Indirect Encodings of Artificial Neural Networks

Jan Drchal
drchajan@fel.cvut.cz

COMPUTATIONAL
INTELLIGENCE
GROUP

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

Overview

- Large-scale Artificial Neural Networks.
- Computational Development.
- Indirect Encodings of ANNs.
- Hyper-cube based encoding.

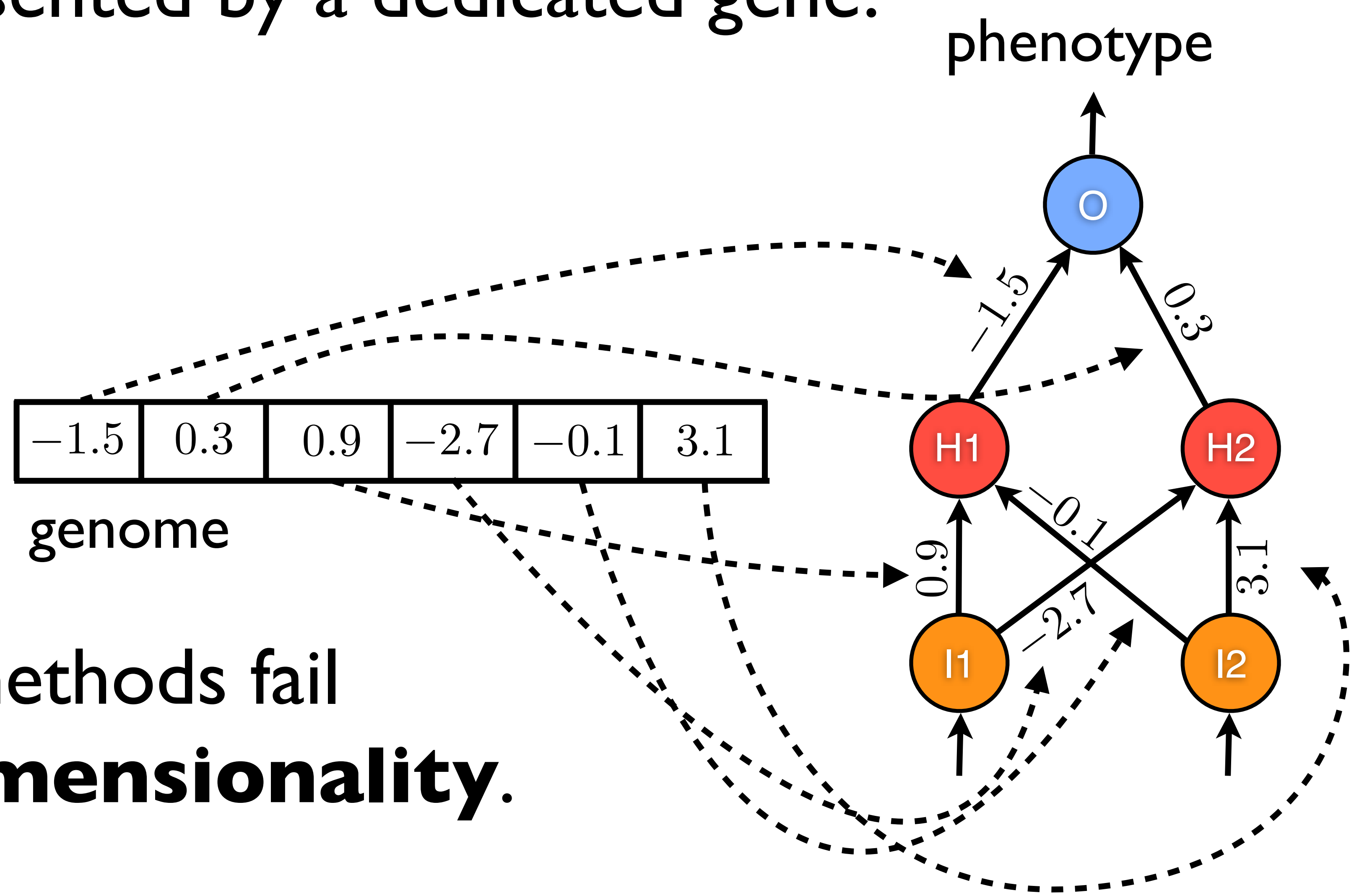
Evolving Large-scale ANNs

- 1000+ neurons (& corresponding # of links).
- **Why to do that?**
  - Complex models,
  - ability to process huge amount of inputs/outputs without hand-coding features (i.e. pattern recognition)...

