

Artificial Neural Networks

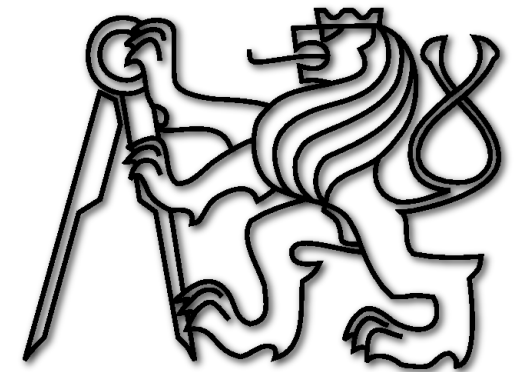
NeuroEvolution = ANN + EA



Jan Drchal

drchajan@fel.cvut.cz

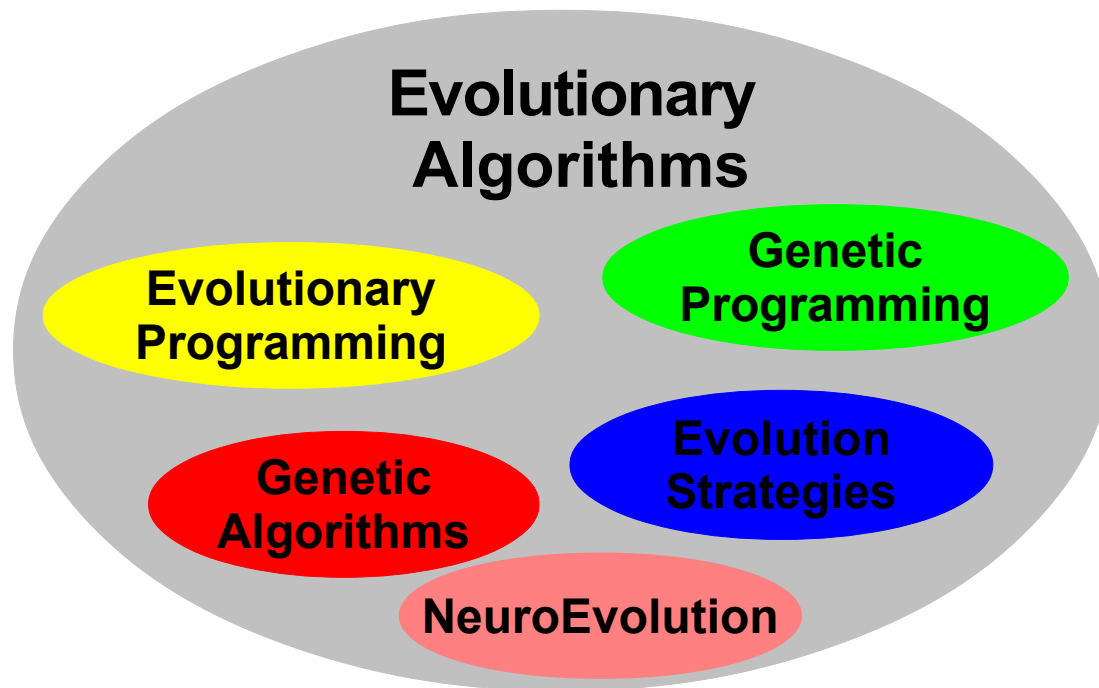
*Computational Intelligence Group
Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague*



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)



