Artificial Neural Networks
NeuroEvolution = ANN + EA

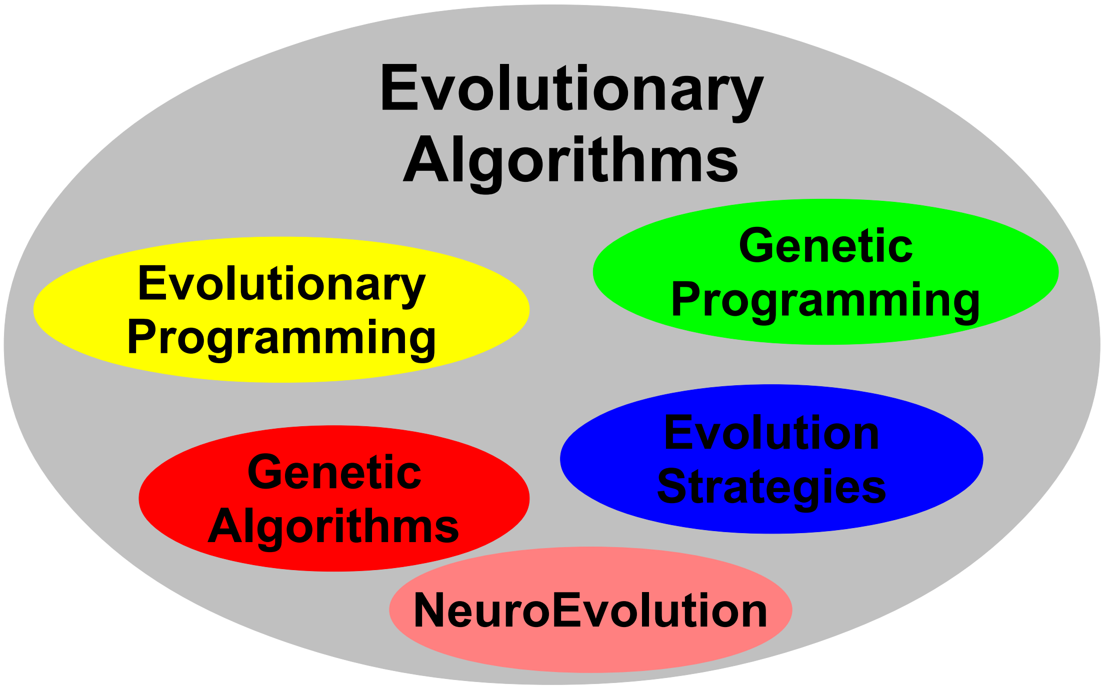
Jan Drchal
drchajan@fel.cvut.cz

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 networks.
- Use of **Evolutionary Algorithms** → slower, but more robust than classic gradient methods like Back-Propagation.

Evolutionary Algorithms (EAs)



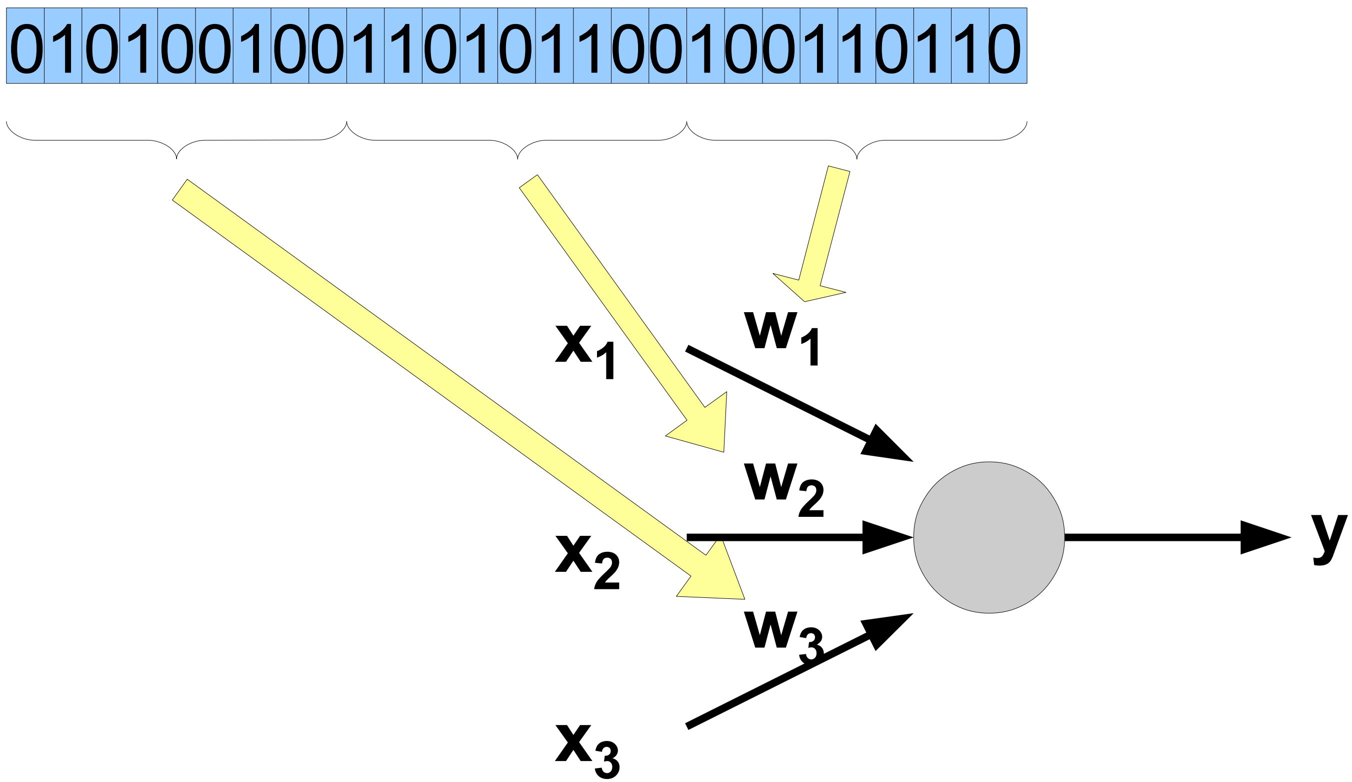
- **Genetic Algorithms:** binary strings
- **Evolutionary Strategies:** real vectors, only mutation.
- **Genetic Programming:** evolution of program trees.
- **Evolutionary Programming:** evolving FSMs.
- **NeuroEvolution**

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

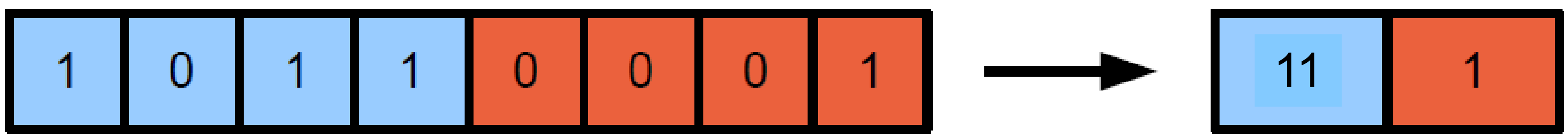
- Directly encode the weights as a bit string:



- Can we do it better? Yes.

Floating-Point Encoding

- Motivation: simplicity, precision

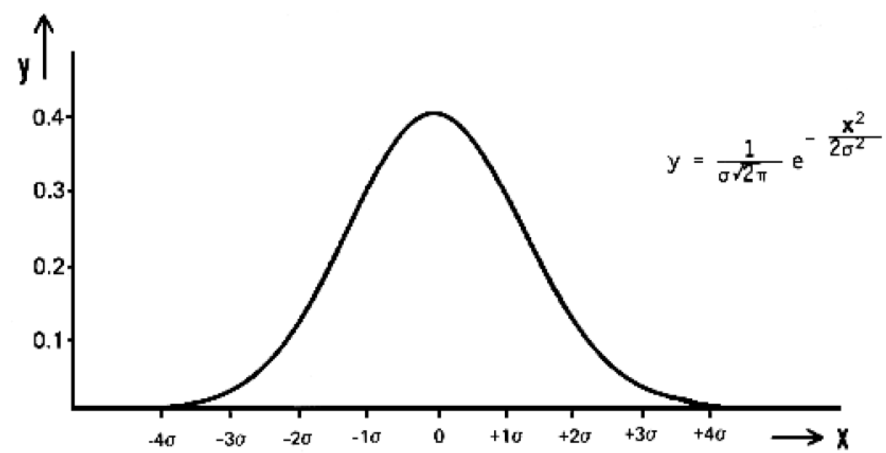


Binary string encodes vector of 2 numbers .

The same encoded using floating-point encoding.

What about mutation? -> Gaussian noise.

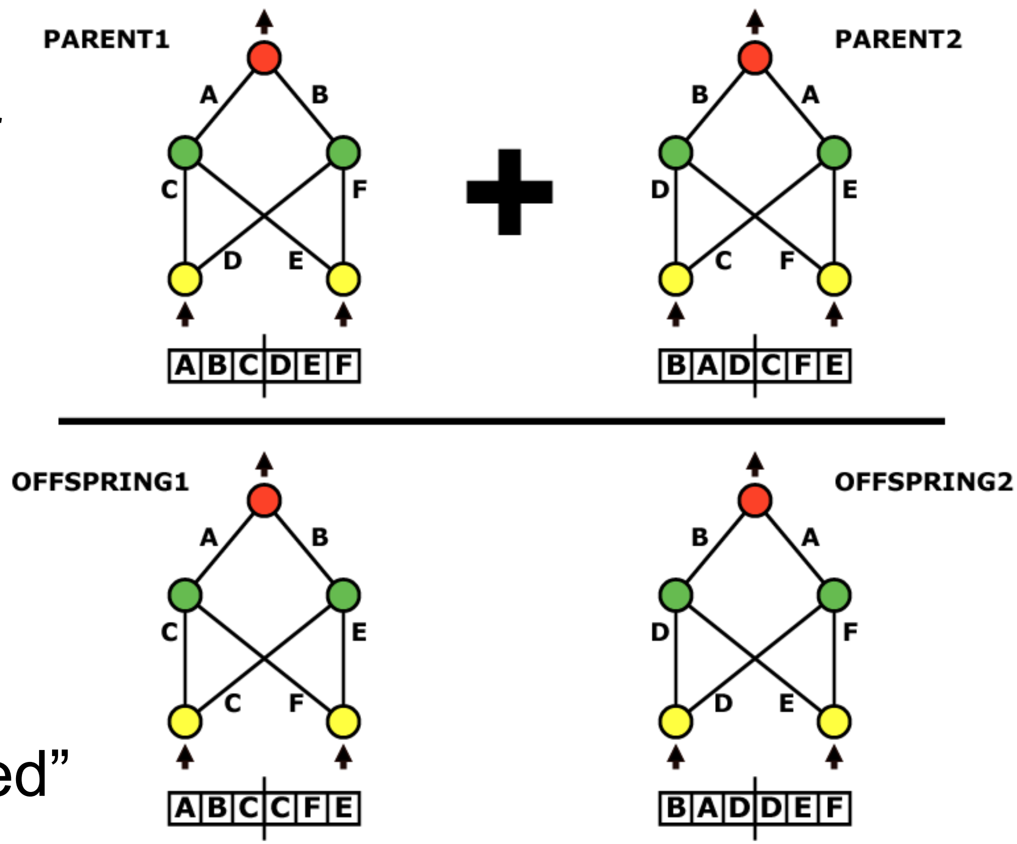
Idea:
small changes with *higher* probability,
large changes with *lower*.



- Useful for integers and floats ...

Competing Conventions Problem

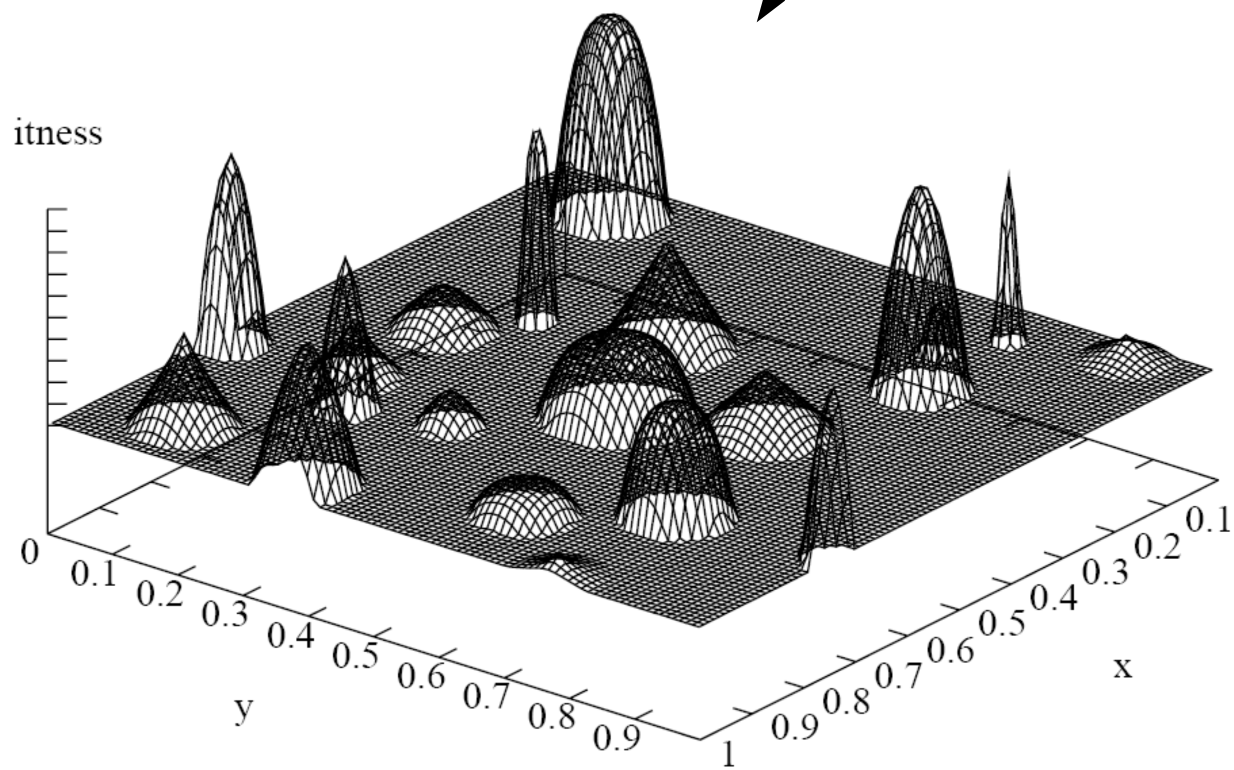
- Problem of competing conventions:
  - Same solution can be represented by many genomes → ordering of weights matters.
  - Error (fitness) landscape contains many optima representing the same solution.
  - Crossover of two such individuals will most probably lead to “crippled” offspring.



Multimodal Domains

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

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.**

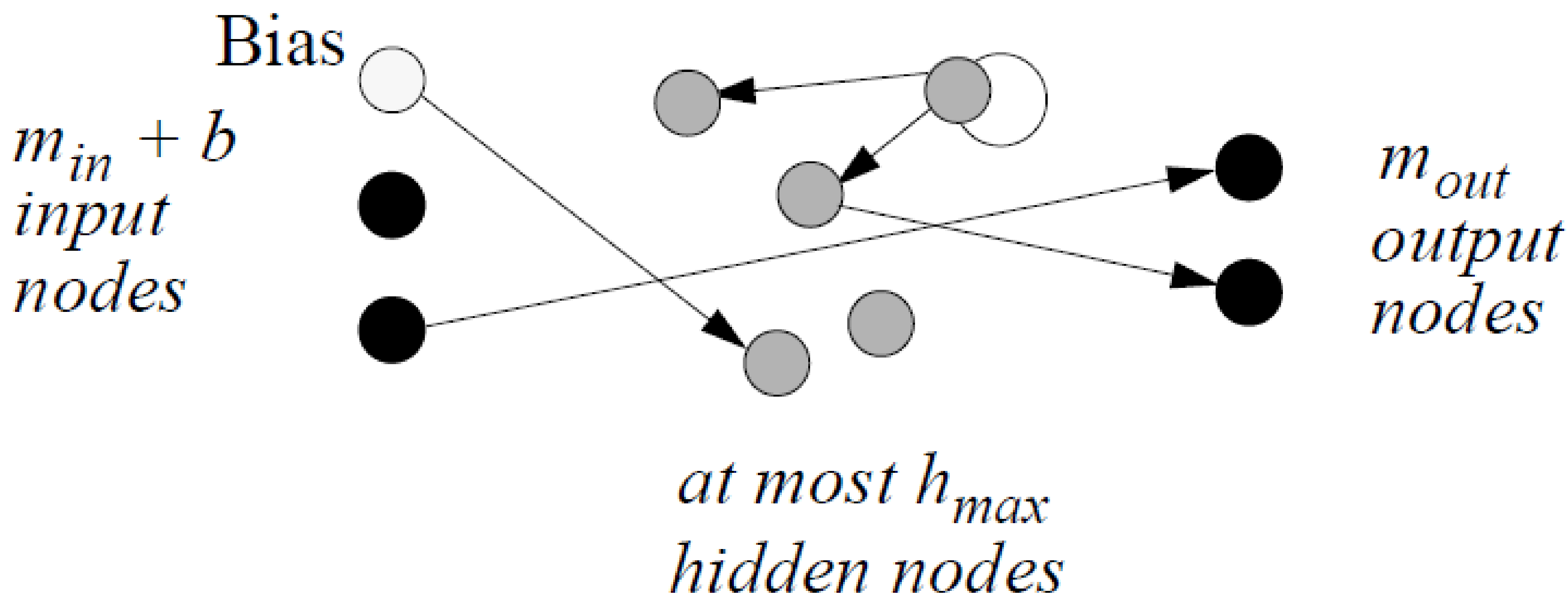
GNARL

- **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.

