COEVOLUTION NEUROEVOLUTION

Luca Manzoni

COEVOLUTION

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

- Usually we are interested in maximising/minimising a certain fitness value…
- …but now the fitness is influenced by the other individuals
- **Absolute fitness**: the fitness that we actually want to optimise
- **Relative fitness**: the fitness that the algorithm is optimising, hopefully also improving the absolute fitness

ONE-POPULATION COMPETITIVE COEVOLUTION

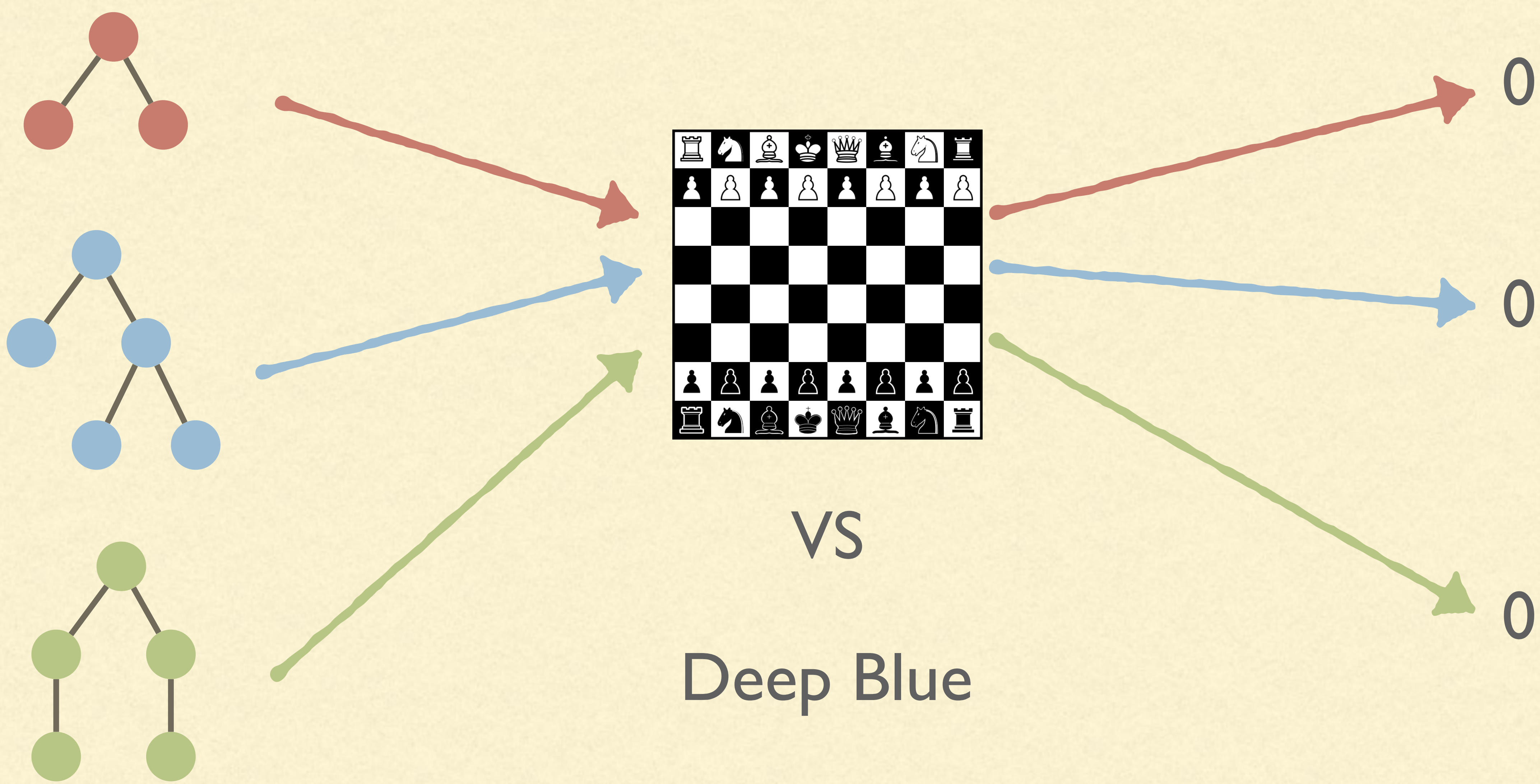
- Usually employed when producing individuals that play games
- Works when evolving agents where the quality of the solution can be assessed by making them play one against the other
- Here evaluating the "real fitness" might be difficult

WHY?

Fitness evaluation with a “competent adversary”

Population at generation 0

Fitness
Number of victories

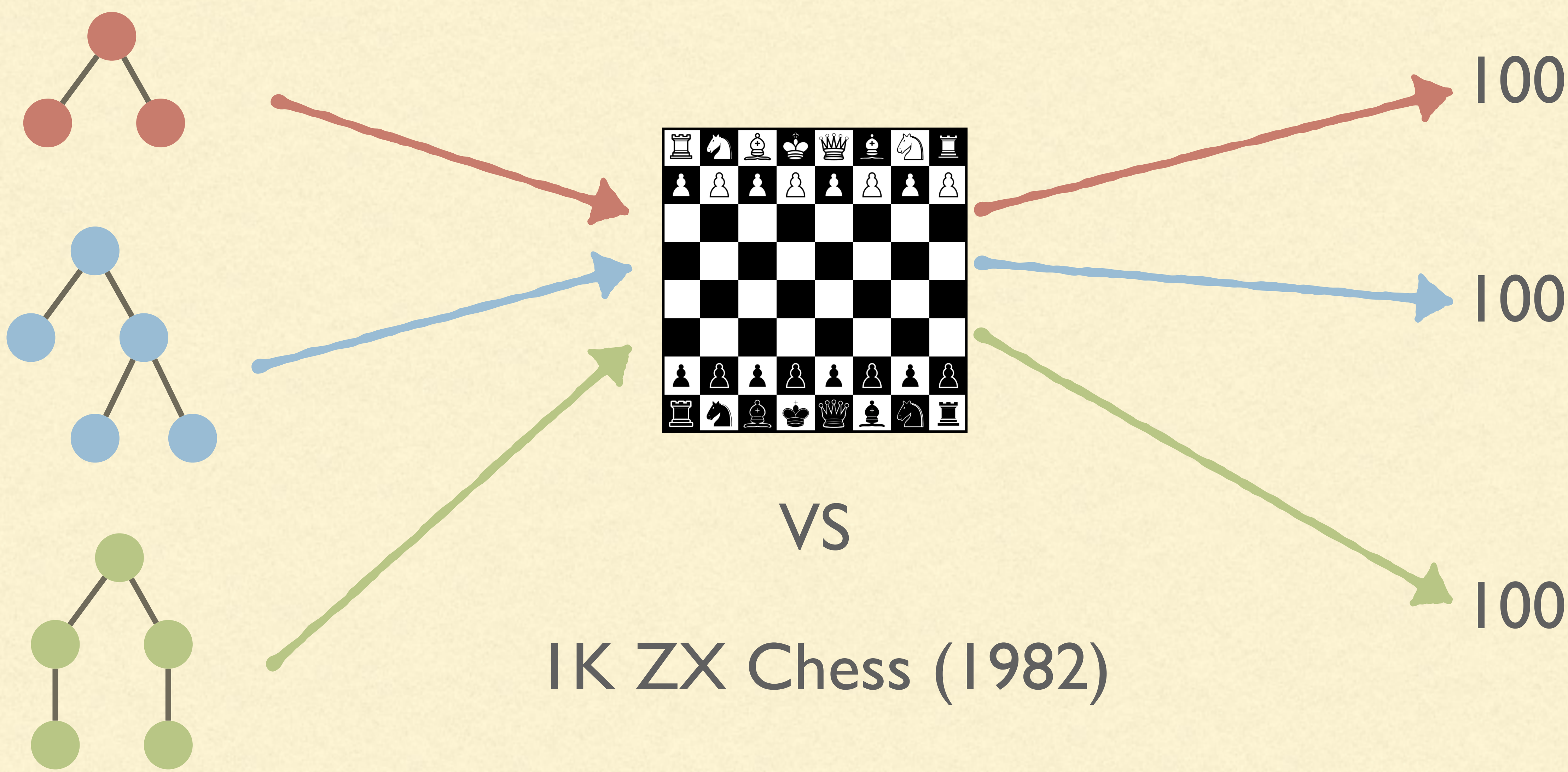


WHY?