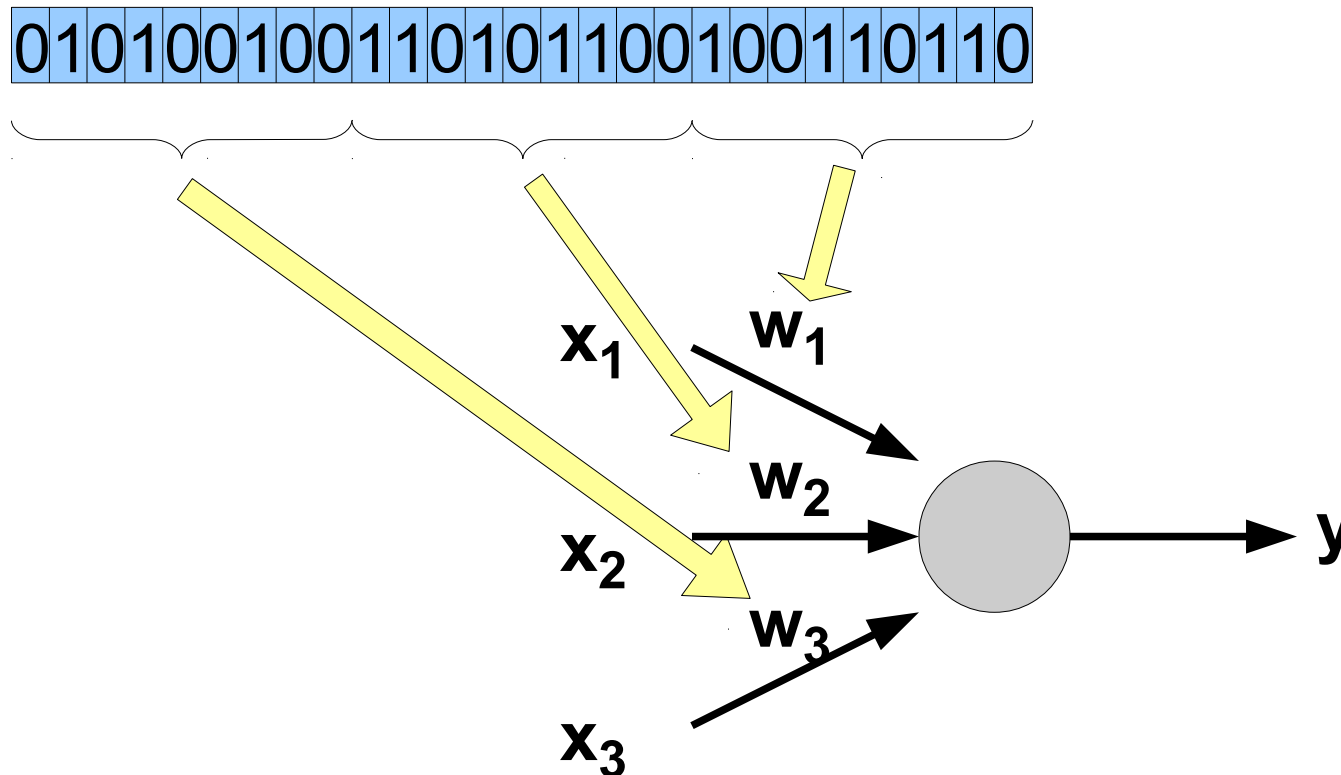
- **Genetic Algorithms:** binary strings
- **Evolutionary Strategies:** real vectors, only mutation.
- **Genetic Programming:** evolution of program trees.
- **Evolutionary Programming:** evolving FSMs.
- **NeuroEvolution**

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

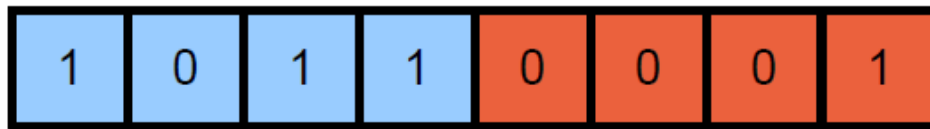
- Directly encode the weights as a bit string:



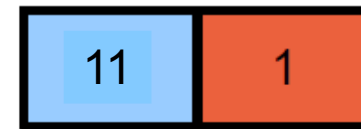
- Can we do it better? Yes.

Floating-Point Encoding

- Motivation: simplicity, precision



Binary string encodes vector of 2 numbers .