Direct Encoding

- **Direct encoding** → each structural part (neuron/link) is represented by a dedicated gene.
- Not suitable for Large-scale ANN's:

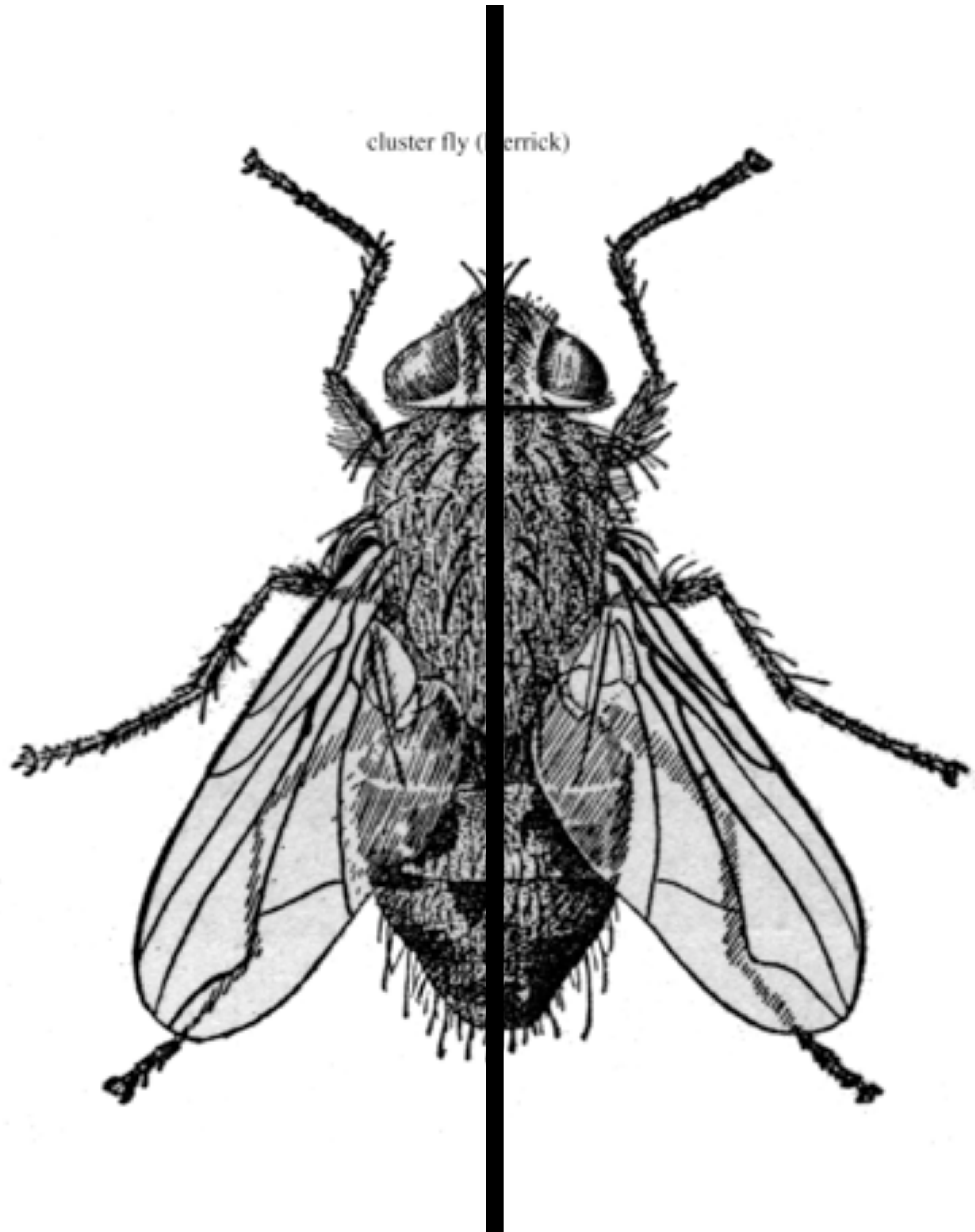
Direct optimization methods fail
→ **the curse of dimensionality**.



