NEAT and HyperNEAT

Michal Pospěch & Daniel Crha

March 21, 2019

Faculty of Mathematics and Physics, Charles University

Neuroevolution

Fixed Topology Evolution

- Searching the space of connection weights
- Topology is given, does not change during evolution

Evolving Topology

- Technical challenges:
  - good representation
  - not removing non-optimized network to early
  - minimisation of networks without need for a complexity function
- TWEANNs - Topology and Weight Evolving Artificial Neural Networks

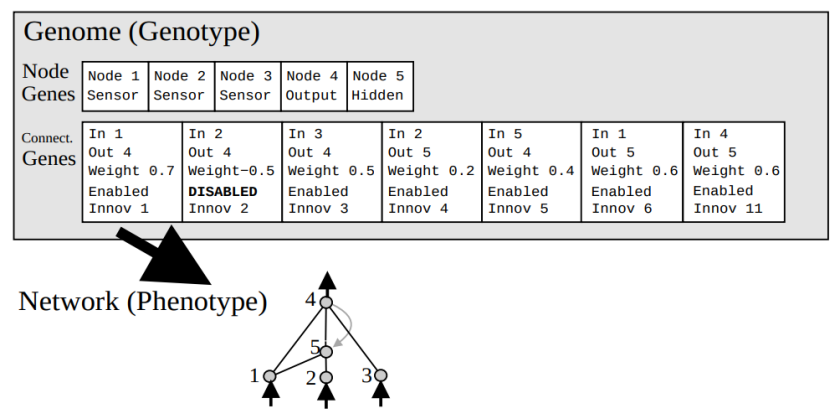
NEAT

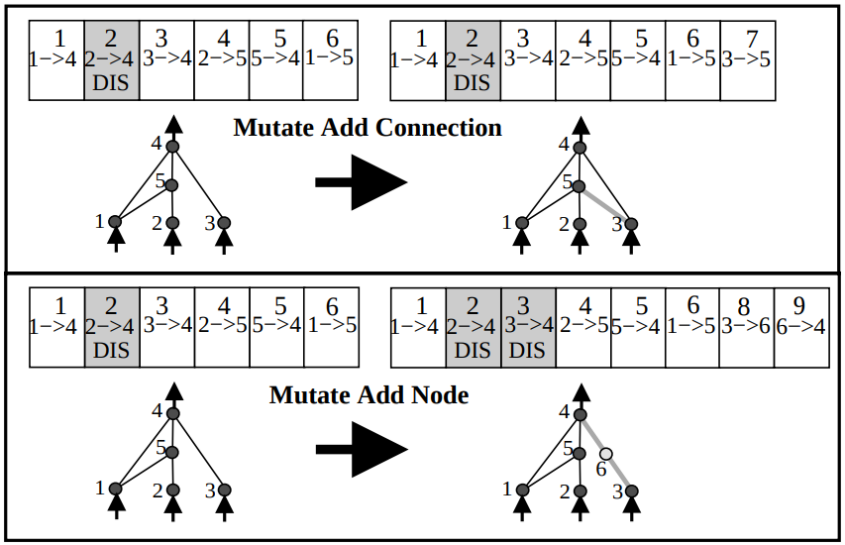
NEAT

- NeuroEvolution of Augmenting Topologies
- Stanley and Miikkulainen, 2002
- solves all the issues aforementioned issues

Encoding and Mutation

- linear representations of network connectivity
  - 2 types of genes (nodes and connections)
  - innovation number
  - node
- 3 types of mutation
  - connection weight mutation
  - new node
  - new connection