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

Population at generation 100

Fitness
Number of victories



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…

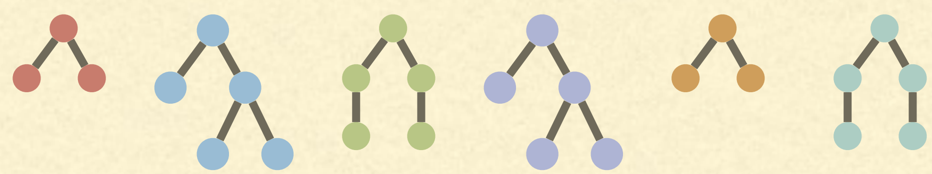
Internal fitness

External fitness

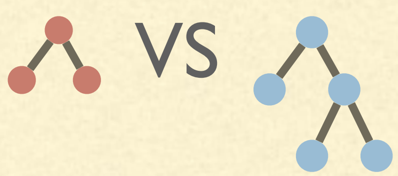
…but not agains real opponents

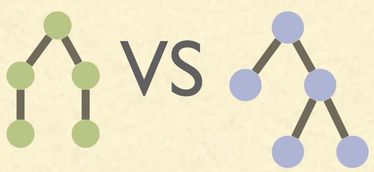
Progress should be evaluated with both fitnesses

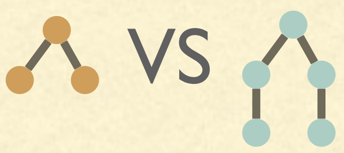
EVALUATION: PAIRWISE



Only make individual n compete with individual n+1



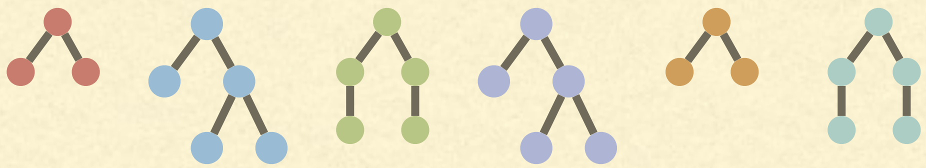




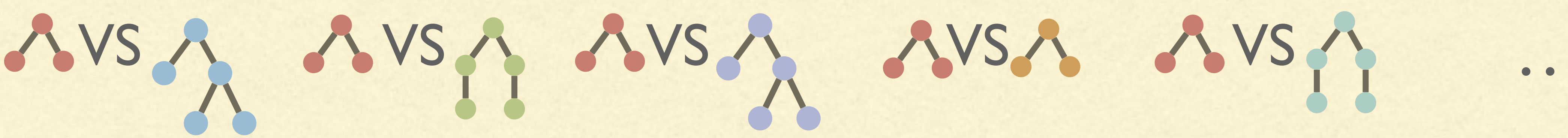
+only O(n) tests

-very dependant on the adversary

EVALUATION: COMPLETE



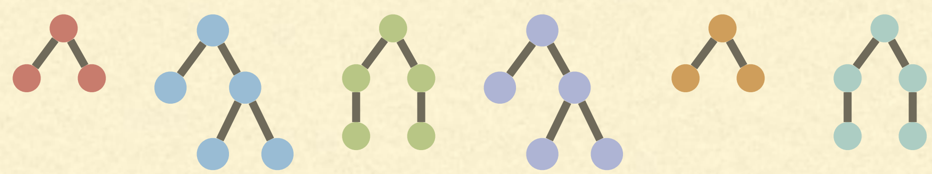
Every individual competes with every other individual



+precise assessment of the fitness

-with $O(n^2)$ tests

EVALUATION: K-FOLD

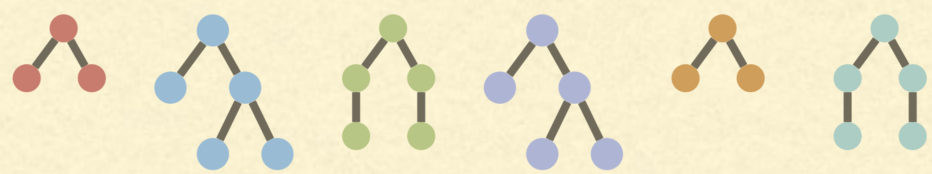


Every individual competes with k randomly selected individual

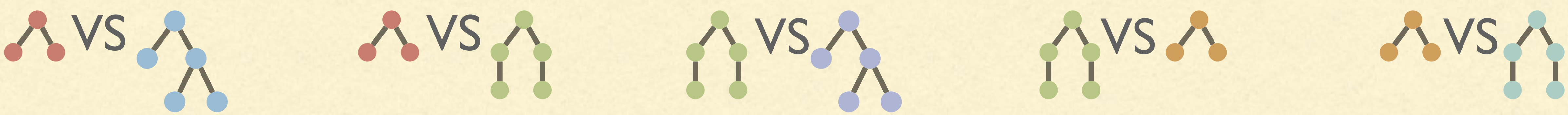
+tunable number of tests

-some individuals might be tested more than others

EVALUATION: SINGLE-ELIMINATION TOURNAMENT



Every individual proceeds with the competitions until it is beaten

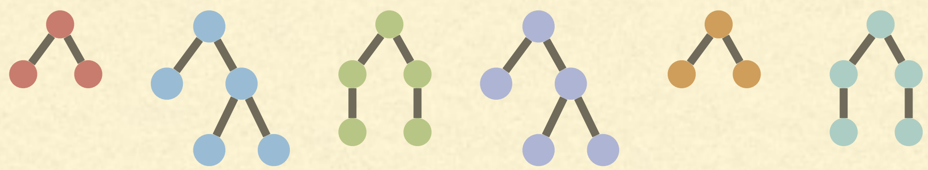


+Only O(n) tests

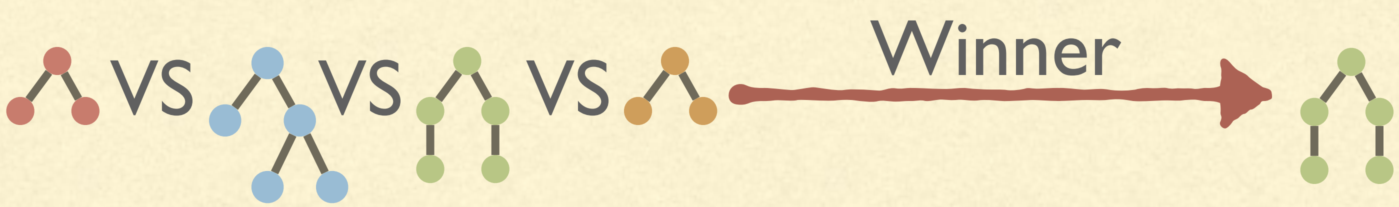
+Better individuals are tested more times