Indirect Encoding: the Way it Works in Nature

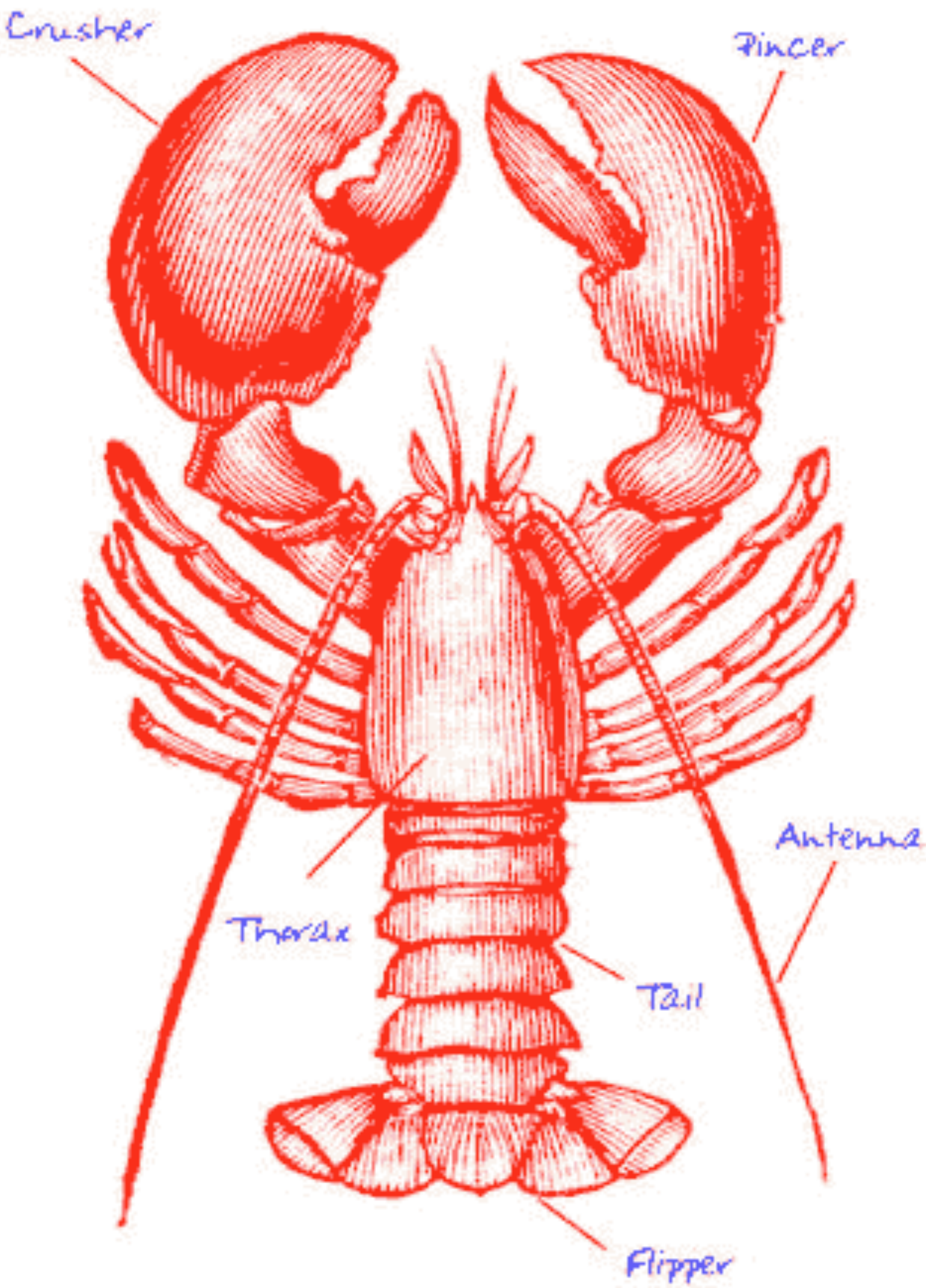
- Human genome → **20 000 - 25 000 genes describing almost 100 billion neurons each linked to as many as 7 000 others** (plus the rest of organism!).
- We need some kind of **compression**:
  → **indirect encoding.**
- But we also need a **regularity** in data being compressed.
- **Q:** What are the regularities found in living organisms?