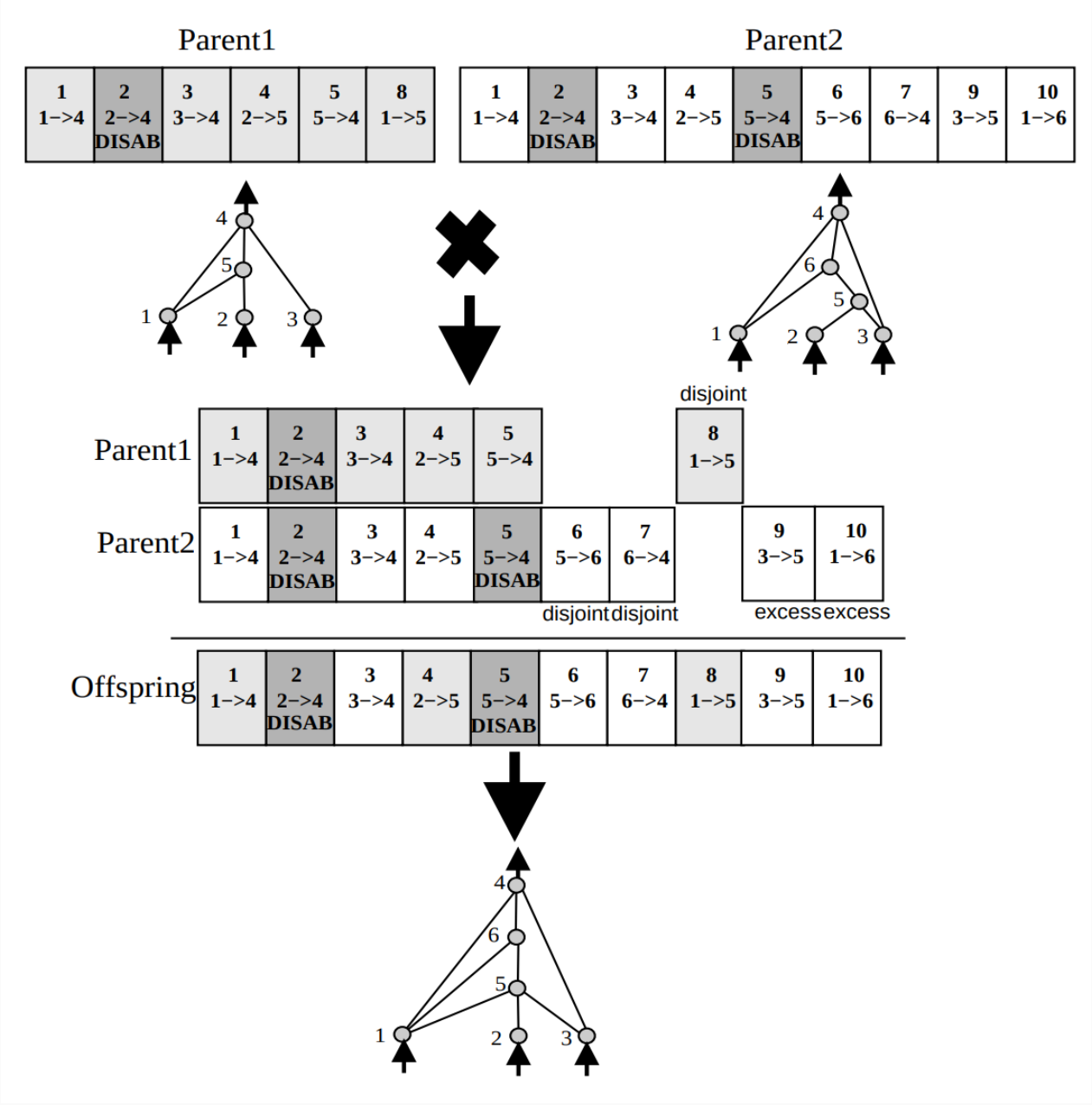




Historical Markings and Crossover

- innovation number
  - new gene via mutation → global innovation number++
  - used to line-up genomes during crossover
- crossover
  - matching genes randomly
  - all disjoint and excess genes

Crossover


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

Speciation

- population is divided into species based on compatibility history

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \overline{W}$$

  and compatibility threshold δ_t
- each population is assigned number of offsprings based on sum of its *adjusted* fitnesses

