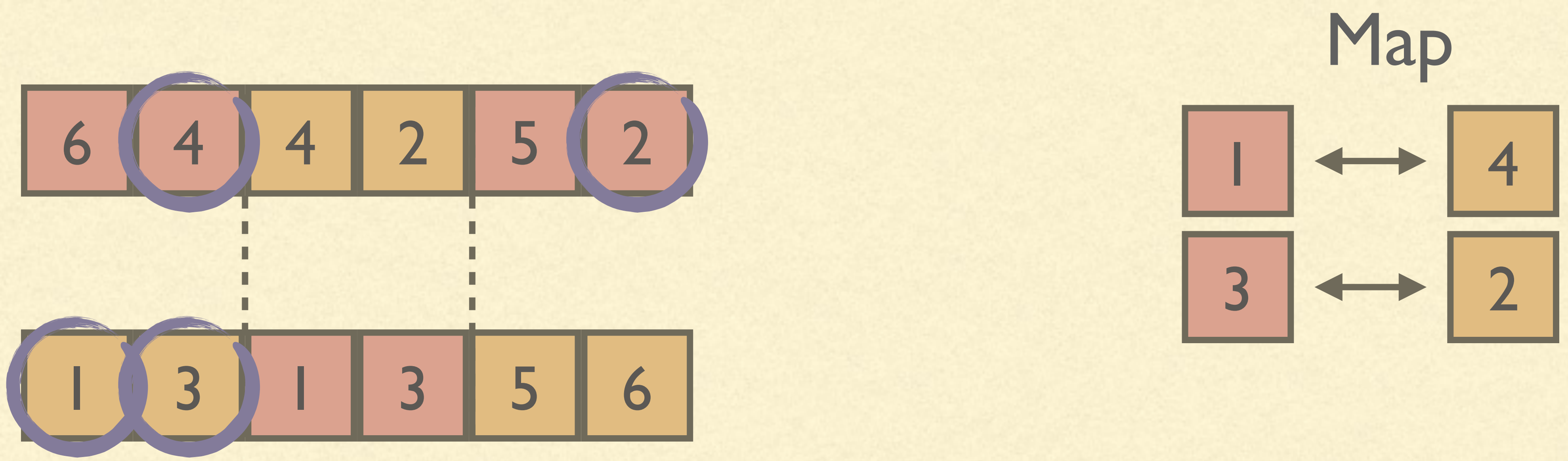
PARTIALLY MAPPED CROSSOVER (PMX)





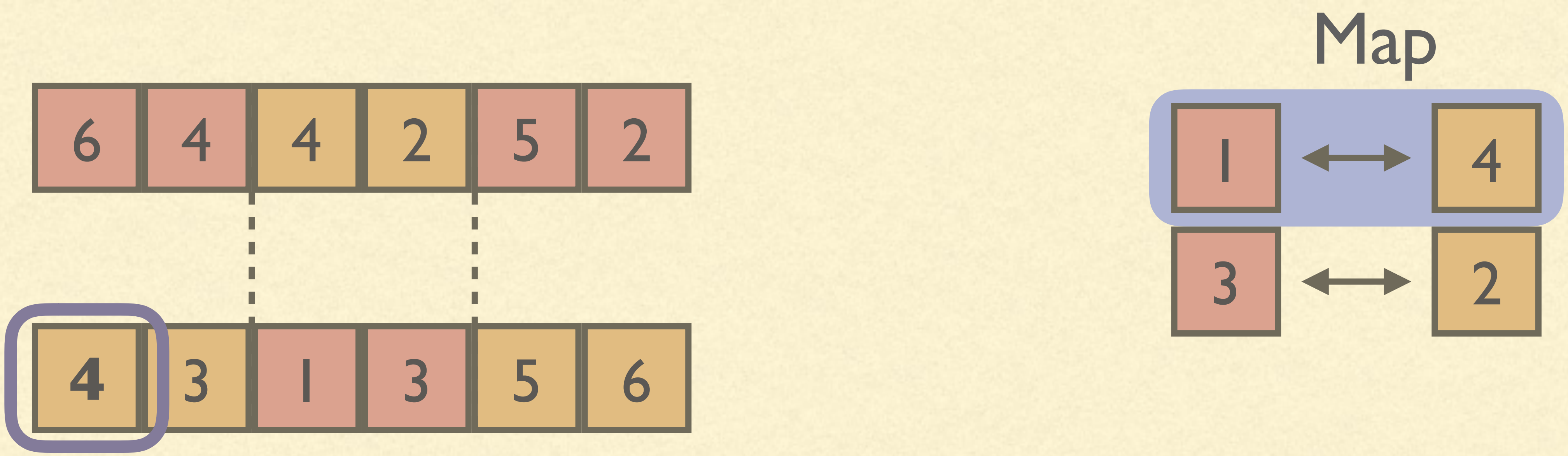
We select two crossover point and we build a map

PARTIALLY MAPPED CROSSOVER (PMX)



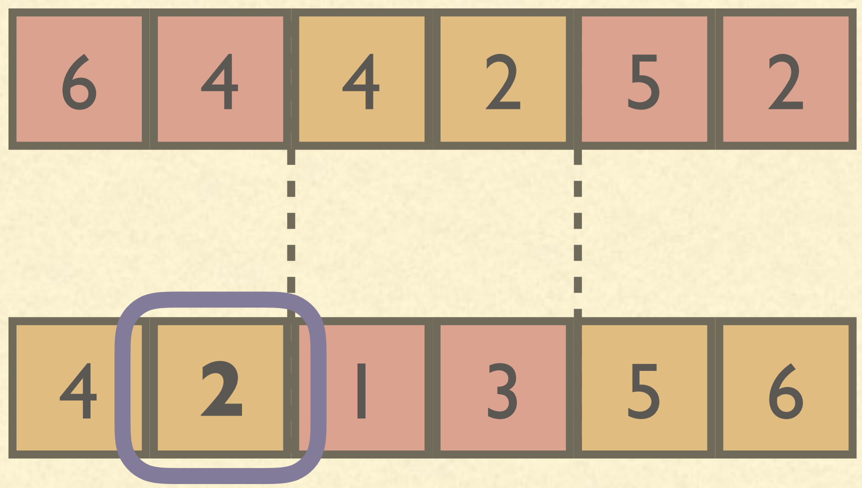
We perform the exchange but the offspring are not valid