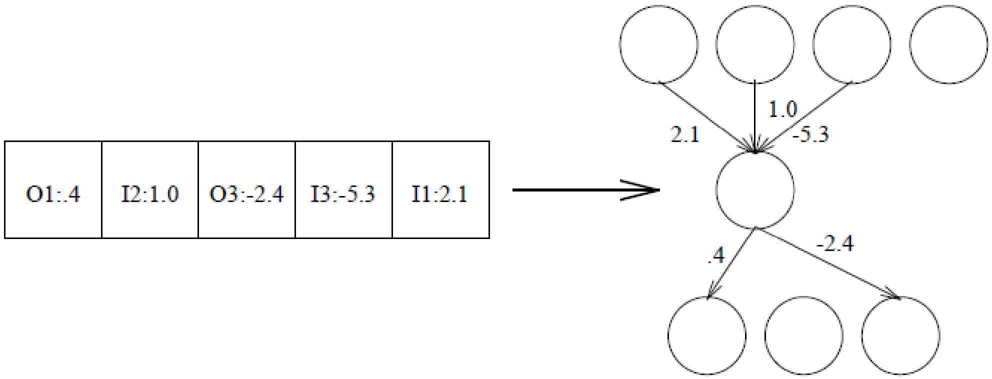
SANE

- **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

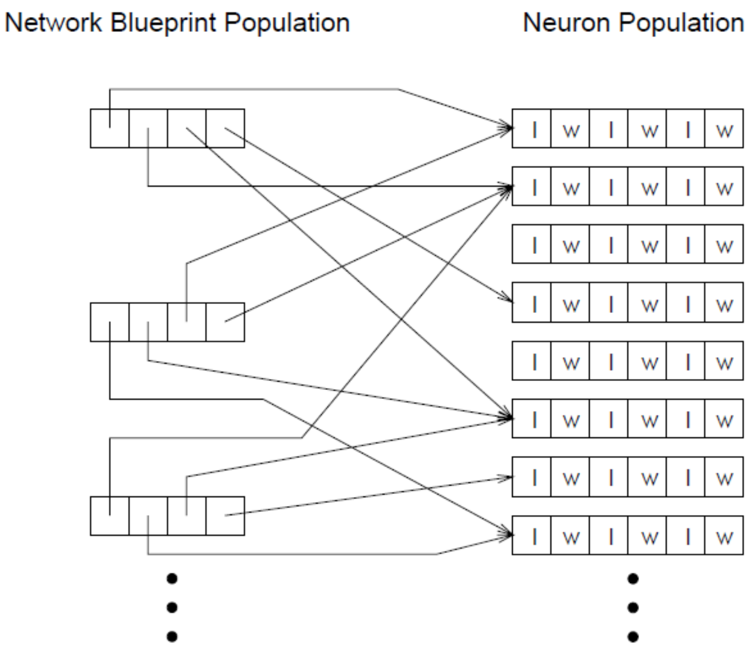
David E. Moriarty, Risto Miikkulainen: **Forming Neural Networks Through Efficient and Adaptive Coevolution**

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 encoded network:

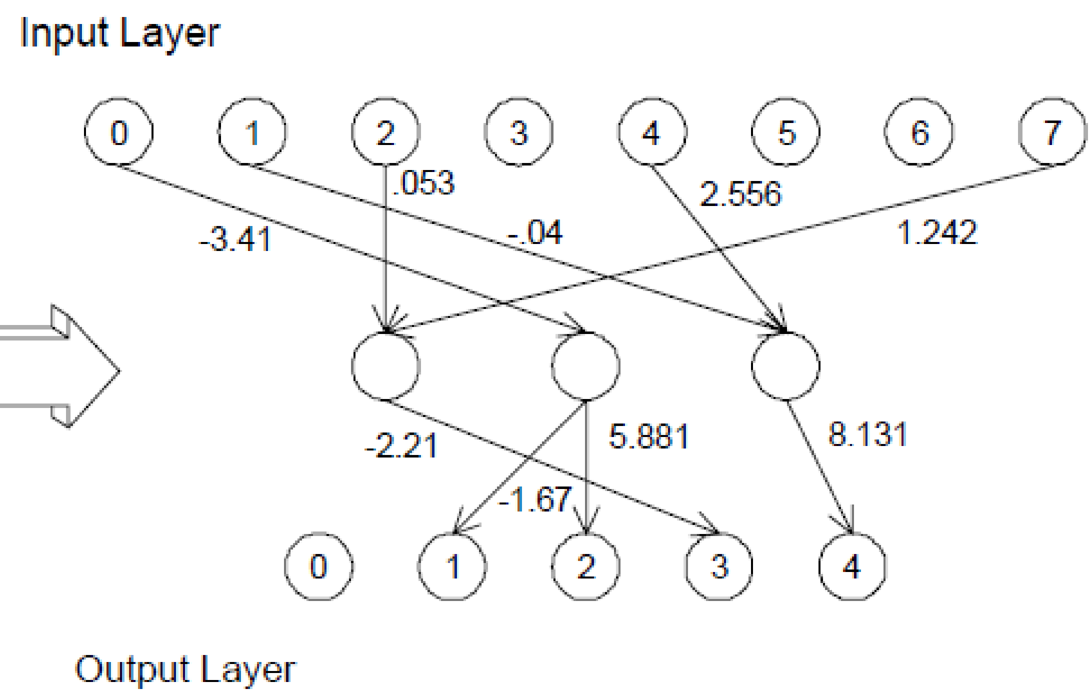
connect to input if ≤ 127

connect to output if > 127

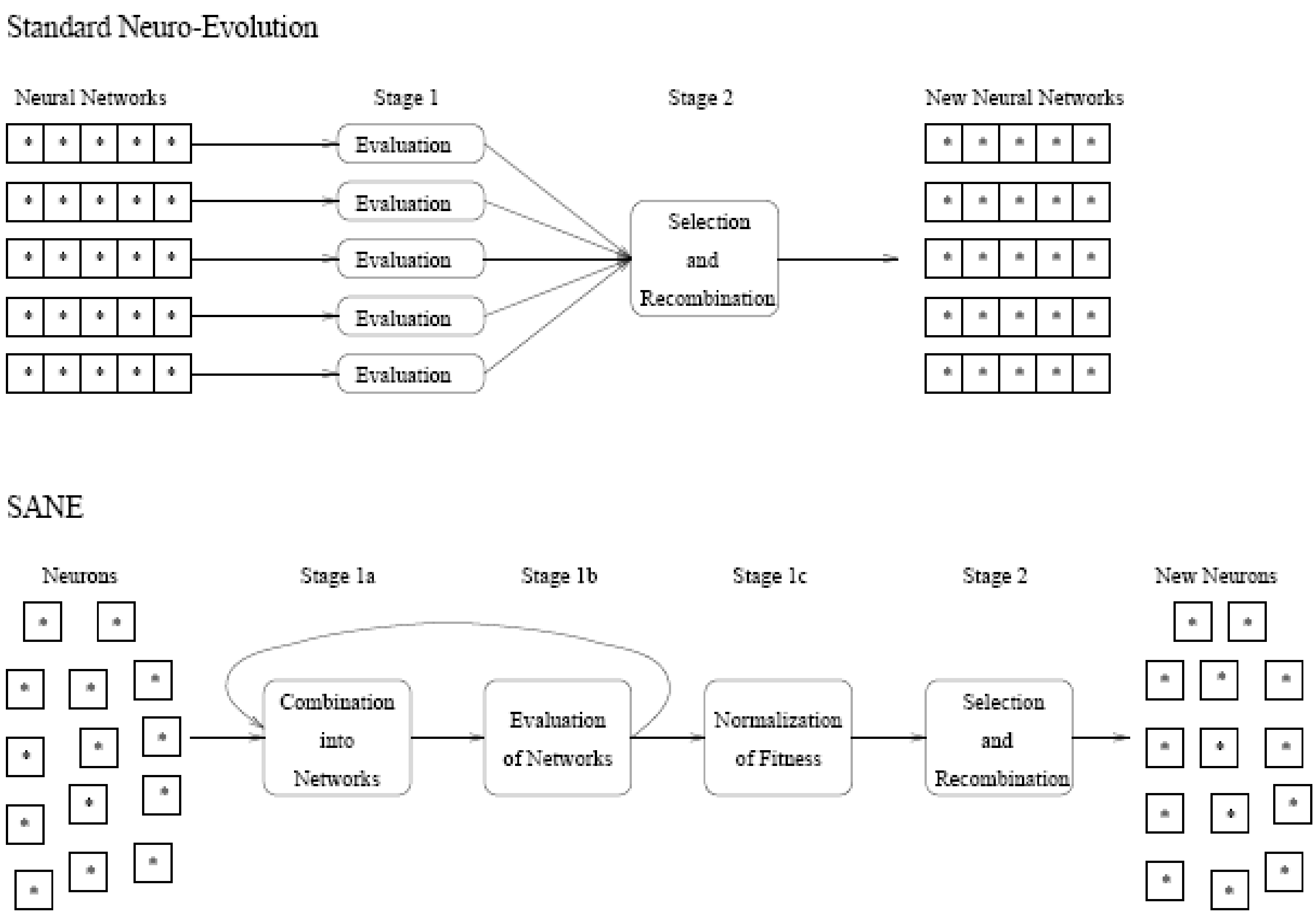
label	weight				
15	1.242	143	-2.21	2	.053
212	5.811	32	-3.41	151	-1.67
65	-.04	100	2.556	134	8.131

65 mod (# of inputs)

134 mod (# of outputs)

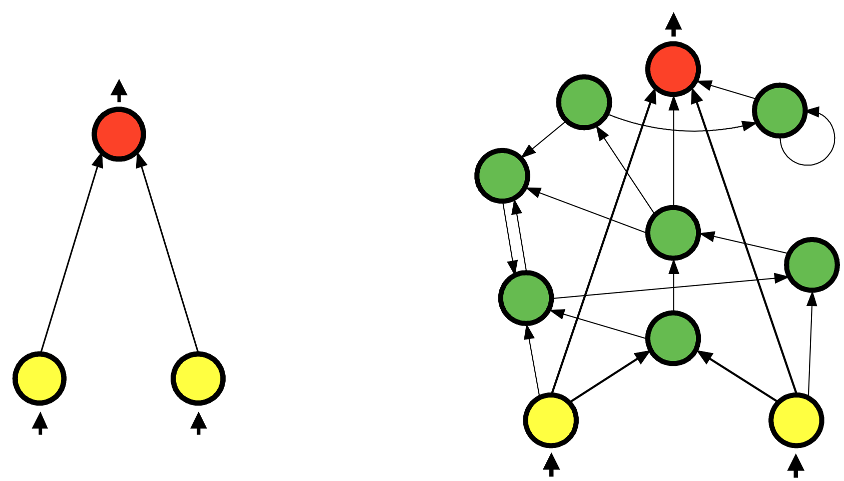


SANE 4


Standard Neuro-Evolution
Neural Networks
Stage 1
Stage 2
New Neural Networks
Evaluation
Evaluation
Evaluation
Evaluation
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.

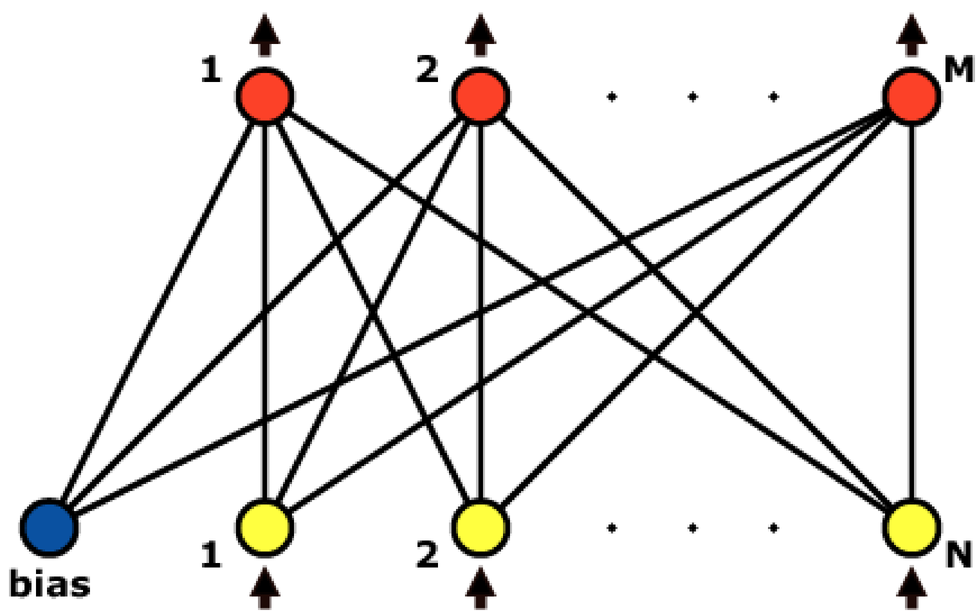
  → Variable genome length.

- Mutations:
  - parametric – Gaussian noise,
  - structural – adding neurons & links (no pruning), switch on/off links.
- Mating – special crossover two parents → single child.

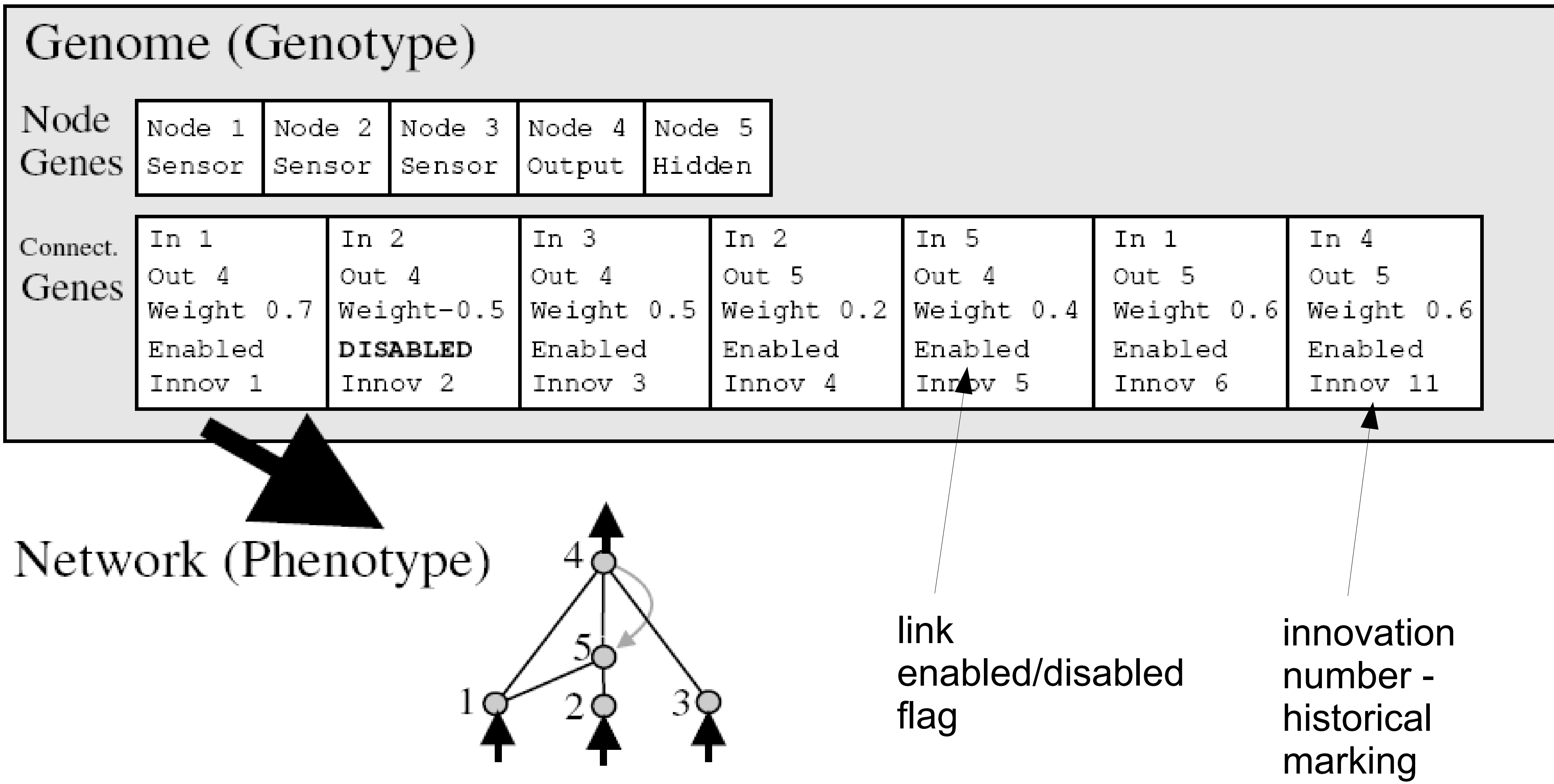
Note, some newer implementations use pruning However, it is not essential.

Minimal Substrate

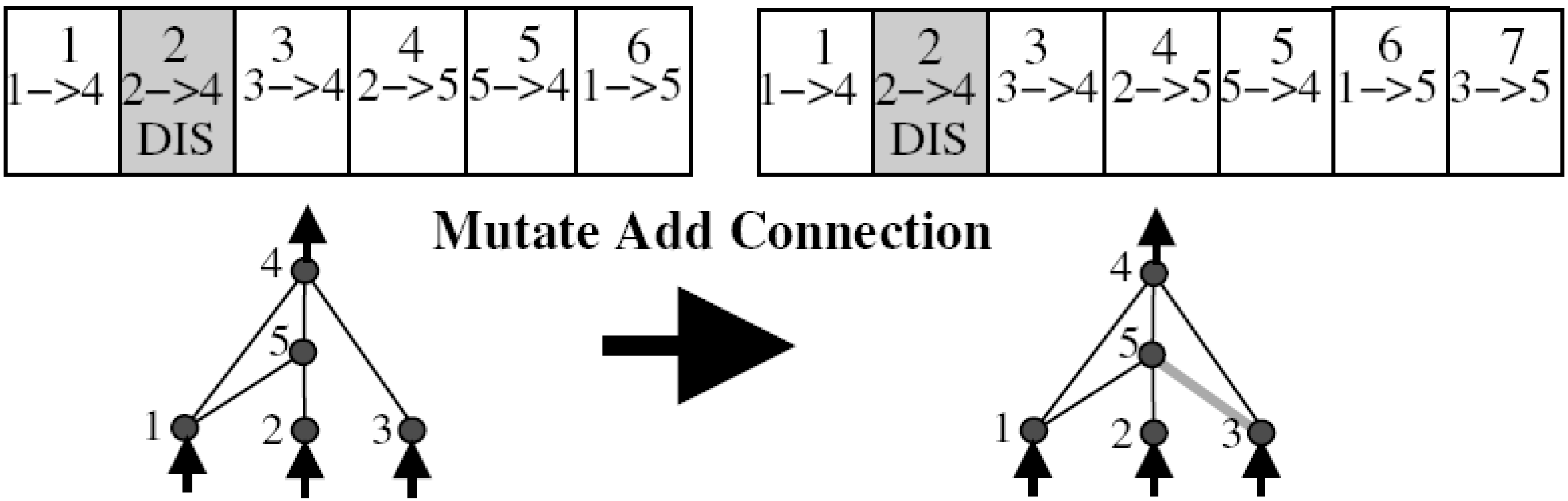
- Initial population is formed of the simplest topologies: fully connected feed-forward networks without hidden layers: the minimal substrate.