Symmetry



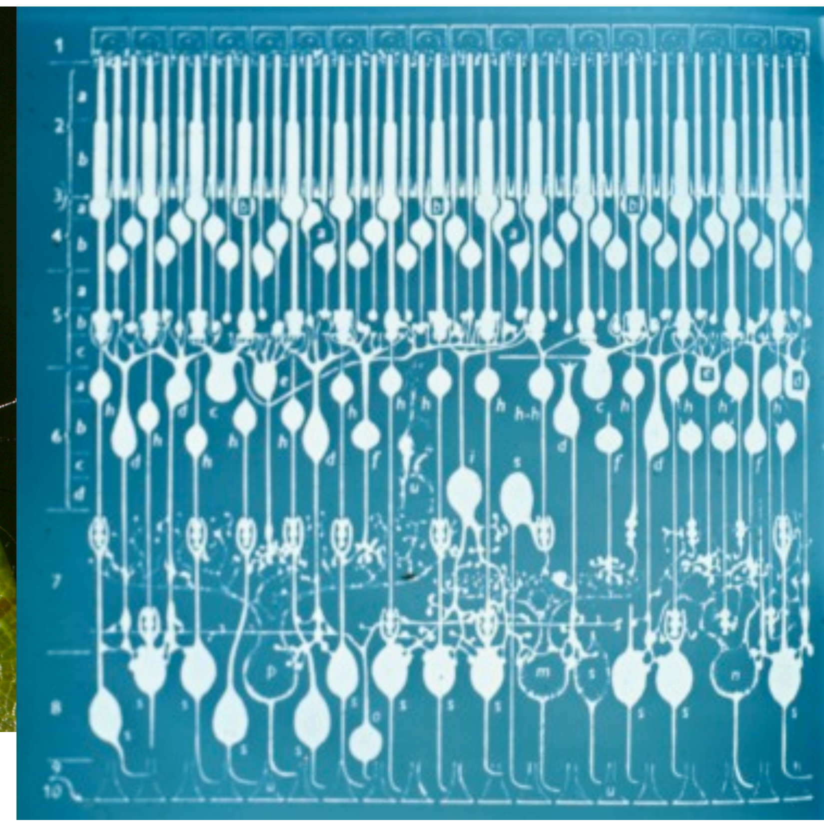
(wikimedia commons)

Imperfect Symmetry



(wikimedia commons)