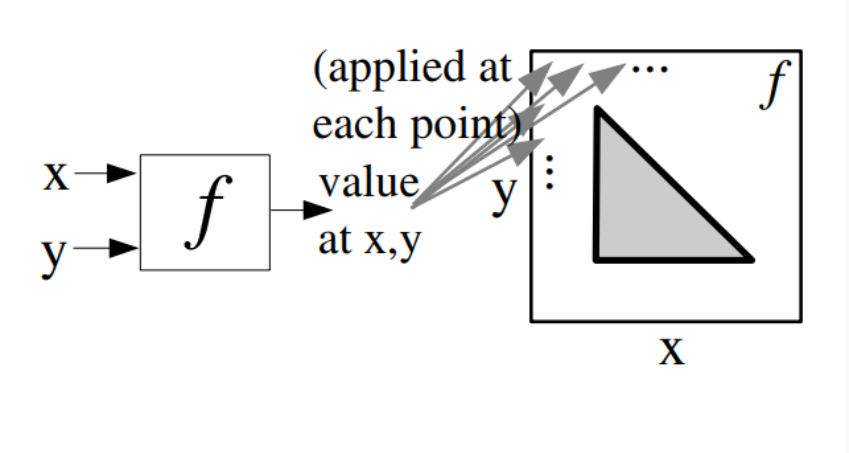
$$f'_i = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))}$$

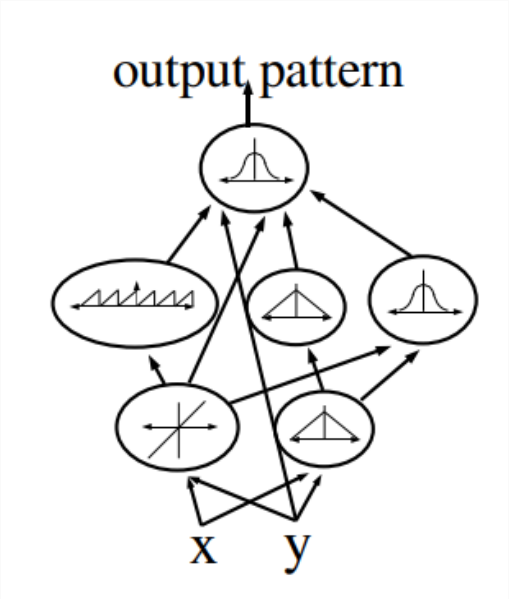
- novel topologies are protected from extinction

HyperNEAT

Compositional Pattern Producing Networks

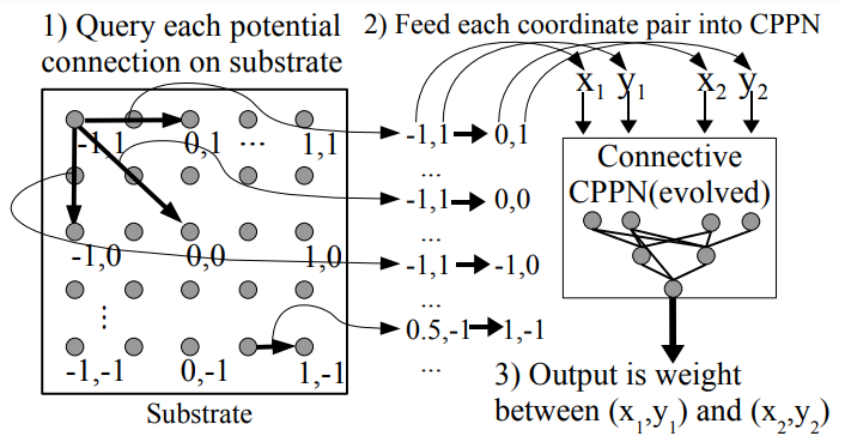
- represent repeating patterns in cartesian space
- nodes are functions
- simple functions can be composed into networks producing complex patterns (repetition, symmetry)





HyperNEAT

- CPPNs evolved via NEAT
- nodes are given (2D grid)
- input: 2 points , output: weight of connection



Substrate

- types
  - 2D grid
  - 3D grid
  - sandwich
  - circular
- placement of inputs and outputs can be exploited
- can be up/down-scaled