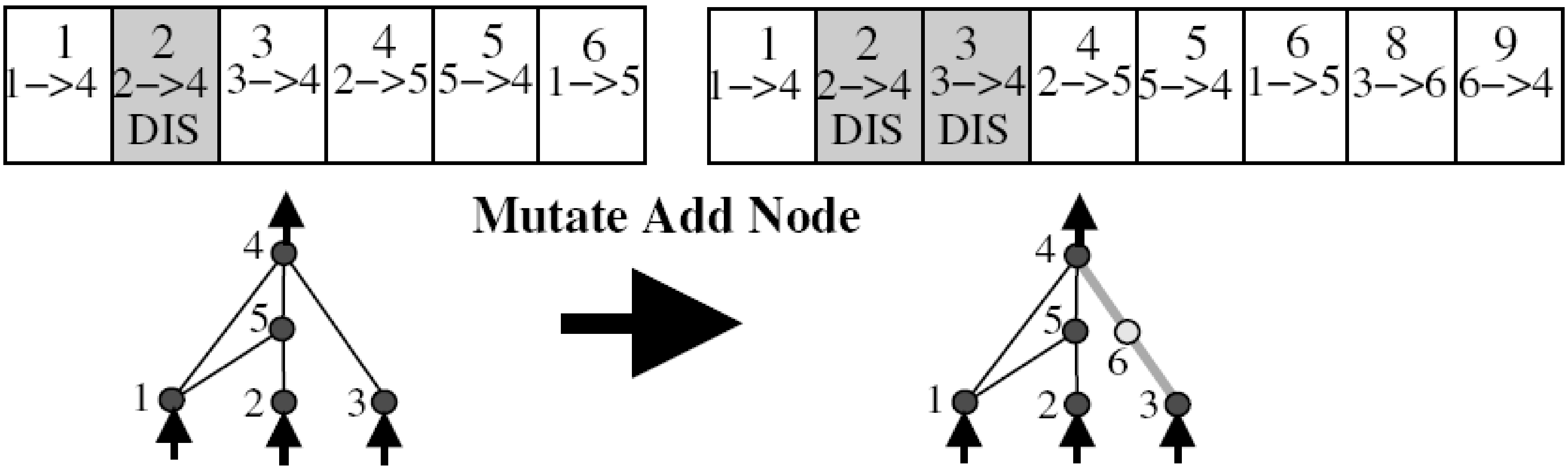
NEAT Genome


Genome (Genotype)
Node Genes
Node 1 Sensor
Node 2 Sensor
Node 3 Sensor
Node 4 Output
Node 5 Hidden
Connect. Genes
In 1 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)
4
5
1
2
3
link enabled/disabled flag
innovation number - historical marking

Add Link Mutation



Add Neuron Mutation



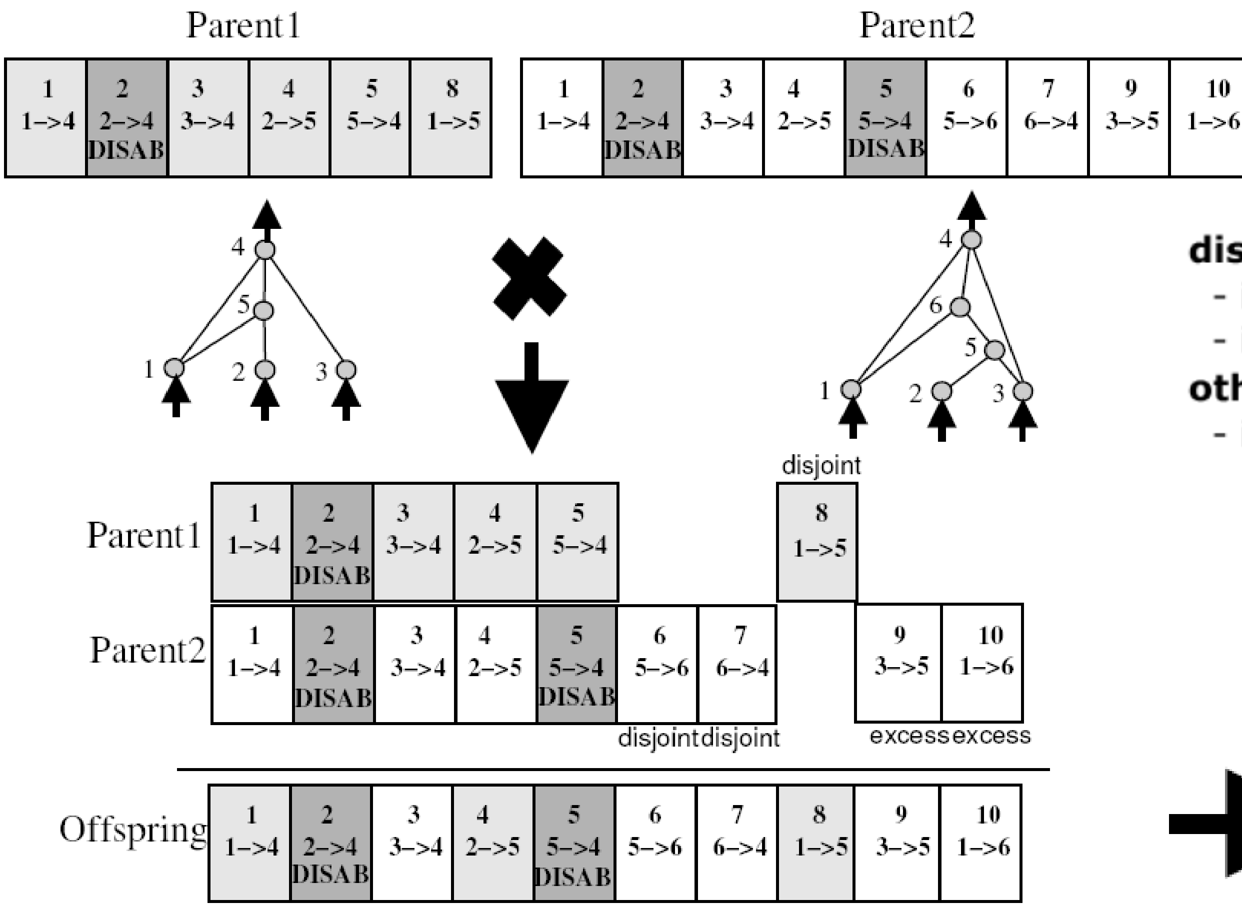
The weights of new neuron's incoming/outgoing links are set in a way which minimizes the difference between original and mutated networks.

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 (i.e. adding a link between neurons #3 and #4).

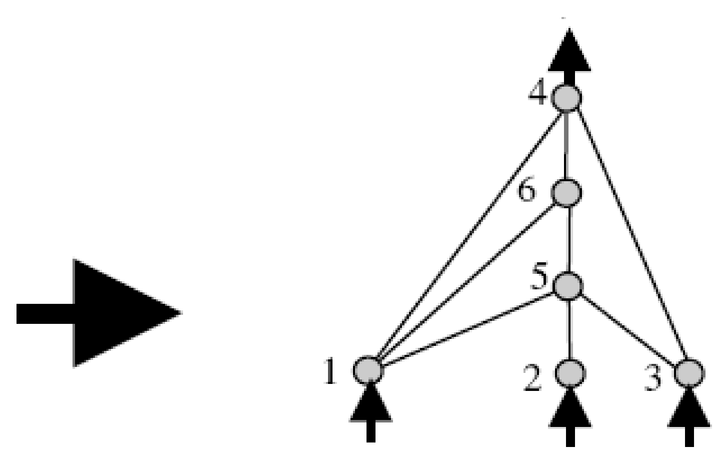
Mating

Let's use historical marking.

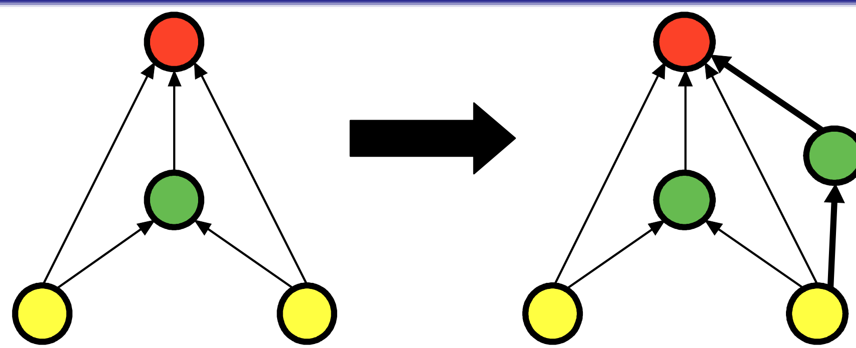


disjoint or **excess**
- inherit more fit
- if equal fitness inherit randomly

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

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

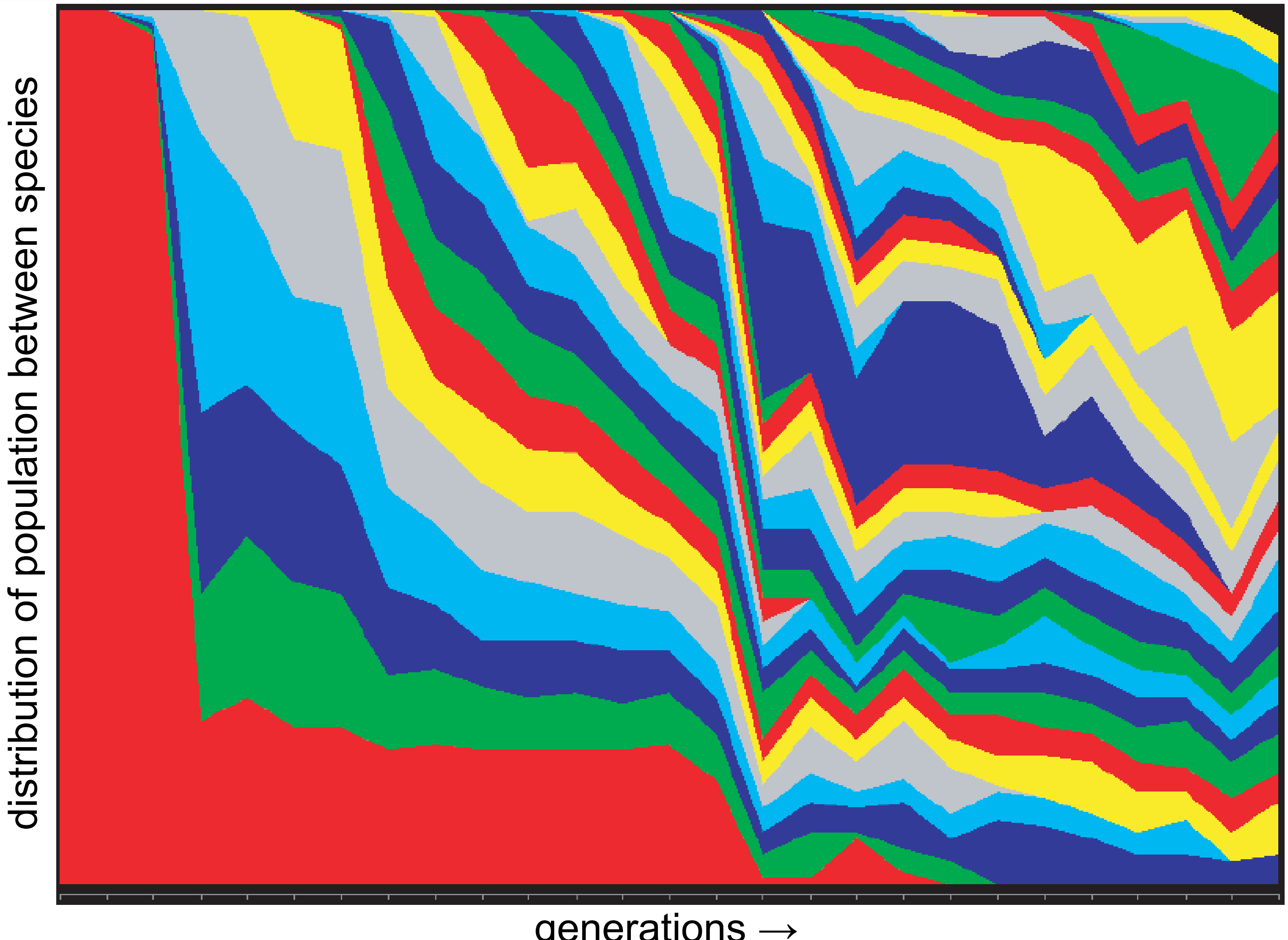
- Using historical markings again.
- *E* … # of excess genes,
- *D* … # of disjoint genes,
- *W* … averaged difference of matching weights,
- *N* … the length of the longer genome,
- c_1, c_2, c_3 … balancing constants.

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.

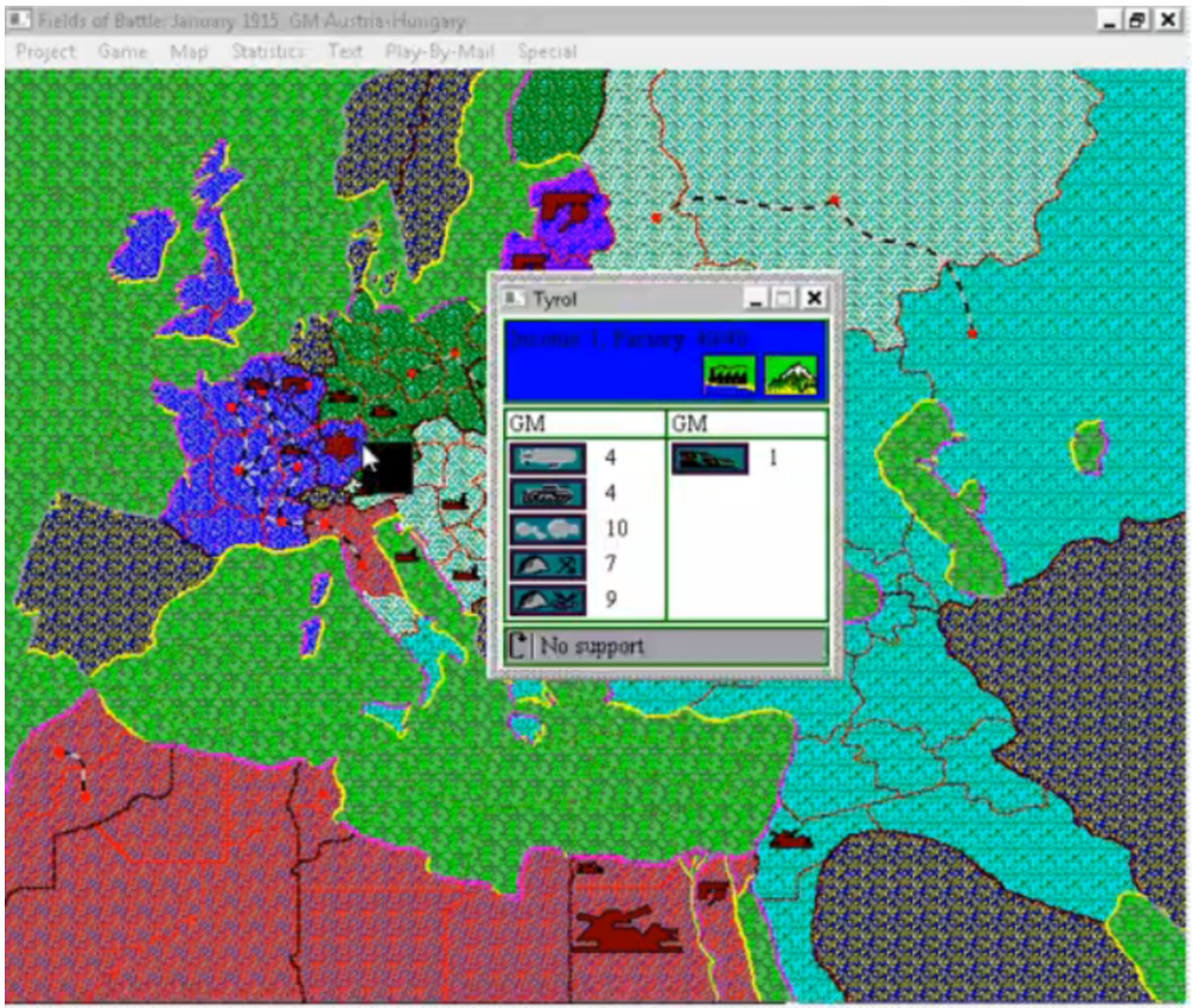
The Three Most Important Ideas Behind NEAT

- **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.

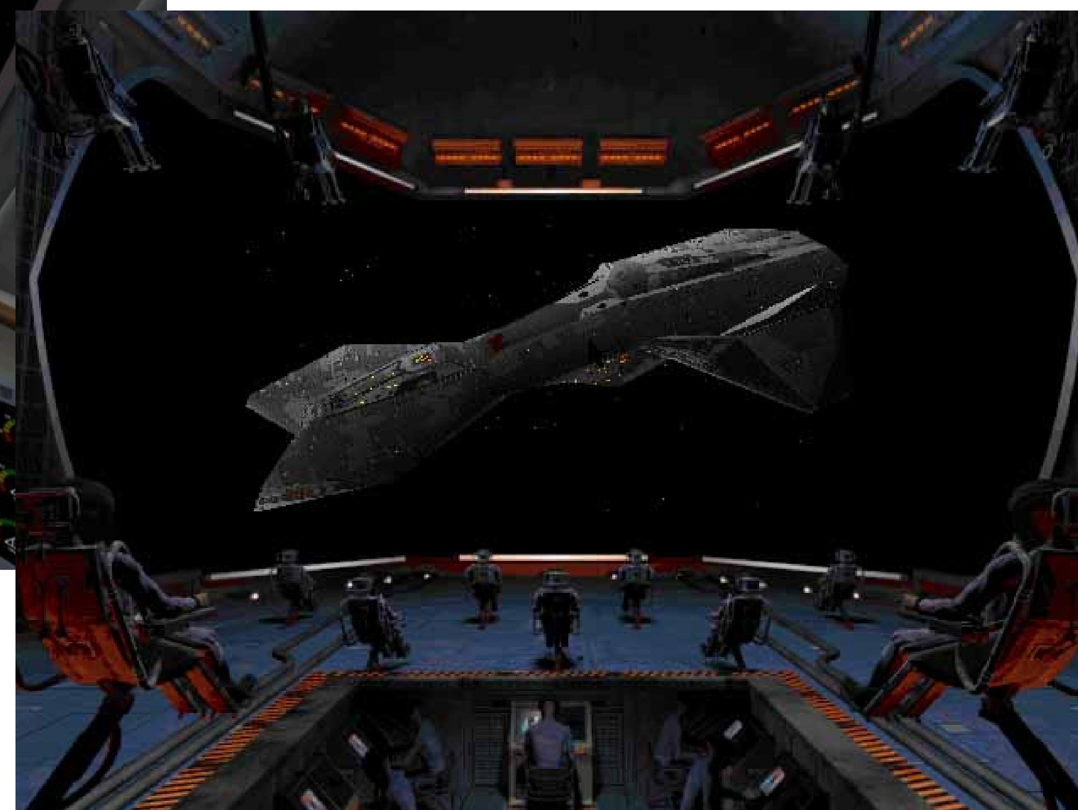
ANNs and Game AI

- AI opponents.
- Modeling player behaviour.
- Strategy estimation.
- Realistic motion.