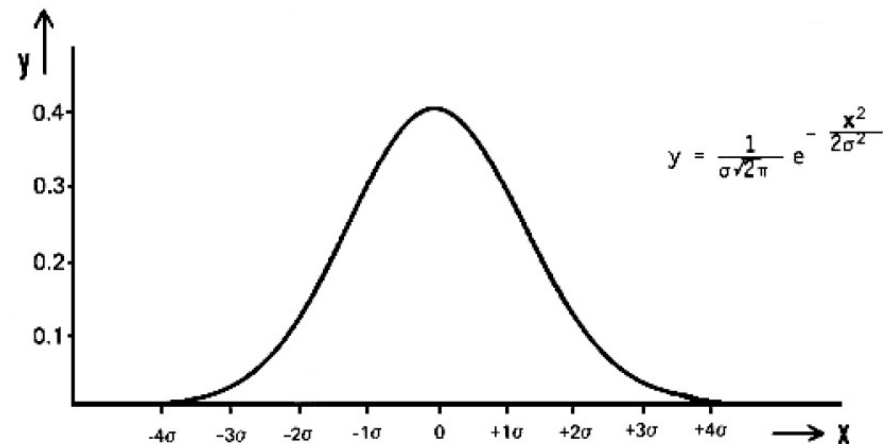


The same encoded using floating-point encoding.

What about mutation? -> Gaussian noise.

Idea:

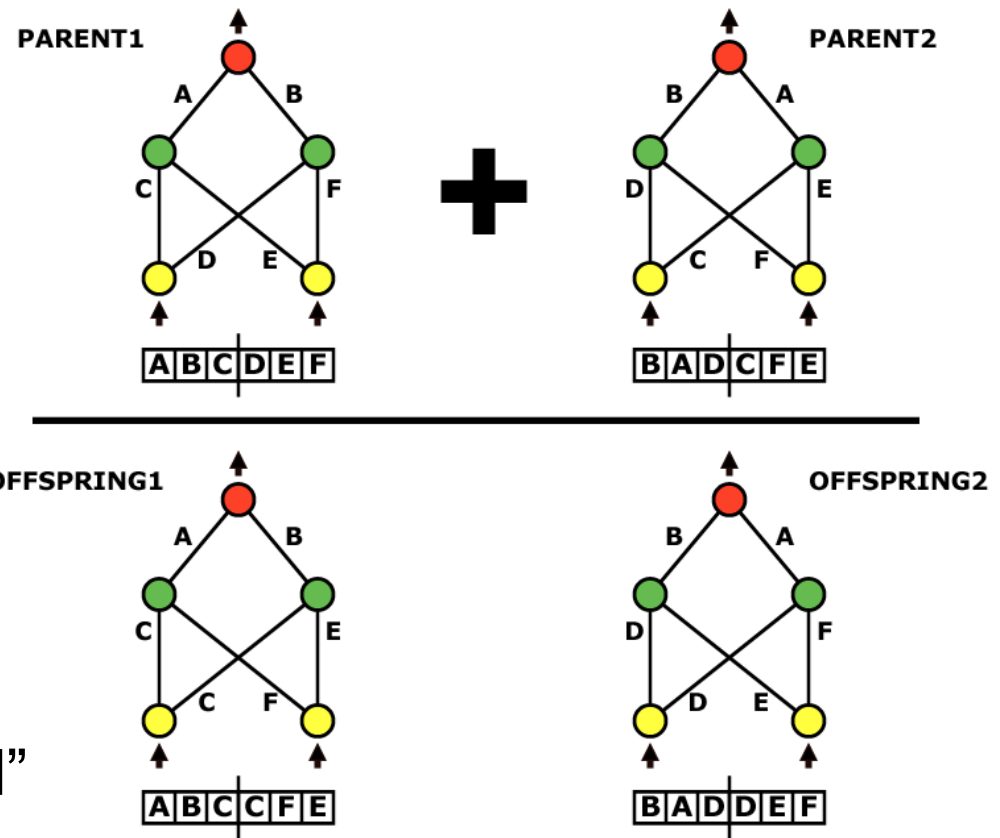
small changes with *higher* probability,
large changes with *lower*.



- Useful for integers and floats ...