Repetition with Variation

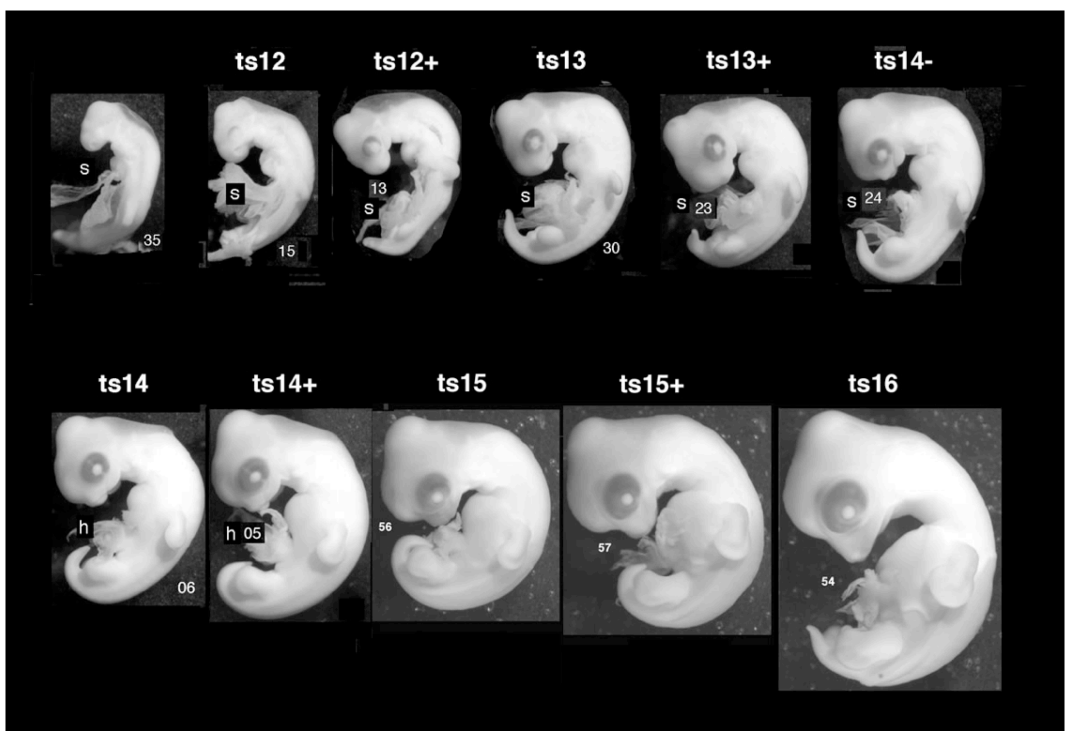


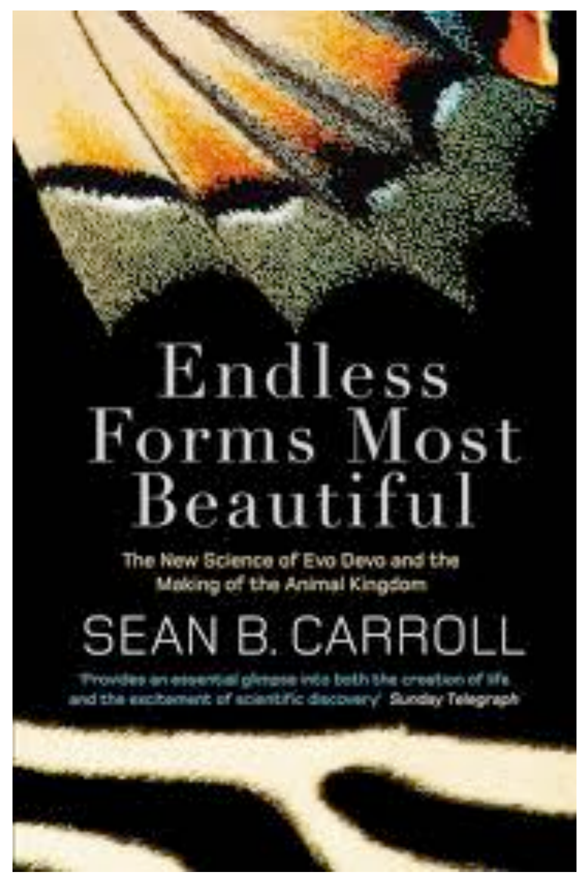
- Note that all these regularities happen at **all scales** of an organism.

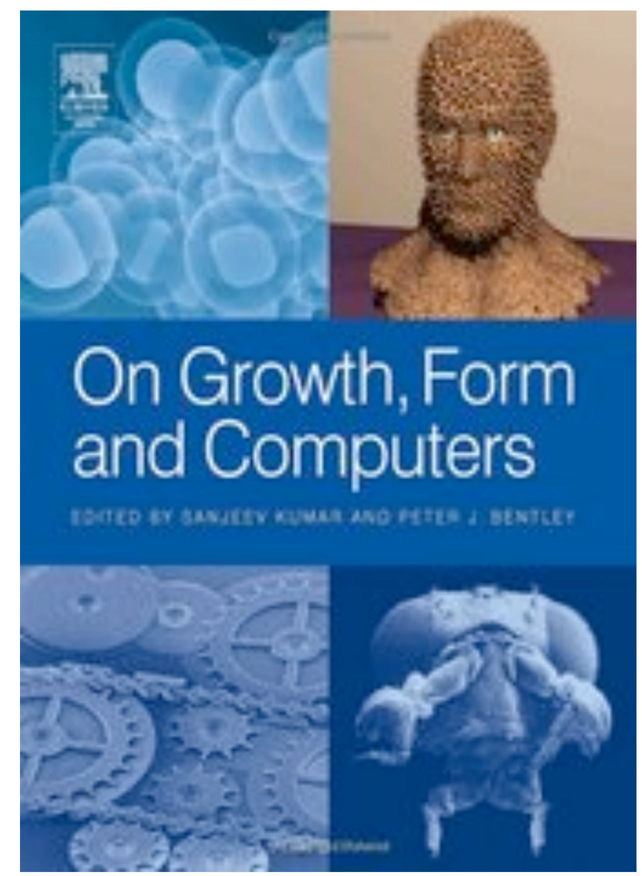
(wikimedia commons)

How Are Organisms Built?