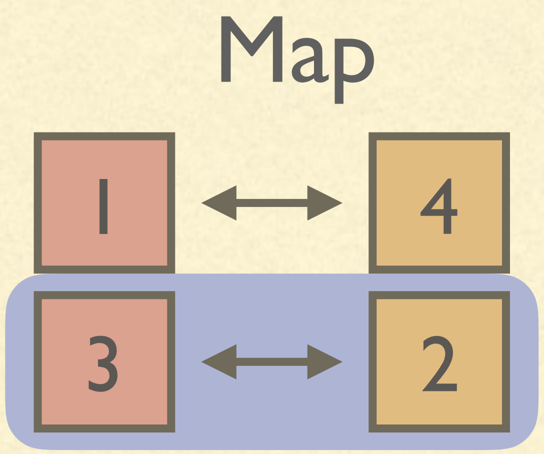
PARTIALLY MAPPED CROSSOVER (PMX)



We iterate the map until we have resolved the conflicts

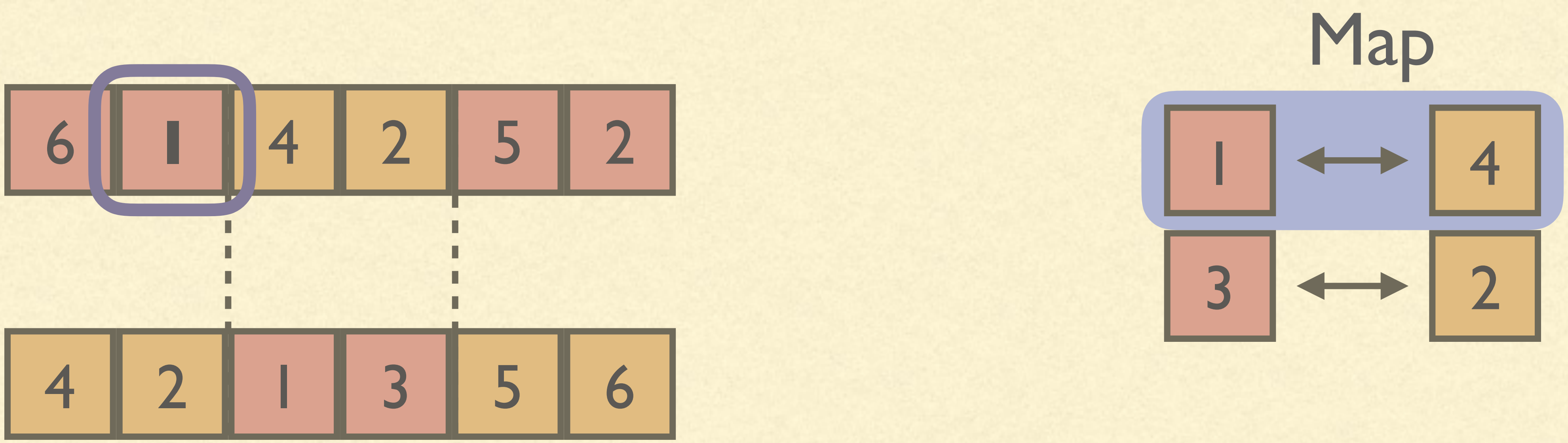
PARTIALLY MAPPED CROSSOVER (PMX)





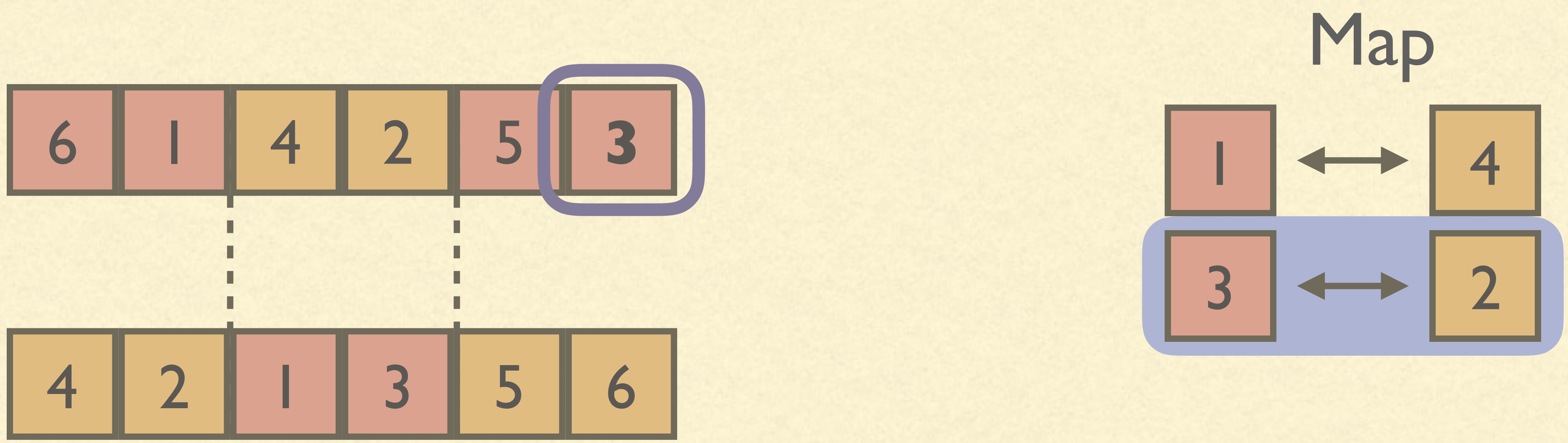
We iterate the map until we have resolved the conflicts

PARTIALLY MAPPED CROSSOVER (PMX)