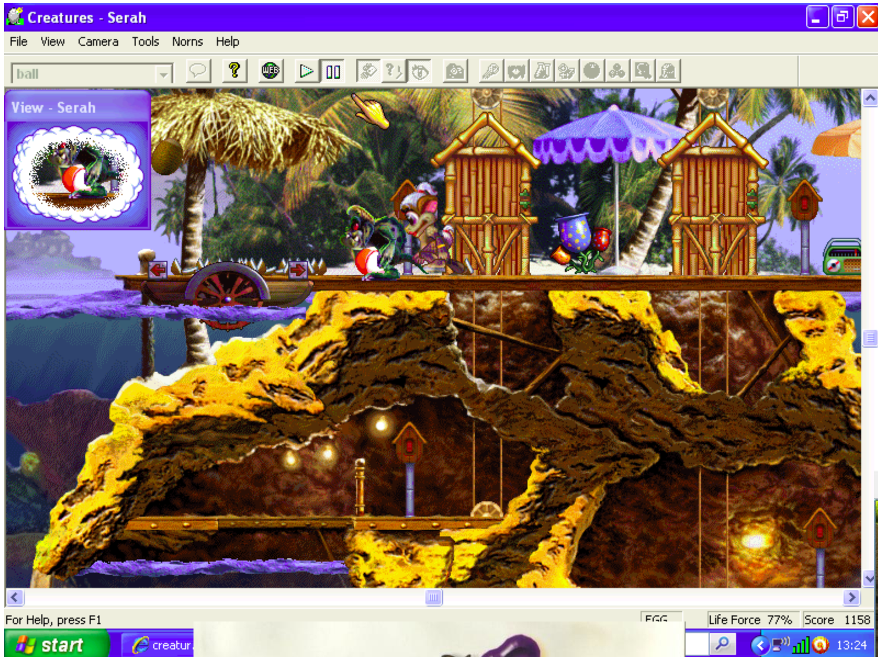
Fields of Battle (1995)

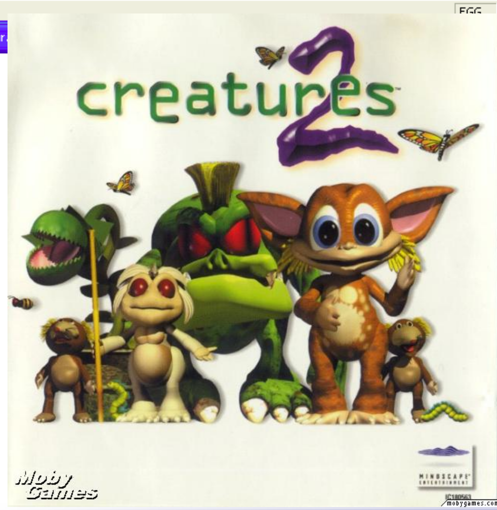


Battlecruiser 3000AD (1996)



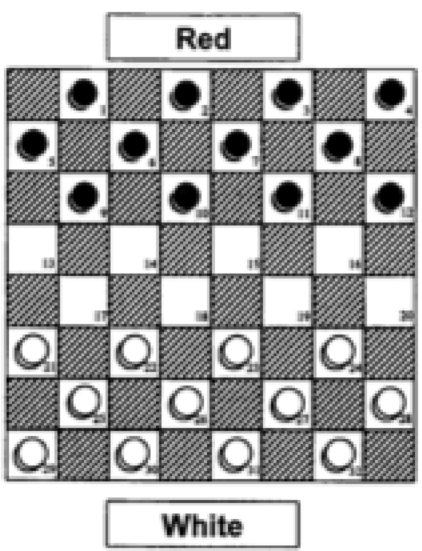
Creatures (1996-)

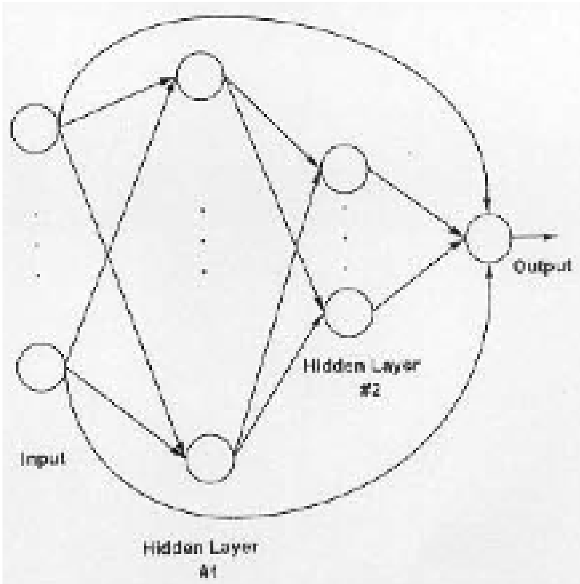




Blondie 24 Chess (1999)

- David B. Fogel.
- Combination of minmax. & neuroevolution.
- Coevolution.
- Defeated 99.61% of 165 online players.





Colin McRae Rally 2.0 (2001)

- Opponents AI.
- MLPs, RPROP learning.
- ANN driving model follows optimal track.
- Different models for different cars and road conditions
- Different networks for steering and speed control.

http://www.ai-junkie.com/misc/hannan/hannan.html

Black & White (2001)

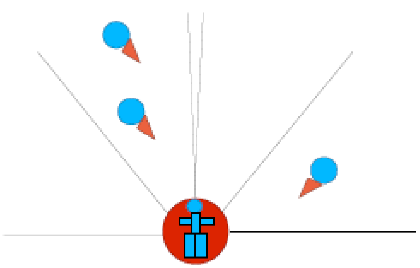
Forza Motorsport 2 (2007)

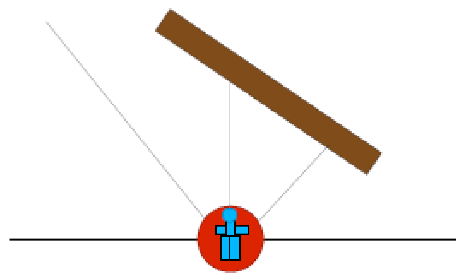
- Opponent AI.

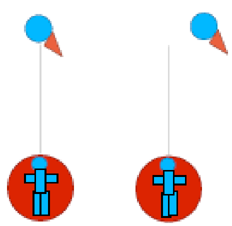
NERO

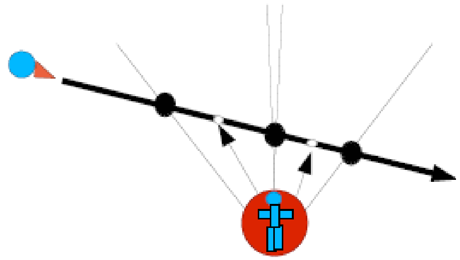
- NeuroEvolving Robotic Operatives.
- *Evolve your own robot army by tuning their artificial brains for challenging tasks, then pit them against your friends' teams in online competitions!*
- http://www.nerogame.org/

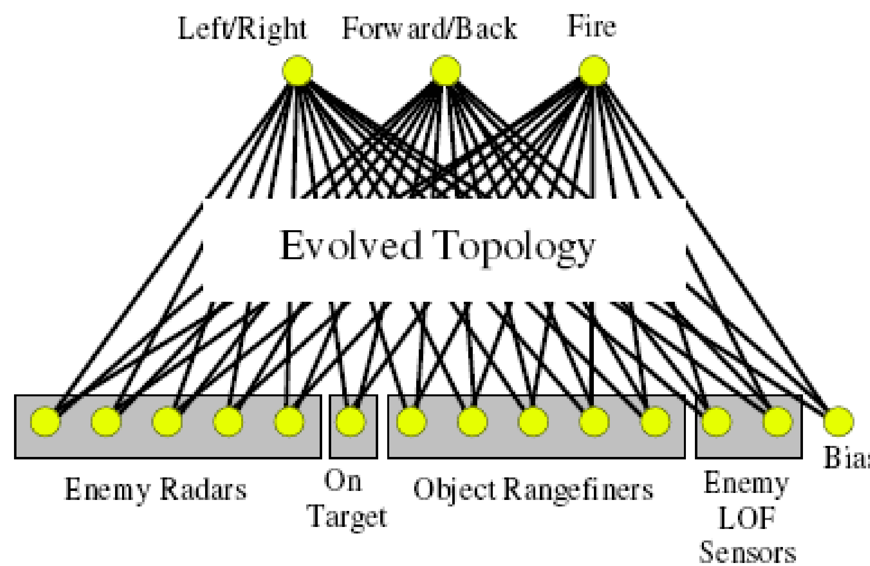
NERO: Robots


(a) Enemy Radars

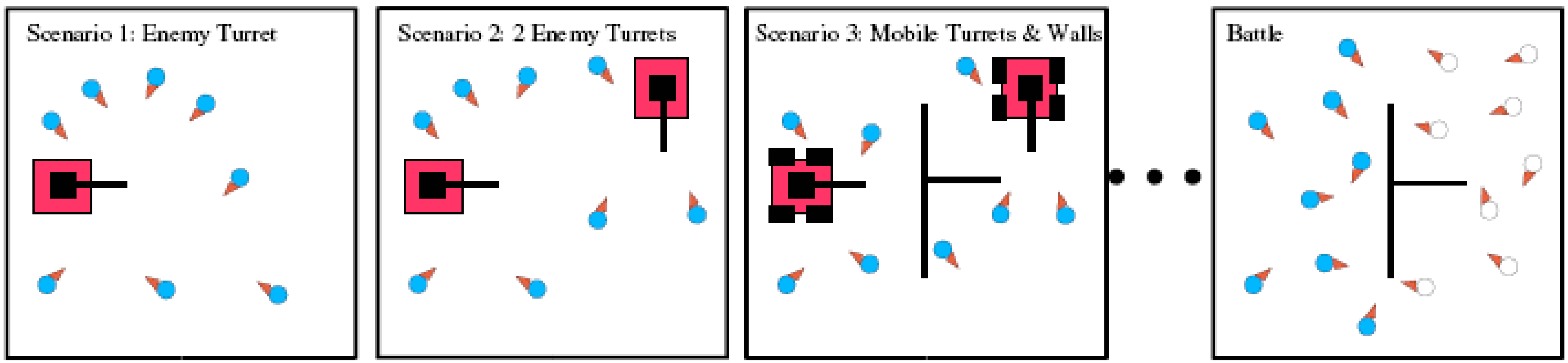

(b) Rangefinders

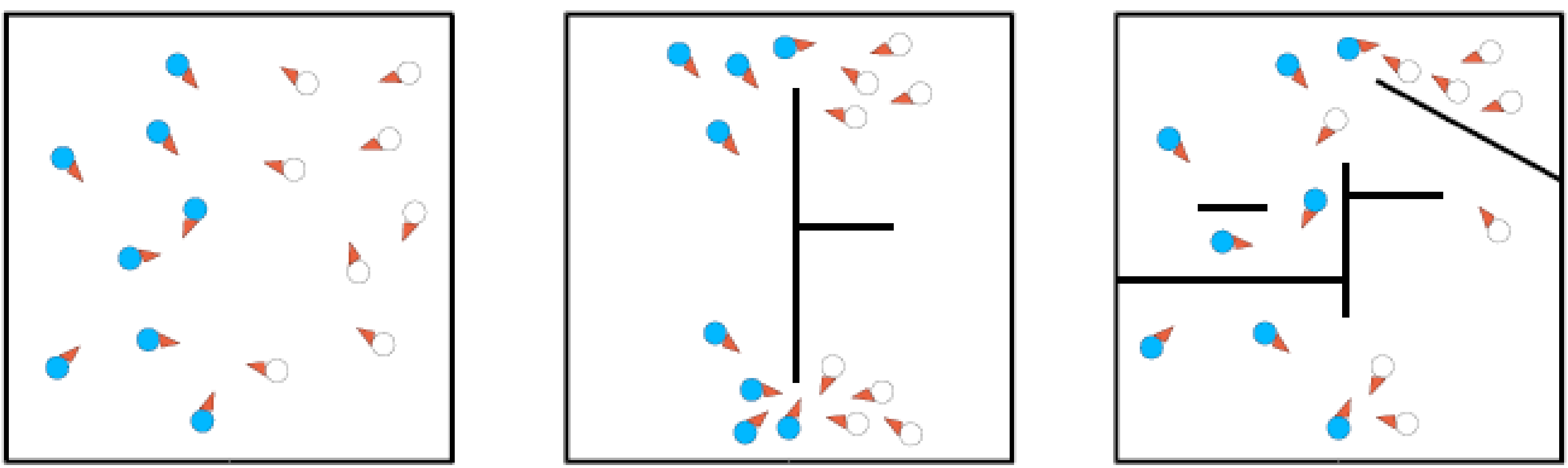

(c) On-Target Sensor


(d) Line-of-fire sensors



NERO: Arena & Task Complexification

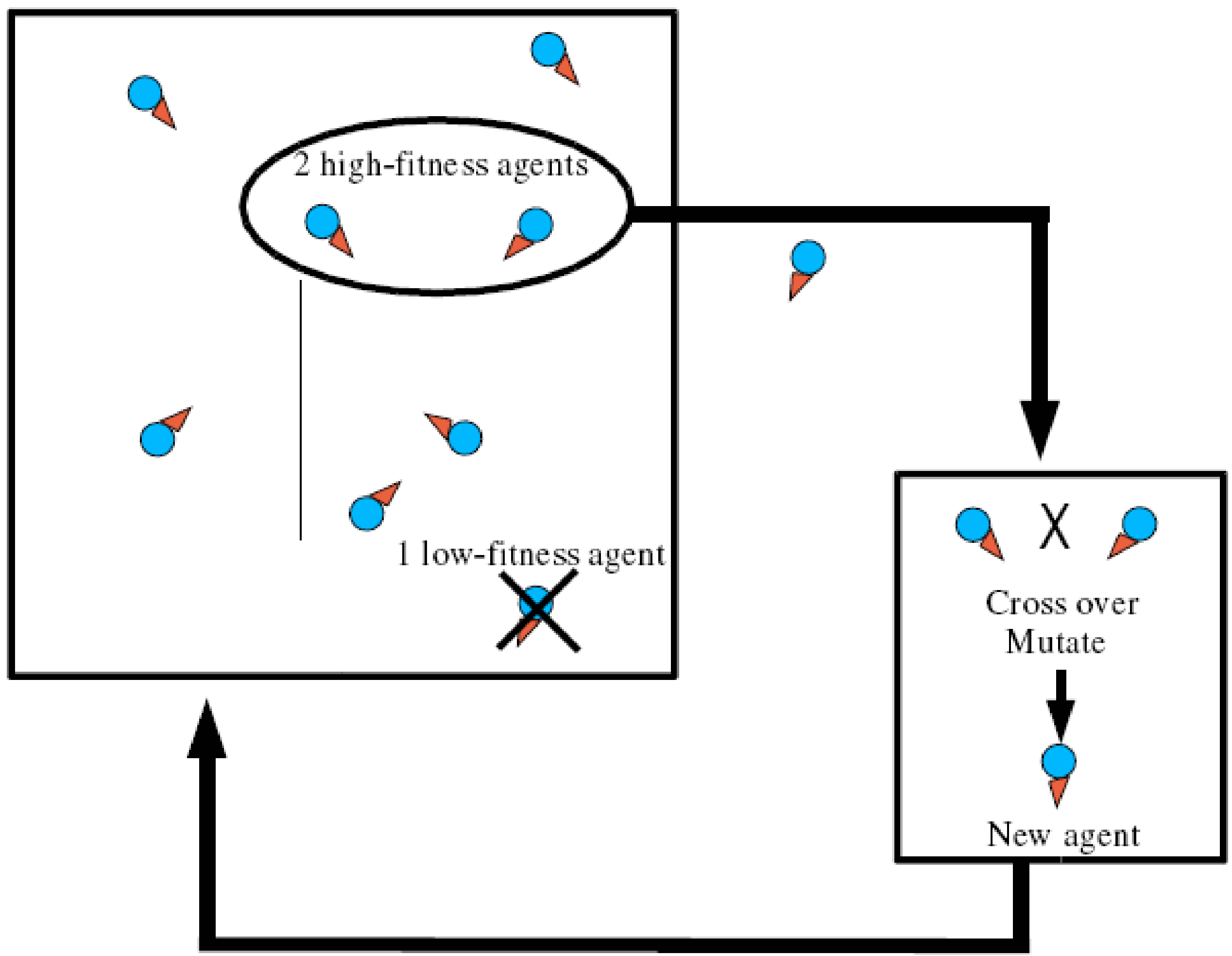




NERO: Fitness

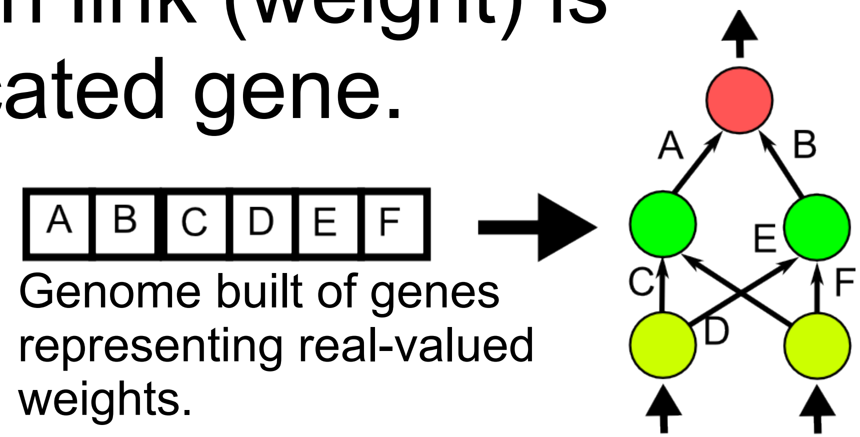
- No separation between generations – robots are bred and die continuously.
- Fitness computed also continuously → based on fulfilling tasks (which can by dynamically changed).
- User interactively influences the shape of the fitness function (i.e. prefer attackers/defenders): **interactive evolution**.

RTNEAT (RealTime NEAT)



Direct vs. Indirect Encodings

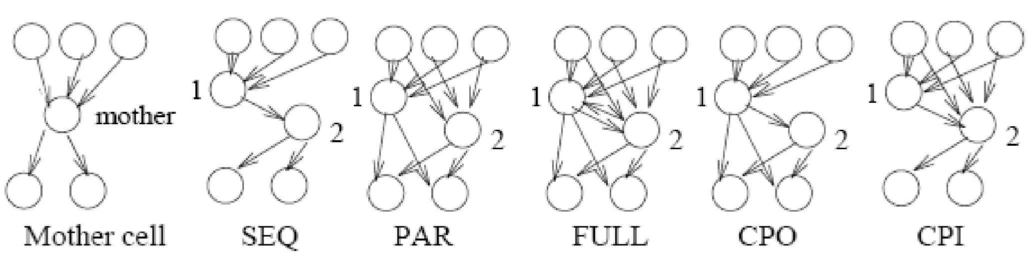
- Direct encoding → each link (weight) is represented by a dedicated gene.
- Not suitable for Large-scale ANN's.