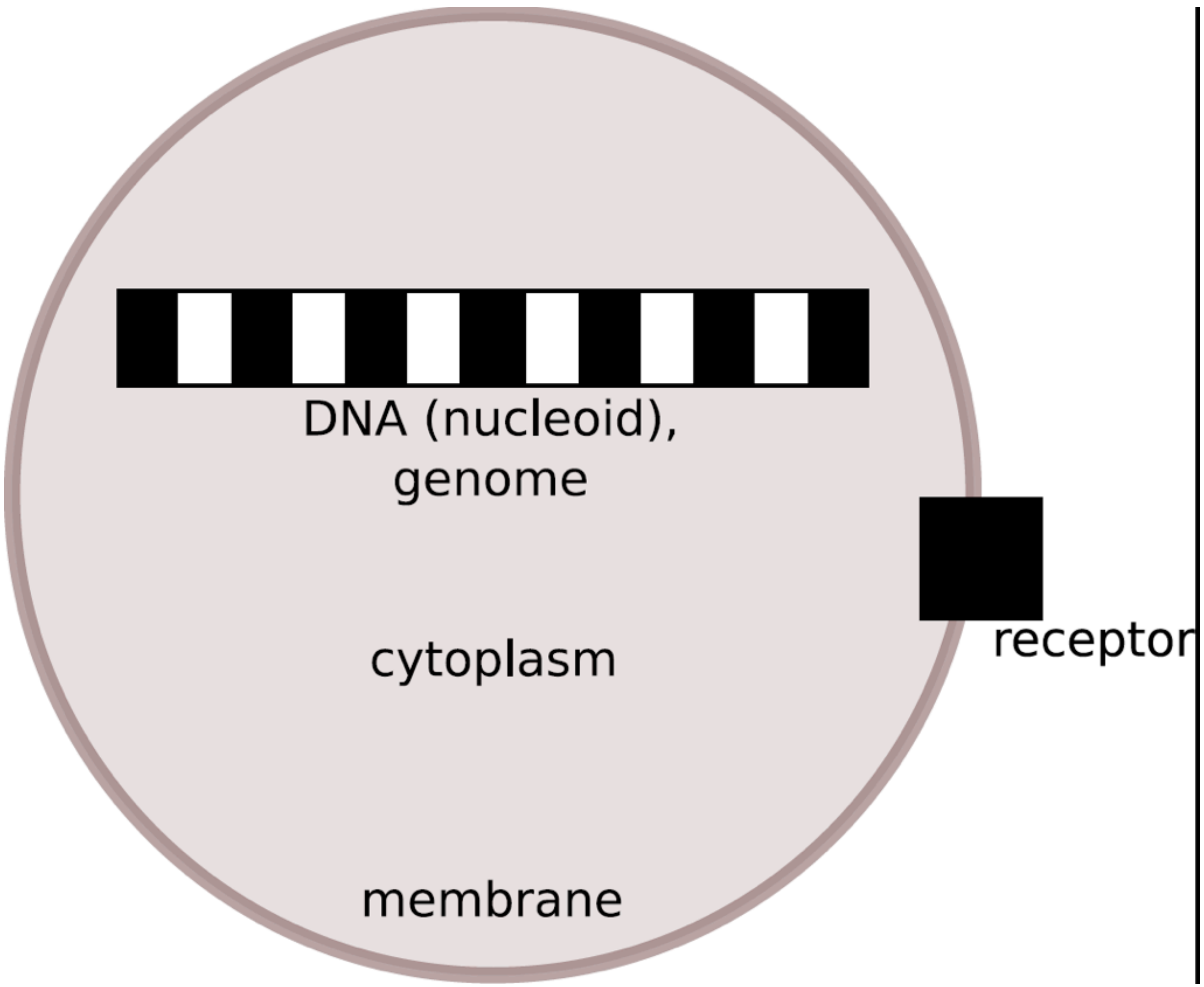
- Development from a single cell (zygote).
- Evolutionary Development "Evo-Devo".