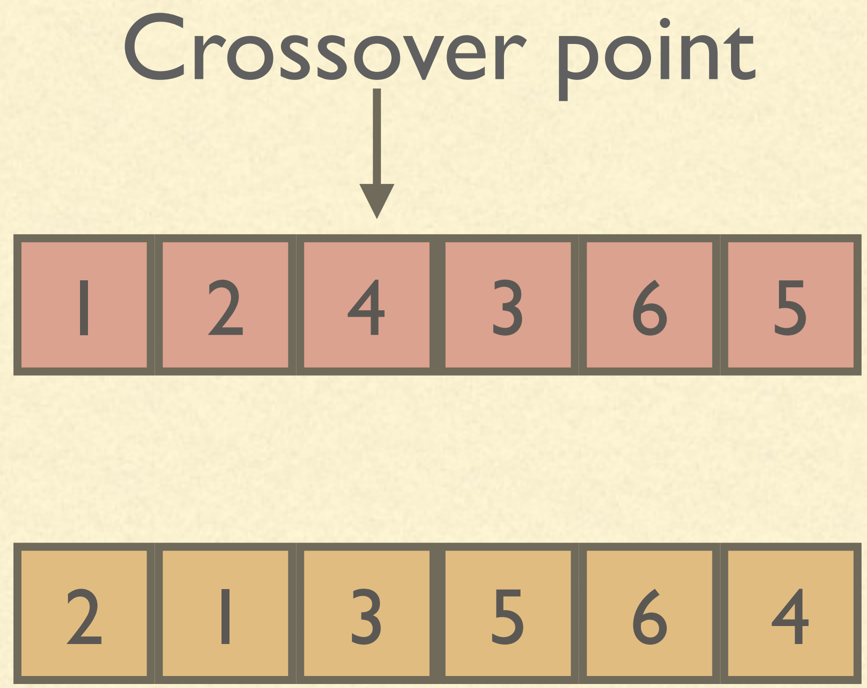
We iterate the map until we have resolved the conflicts

PARTIALLY MAPPED CROSSOVER (PMX)



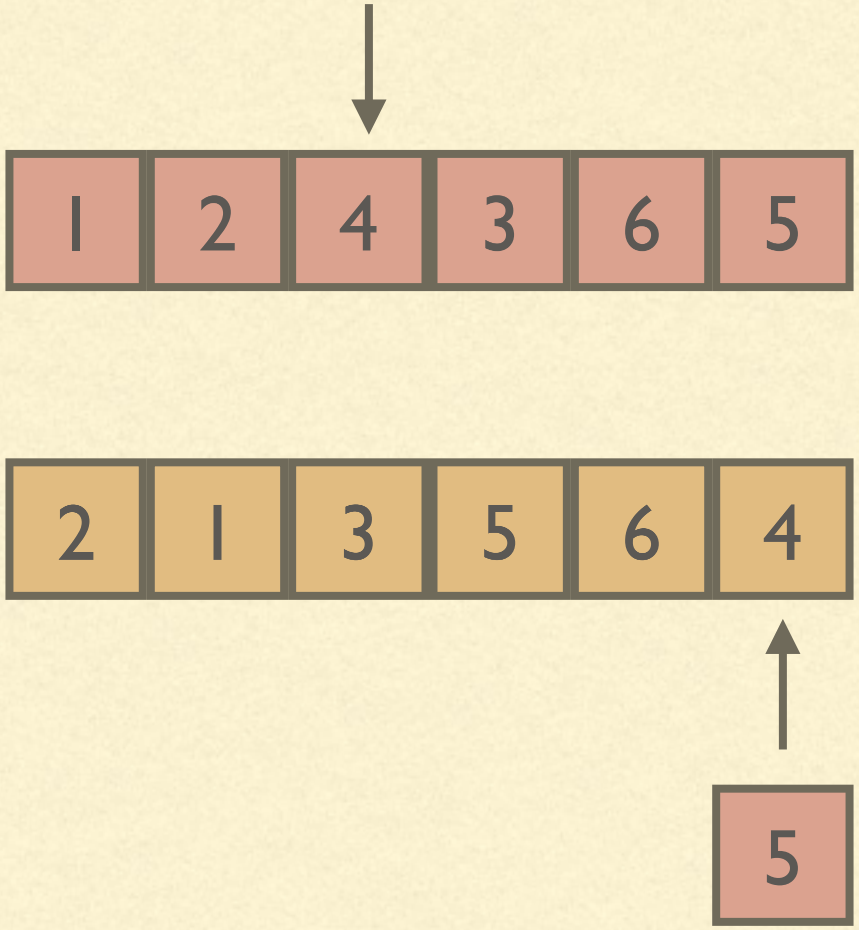
We iterate the map until we have resolved the conflicts

CYCLE CROSSOVER



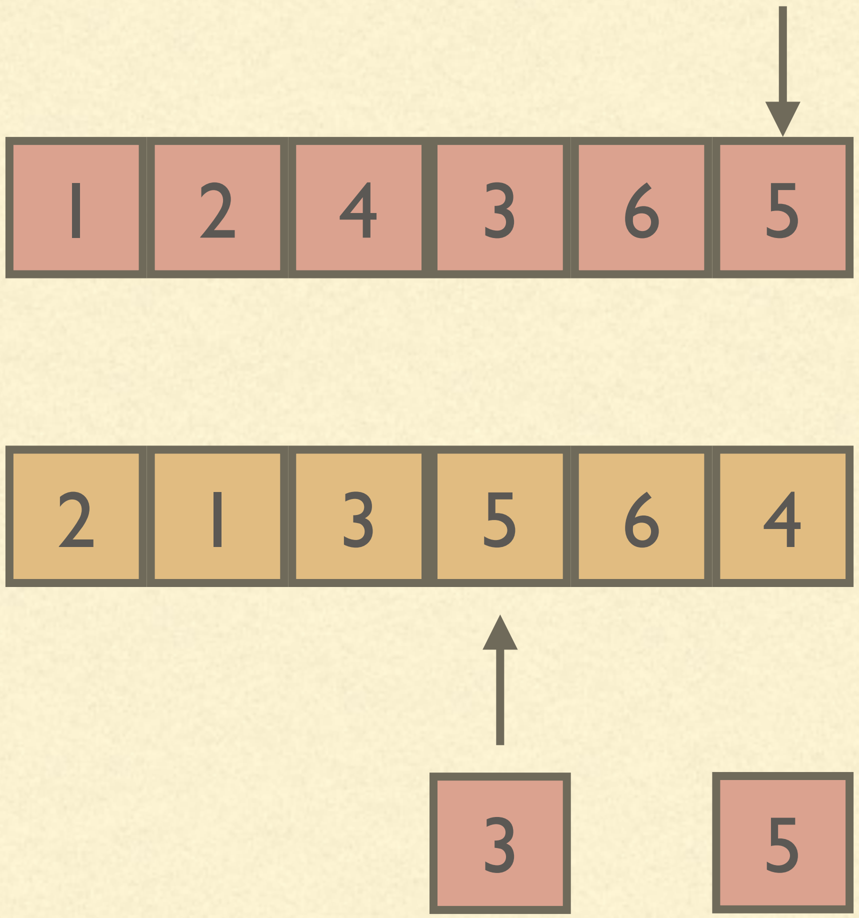
We select a single starting point
and we search che same value in the second parent