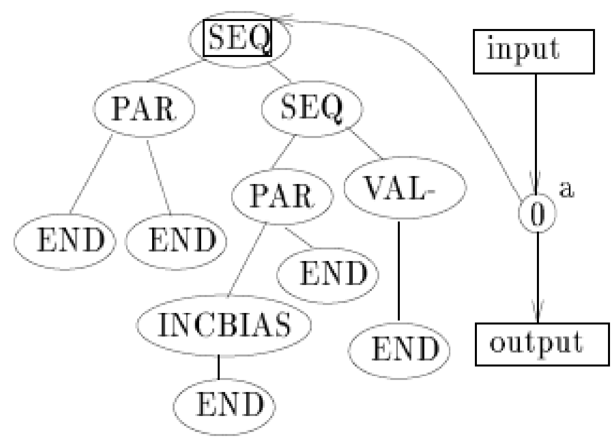


- Indirect encoding → developmental approaches:
  - Cellular encoding,
  - HyperNEAT/HyperGP.

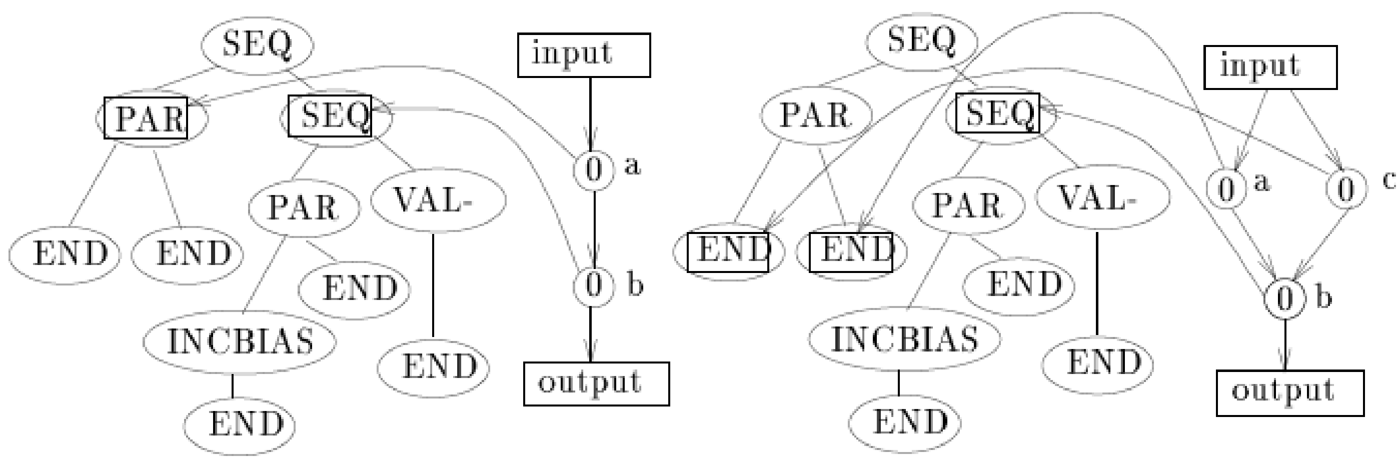
Cellular encoding (CE)

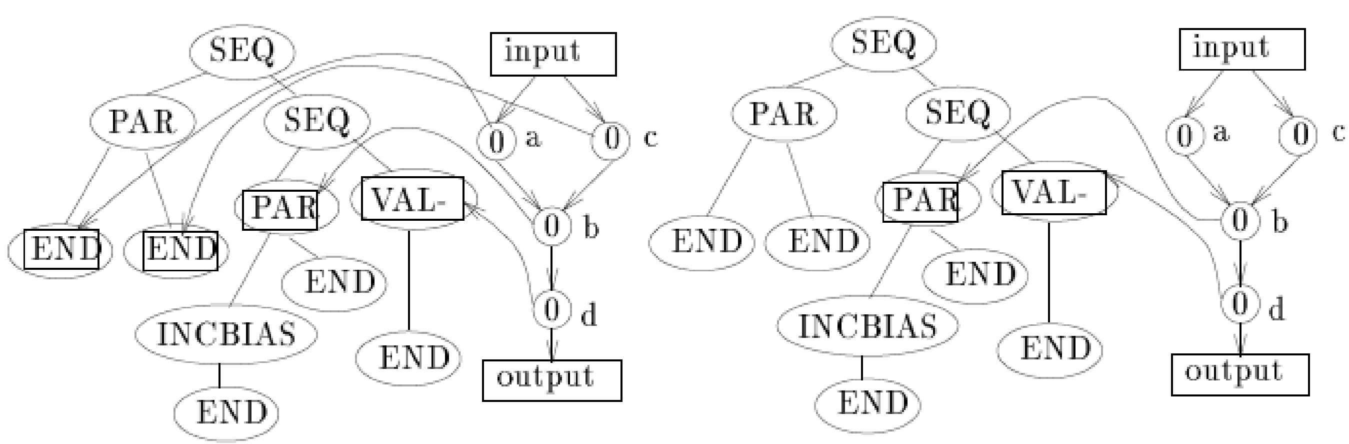
- 1993, Fréderic Gruau: indirect encoding example.
- Inspiration in embryo-genesis (cell division and differentiation). Cells → neurons.
- Program to “grow” ANN is represented by a tree (Genetic Programming).
- Operations: parallel/sequential divisions, connections change, change of weights/bias...





Cellular Encoding 2





Cellular Encoding 3

- May use operation which reads a sub-tree repeatedly → evolved a network representing parity of arbitrary number of inputs.
- Allows ANNs of arbitrary size: **neural module reuse**.

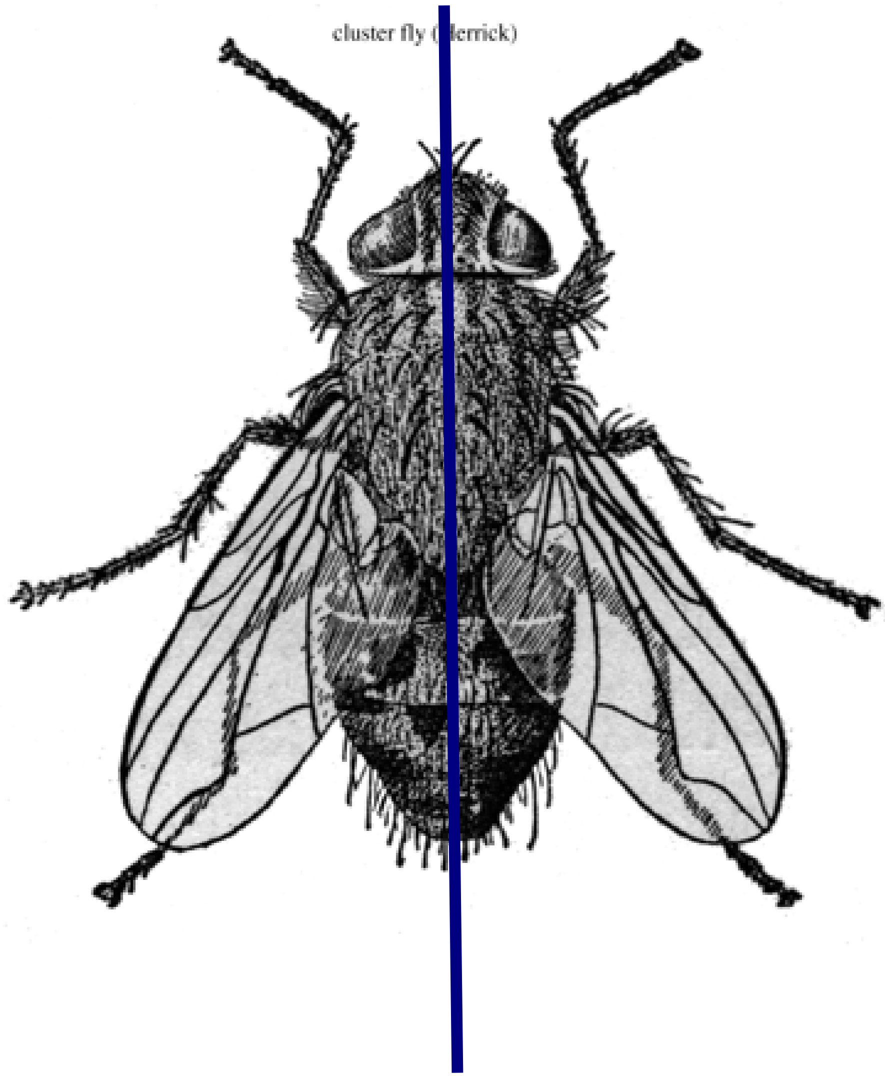
Evolving Large-scale ANNs

- 1000+ neurons (& corresponding # of links).
- Most optimization methods fail → the curse of dimensionality.
- Modularity, regular patterns.
- Why to do that?
  - complex models,
  - ability to process huge amount of input/outputs, without hand-coding features (i.e. pattern recognition)...
- HyperNEAT can do this...

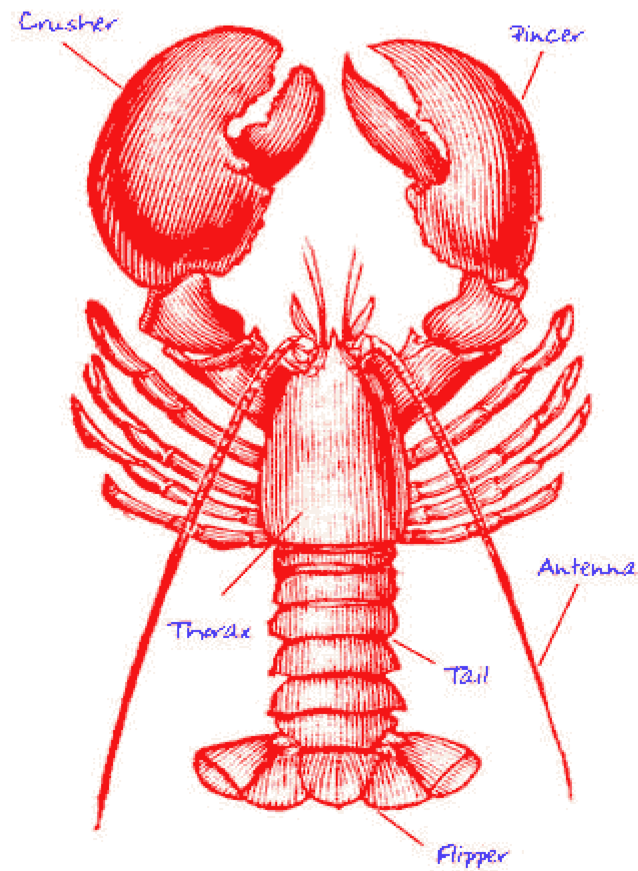
How does it work in nature?

- Human genome -> 30 000 genes describing 100 billion neurons each linked to as many as 10 000 others (plus the rest of organism!).
- We need some kind of compression.
- But we also need a regularity in compressed "data".
- Q: What are the regularities found in living organisms?

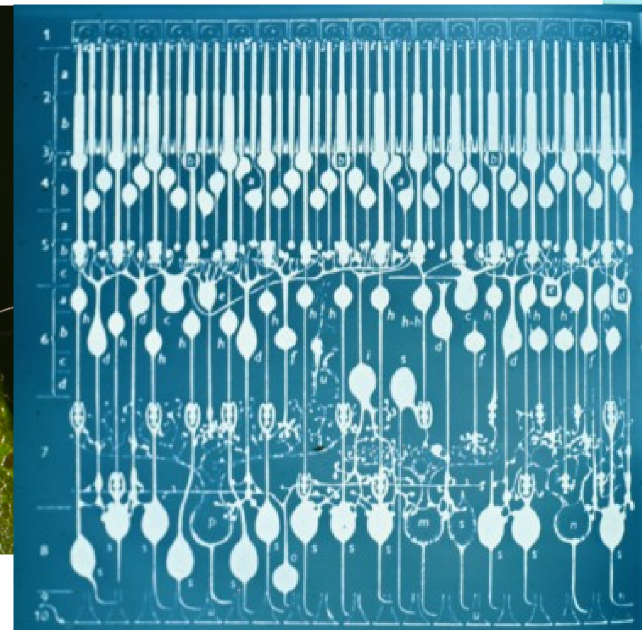
Symmetry



Imperfect Symmetry



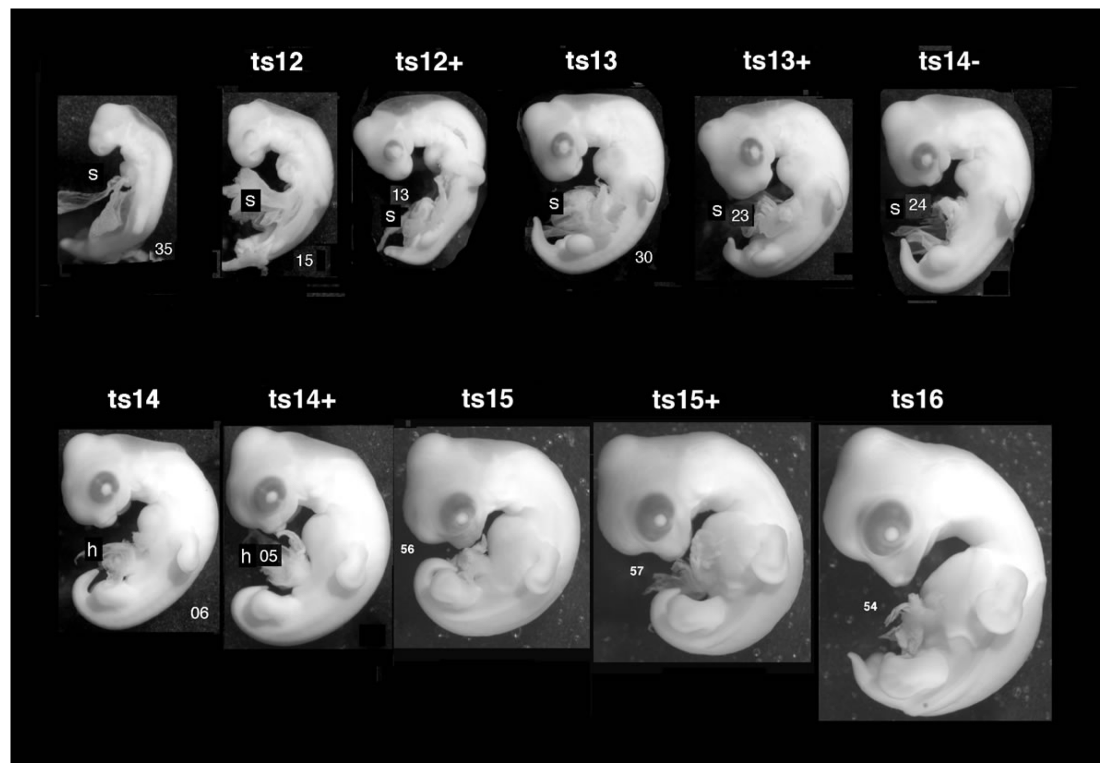
Repetition with Variation



- Note that all these regularities happen at all scales of an organism.

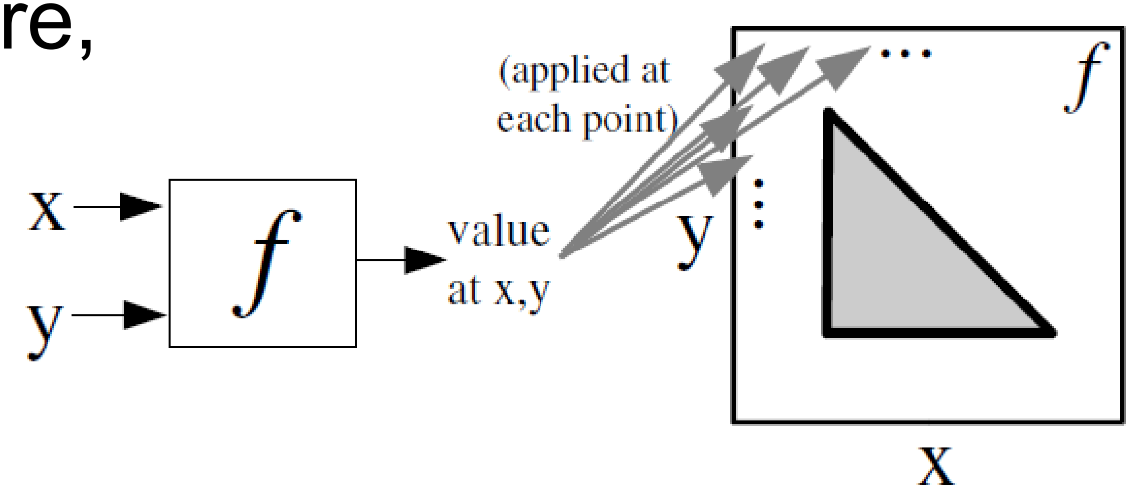
How are organisms built?

- Development from a single cell (zygote).
- Evolutionary Development “Evo-Devo”.



Compositional Pattern Producing Networks (CPPNs)

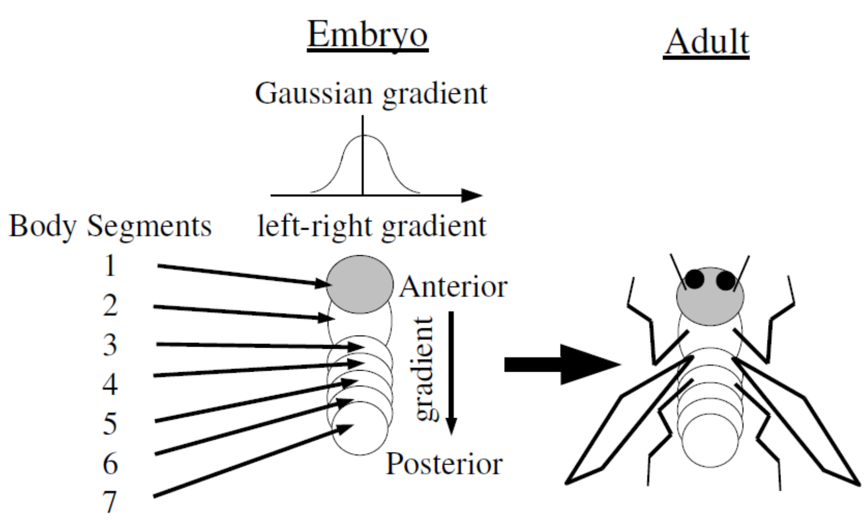
- Stanley 2006
- Can we create such regular patterns without development in time?
- We can ask a special function called CPPN, where the cells are, using absolute coordinates.