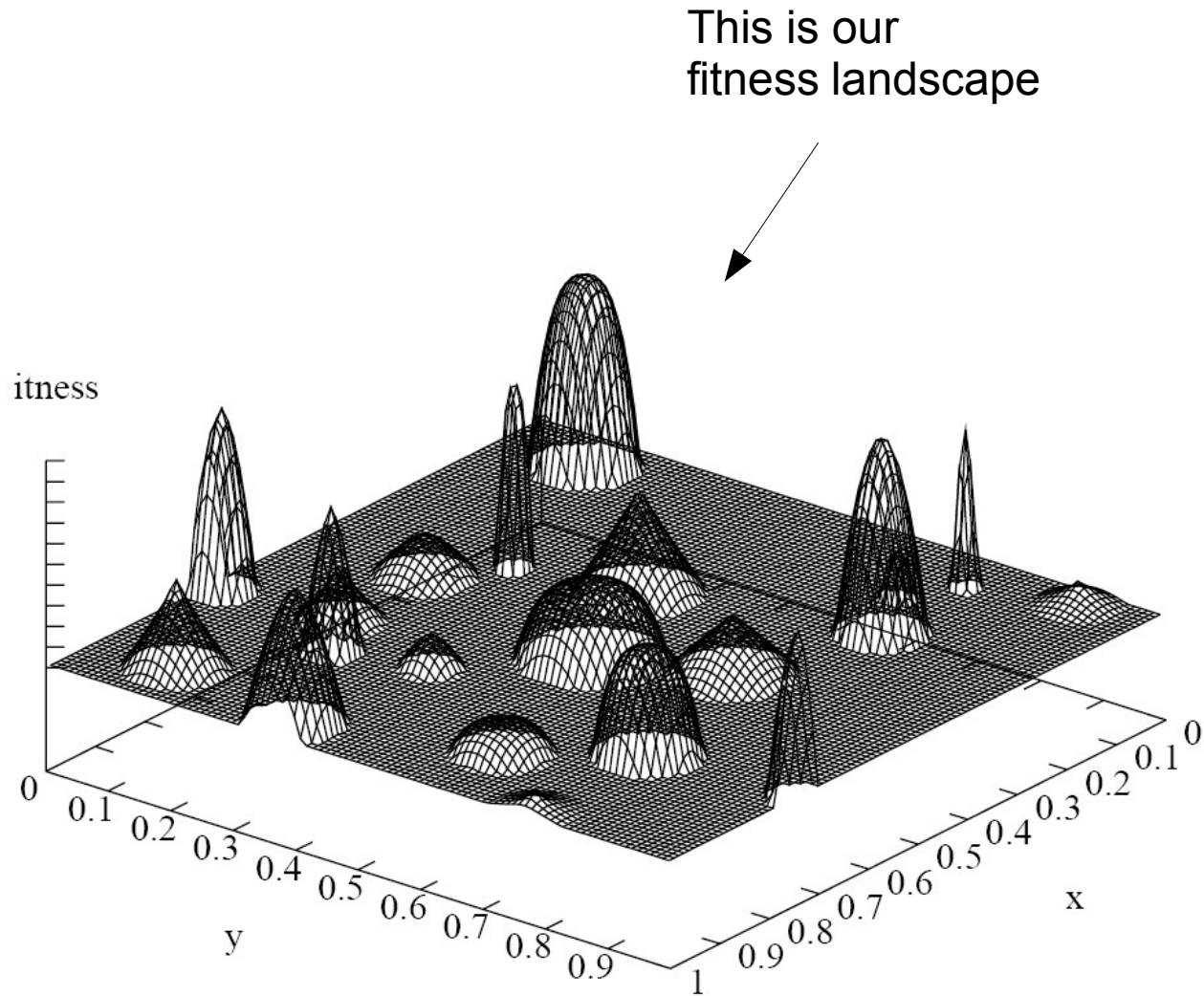
Competing Conventions Problem

- 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.