CYCLE CROSSOVER



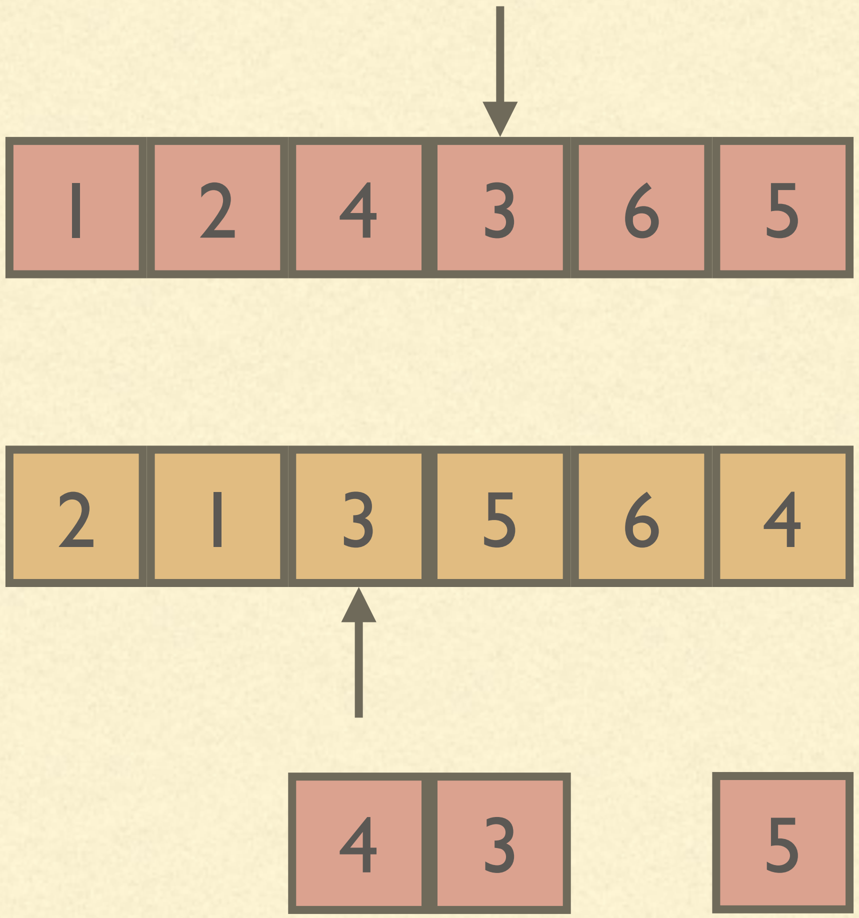
Once found we copy the value from the first parent.
Repeat until we return to the beginning

CYCLE CROSSOVER



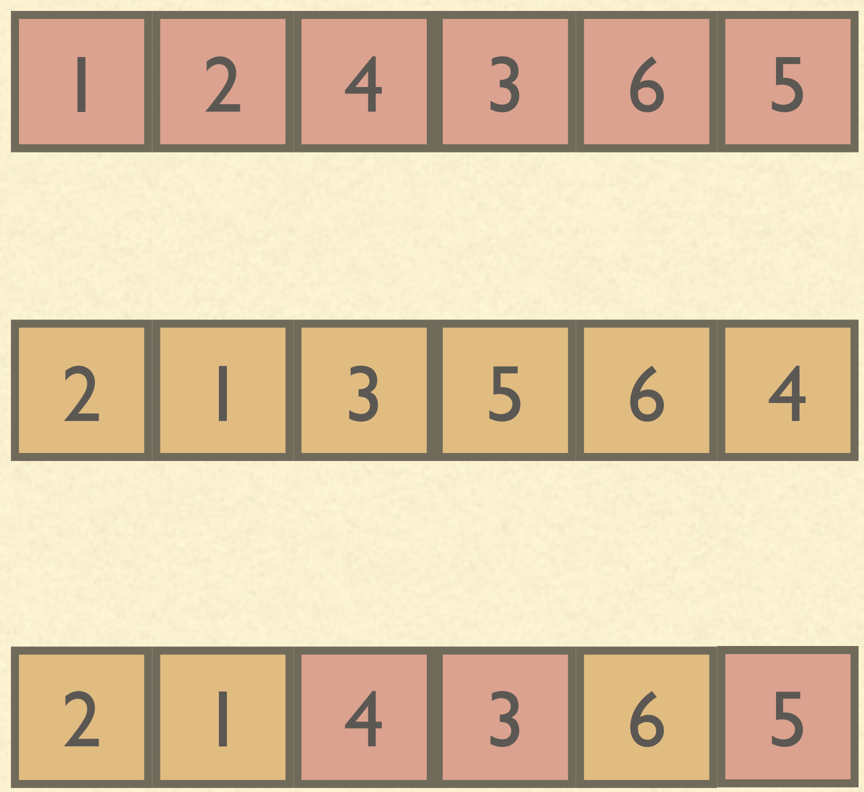
Once found we copy the value from the first parent.
Repeat until we return to the beginning

CYCLE CROSSOVER



Once found we copy the value from the first parent.
Repeat until we return to the beginning

CYCLE CROSSOVER



We copy the remaining elements from the second parent

REPRESENTING GRAPHS

- You might want to represent graphs. Possibly because the are ubiquitous in computer science.
- You can represent graph in two ways:
  - *Direct encoding*. By actually representing vertices and edges
  - *Indirect encoding*. By representing some “device” that builds a graph

ADJACENCY MATRIX

Side of the matrix = max number of nodes

$$\begin{bmatrix} 0.4 & \text{no edge} & 0.6 & -5.3 \\ \text{no edge} & 5.6 & 0.1 & 0.2 \\ 2.4 & 0.8 & 4.1 & 8.3 \\ -0.2 & \text{no edge} & 0.5 & \text{no edge} \end{bmatrix}$$

Special value to represent missing edges

LARGE GRAPHS

$V = \{a, b, c, d, e\}$ We evolve the sets of vertices and edges

$E = \{(a, b), (a, c), (d, a), (e, e)\}$

Possible mutations:

- Add an edge
- Add a node
- Remove an edge
- Remove a node and all its edges
- …