-Bad pairing might penalise some individuals

FITLESSNESS SELECTION



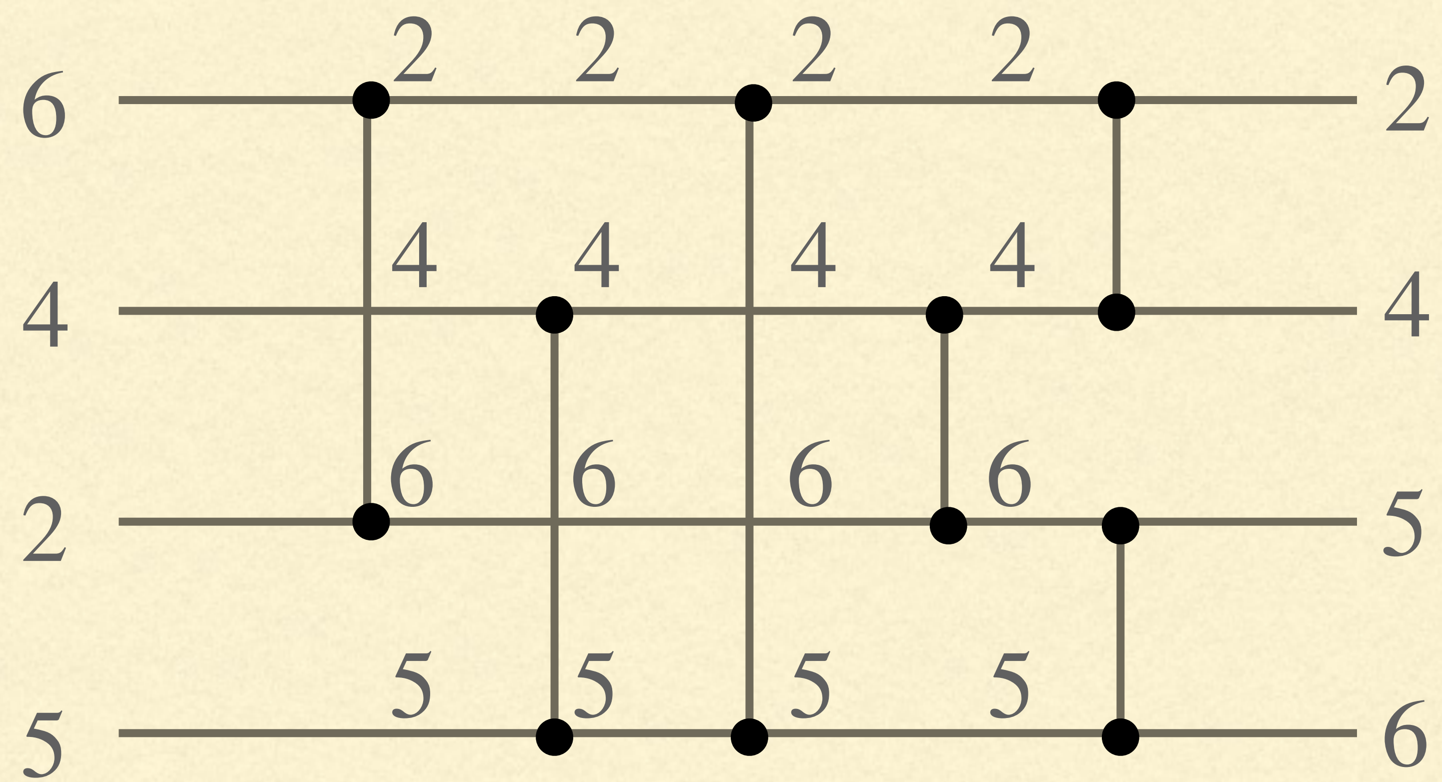
We never compute the fitness:
in tournament selection the selected individual is
the winner of the tournament



