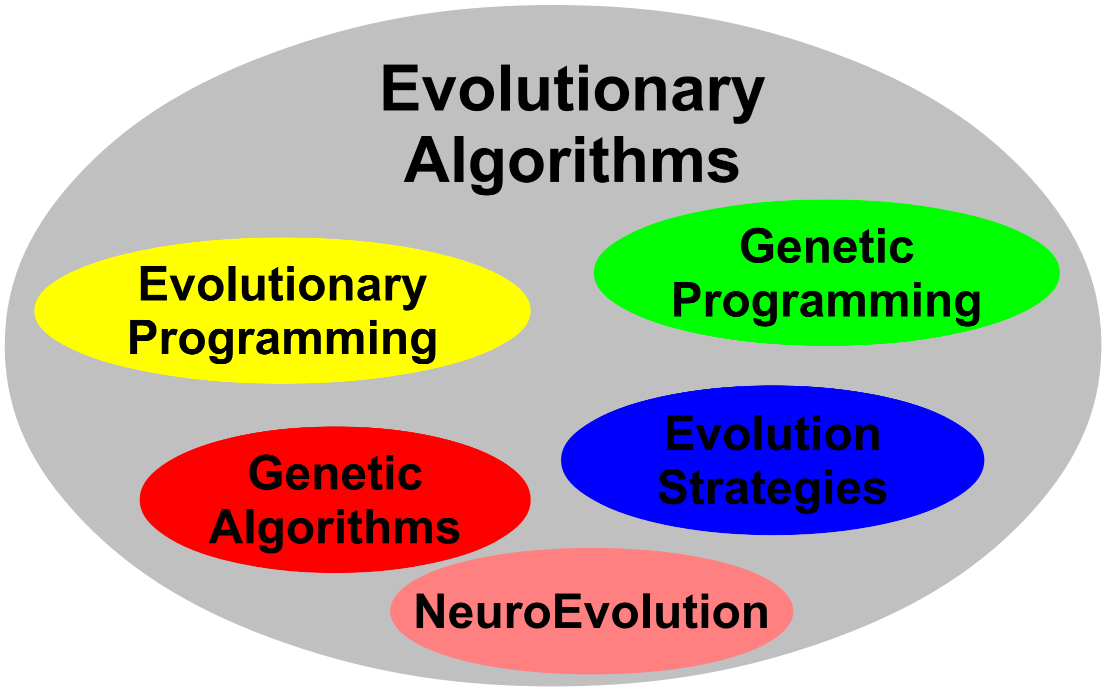
Motivation

- Learning ANNs = optimization of weights or potentially architecture.
- Problem of local extremes → unable to learn hard task/large network.
- Use of **Evolutionary Algorithms** → slower, but more robust than classic gradient methods like Back-Propagation.

Evolutionary Algorithms (EAs)



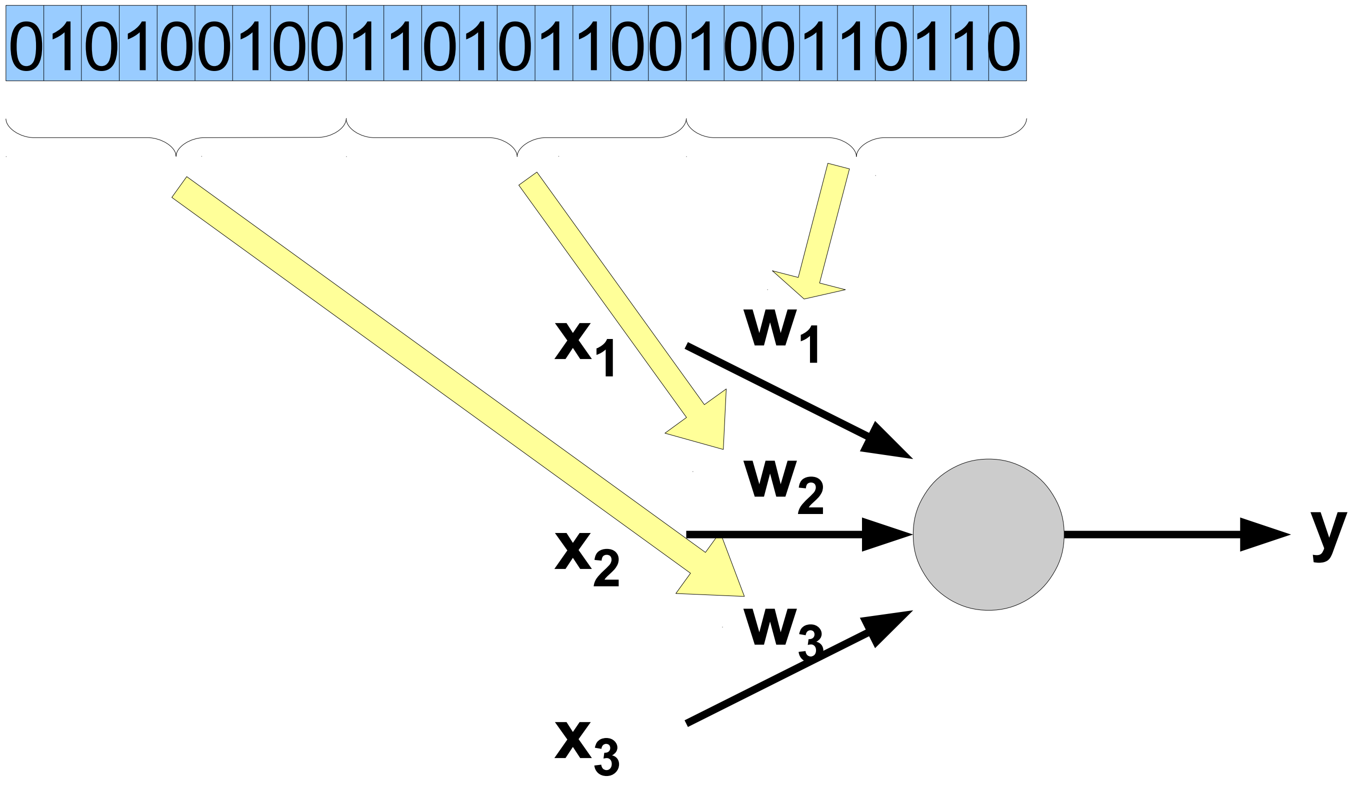
- **Genetic Algorithms:** binary strings
- **Evolutionary Strategies:** real vectors, only mutation.
- **Genetic Programming:** evolution of program trees.
- **Evolutionary Programming:** evolving FSMs.
- **NeuroEvolution**

What is Neuro Evolution?

- **Neuro-evolutionary algorithm is just another special kind of EA** → the task is to evolve (learned) neural networks.
- Both parameters (weights) and topology can be optimized by evolution.
- **But how to encode a network into a genome?** → A network with fixed topology is described by a vector/matrix of all its weights (real numbers)...

Direct Encoding of Neural Network

- Directly encode the weights as a bit string:



- Can we do it better? Yes.