Each one can have
a different probability

Crossover is difficult
to define and you might
decide not to use it

PRODUCTION RULES

- There are terminal symbols and non-terminal symbols
- Production rules map a non-terminal symbol into a sequence/matrix of non-terminal and terminal symbols
- We continue the expansion until the configuration is composed only of terminal symbols
- By using adeguate production rules we can encode indirectly a graph (i.e., rules that build a graph)

PRODUCTION RULES

$$S \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$A \rightarrow \begin{bmatrix} c & p \\ a & c \end{bmatrix} \; B \rightarrow \begin{bmatrix} a & a \\ a & e \end{bmatrix} \; C \rightarrow \begin{bmatrix} a & a \\ a & a \end{bmatrix} \; D \rightarrow \begin{bmatrix} a & a \\ a & b \end{bmatrix}$$

$$a \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \; b \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \; c \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \; e \rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \; p \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Ruleset used in an example in:

Hiroaki Kitano, Designing neural networks using a genetic algorithm with a graph generation system

PRODUCTION RULES

Starting from an axiom [S] we can iterate the production rules

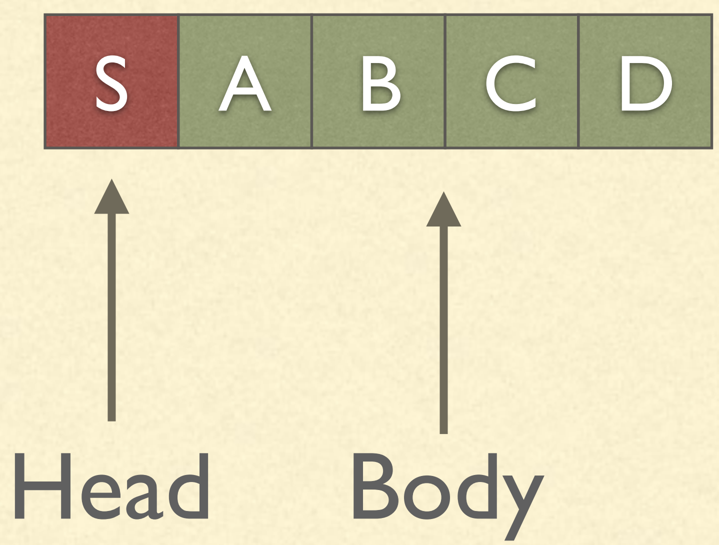
$$[S] \longrightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix} \longrightarrow \begin{bmatrix} c & p & a & a \\ a & c & a & e \\ a & a & a & a \\ a & a & a & b \end{bmatrix} \longrightarrow \cdots$$

PRODUCTION RULES

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

A graph of 5 vertices
(8 encoded but 3 of them
are not connected to anything)

PRODUCTION RULE: ENCODING



encodes $$S \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Since now we have a vector of fixed length,
to perform the evolution
we can apply traditional GA operators

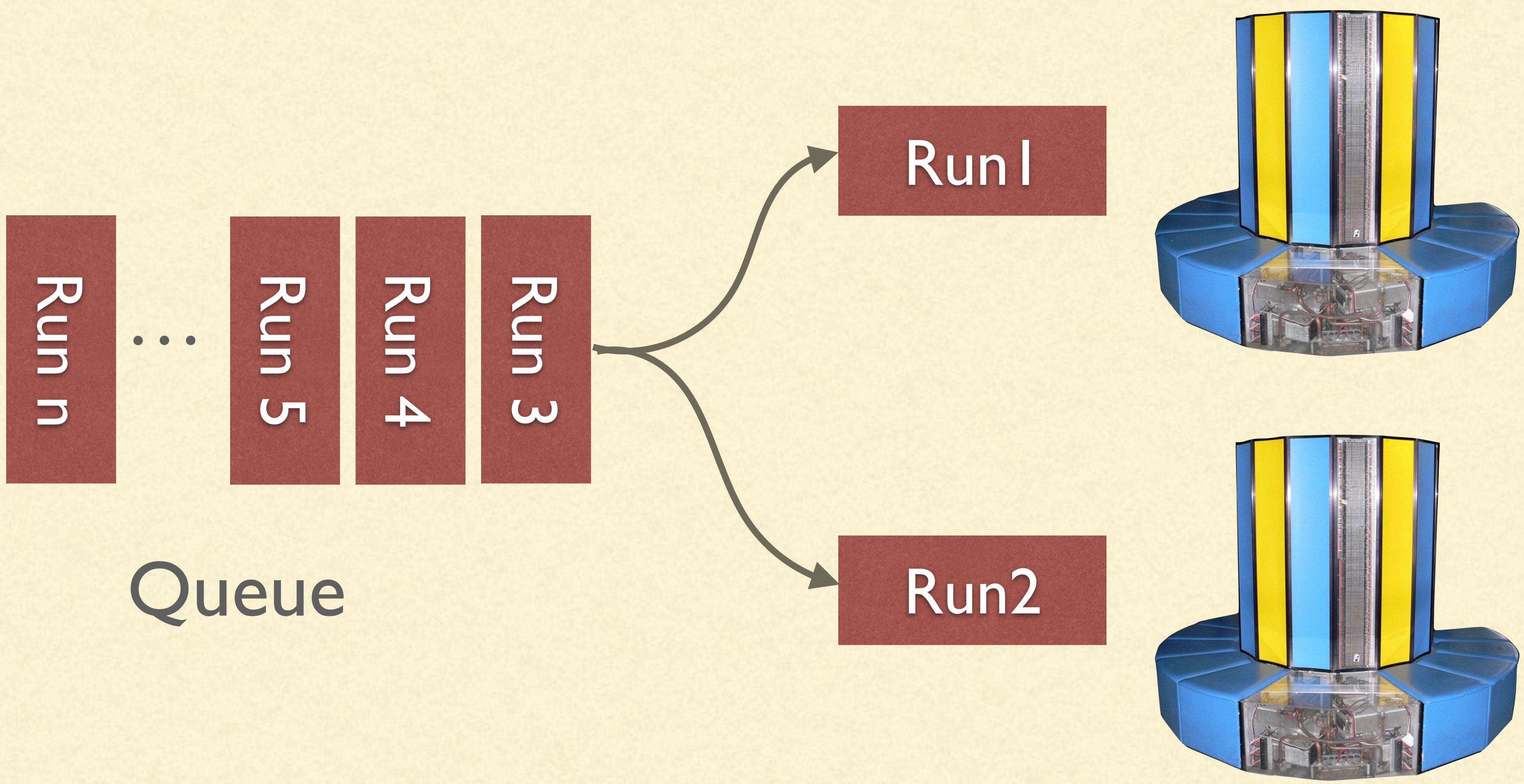
PARALLEL AND DISTRIBUTED METHODS

WHY?