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

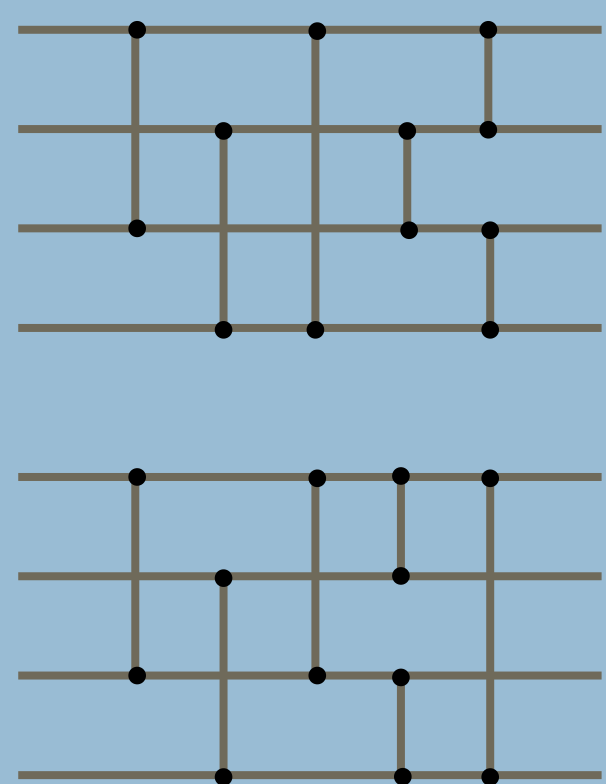
SORTING NETWORKS



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

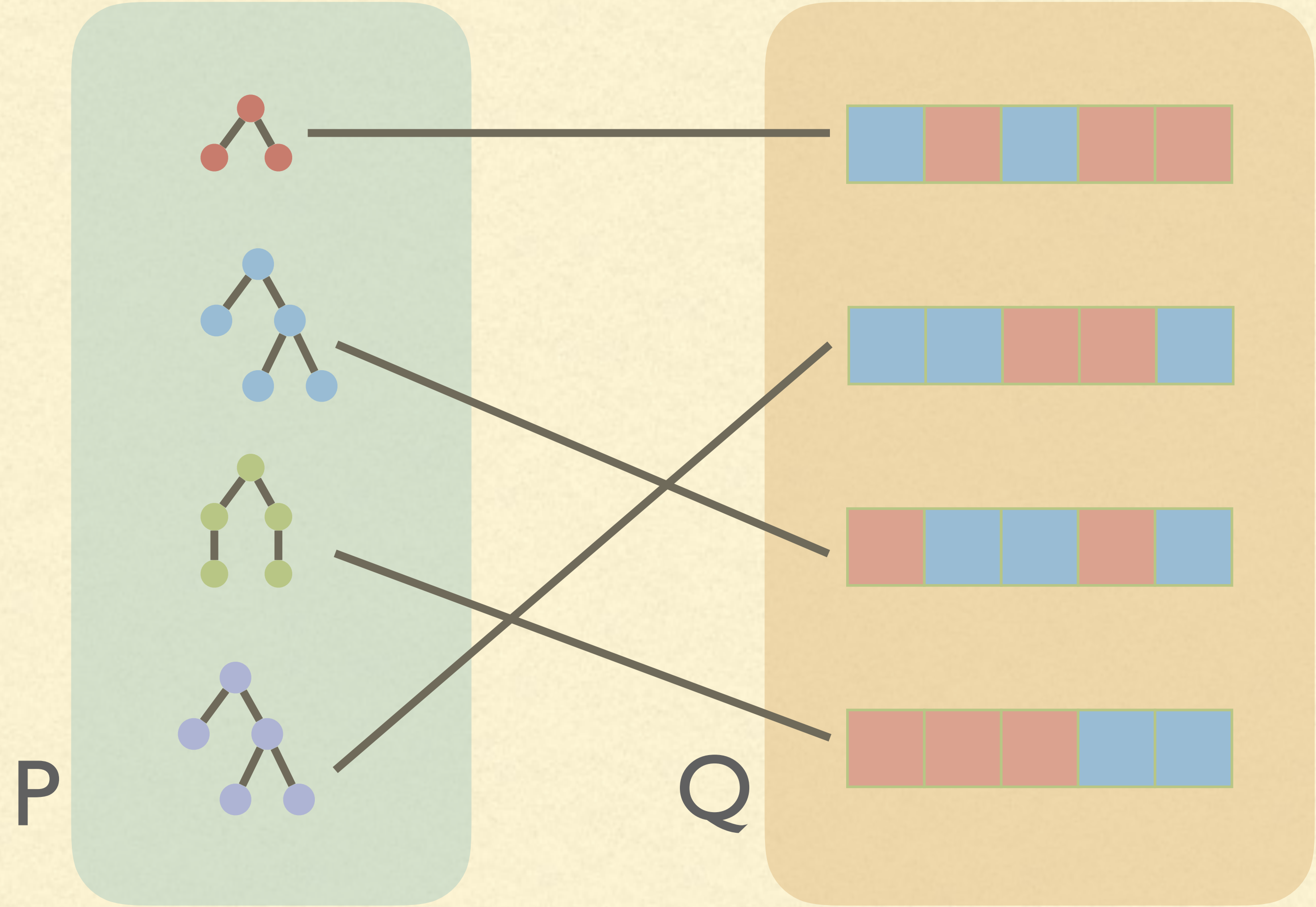
↓

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

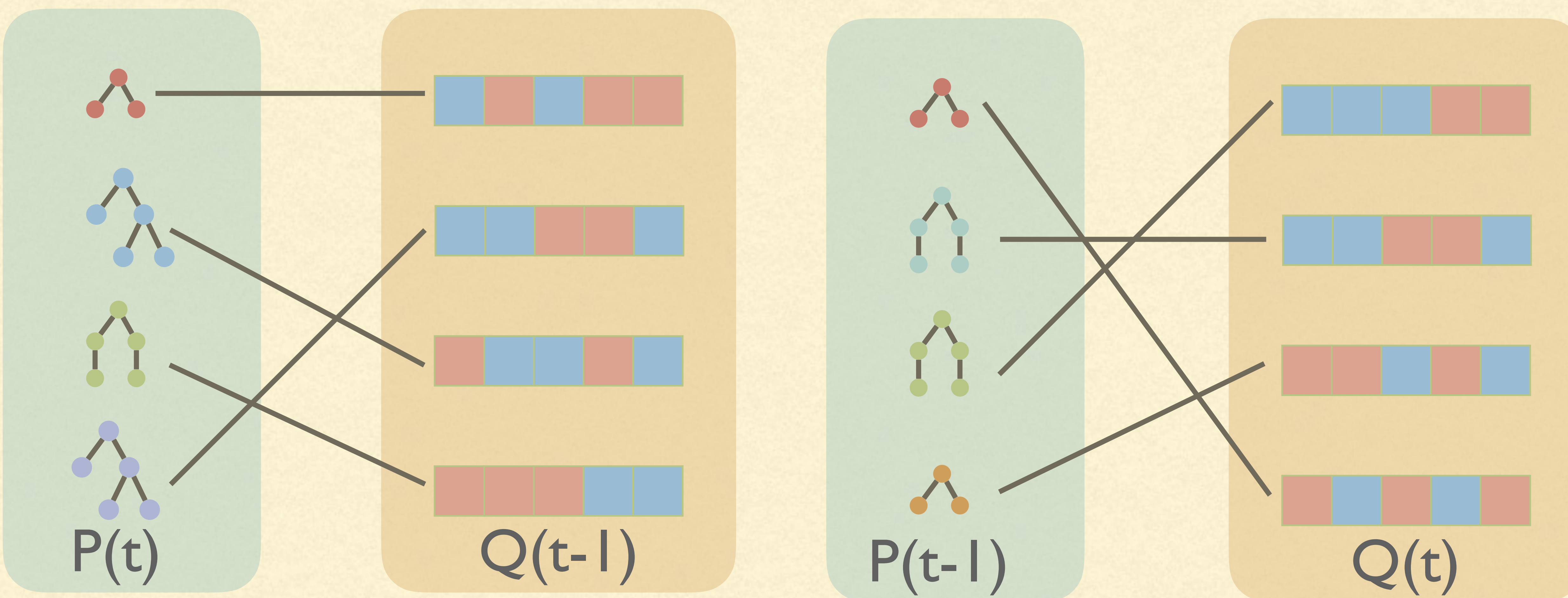
Repeat k times

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



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

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 Optimisation

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

- Overgeneralisation: 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

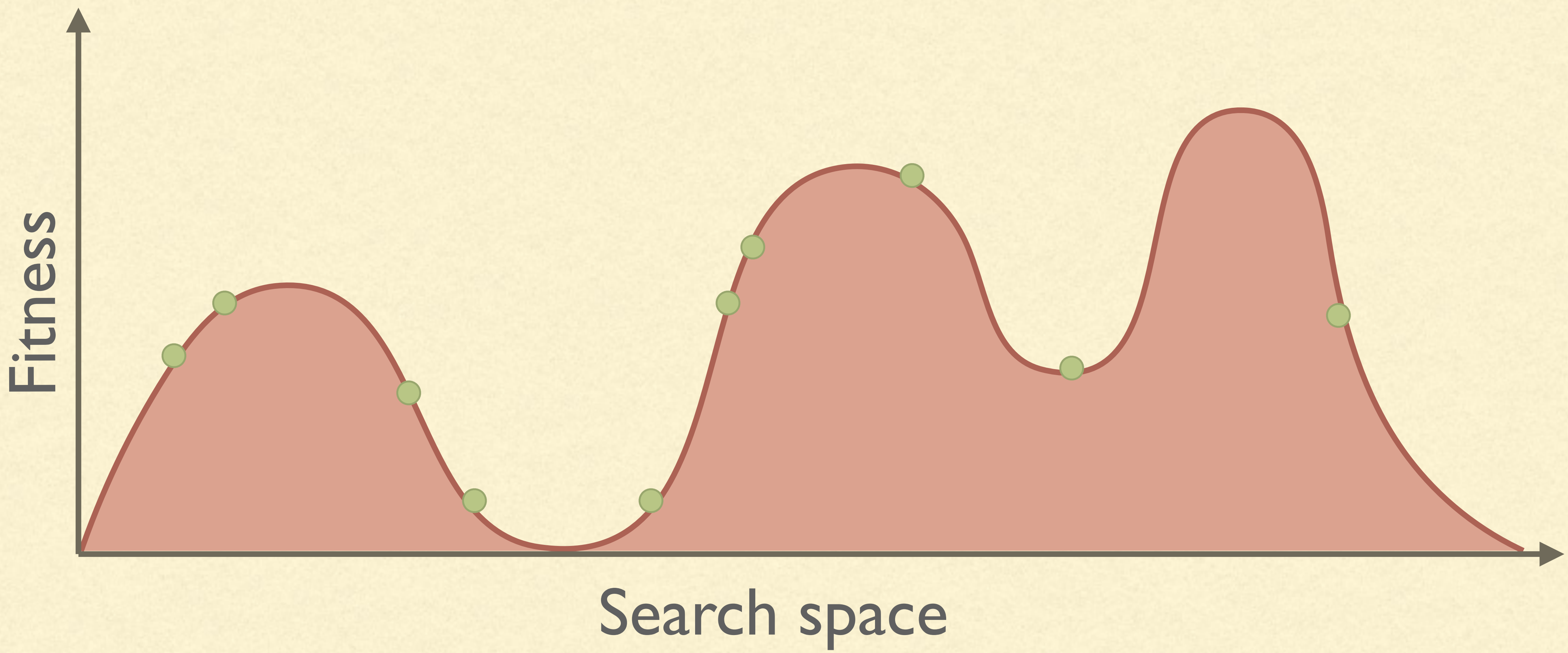
Behaviours for individuals in population 1