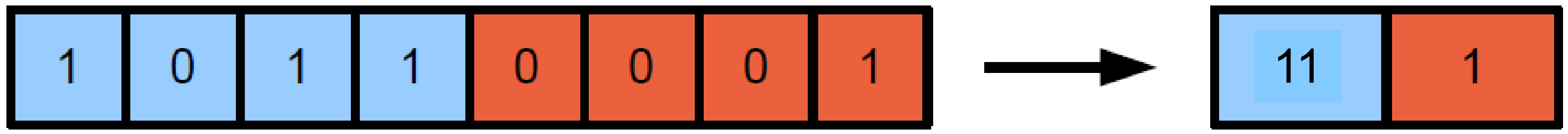
Artificial Neural Networks
NeuroEvolution = ANN + EA

Jan Drchal
drchajan@fel.cvut.cz

Computational Intelligence Group
Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague

Floating-Point Encoding

- Motivation: simplicity, precision

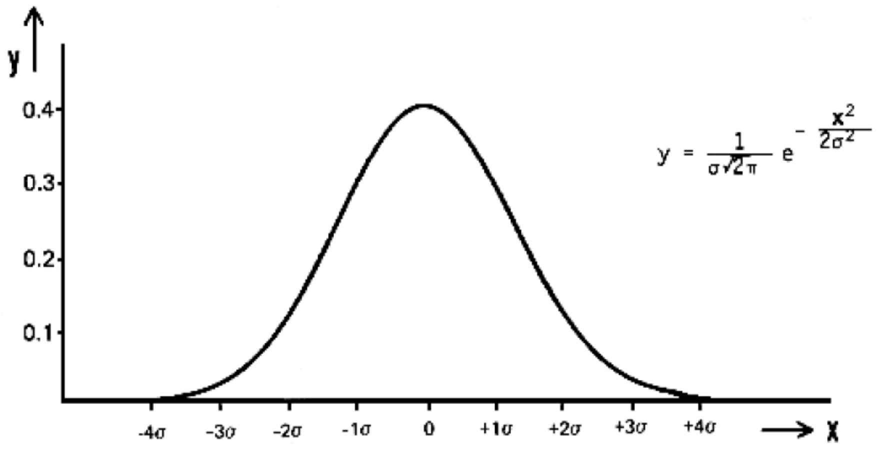


Binary string encodes vector of 2 numbers .

The same encoded using floating-point encoding.

What about mutation? -> Gaussian noise.

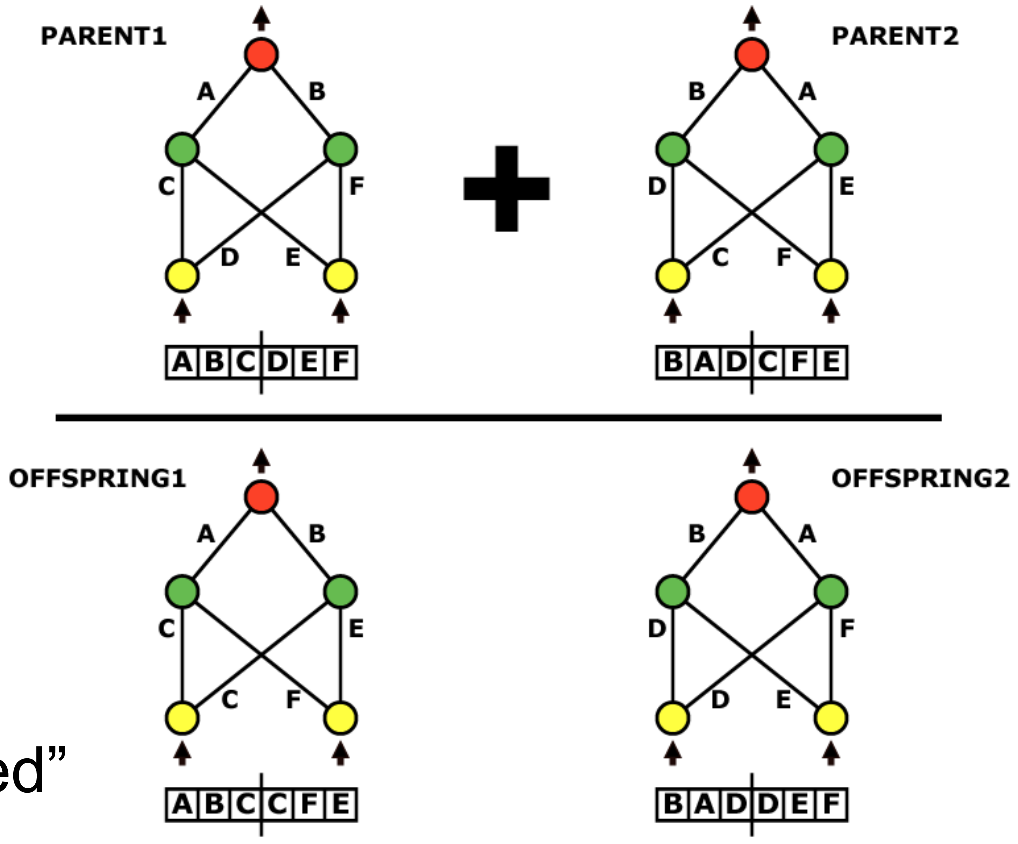
Idea:
small changes with *higher* probability,
large changes with *lower*.



- Useful for integers and floats ...