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

THE SIMPLEST WAY

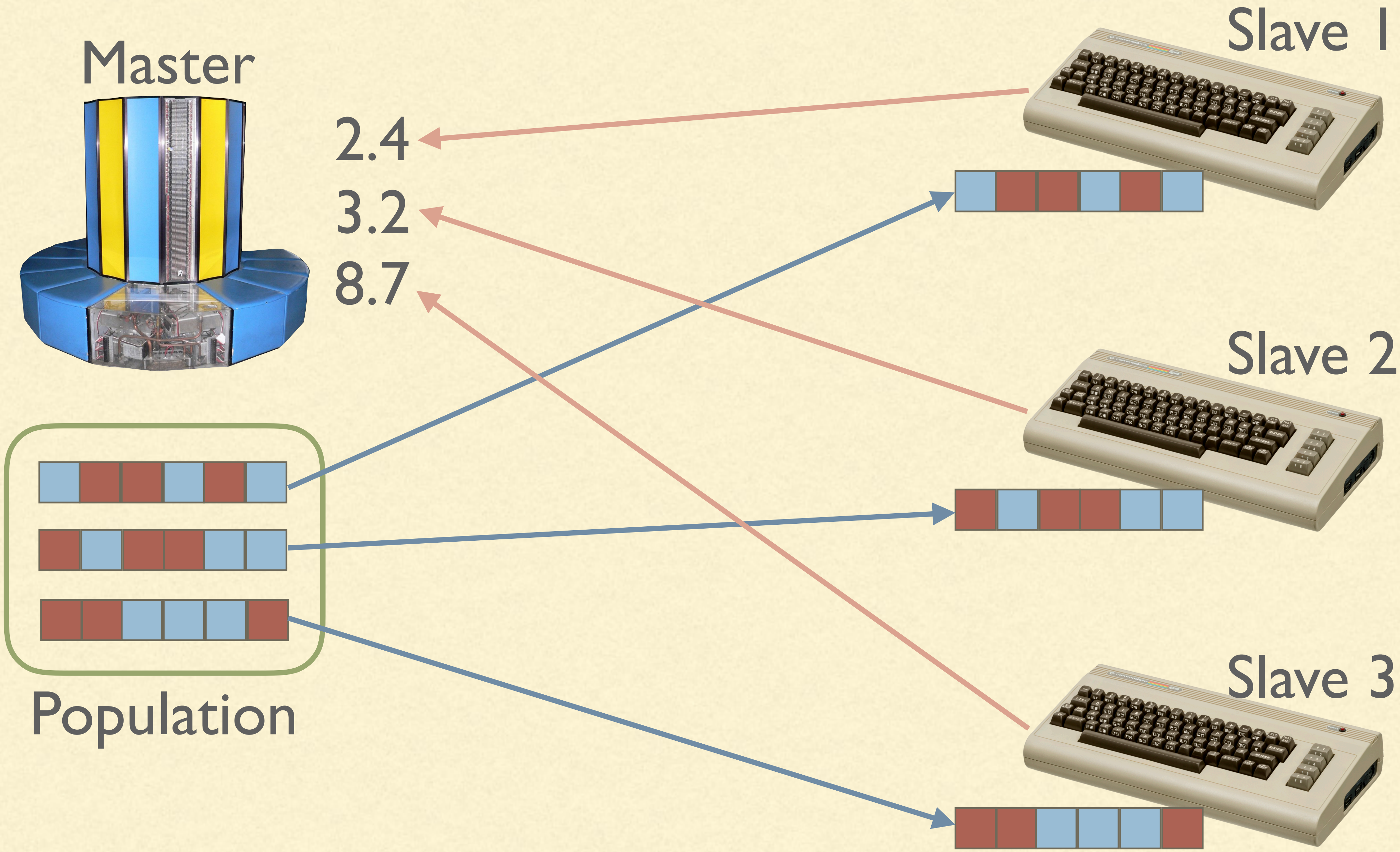
You have to perform **n** runs,
and you have multiple cores/computers



DISTRIBUTED FITNESS ASSESSMENT

- Also known as client-slave or master-server
- Idea fitness evaluation can be (by far) the most expensive operation
- Keep the evolution process inside a single node (the master)…
- …but move the fitness evaluation among a set of “slave” nodes