Behaviours for individuals in population 2

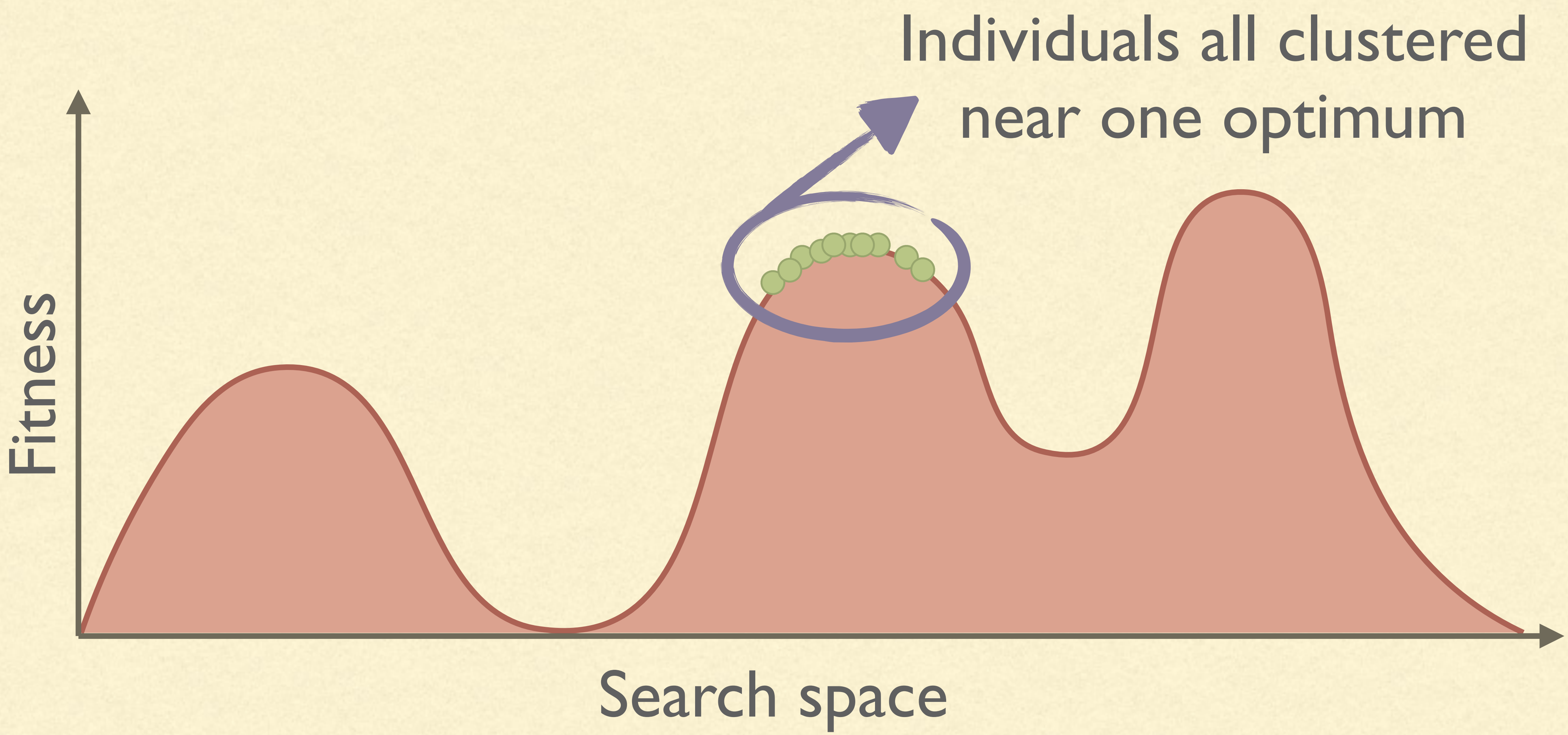
	A	B
A	5	0
B	0	20

To move from the local optimum 5 the two population must change behaviour "at the same time"

DIVERSITY MAINTENANCE

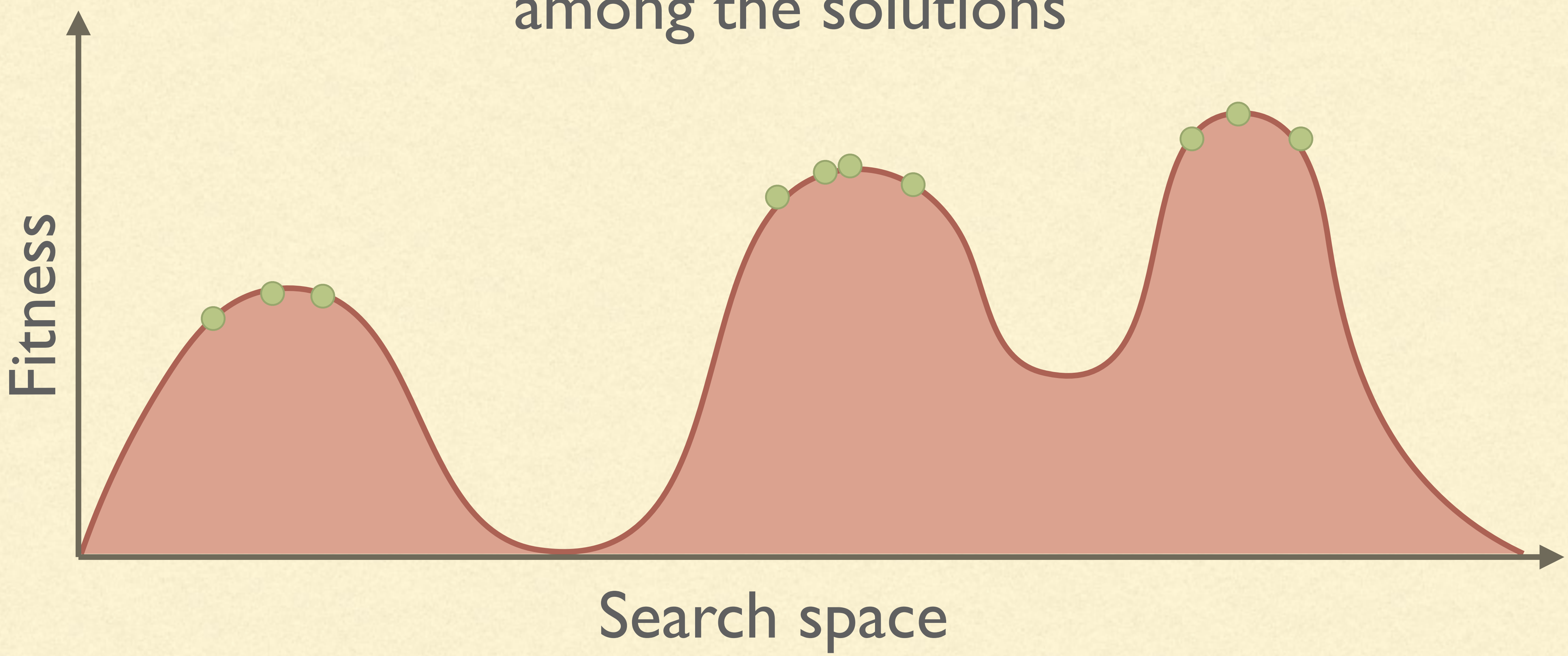


DIVERSITY MAINTENANCE



DIVERSITY MAINTENANCE

But we would like to preserve some diversity among the solutions



DIVERSITY MAINTENANCE

- One of the problems to avoid in optimisation 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 fo diversity in the population.