Competing Conventions Problem

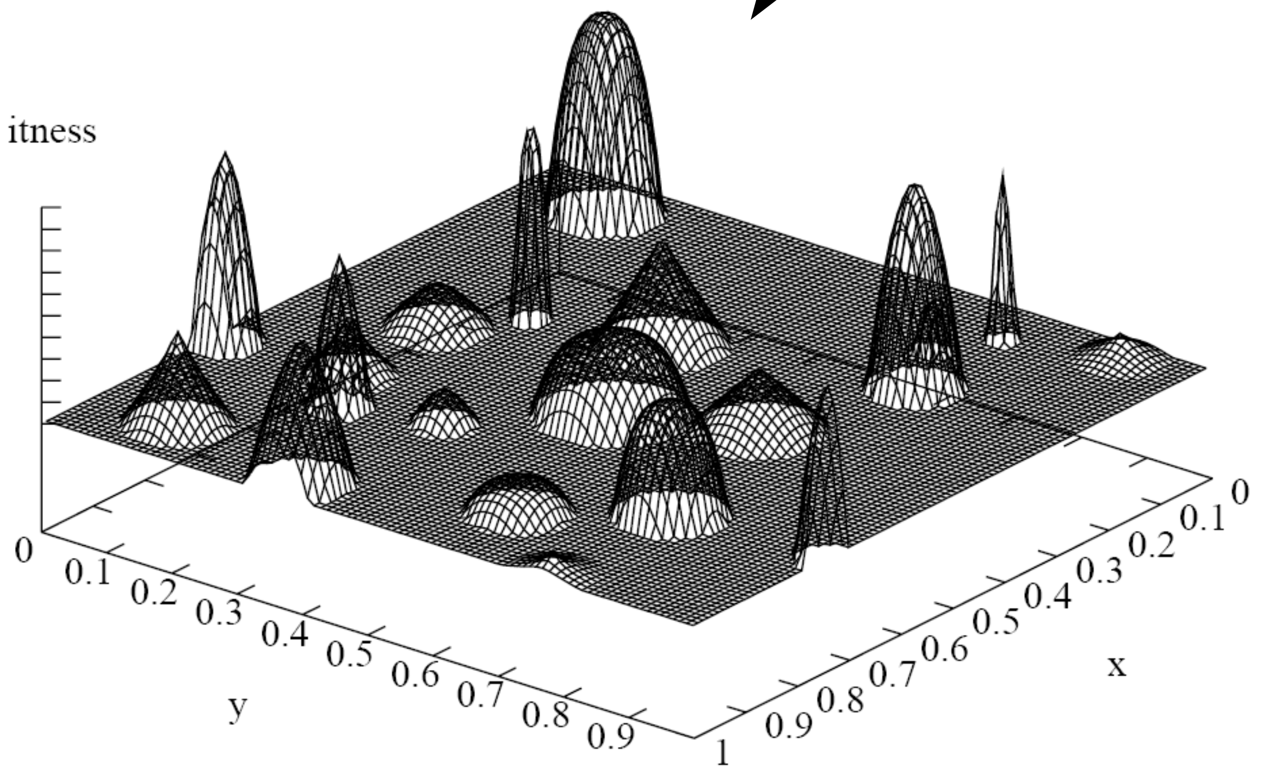
- Problem of competing conventions:
  - Same solution can be represented by many genomes → ordering of weights matters.
  - Error (fitness) landscape contains many optima representing the same solution.
  - Crossover of two such individuals will most probably lead to “crippled” offspring.



Multimodal Domains

- Multimodal functions:
  - multiple optima,
  - many local.
- Too many attractors → hard optimization :(
- ANN fitness/error landscapes look like this.

This is our fitness landscape



Reinforcement Learning & EAs

- *Supervised learning* (Back-Propagation).
- *Unsupervised learning* (SOM).
- **Reinforcement learning** – typical for control tasks.
- Unlike in supervised learning, we don't know the desired output signal, we have only *signal* determining the state of a system.
- **EA is an ideal tool for reinforcement learning.**

Why to Evolve the Topology?

- Motivation:
  - spare experimenters time → finding correct number of layers/neurons,
  - well designed algorithm can find globally optimal topology (smallest but sufficient).
- **TWEANNs: Topology & Weight Evolving Artificial Neural Networks.**

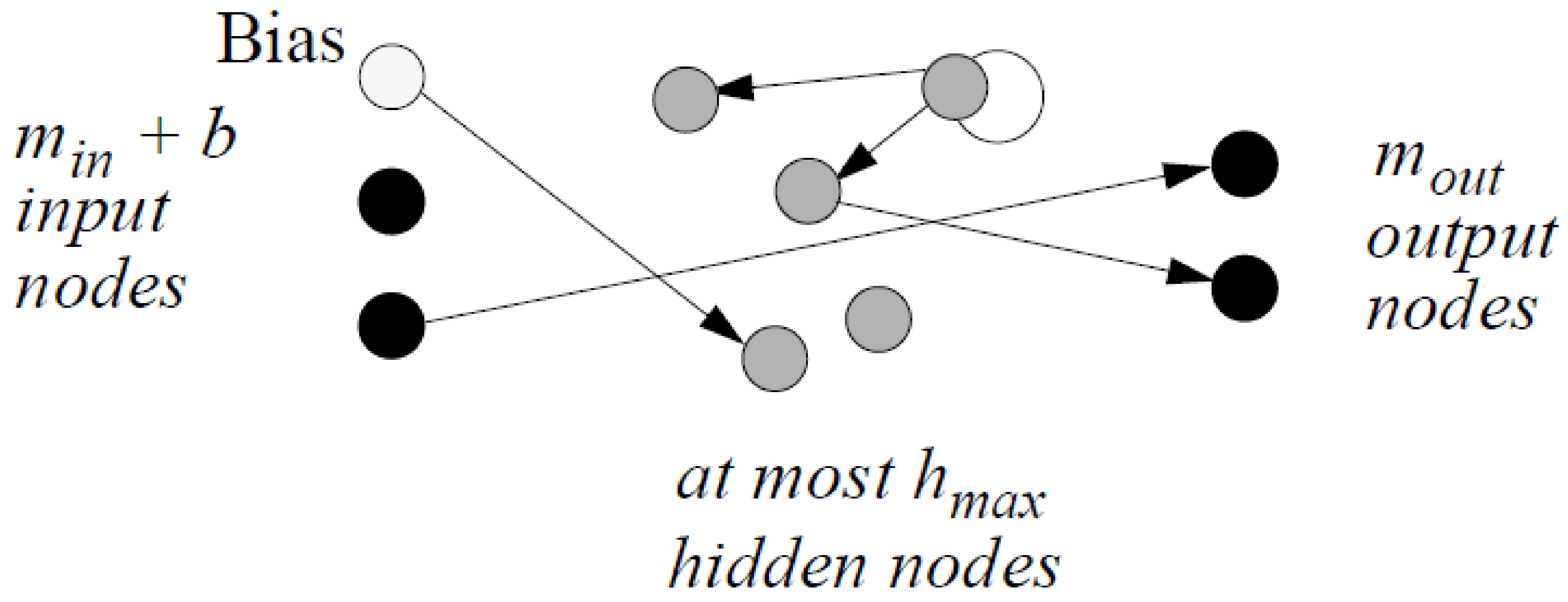
GNARL

- **G**e**N**eralized **A**quisition of **R**ecurrent **L**inks.
- 1994: Angeline, Saunders, Pollack.
- Evolution of general recurrent networks.
- Based on evolutionary strategies → no crossover operator→ no competing conventions.
- Starts with population of random networks.
- **Two kinds of mutation operators:**
  - **Parametric** – weight mutations (Gaussian noise),
  - **Structural** – add/remove neurons/links between.

Angeline, P.J. Saunders, G.M. Pollack, J.B. : **An Evolutionary Algorithm That Constructs Recurrent Neural Networks**

GNARL 2

- Sample of GNARL's initial random network:



Note, disconnected neuron does not affect network evaluation, it is available as a resource for future structural mutations.

