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

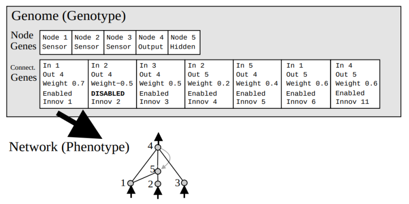
NEAT

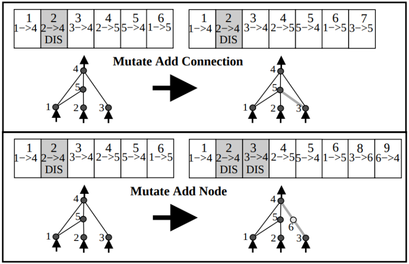
NEAT

- 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

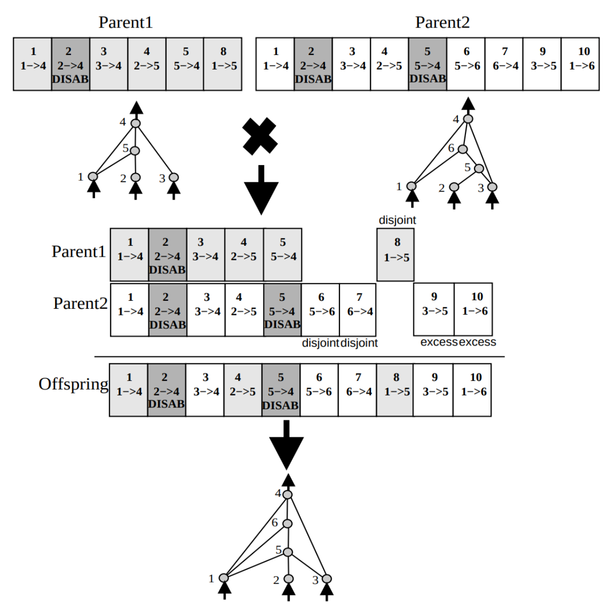




Historical Markings and Crossover

- 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
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

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

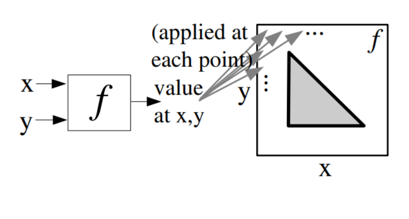
$$f_i' = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))}$$

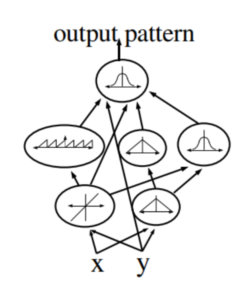
- novel topologies are protected from extinction

HyperNEAT

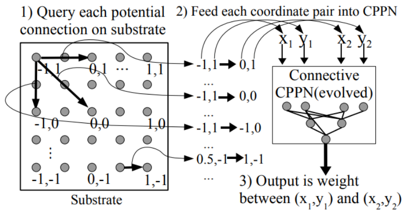
Compositional Pattern Producing Networks

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





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



Substrate

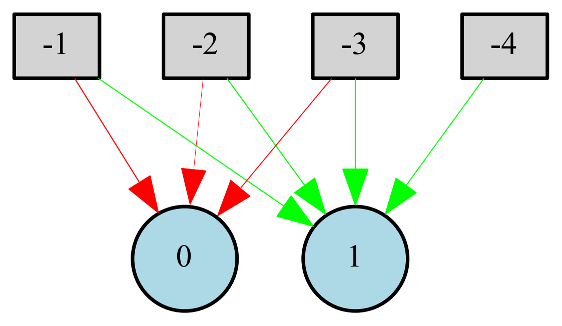
- types
  - 2D grid
  - 3D grid
  - sandwich (*state-space sandwich*)
  - circular
- placement of inputs and outputs can be exploited
- can be up/down-scaled

Performance and examples

Evaluation

- used environment - OpenAI Gym, Cartpole-v1
- our results (GIFs) - https://imgur.com/a/4nLJ4oV
- other methods - https://github.com/adibyte95/CartPole-OpenAI-GYM

NEAT and cartpole



HyperNEAT and cartpole

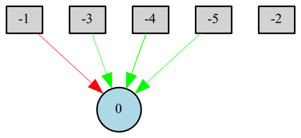


Figure 1: CPPN

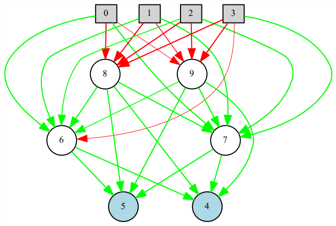


Figure 2: ANN

Comparison

Method	Evaluations	Generations	No. Nets
Ev. Programming	**307,200**	150	2048
Conventional NE	**80,000**	800	100
SANE	**12,600**	63	200
ESP	**3,800**	19	200
NEAT	**3,600**	24	150

Figure 3: Pole balancing results

Method	Evaluations	Generalization	No. Nets
CE	**840,000**	300	16,384
ESP	**169,466**	289	1,000
NEAT	**33,184**	286	1,000

Figure 4: Double pole balancing results

NEAT ablations

Method	**Evaluations**	**Failure Rate**
No-Growth NEAT (Fixed-Topologies)	**30,239**	80%
Nonspeciated NEAT	**25,600**	25%
Initial Random NEAT	**23,033**	5%
Nonmating NEAT	**5,557**	0
Full NEAT	**3,600**	0

Summary

- all components of NEAT are important
- NEAT is better than other EAs in imitating evolution by *"both optimizing and complexifying solutions"*
- good for complex problems
- HyperNEAT - good input of high dimension (Cartpole is too simple)
-