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.**

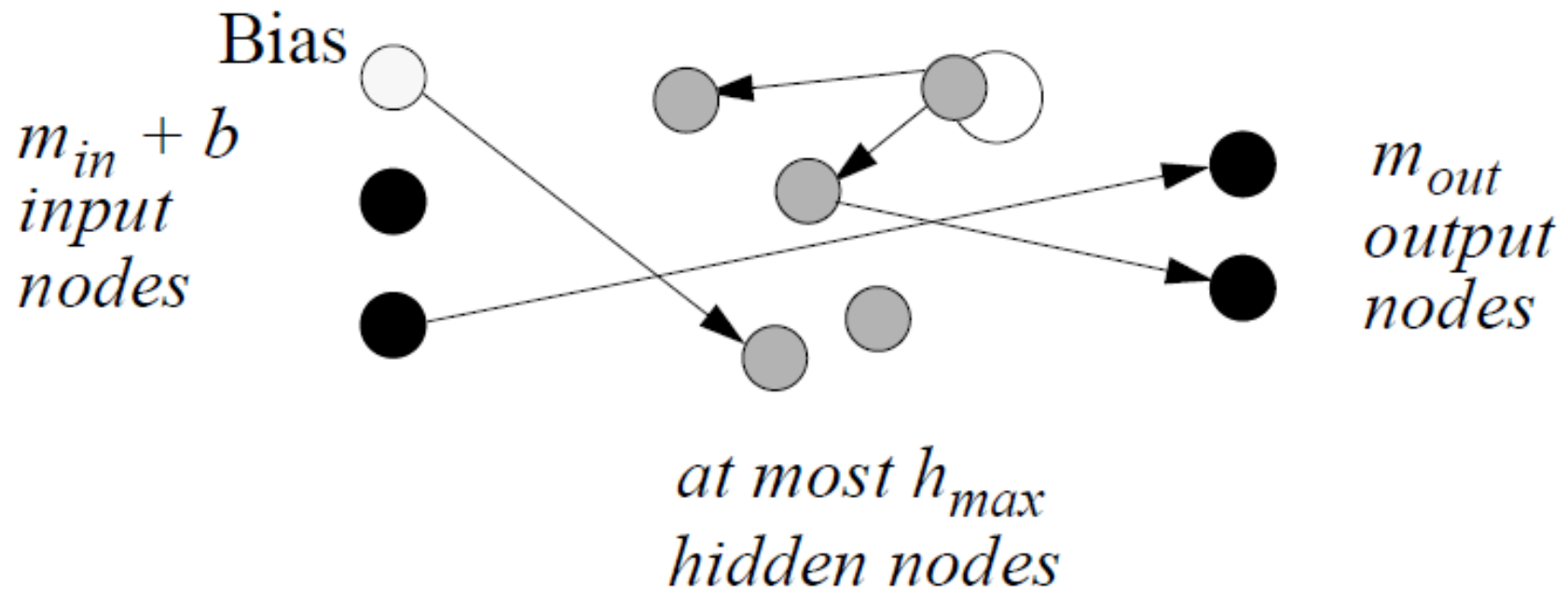
GNARL

- **GeNeralized Aquisition of Recurrent Links.**
- 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

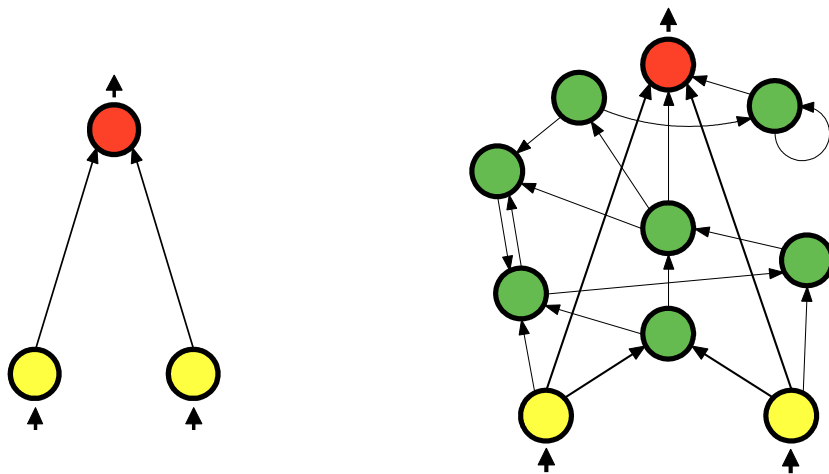
- Sample of GNARL's initial random network:



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

NEAT

- **NeuroEvolution of Augmenting Topologies:**
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

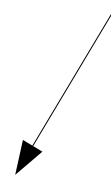
- Topology is augmented by adding neurons and links between.