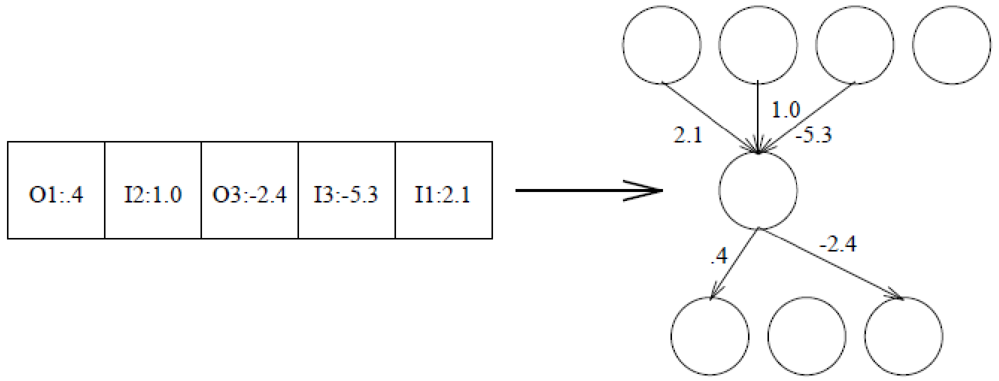
SANE

- **S**ymbiotic, **A**daptive **N**euro-**E**volution.
- 1998: Moriarty, Miikkulainen.
- Based on **coevolution**:
  - simultaneous evolution of multiple populations, mutually influencing each other:
  - **neurons –** weights of links incoming to neuron,
  - **blueprints** - „plans“ of connecting neurons to whole networks.
- How to compute fitness:
  - **neuron** – fitness of 5 best networks, which it appeared in
  - **blueprint** – fitness of the describing network.
- Evolution at the level of neurons – no problem with competing conventions

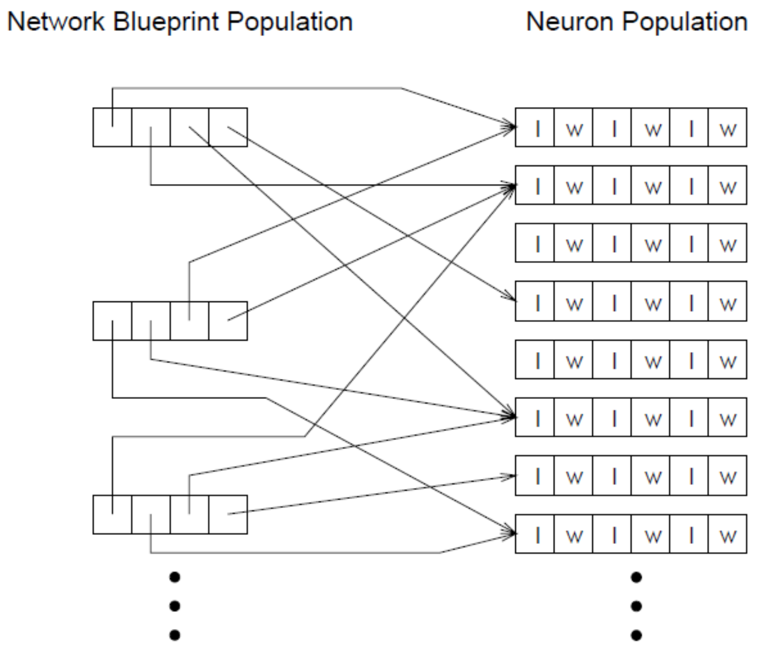
David E. Moriarty, Risto Miikkulainen: **Forming Neural Networks Through Efficient and Adaptive Coevolution**

SANE 2

This way the neurons are encoded. SANE works with networks having single hidden layer and fixed input/output layers.



Blueprints contain list of pointers to neurons.



SANE 3

- Example of an encoded network:

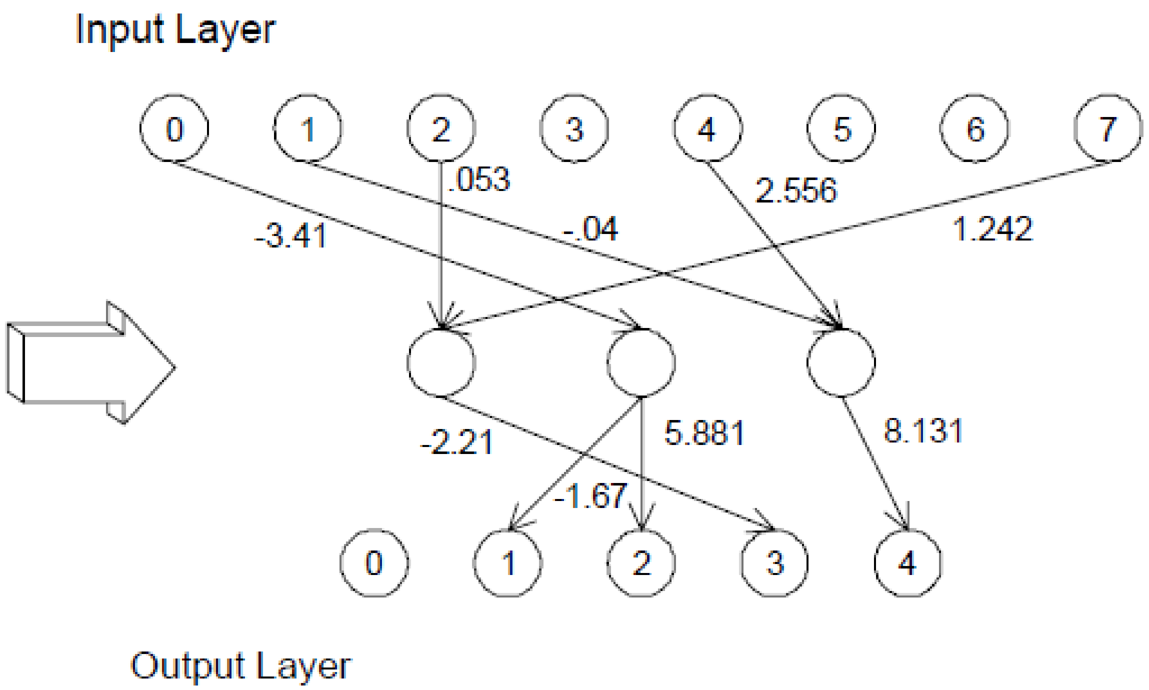
connect to input if ≤ 127

connect to output if > 127

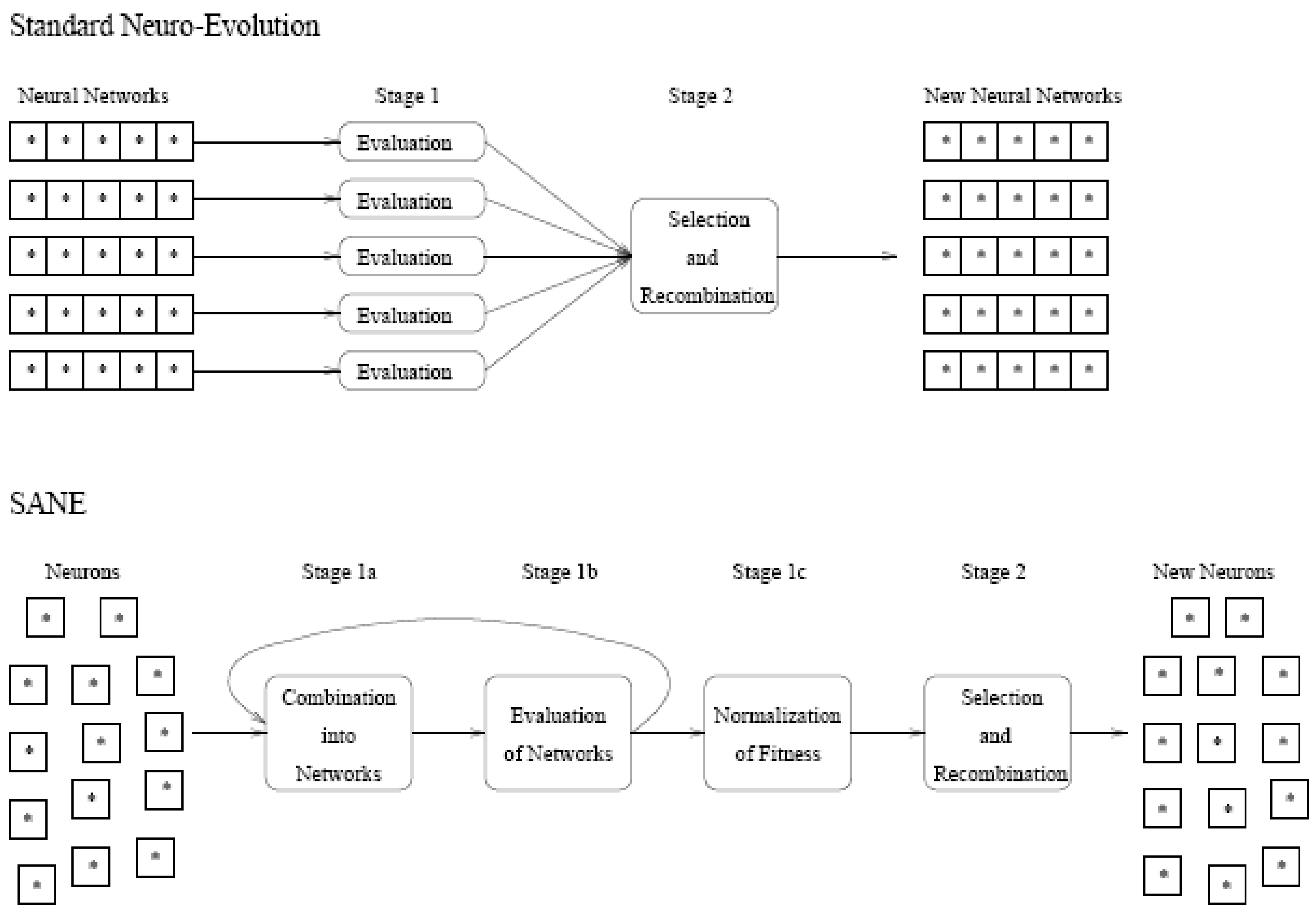
label	weight				
15	1.242	143	-2.21	2	.053
212	5.811	32	-3.41	151	-1.67
65	-.04	100	2.556	134	8.131

65 mod (# of inputs)

134 mod (# of outputs)

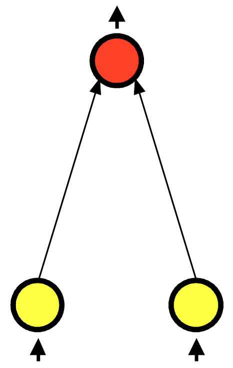


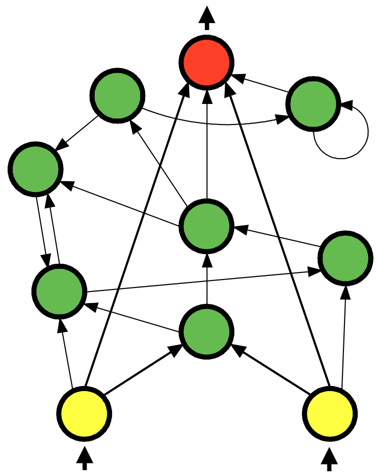
SANE 4


Standard Neuro-Evolution
Neural Networks
Stage 1
Stage 2
New Neural Networks
Evaluation
Selection and Recombination
SANE
Neurons
Stage 1a
Stage 1b
Stage 1c
Stage 2
New Neurons
Combination into Networks
Evaluation of Networks
Normalization of Fitness
Selection and Recombination

NEAT

- **N**euro**E**volution of **A**ugmenting **T**opologies: Kenneth O. Stanley, 2001, The University Of Texas at Austin
- Complexification – start from small topologies: evolution add neurons/links as needed by task.





Kenneth O. Stanley and Risto Miikkulainen: **Evolving Neural Networks Through Augmenting Topologies**

NEAT 2

- Topology is augmented by adding neurons and links between.

  → Variable genome length.

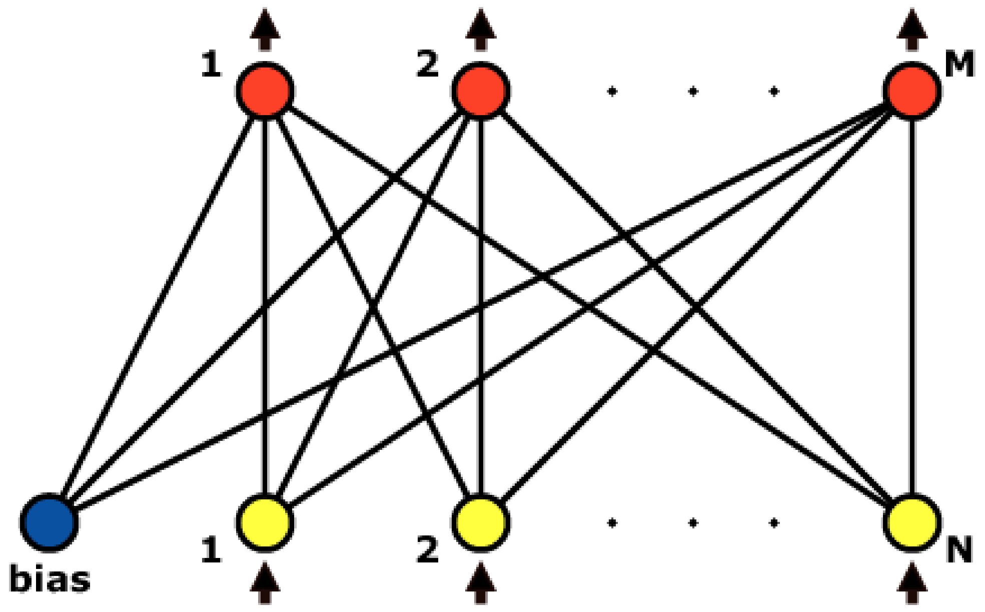
- Mutations:
  - parametric – Gaussian noise,
  - structural – adding neurons & links (no pruning), switch on/off links.

- Mating – special crossover two parents → single child.