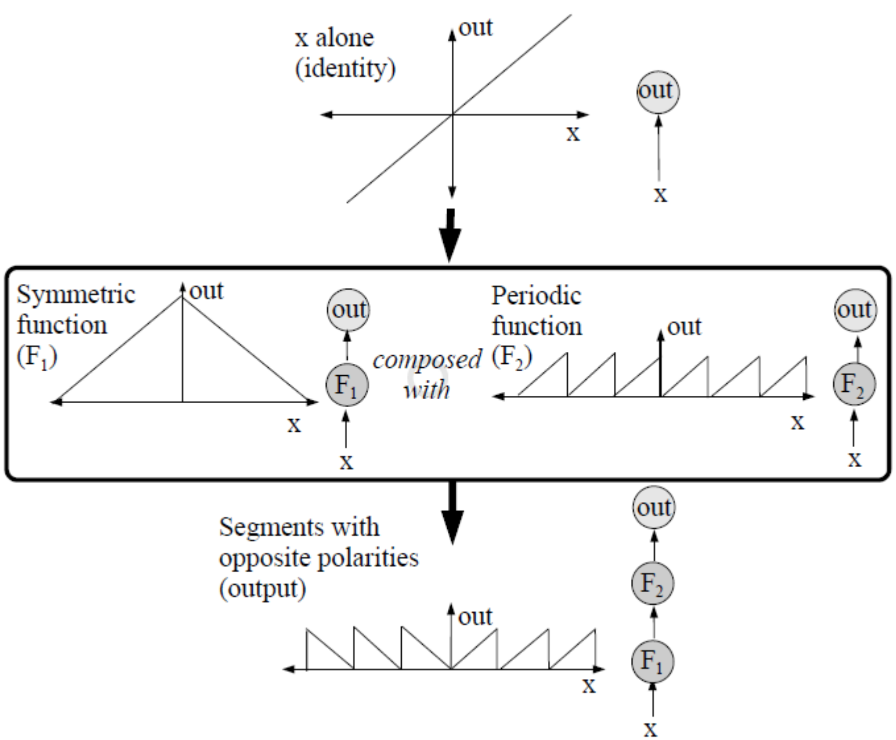


Kenneth O. Stanley (2006):
Compositional Pattern Producing Networks: A Novel Abstraction of Development

Regularities by CPPNs

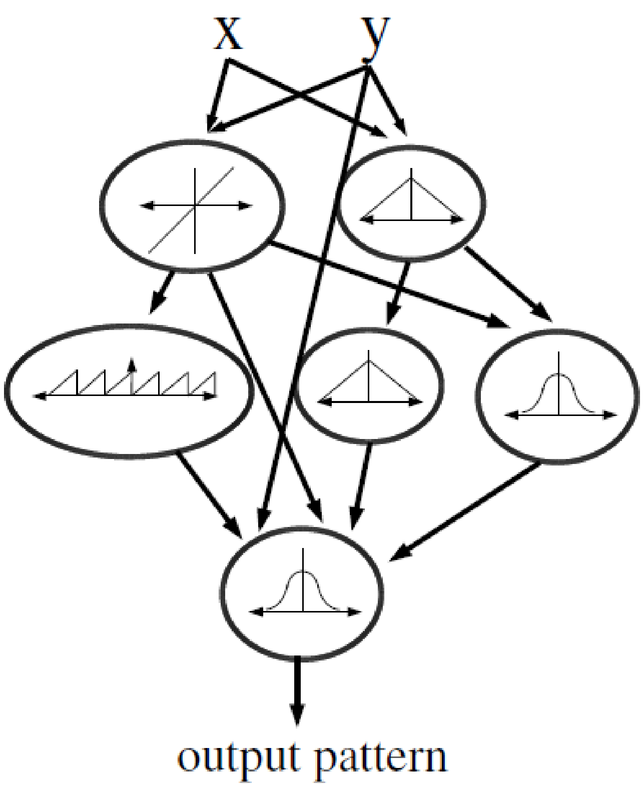
- Nature uses concentration gradients of regulatory proteins to determine position.
- CPPN is a composition of symmetric, periodic and other functions.





Regularities by CPPNs II

- Common CPPN building block functions:



Name	Equation
Bipolar Sigmoid	$\frac{2}{1+e^{-4.9\,x}} - 1$
Linear	x
Gaussian	$e^{-2.5\,x^2}$
Absolute value	$\lvert x \rvert$
Sine	$sin(x)$
Cosine	$cos(x)$

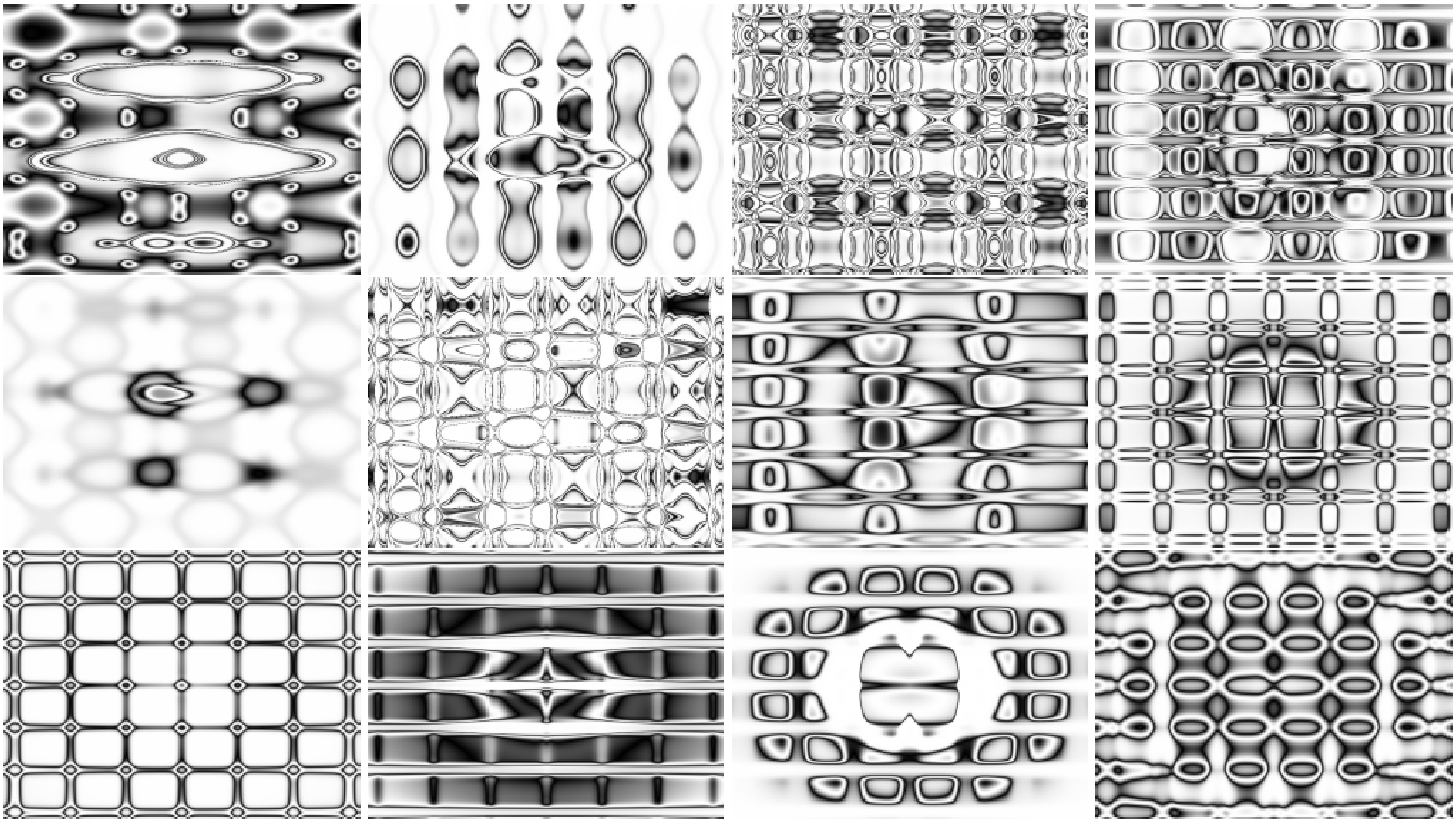
Picbreeder

- Interactive evolution of images.
- CPPN output: level of grey.
- CPPNs evolved using NEAT.
- http://picbreeder.org/

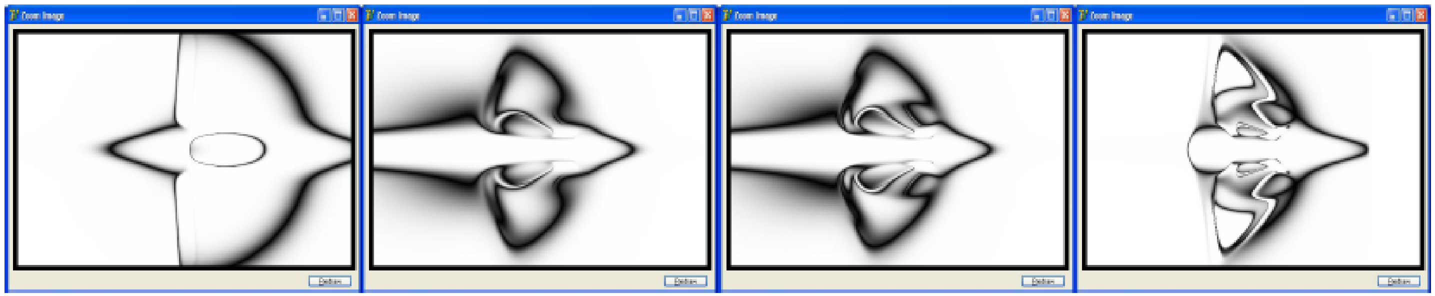
(a) Eye warped left (b) Symmetric eye (c) Eye warped right

K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 2007.

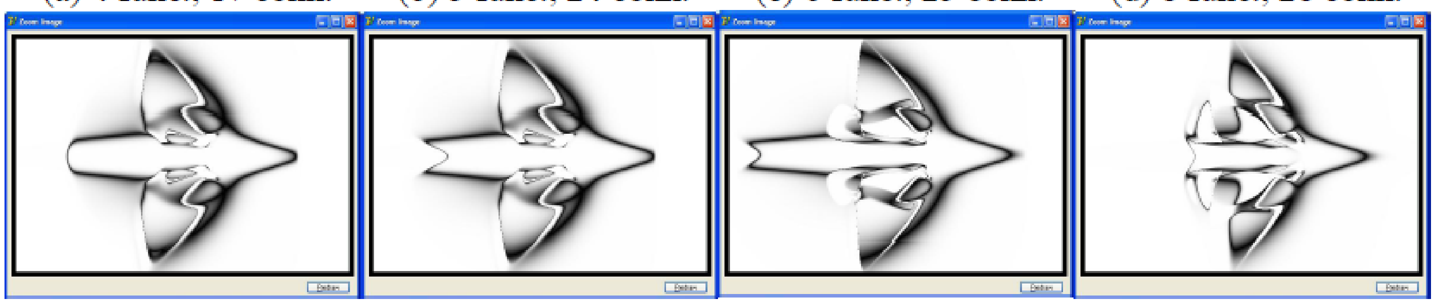
Picbreeder: Repetition with Variation



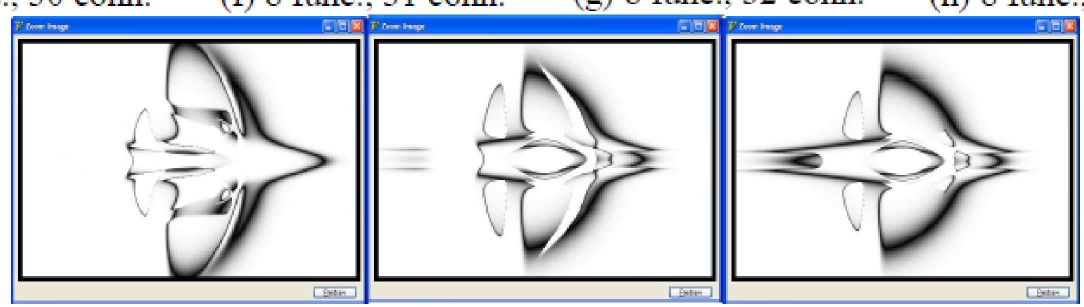
Picbreeder: Spaceship



(a) 4 func., 17 conn. (b) 5 func., 24 conn. (c) 6 func., 25 conn. (d) 8 func., 28 conn.



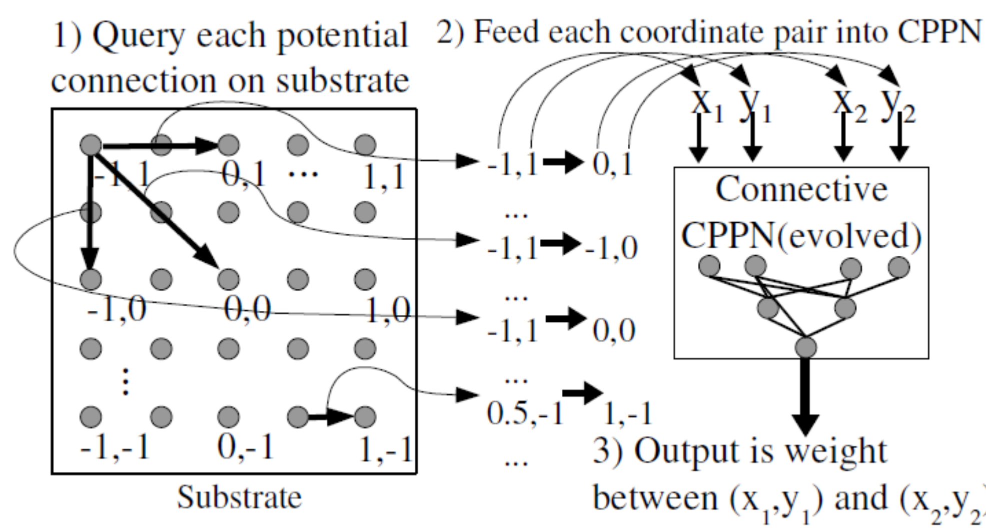
(e) 8 func., 30 conn. (f) 8 func., 31 conn. (g) 8 func., 32 conn. (h) 8 func., 34 conn.



(i) 8 func., 36 conn. (j) 9 func., 36 conn. (k) 9 func., 38 conn.

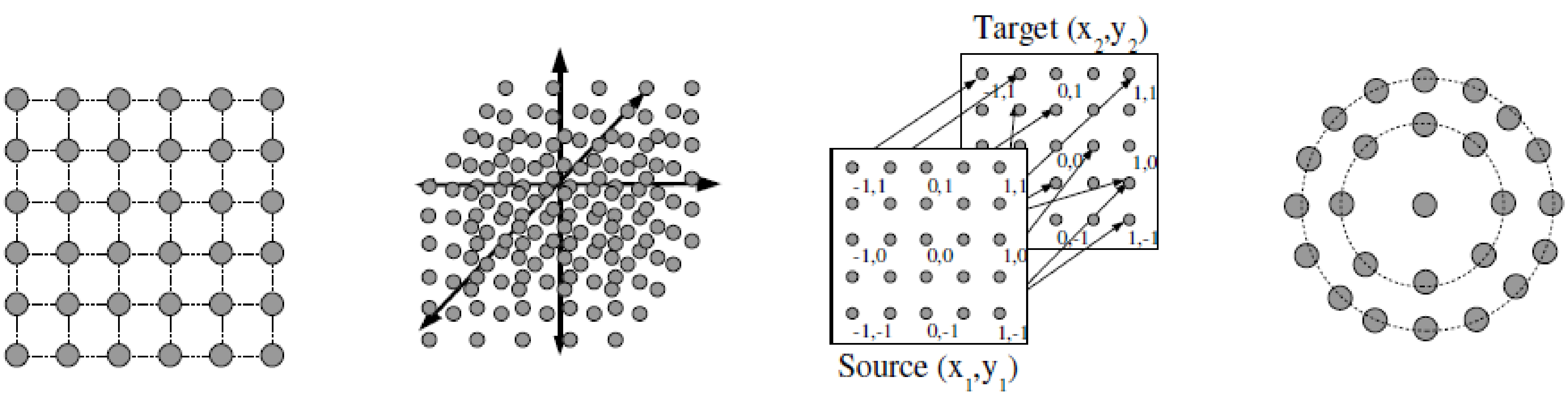
HyperNEAT

- Stanley 2007
- Uses CPPNs in a similar way to Picbreeder: creates Connectivity Patterns.
- Places neurons on a substrate assigning them spatial coordinates.
- CPPN takes coordinates of two neurons and computes the weight of a connection.



What Exactly is the Substrate?

- The list of neurons' coordinates along with possible connections between them.



(a) Grid (b) Three-dimensional (c) Sandwich (d) Circular

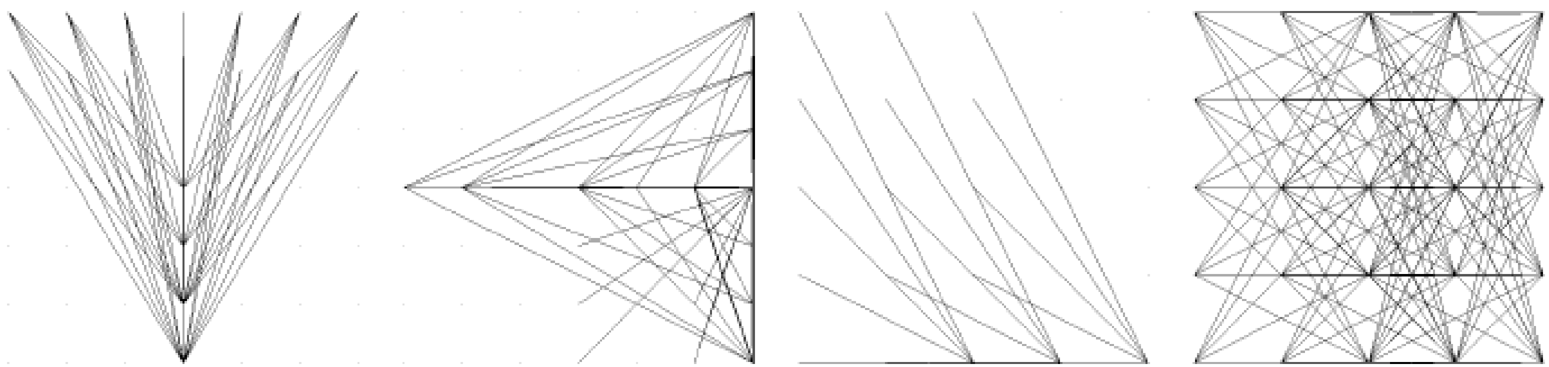
Create or not to Create a Link?

- Substrates are often fully connected → lots of links → computationally infeasible → pruning is used.
- If CPPN outputs weights in range *[-3; 3]* then
  - links with weights < *0.2* are not expressed,
  - >= *0.2* are scaled to magnitude between *0* and *3*.

→ when using this approach the final ANN is a sub-graph of a substrate.