MEASURING SIMILARITY

- **Genotype**: similar construction (e.g., small Hamming distance for sequences of bits)
- **Phenotype**: similar behaviour (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

Degree of punishment

$\alpha > 0$

Threshold

$$s(x,y) = \begin{cases} 1 - \left(\frac{d(x,y)}{\sigma}\right)^{\alpha} & \text{if } d(x,y) \le \sigma \\ 0 & \text{otherwise} \end{cases}$$

Punishment of x for being too similar to y

FITNESS SHARING

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

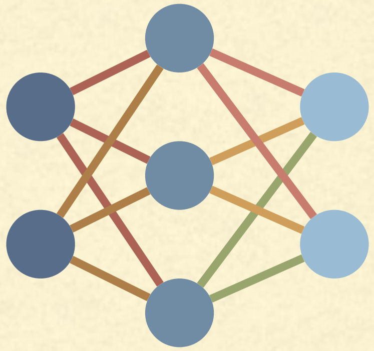
Raw fitness

Scaling factor
$\beta > 1$

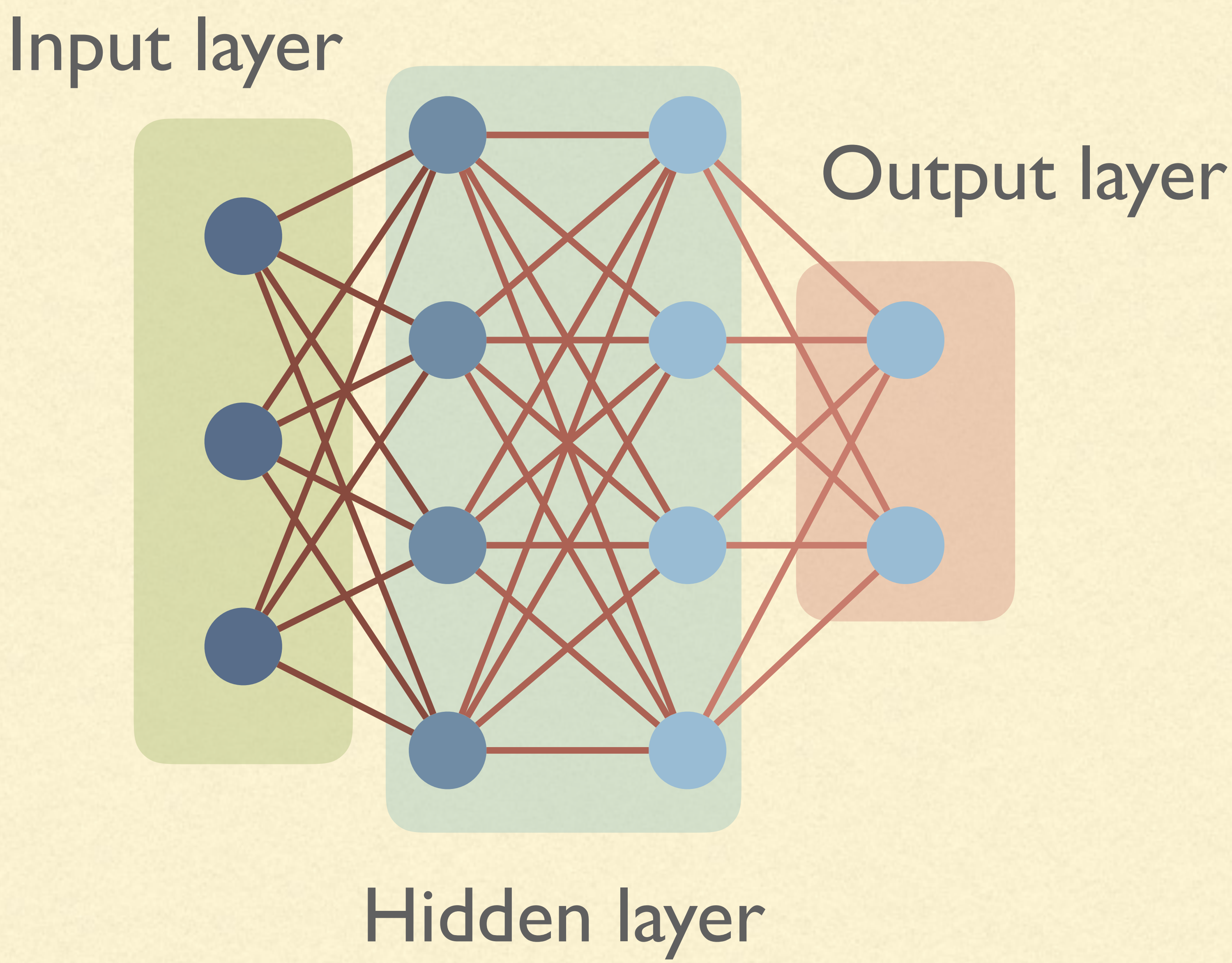
$$f(x) = \frac{r(x)^{\beta}}{\sum_y s(x, y)}$$