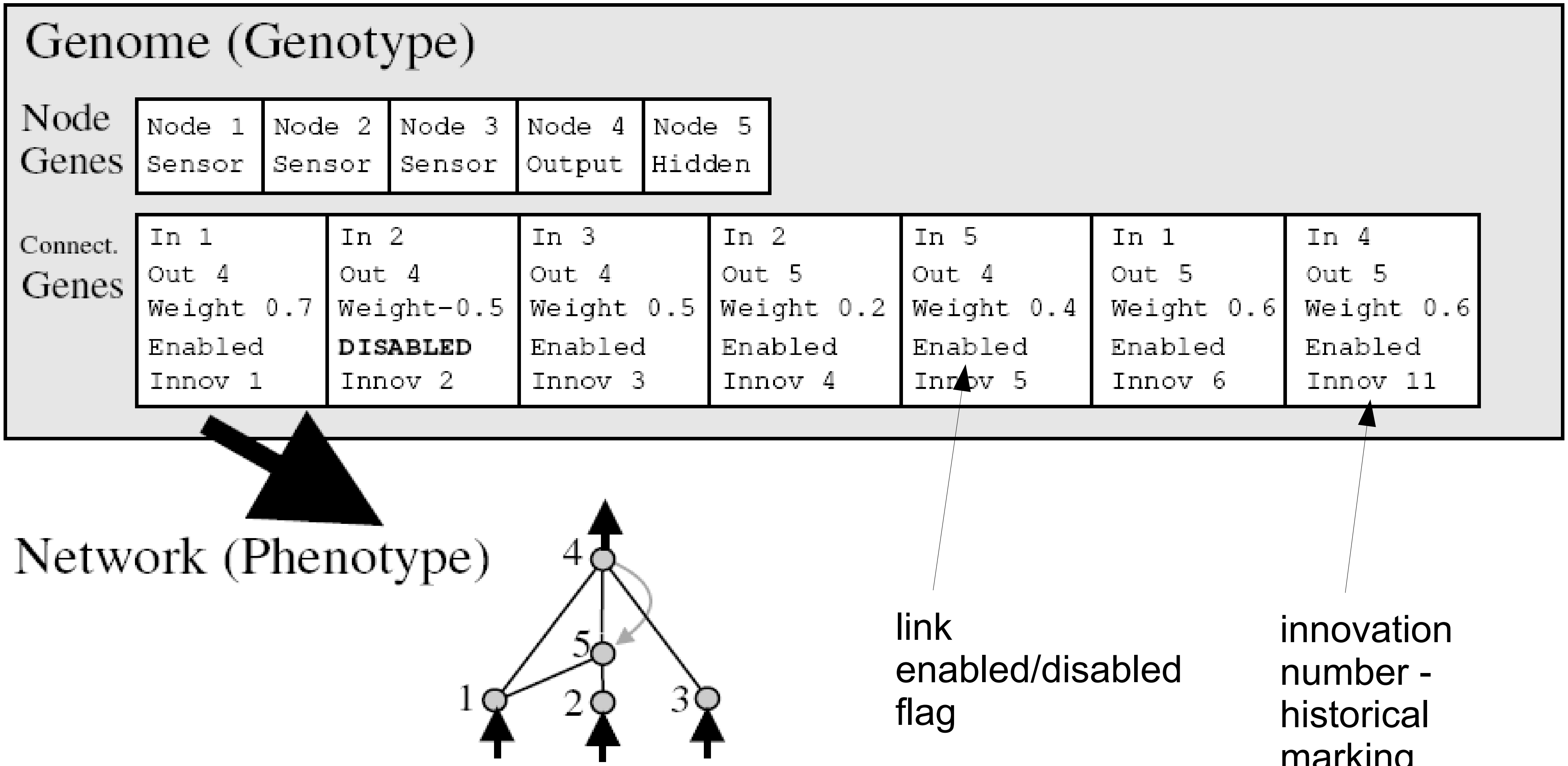
Note, some newer implementations use pruning However, it is not essential.

Minimal Substrate

- Initial population is formed of the simplest topologies: fully connected feed-forward networks without hidden layers: the minimal substrate.



NEAT Genome



Genome (Genotype)

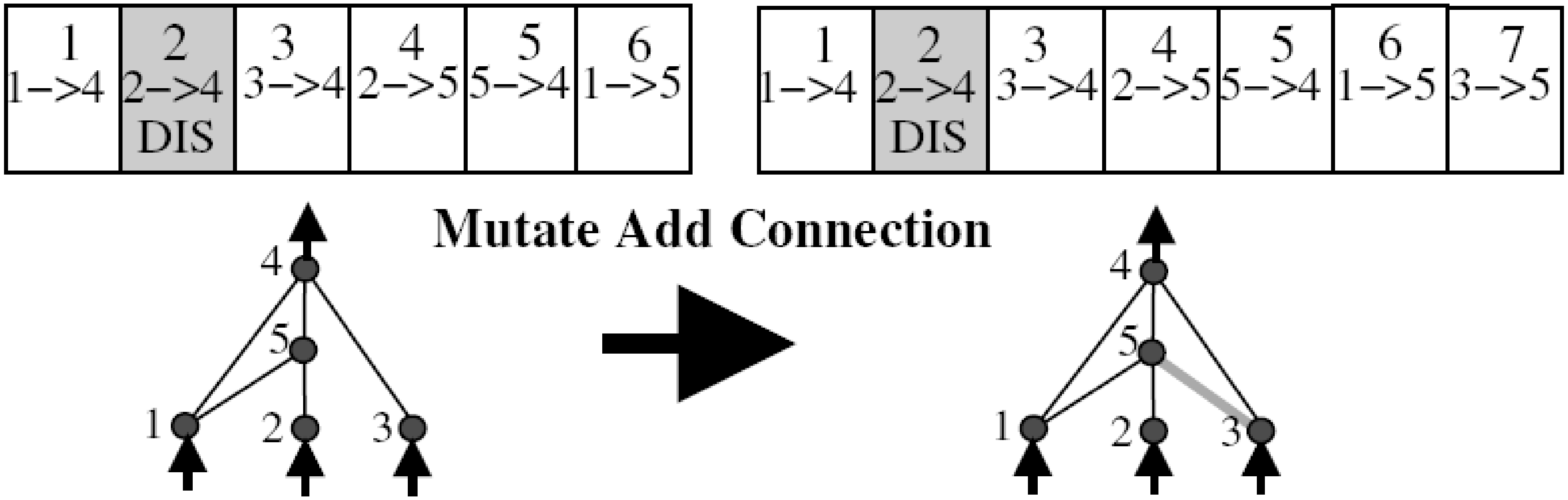
Node Genes					
	Node 1 Sensor	Node 2 Sensor	Node 3 Sensor	Node 4 Output	Node 5 Hidden

Connect. Genes							
In	1	2	3	2	5	1	4
Out	4	4	4	5	4	5	5
Weight	0.7	-0.5	0.5	0.2	0.4	0.6	0.6
	Enabled	**DISABLED**	Enabled	Enabled	Enabled	Enabled	Enabled
Innov	1	2	3	4	5	6	11