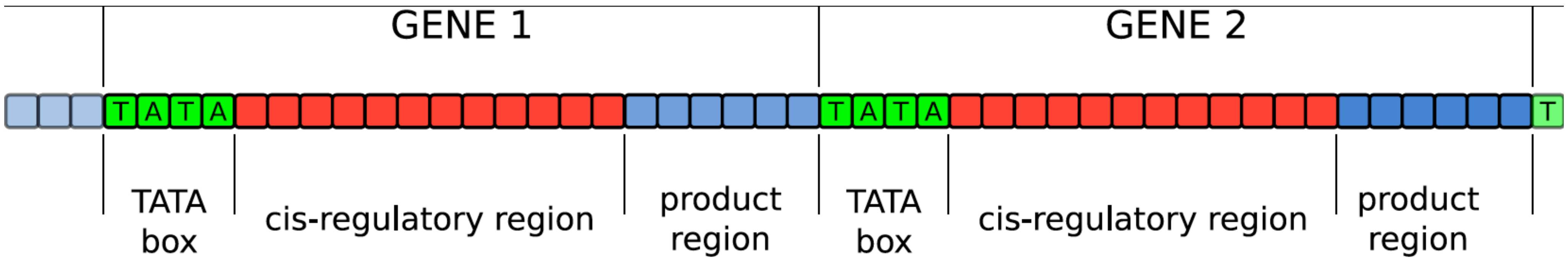




The Cell

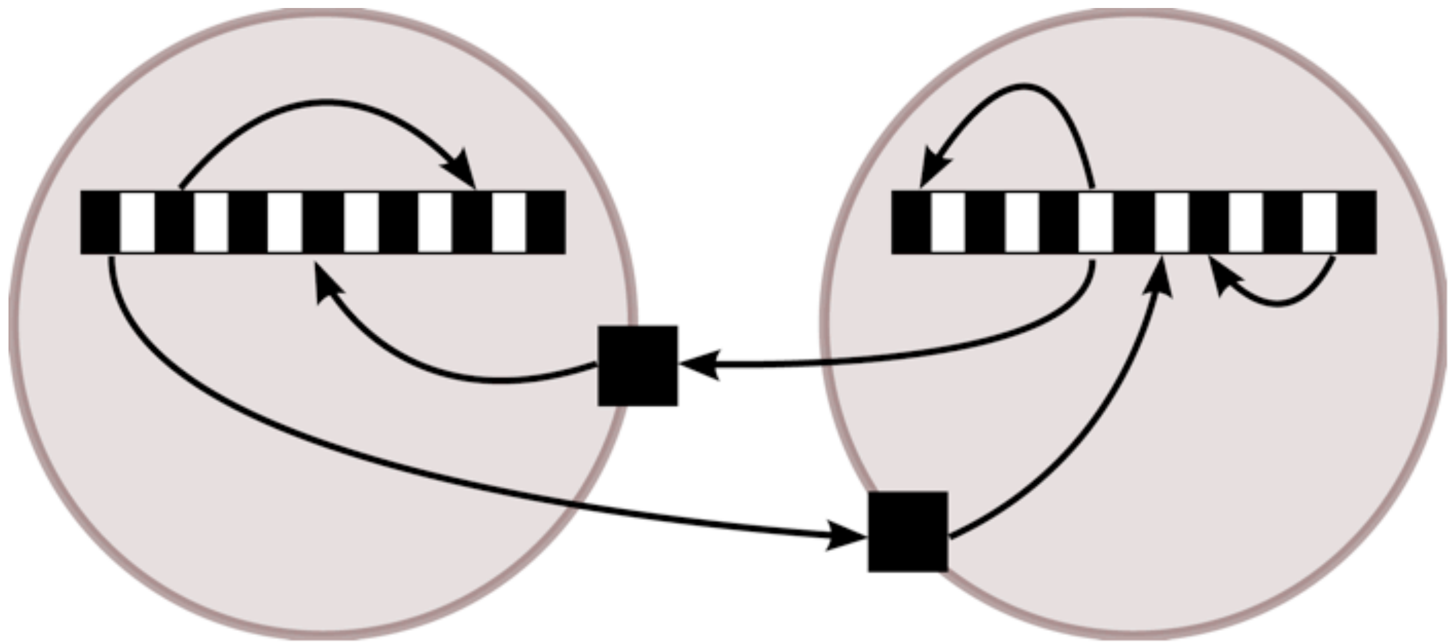


Genome: A Closer Look



- *TATA box* – marks the start of a gene
- *(cis-)regulatory region* – composed of binding sites.
- *binding site* – binds regulatory proteins → gene activation/ inhibition
- *product region* – when gene is active a protein is produced:
- *special*: cell division, differentiation,
- *regulatory*: can bind to binding sites of other genes,
- *structural*.