A total of 3 parameters: α, β, σ

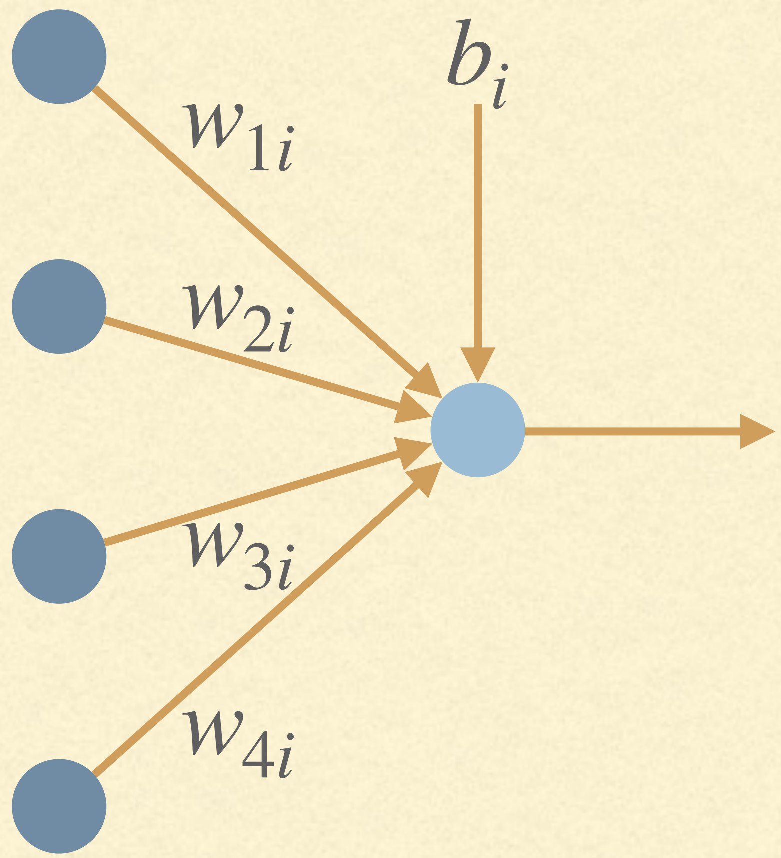
NEUROEVOLUTION



A QUICK AND NOT-TOO-INACCURATE RECAP ON NEURAL NETWORKS



A QUICK AND NOT-TOO-INACCURATE RECAP ON NEURAL NETWORKS



$$f\Big(b_i + \sum_{j:j\to i} w_{ji} x_j\Big)$$

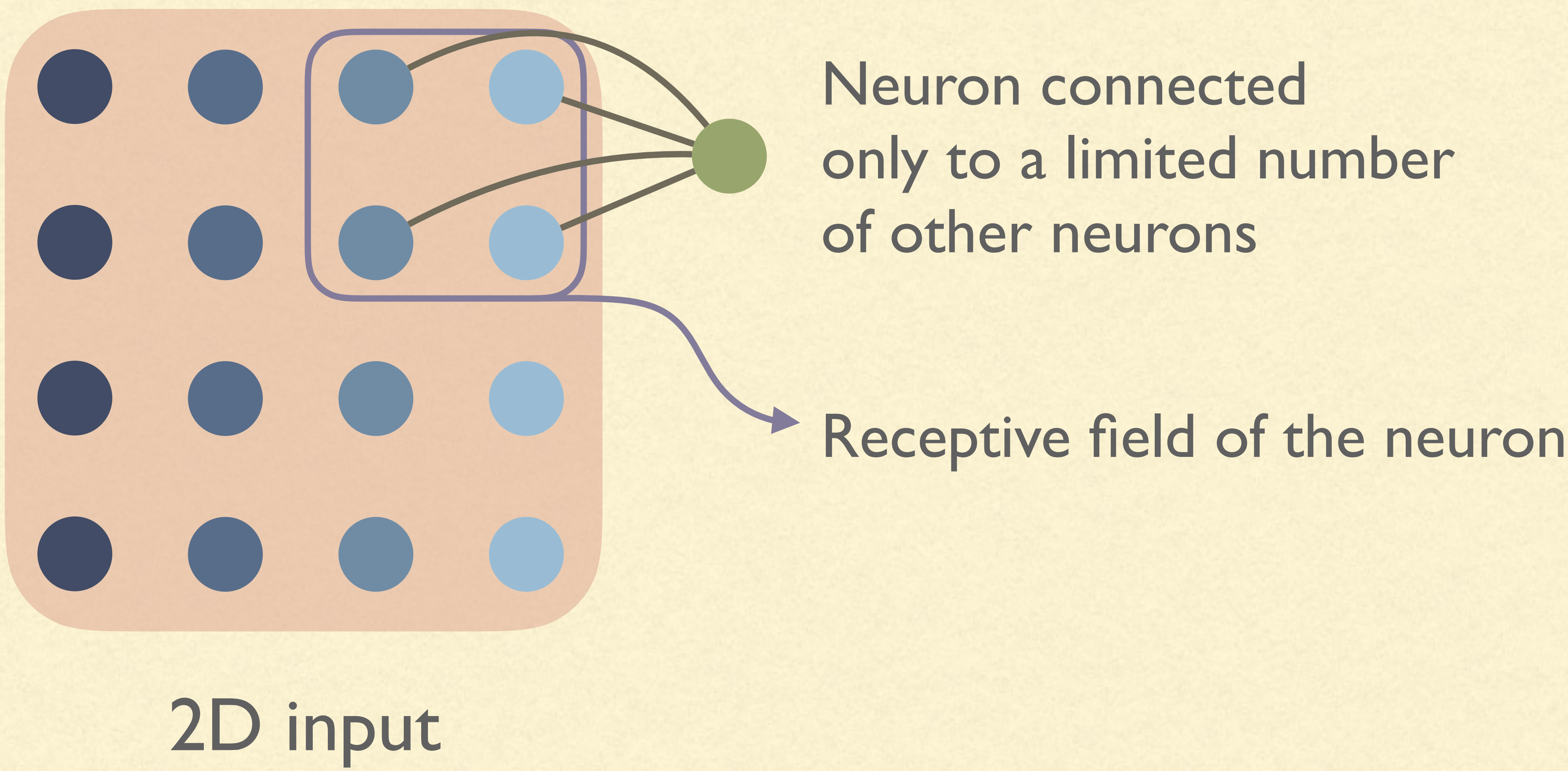
Activation function

Bias

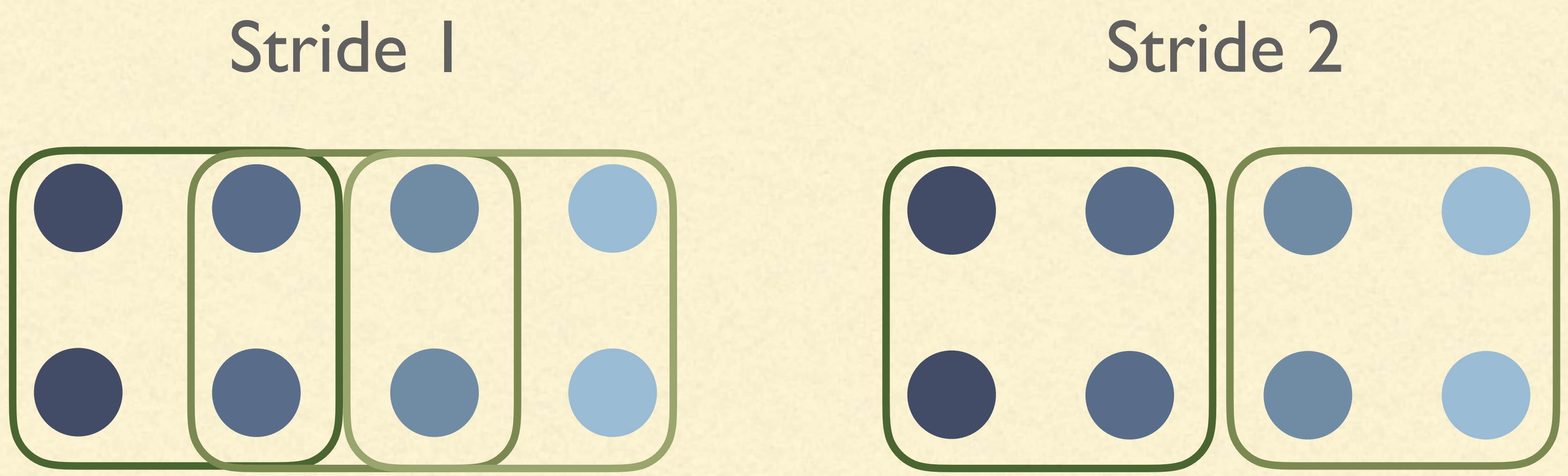
Weighted sum of the inputs

A QUICK AND NOT-TOO-INACCURATE RECAP ON NEURAL NETWORKS

Convolution layer



A QUICK AND NOT-TOO-INACCURATE RECAP ON NEURAL NETWORKS



All neurons in the convolution layer
share the same weights

This is in contrast with fully connected layers, where
1) all connections are present
2) weight can be different across neurons