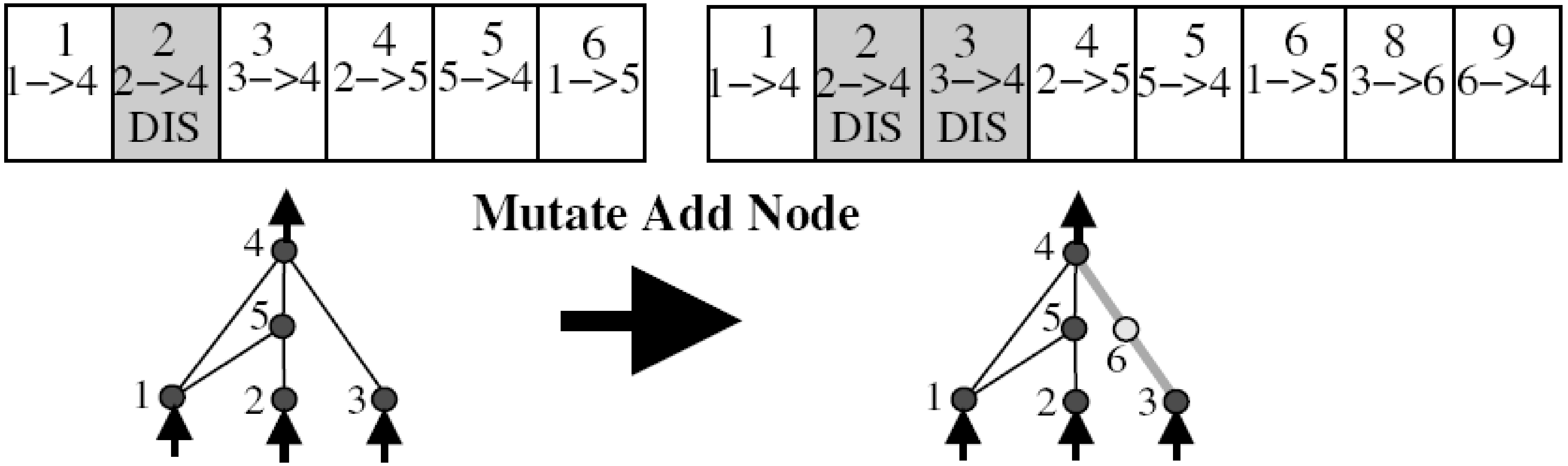
link enabled/disabled flag

innovation number - historical marking

Add Link Mutation



Add Neuron Mutation



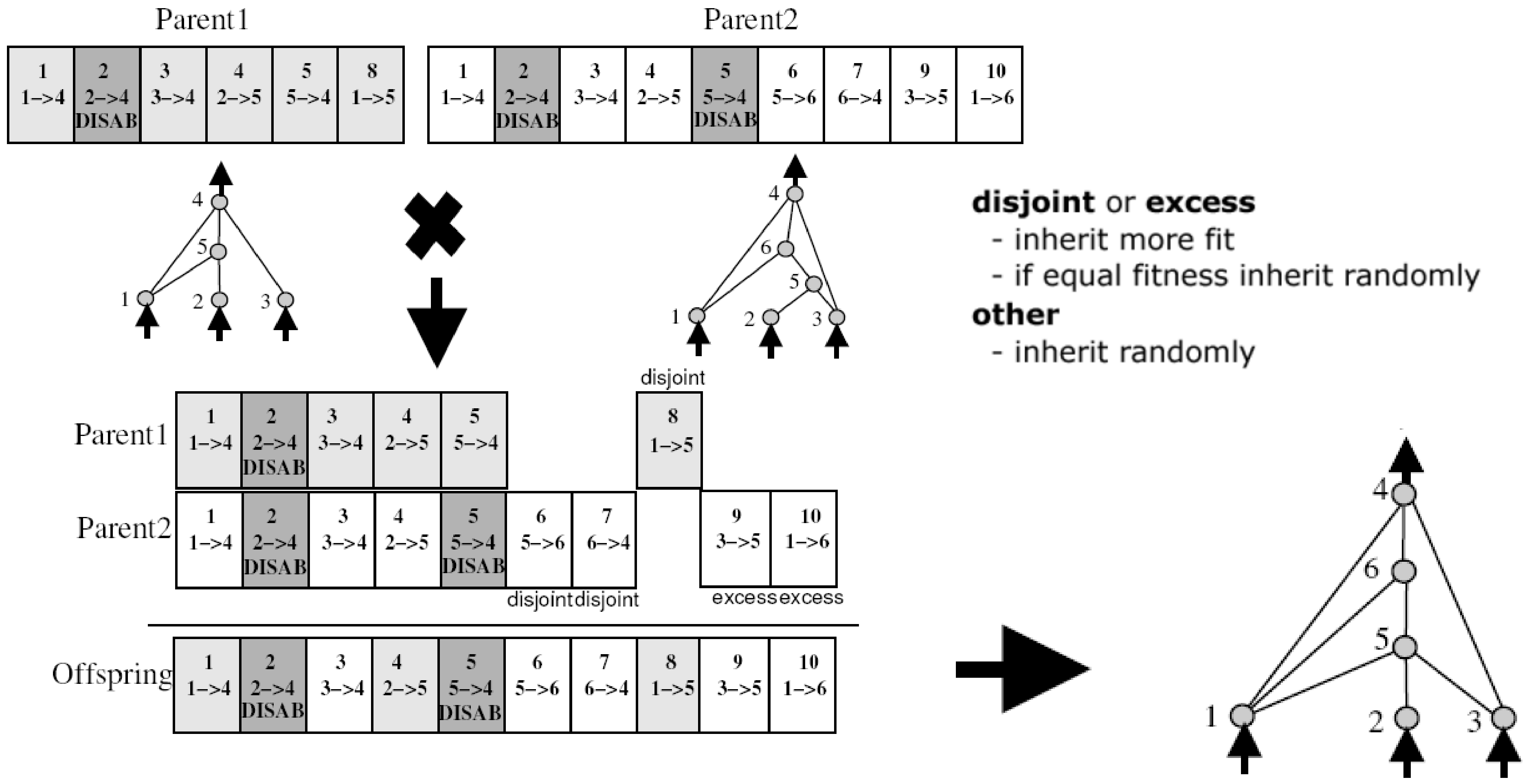
The weights of new neuron's incoming/outgoing links are set in a way which minimizes the difference between original and mutated networks.

Historical Markings

- **Q**: How to *align* two genomes of different size representing two different networks?
- **A**: let's use “the creation date” of a particular gene (caused by a structural mutation) – **historical marking (innovation number)**.
- Aligning two genomes:
  - when two genes with matching HMs are found, it is likely they have similar function in the network.
- HM is a counter, the same value is assigned for the same innovation within a single generation or more generations (i.e. adding a link between neurons #3 and #4).