Connectivity Patterns by HyperNEAT

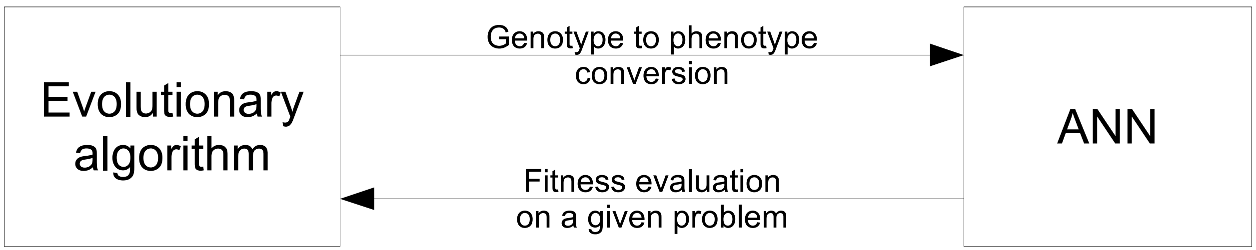
- Patterns evolved using interactive evolution:



(a) Sym. (b) Imperf. (c) Repet. (d) Var.

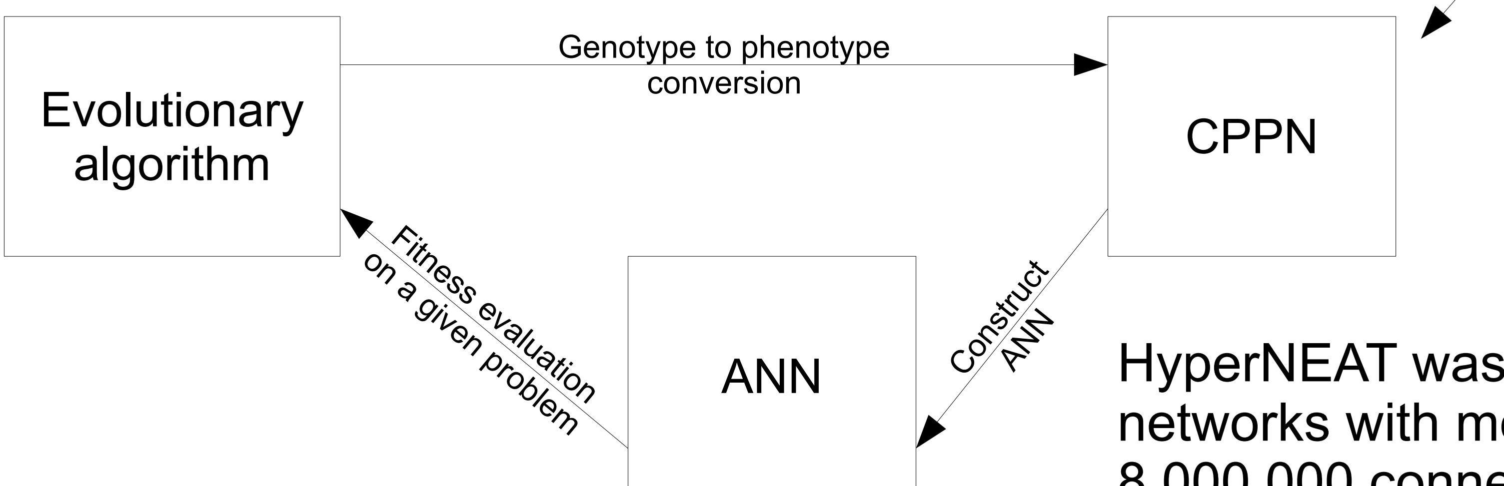
HyperNEAT vs. Standard Evolution of ANNs

Common approach: Evolution of ANNs



A special network, which can represent the regularities efficiently. It constructs the final network.

HyperNEAT



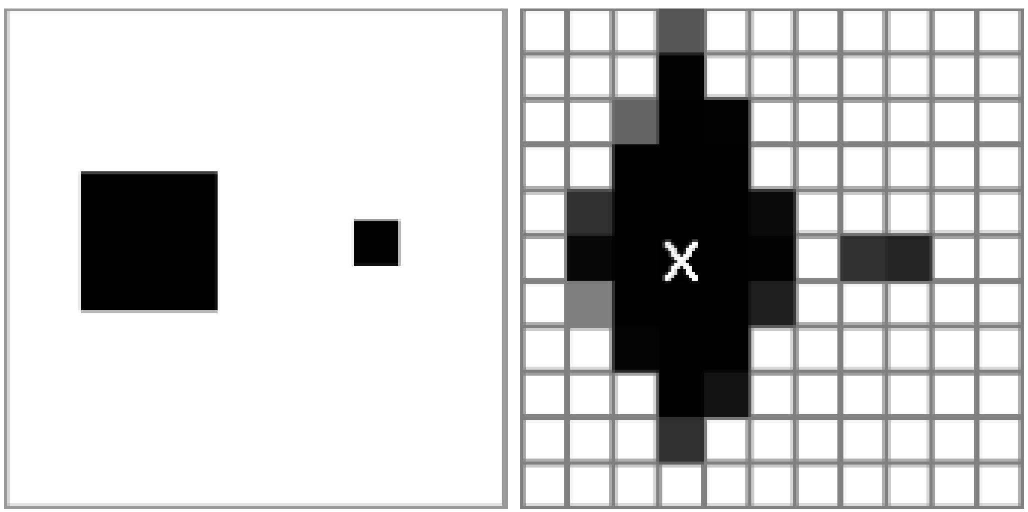
HyperNEAT was used to build networks with more then 8 000 000 connections!

Spatial Representation

- HyperNEAT exploits spatial representation of a problem. The same happens in the nature:
  - connection of eyes to brain hemispheres,
  - similar things processed nearby.
- We have to assign coordinates. Does every problem have a reasonable spatial representation?
  - It seams that most problems have.
  The others would not probably benefit from regularities in ANNs.

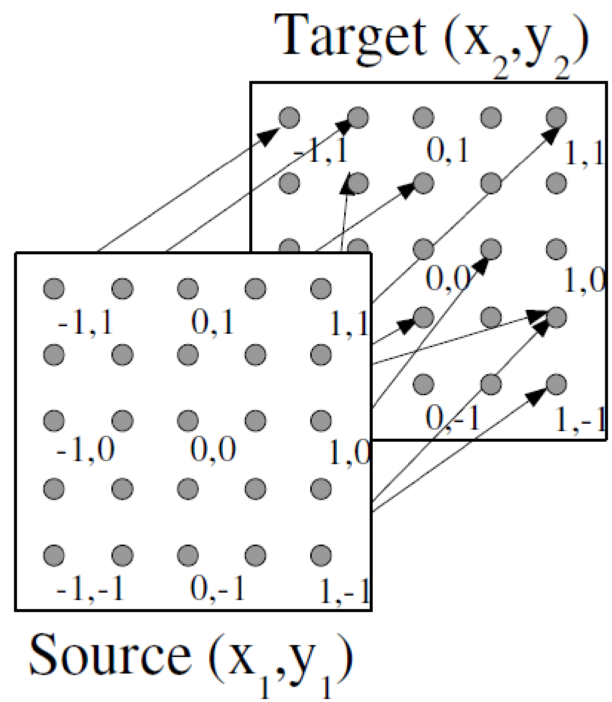
Object Targeting with HyperNEAT

- Visual targeting: distinguish the larger object.
- “Sandwich substrate”.



(a) Sensor Field (object placement)

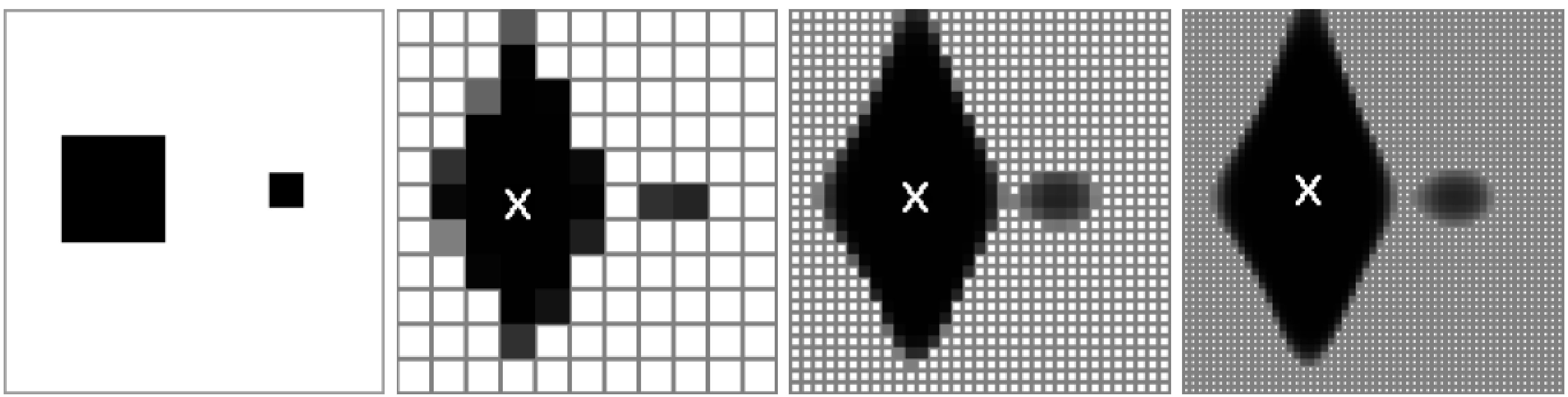
(b) 11x11 Target Field (12,827 connections)



Jason J. Gauci and Kenneth O. Stanley (2007):
Generating Large-Scale Neural Networks Through Discovering Geometric Regularities

Object Targeting II: Scaling the Substrate

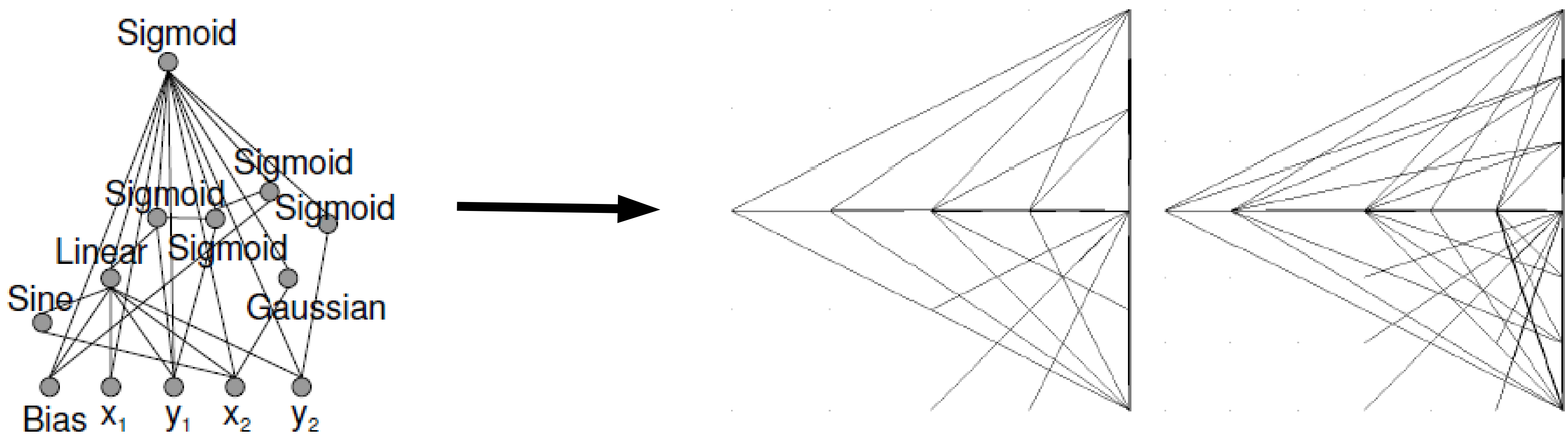
- The substrate density can be scaled.
- The function of final ANN is approximately preserved.
- We can train on small → get large.



(a) Sensor Field (object placement) (b) 11x11 Target Field (12,827 connections) (c) 33x33 Target Field (1,033,603 connections) (d) 55x55 Target Field (7,970,395 connections)

Object Targeting III: Scaling the Substrate

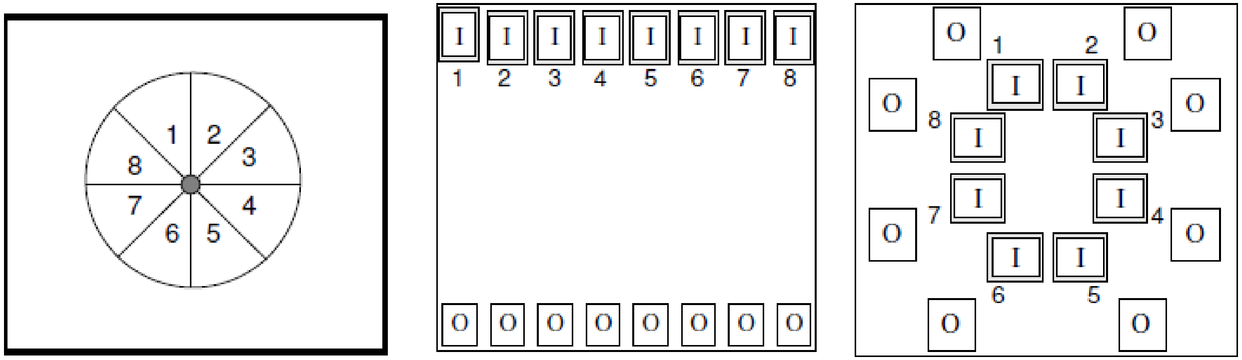
- An equivalent connectivity concept at different substrate resolutions.



(a) 5×5 Concept (b) 7×7 Concept

Food Gathering Problem

- Range-finder sensors detect food.
- More food eaten → higher fitness.
- Experiments with different sensor/effector placement – exploiting geometric relationships with "outer world".



(a) Robot (b) Parallel (c) Concentric

David B. D'Ambrosio and Kenneth O. Stanley (2007)
A Novel Generative Encoding for Exploiting Neural Network Sensor and Output Geometry

Food Gathering Problem II

- Parallel worked better than Concentric because less computation is needed for CPPN.
- New CPPN inputs added: the distances

  (x_1-x_2) and (y_1-y_2)

- When CPPN is provided the distances, both work the same.

Our work

- CPPF
- HyperGP
- RoboNEAT

Drchal, Koutník and Šnorek (2009):
HyperNEAT Controlled Robots Learn How to Drive on Roads in Simulated Environment

Buk, Koutník and Šnorek (2009):
NEAT in HyperNEAT Substituted with Genetic Programming

Drchal, Kapraľ, Koutník and Šnorek (2009):
Combining Multiple Inputs in HyperNEAT Mobile Agent Controller

HyperGP + Compositional Pattern Producing Functions

- CPPN = function represented by a network.
- **CPPF** (Compositional Pattern Producing Function) = general representation.
- **HyperGP** = NEAT replaced by Genetic Programming (GP):
  - faster than HyperNEAT
  - nodes:

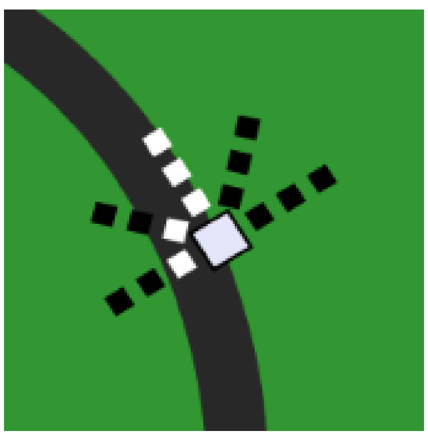
$$x+y,\ x-y,\ xy,\ \sin(x),\ \cos(x),\ \arctan(x),\ \sqrt{|x|},\ |x|,\ e^{-x^2},\ e^{-(x-y)^2}$$

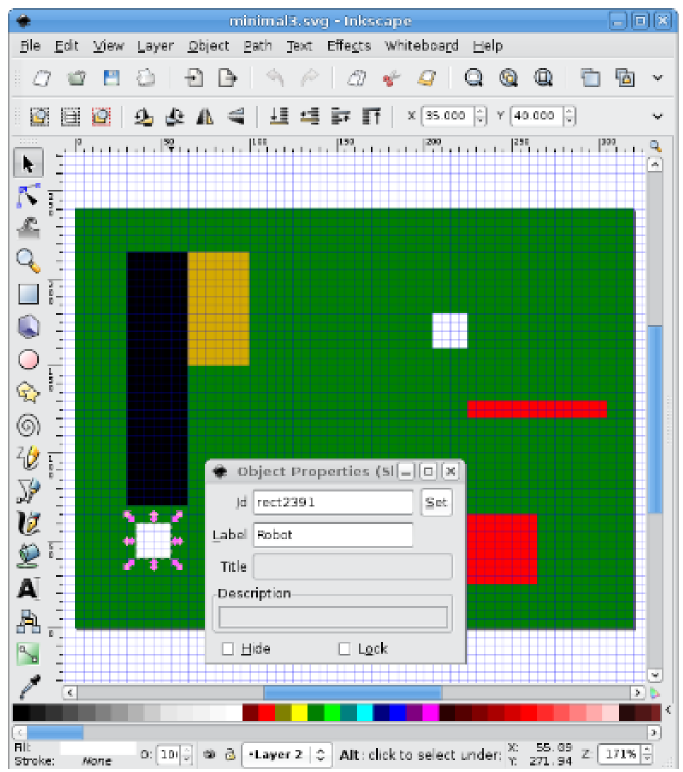
  - atoms:

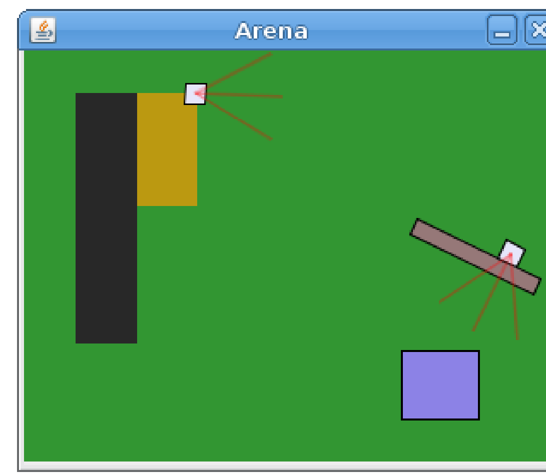
$$x_1,\ x_2,\ y_1,\ y_2,\ random(-5{,}5)$$

RoboNEAT

- HyperNEAT/HyperGP for robot control.
- ViVAE Simulated 2D environment with rigid body physics.