→ Variable genome length.

- Mutations:

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

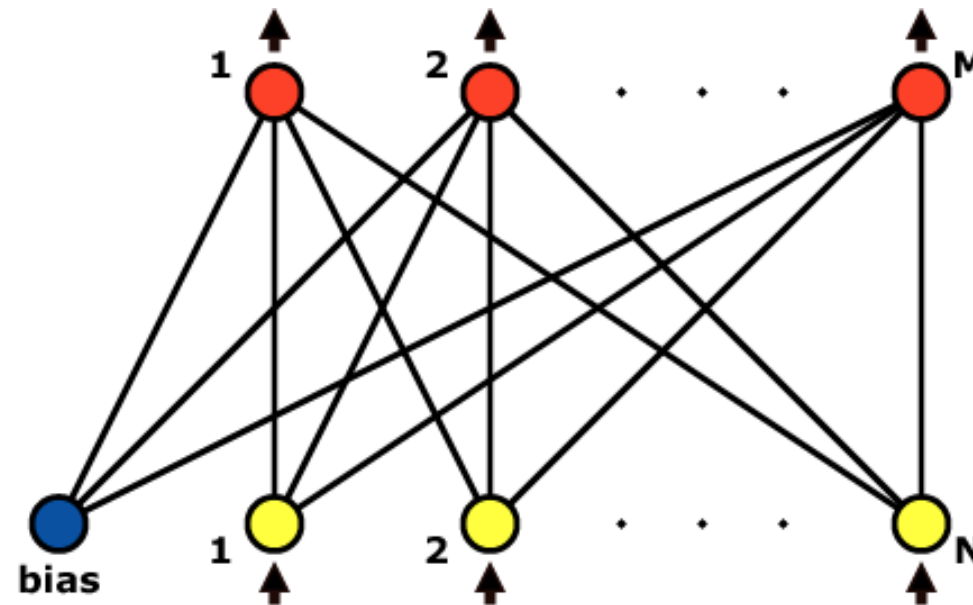
Note, some newer implementations use pruning However, it is not essential.



- Mating – special crossover two parents → single child.

Minimal Substrate

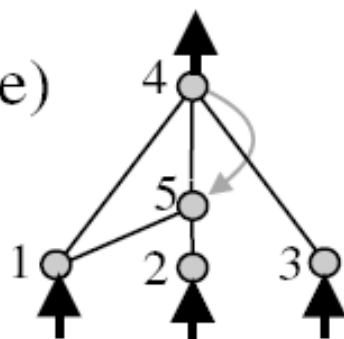
- Initial population is formed of the simplest topologies: fully connected feed-forward networks without hidden layers: the minimal substrate.



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

Network (Phenotype)