DISTRIBUTED FITNESS ASSESSMENT



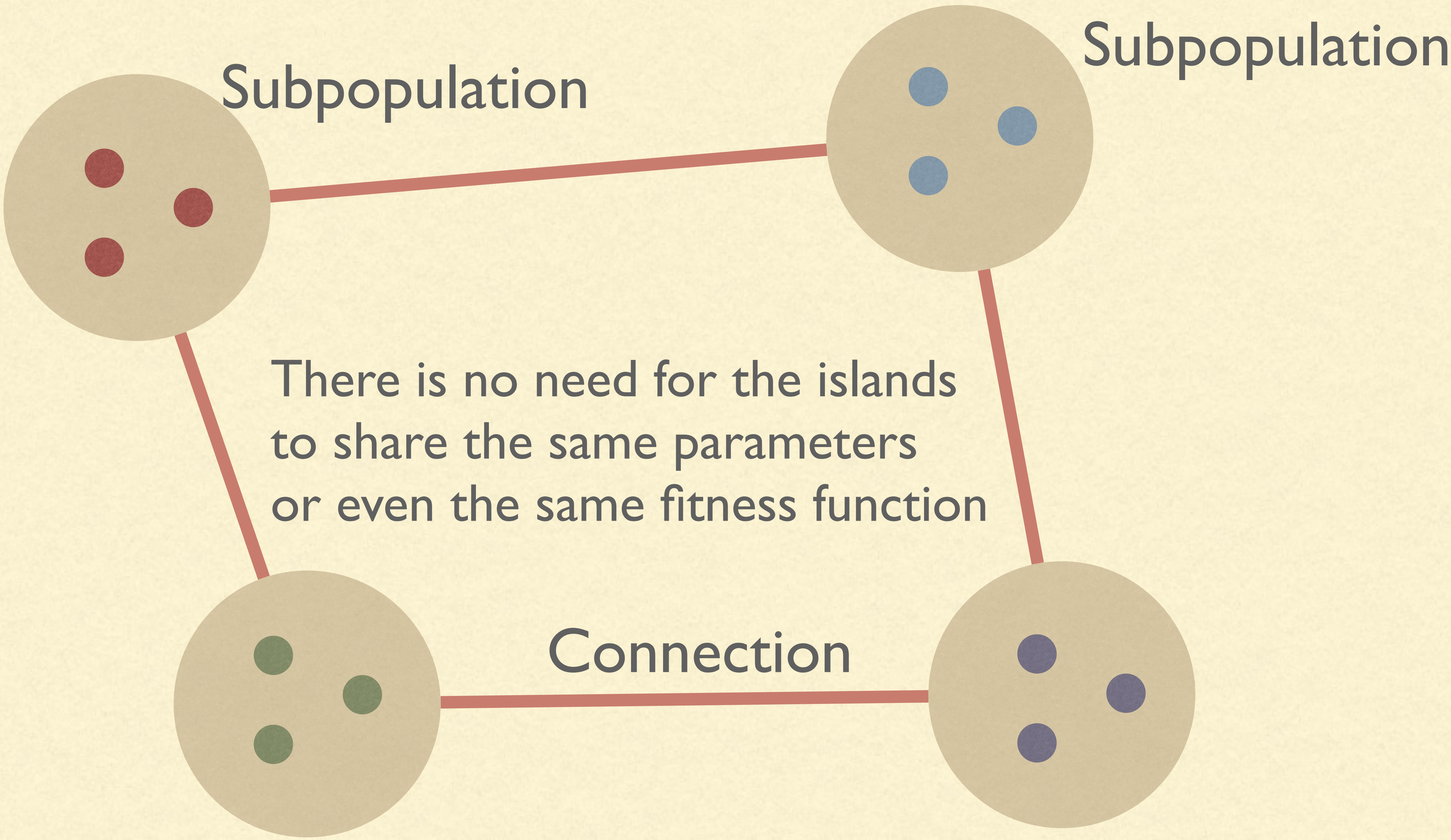
ADVANTAGES AND DISADVANTAGES

- If the fitness assessment is long then this method scales well
- But if the time to transmit the individual to the slaves is in the same order of magnitude of the fitness evaluation then there is no advantage
- You only need to transmit the individuals (possibly compressed) and to receive the fitness value
- Resistant to failures of the slave nodes
- Possible to add/remove nodes when needed

ISLAND MODEL

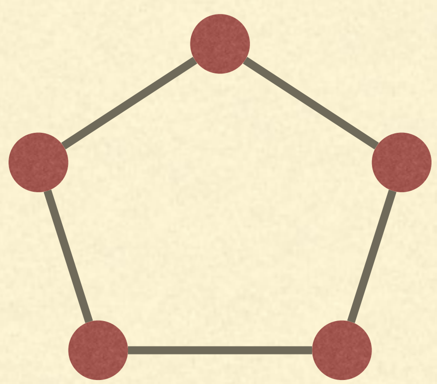
- Distributed evolution: like in real-world islands, populations evolve independently with the occasional exchange of individuals
- New additional parameters:
  - The number of the islands
  - The size of the populations on the islands
  - How the islands can talk (the topology)
  - Which individuals they exchange and how frequently.

ISLAND MODEL

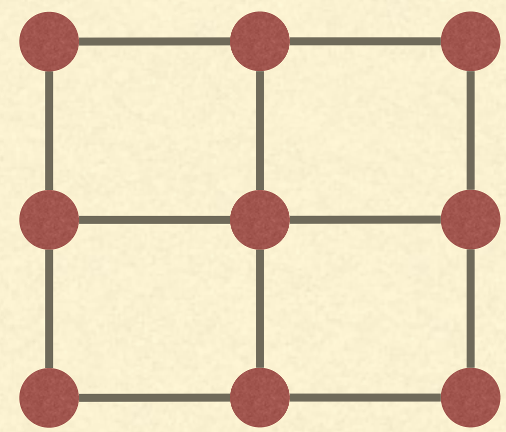


There is no need for the islands to share the same parameters or even the same fitness function

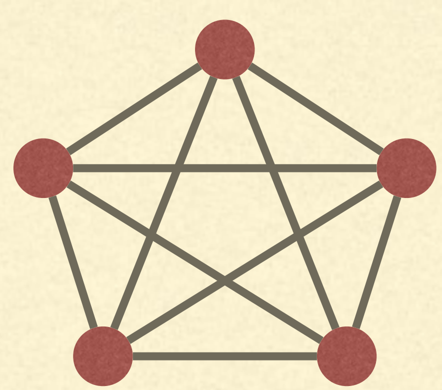
ISLAND MODEL: TOPOLOGIES



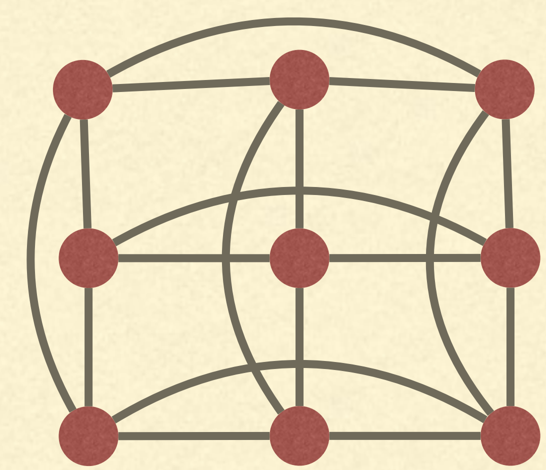
Ring



Grid



Fully connected



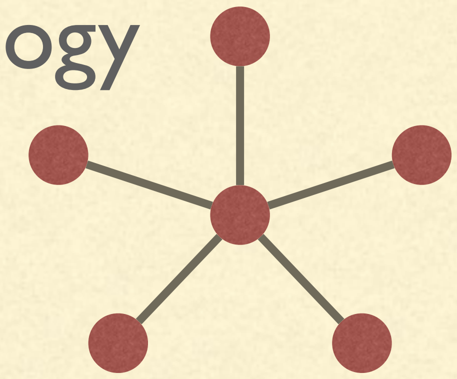
Toroid

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

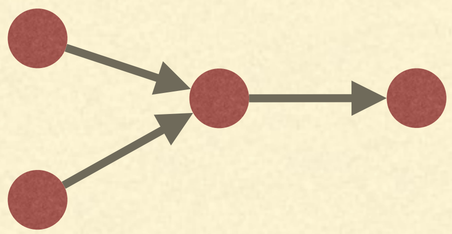
ONE "MASTER" POPULATION

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 "master" that collects individuals from all the other islands