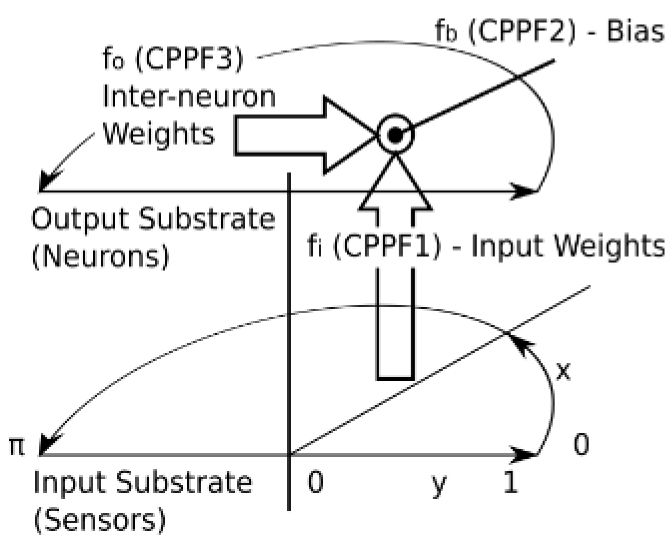


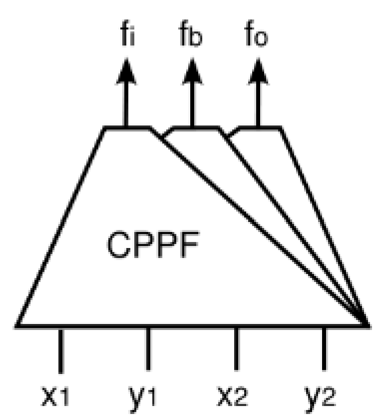


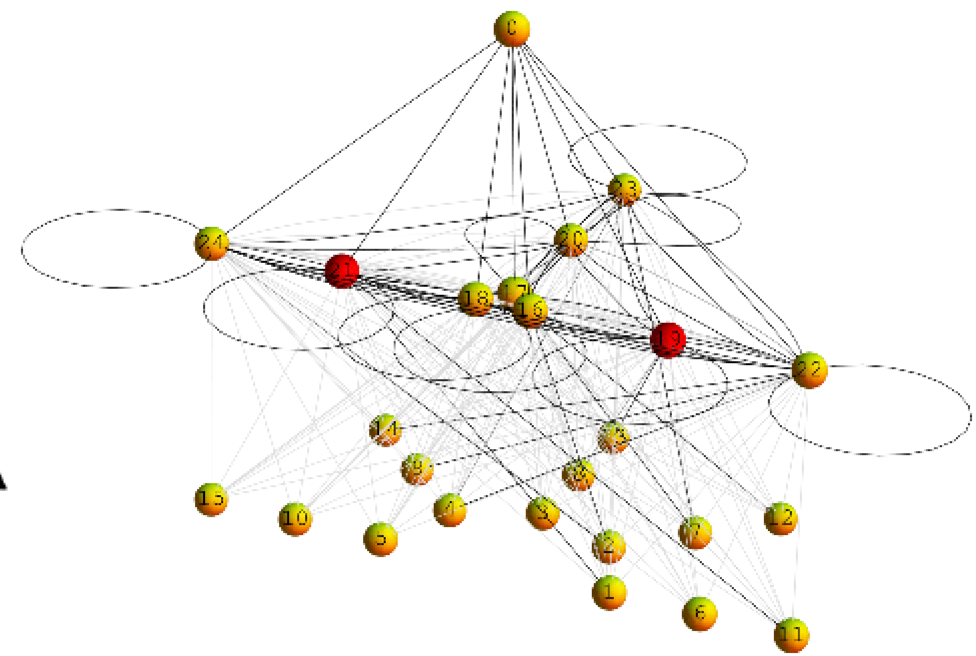


RoboNEAT II

- Substrate uses polar coordinates.
- Input + 1 fully recurrent layer
- See VIDEO...



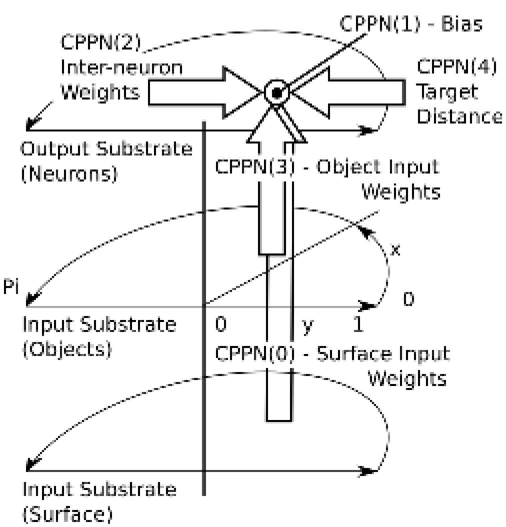


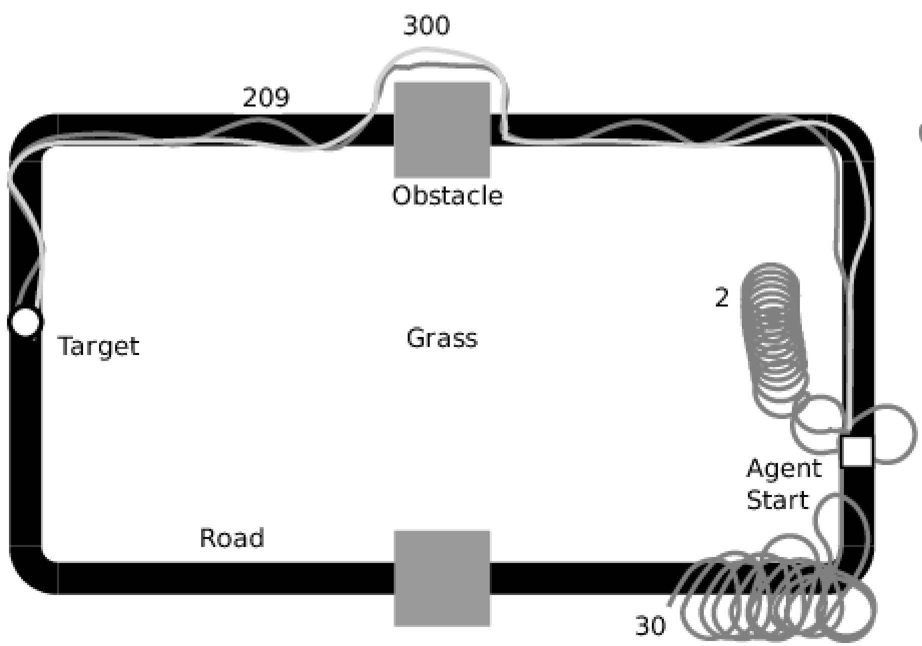


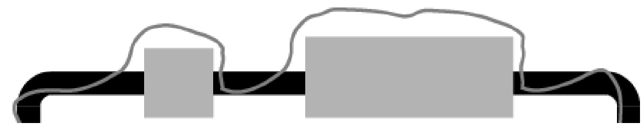
Jan Drchal, drchajan@fel.cvut.cz, http://cig.felk.cvut.cz

RoboNEAT III

- Obstacle avoidance.
- Object sensors added (two input layers)







GENERALIZATION

$$f = \frac{distanceTravelled}{simulationSteps + 1}\left(1 - \frac{targetDistance}{initialDistance}\right)$$

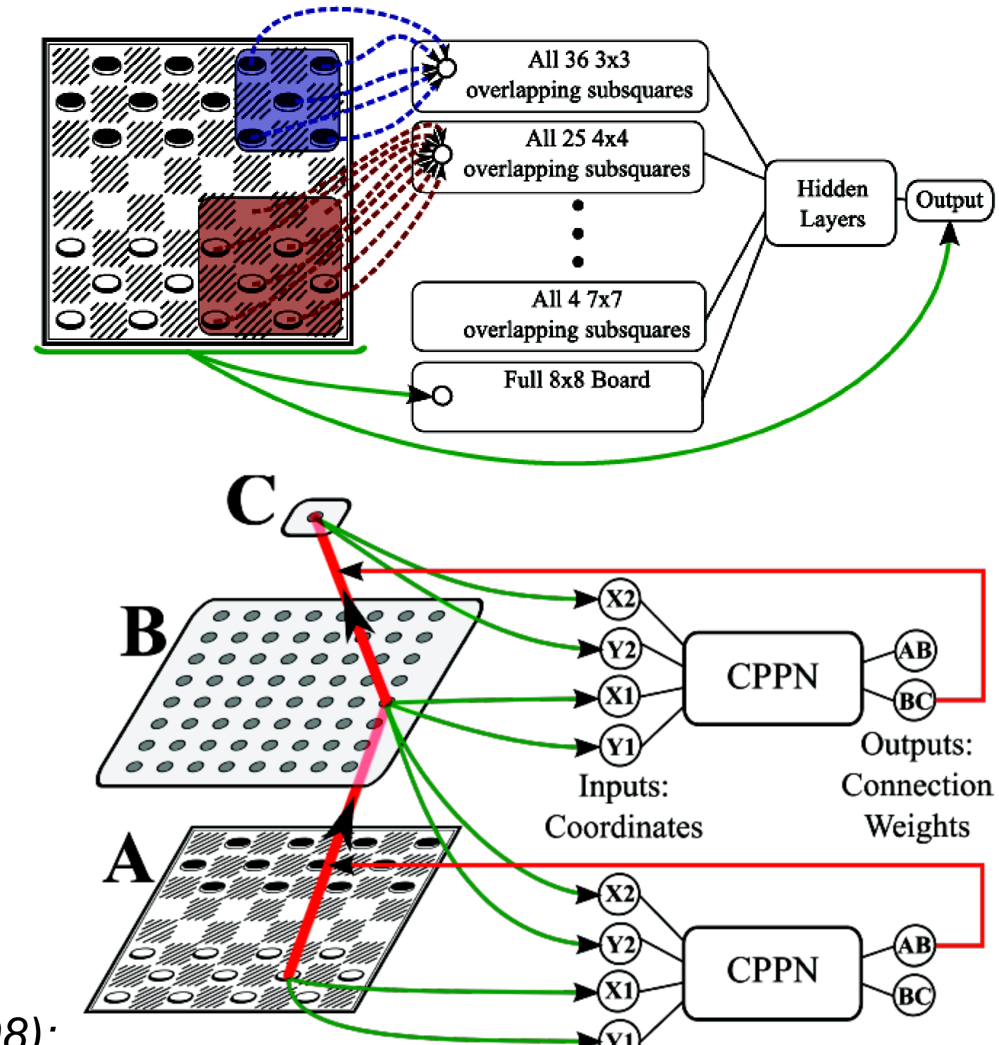
Q&A

Reinforcement Learning Demonstration

- 1994, Karl Sim's: Evolving Virtual Creatures video.
- Evolves both creature bodies (including sensors) and controlling networks...
- You can play with http://www.framsticks.com/

Checkers

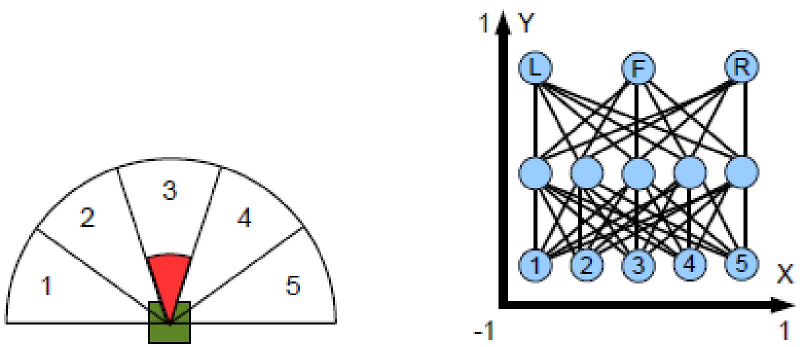
- Comparison with classic NEAT.
- HyperNEAT is faster + generalizes.
- Single CPPN with multiple outputs.
- The output of the final net is a heuristic score for the minimax algorithm.



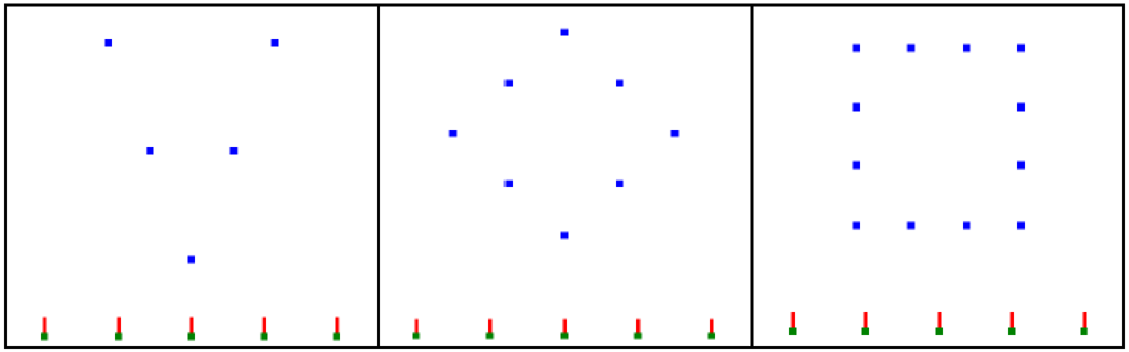
Jason Gauci and Kenneth O. Stanley (2008):
A Case Study on the Critical Role of Geometric Regularity in Machine Learning

Multiagent Predator-Prey

- Predator team tries to catch Prey team in a coordinated fashion.
- Agents cannot see their team mates.



(a) Robot (b) Substrate



(a) Triangle (b) Diamond (c) Square

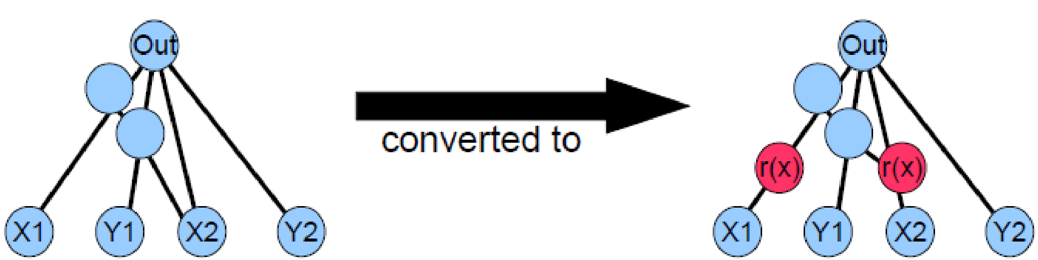
David B. D'Ambrosio and Kenneth O. Stanley (2008)
Generative Encoding for Multiagent Learning

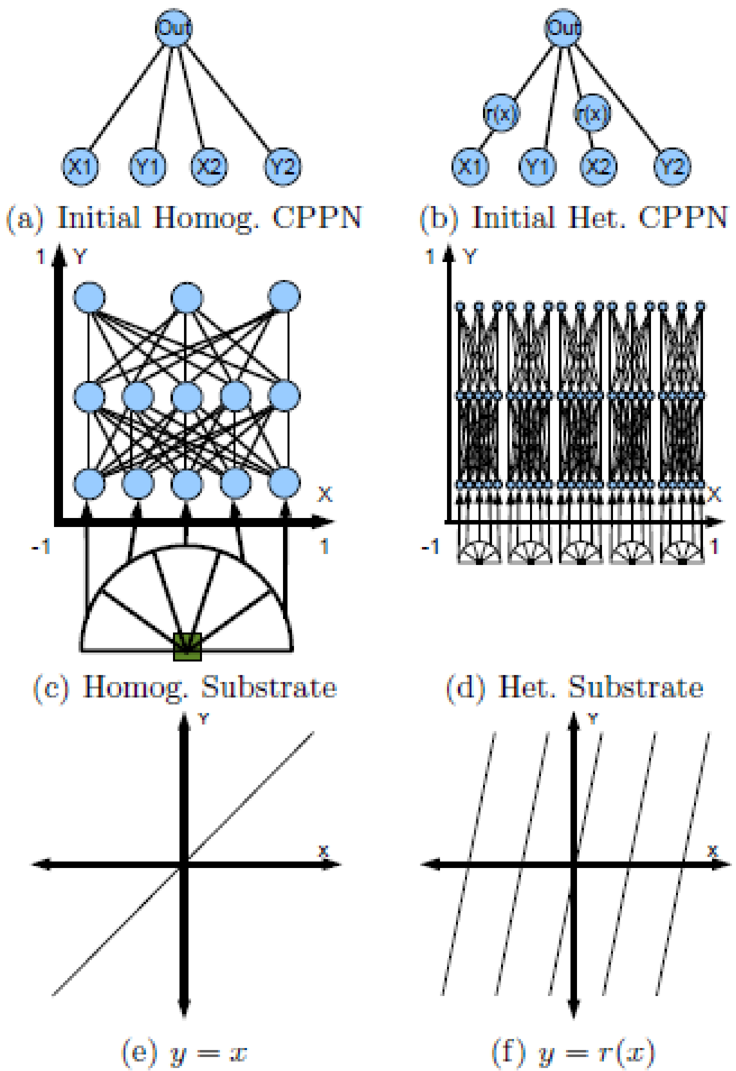
Multiagent Predator-Prey II

- Multiple agent's ANNs using a single CPPN.
- Coordinate repeat *r(x)*.

  VIDEOS:
  http://eplex.cs.ucf.edu/multiagentHyperNEAT/

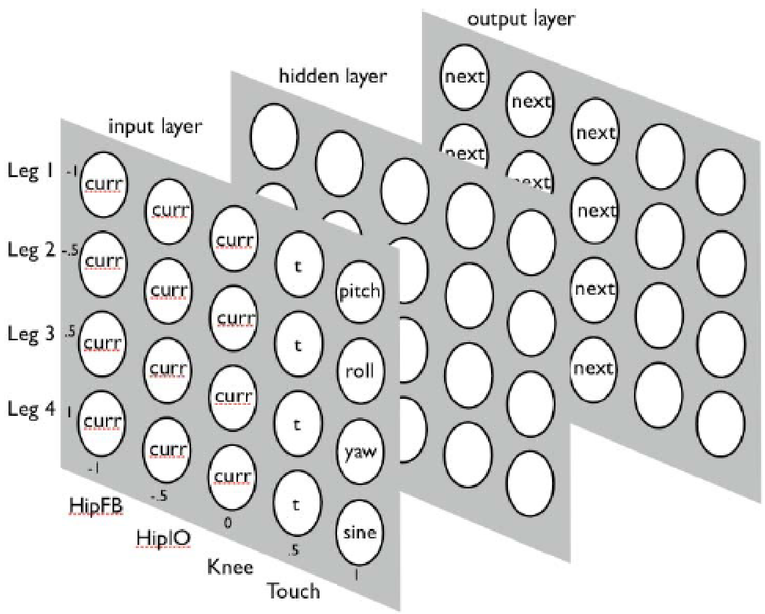
- Heterogeneous seeding.





HyperNEAT Coordinated Quadruped Gaits

Jeff Clune: ***Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding***



- Simulation of four legged walker robot.
- Comparison with classic NEAT.
- Other experiments show that HyperNEAT can deal with random substrates.