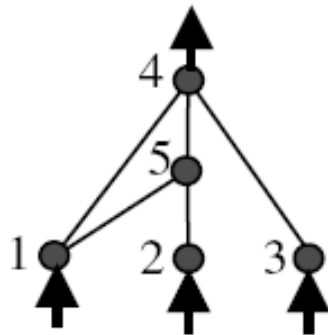
link
enabled/disabled
flag

innovation
number -
historical
marking

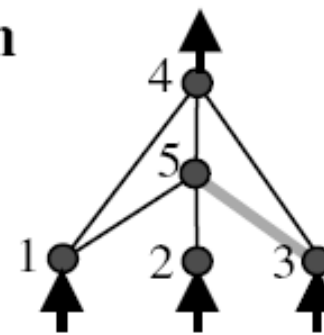
Add Link Mutation

1	2	3	4	5	6
1->4	2->4 DIS	3->4	2->5	5->4	1->5

1	2	3	4	5	6	7
1->4	2->4 DIS	3->4	2->5	5->4	1->5	3->5



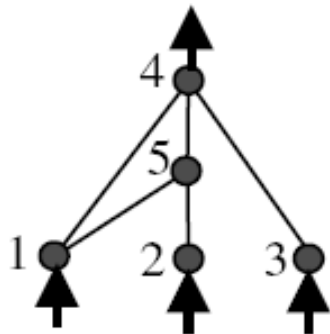
Mutate Add Connection



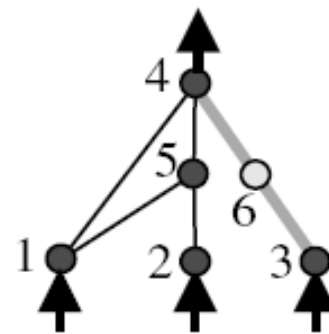
Add Neuron Mutation

1	2	3	4	5	6
1->4	2->4 DIS	3->4	2->5	5->4	1->5

1	2	3	4	5	6	8	9
1->4	2->4 DIS	3->4 DIS	2->5	5->4	1->5	3->6	6->4



Mutate Add Node



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

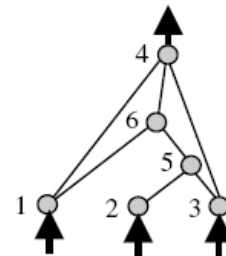
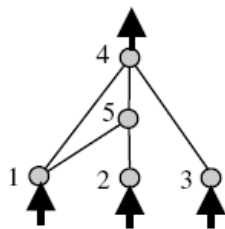
Historical Markings

- **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.

Parent1						Parent2								
1	2	3	4	5	8	1	2	3	4	5	6	7	9	10
1->4	2->4 DISAB	3->4	2->5	5->4	1->5	1->4	2->4 DISAB	3->4	2->5	5->4 DISAB	5->6	6->4	3->5	1->6



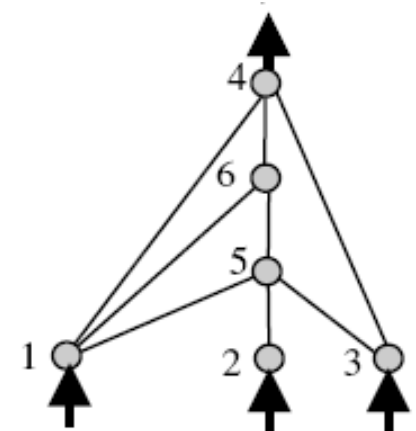
disjoint or excess

- inherit more fit
- if equal fitness inherit randomly

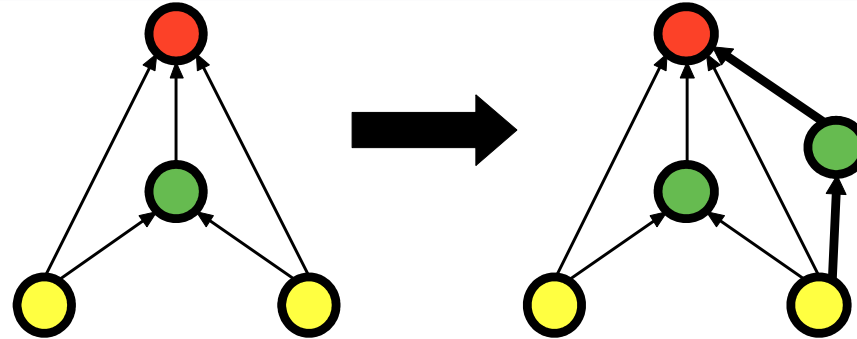
other

- inherit randomly

Parent1	1 1->4	2 2->4 DISAB	3 3->4	4 2->5	5 5->4	disjoint		8 1->5			
Parent2	1 1->4	2 2->4 DISAB	3 3->4	4 2->5	5 5->4 DISAB	6 5->6	7 6->4	excess		9 3->5	10 1->6
Offspring	1 1->4	2 2->4 DISAB	3 3->4	4 2->5	5 5->4 DISAB	6 5->6	7 6->4	8 1->5	9 3->5	10 1->6	



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

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

- 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.

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.