COLLECTOR TOPOLOGY



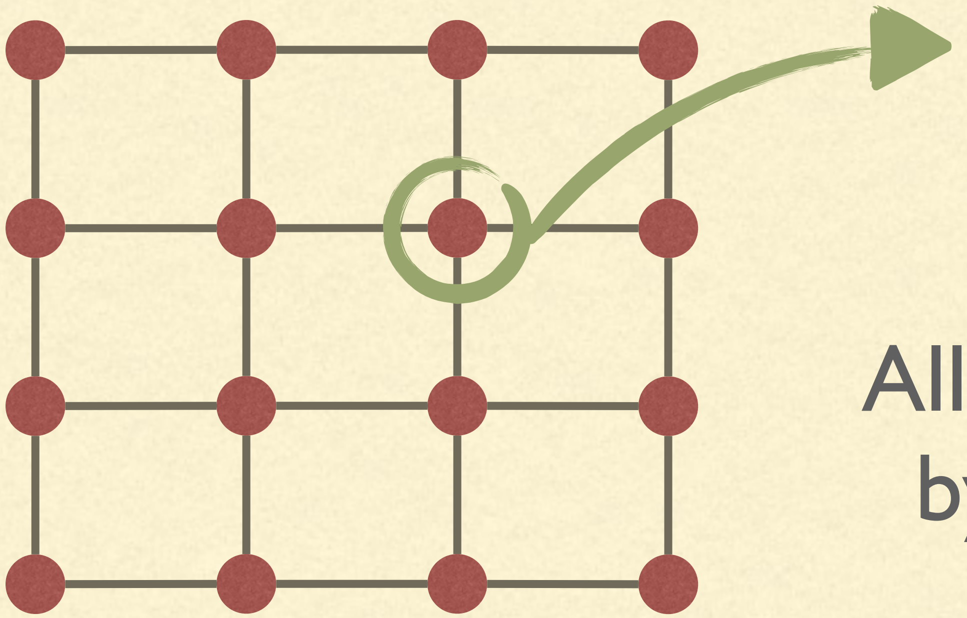
A directed acyclic graph

- Individuals only move in one direction
- This can be useful to optimise 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

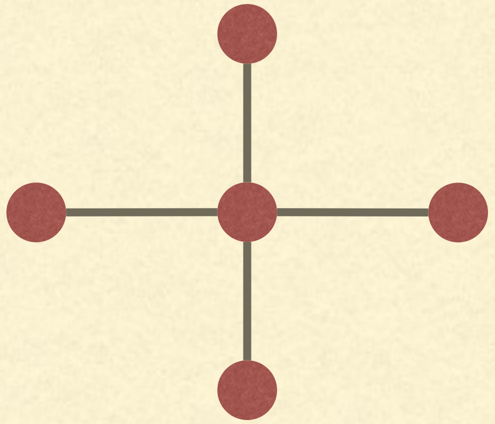
SPATIALLY-EMBEDDED EA



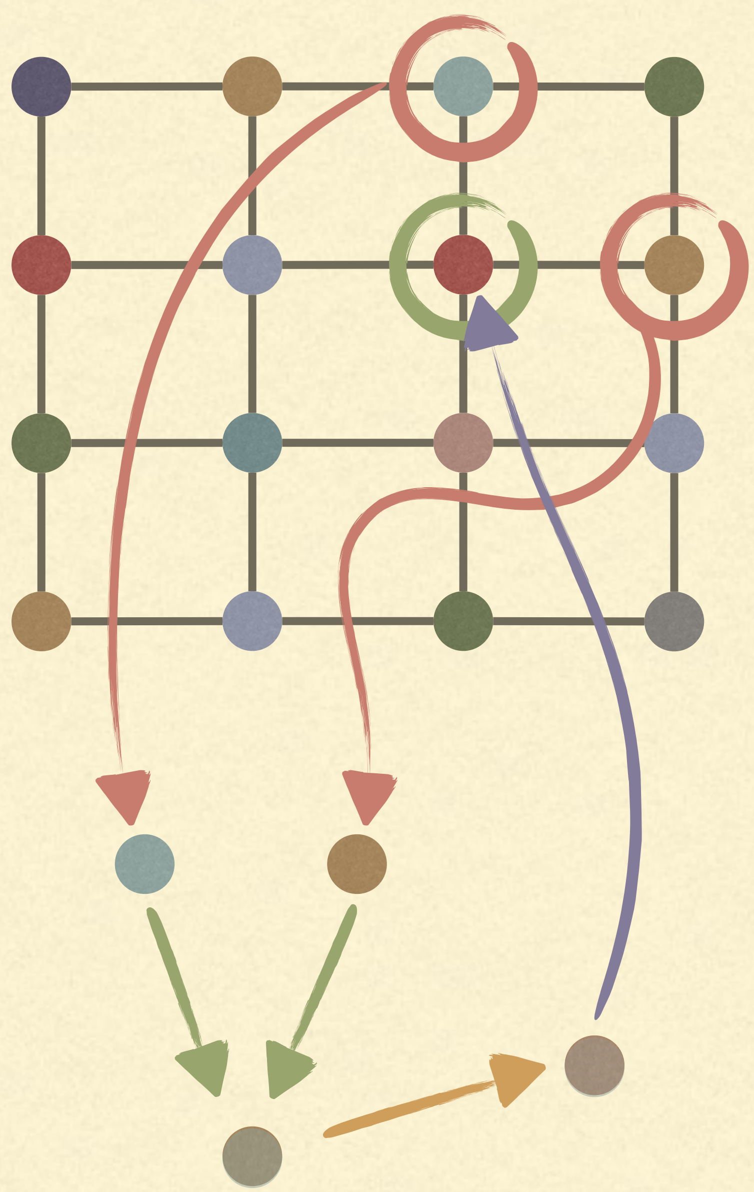
Single individual

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

E.g.:



SPATIALLY-EMBEDDED EA



- Select two parents from the neighbourhood
- Perform the crossover between them
- Mutate the resulting individual
- Replace the individual in this node with the new one