NEURAL NETWORK TRAINING

- Training is usually performed via (stochastic) gradient descend
- Main idea: find in which direction the weights can be modified to reduce the error by differentiating the error w.r.t. the weights
- However, only the weights are learned
- The architecture of the network (i.e., number, size, and type of hidden layers) and the hyperparameters are selected manually

WHAT TO EVOLVE

- Weights (but this can also be done with backpropagation)
- Activation functions
- Hyperparameters (e.g., momentum, learning rate, dropout)
- Architecture (e.g., connections, types of layer)

HISTORY
(CLASSIC NEUROEVOLUTION)

- History: anything before the advent of large/deep networks
- Evolution of weights and topology
- The "unit" was the single neuron/single weight
- Currently unfeasible due to the large number of neurons/weights/layers

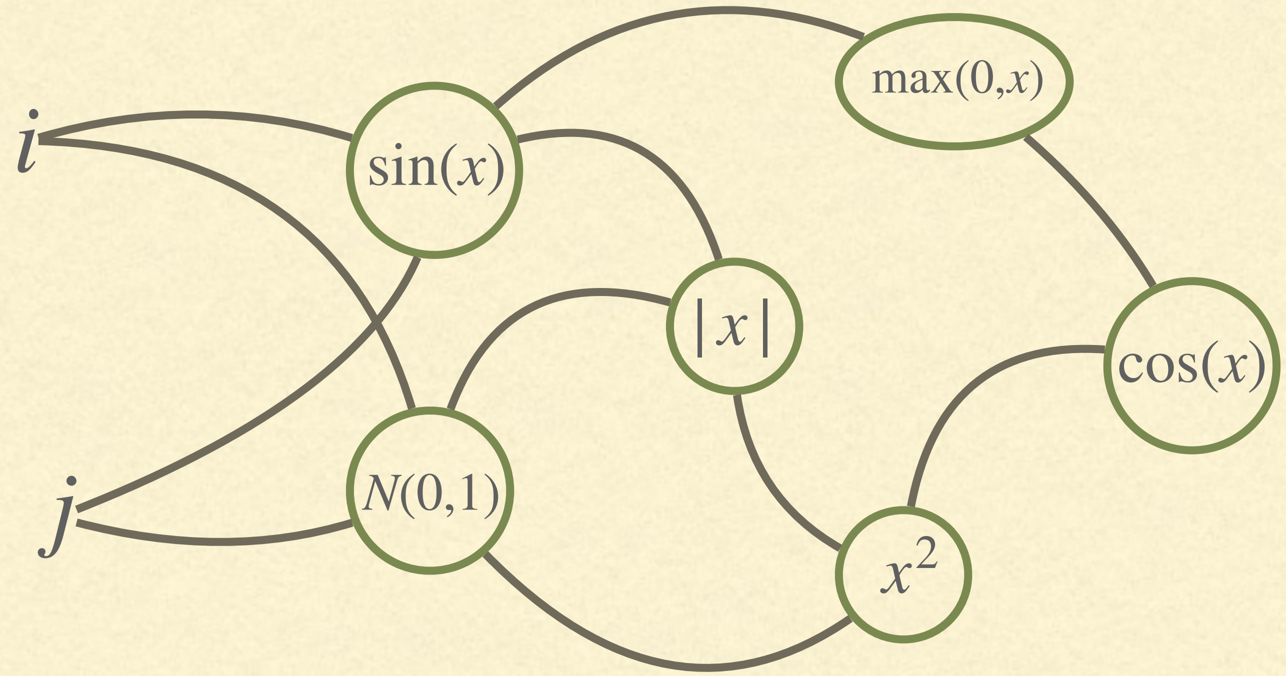
NEAT
NEUROEVOLUTION OF AUGMENTING TOPOLOGIES

- Think "direct encoding of a graph structure" where both the structure of the network and the weight are evolved at the same time
- The initial population start "simple" (no hidden nodes). Evolution might add new nodes
- Each gene has a "birth time" that tracks when it was introduced. This feature is used to allow crossover
- Features are protected with speciation. That is, at each generation only individuals in the same specie can mate

INDIRECT ENCODINGS

- One of the advantages of indirect encoding is a compact representations
- But also the ability to express regularities, as the one present in convolution layers
- We will see two examples without many details: HyperNEAT and DENSER

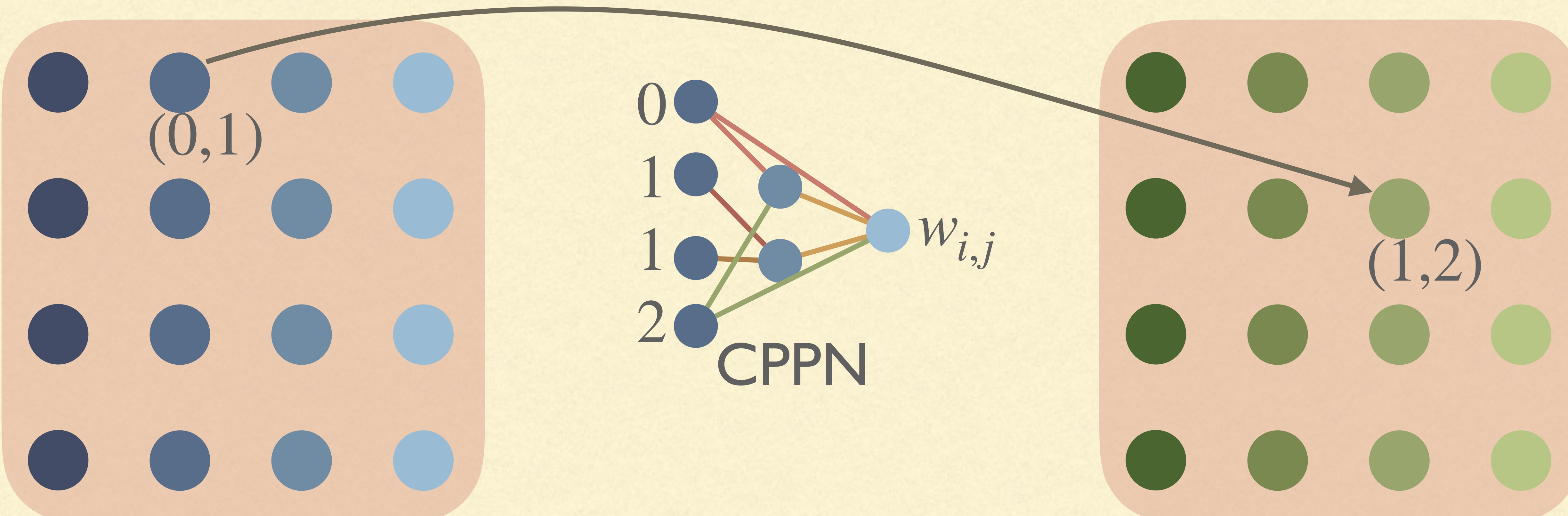
CPPN
COMPOSITIONAL PATTERN-PRODUCING NETWORKS



A “network” of functions that maps a position (i,j) in a 2D plane (more dimensions possible) to a value

This can be evolved as a “classical”network.
So, where's the indirect encoding?

HYPERNEAT



Network layer
(2D disposition of neurons)

Network layer
(2D disposition of neurons)

DENSER

DEEP EVOLUTIONARY NETWORK STRUCTURED REPRESENTATION

- A layer-based approach:
  the number, type, and parameters of the layers are evolved
- The weights are obtained via backpropagation
- A two-levels approach:
  - the sequence of layer and a “general” type is encoded by a GA
  - The parameters of each layer are generated by grammatical evolution (GE), in particular dynamic structured GE
- Only the best-performing networks are trained for more than a few epochs