Mating

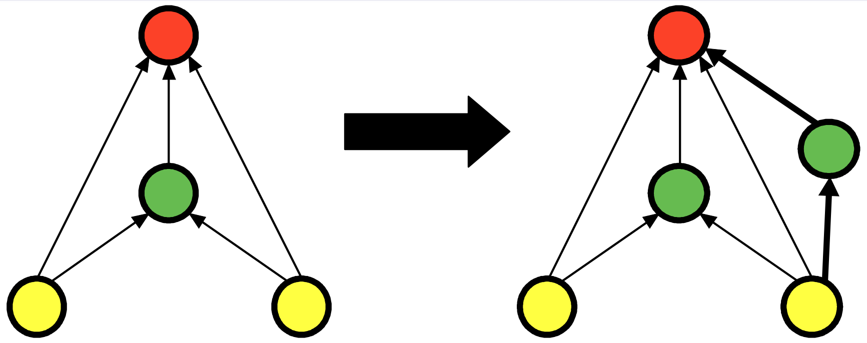
Let's use historical markings.



disjoint or **excess**
- inherit more fit
- if equal fitness inherit randomly

other
- inherit randomly

Niching



- There are networks of different sizes in the population.
- Adding a new structure:
  - likely lowers the fitness,
  - larger networks → longer genome → more time needed to optimize parameters.
- **New topologies must be protected** → niching.
- Here we use Explicit Fitness Sharing:

  Separate the population into species → selection and reproduction only among similar individuals → HMs again used to compute similarity of two genomes.

Similarity – Distance

$$d_{ij} = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W}$$

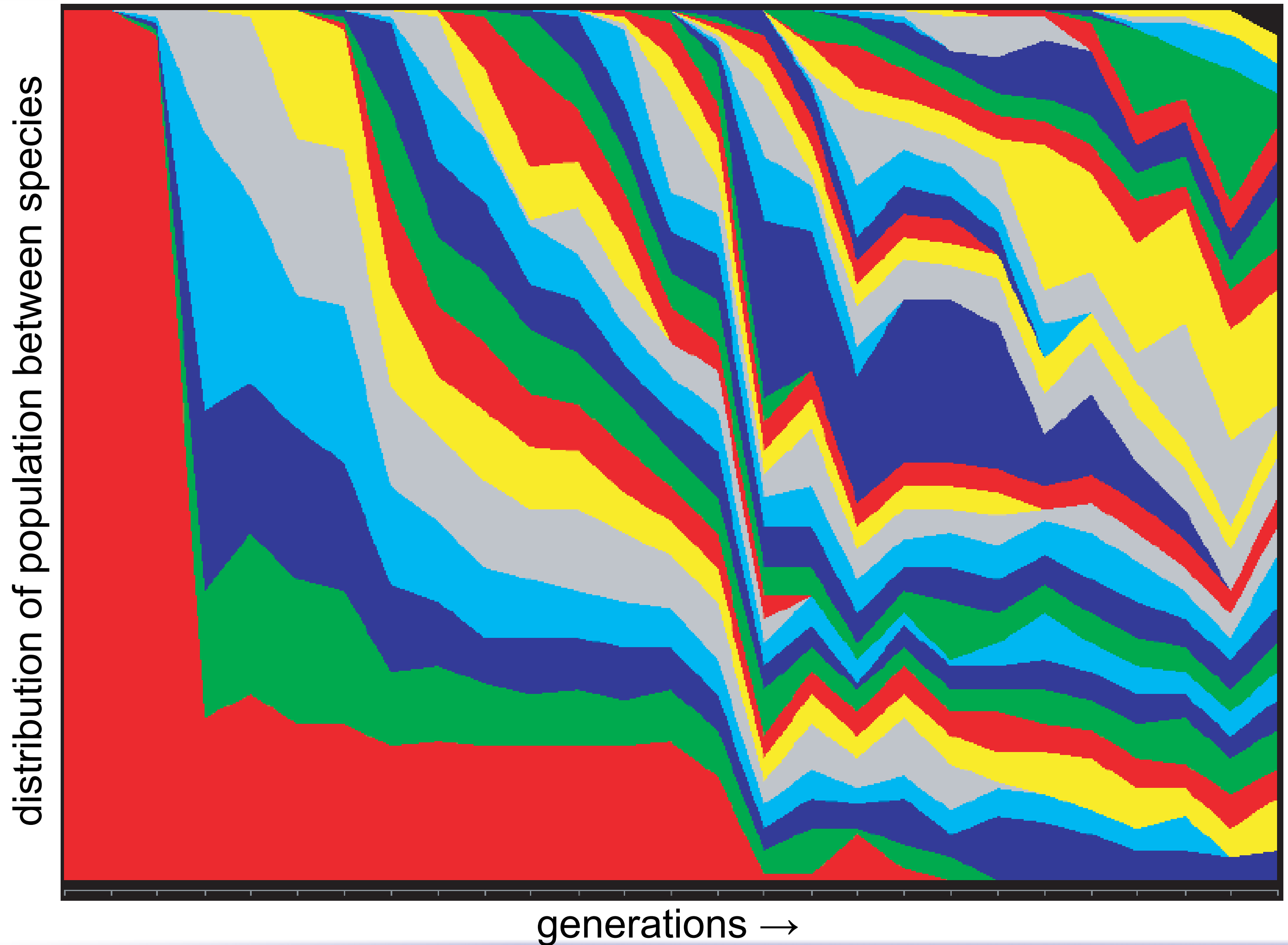
- Using historical markings again.
- *E* … # of excess genes,
- *D* … # of disjoint genes,
- *W* … averaged difference of matching weights,
- *N* … the length of the longer genome,
- c_1, c_2, c_3 … balancing constants.

Explicit Fitness Sharing

- Simplified fitness sharing: $O(n)$ vs. $O(n^2)$.
- Using sharing function *sh*.

1. Start with a single species spread over whole population – choose a random representative.
2. New individual x is assigned to a first appropriate species, satisfying:
   $d(x, representative) < \delta$
3. If no such species exist, create a new one and make x its representative.
4. Adjust fitness: divide it by the species size.
5. Average species fitness determines its offspring count.

Explicit Fitness Sharing 2



NEAT Overview

1. Initialize population.
2. Compute fitness for all individuals.
3. Speciate by means of Explicit Fitness Sharing.
4. Adjust fitness *f’ = f / species_size*.
5. Determine offspring count for all species proportionally to *f'*.
6. Eliminate inferior individuals of current generation.
7. Reproduce – replace current generation by its offspring.
8. While not satisfied, go to 2.

The Three Most Important Ideas Behind NEAT

- **Complexification** – start with small networks, gradually add neurons/links (reminds GMDH or GAME approaches).
- Concept of **historical markings** - cross/match only corresponding genes → **deals with competing conventions.**
- Use of **niching** - allows the survival of larger, recently structurally innovated networks → gives them time to optimize their weights and "show" that the structural innovation was beneficial.

Thanks for Attention

- What's next?
  - large-scale ANNs,
  - direct vs. indirect encodings.