Cell Divisions



- Program same for all cells.
- What differs?
  - Regulatory protein *concentrations*.
- *Receptors* – selectively pass regulatory proteins from inter-cellular space.
- Diffusion, decay, cell differentiation.
- Gene Regulatory Networks (GRNs).

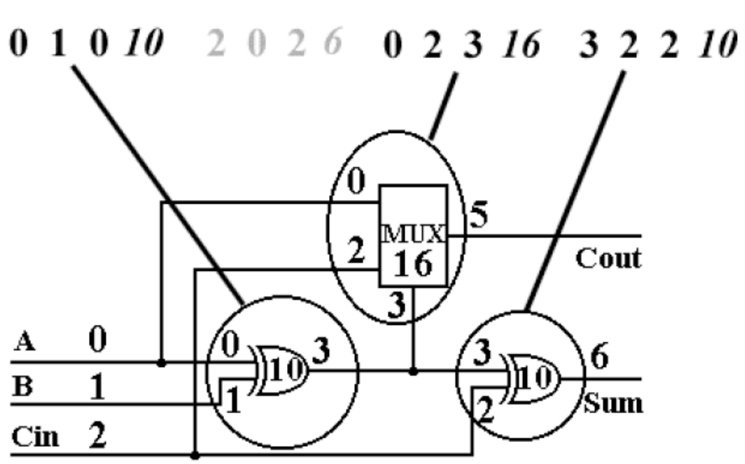
How to Simulate Development?

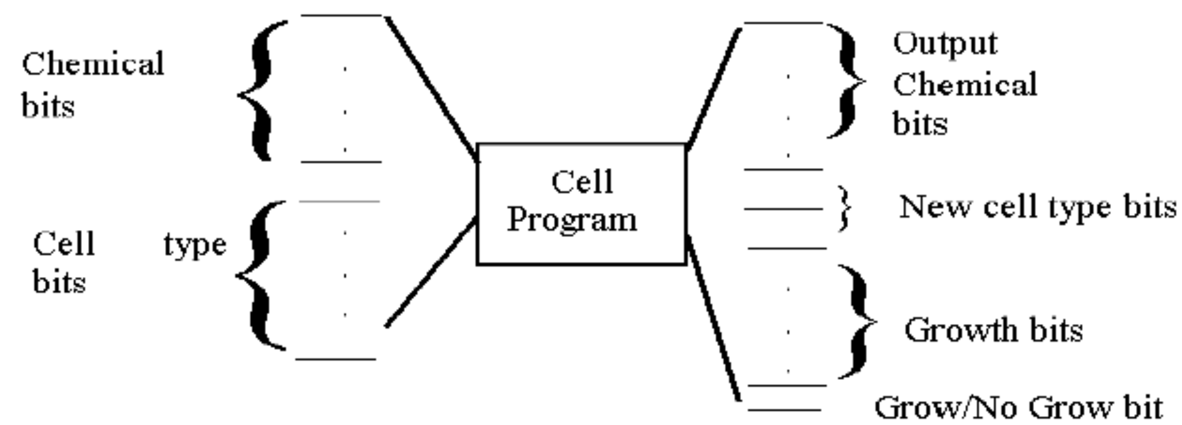
- Cell program – ANN, FSM or other controller:
  - *inputs*: binding sites,
  - *outputs*: one for each gene → gene activity.
- *Physical simulation*: diffusion, decay, receptors...
- Cell division:
  - *copy cell program* from mother → daughter cell,
  - *different concentrations* for mother/daughter.
- This is called: *Computational Development*.

“French Flag” Organism

- Cell program evolved using Cartesian Genetic Programming (CGP).

CGP encoded adder





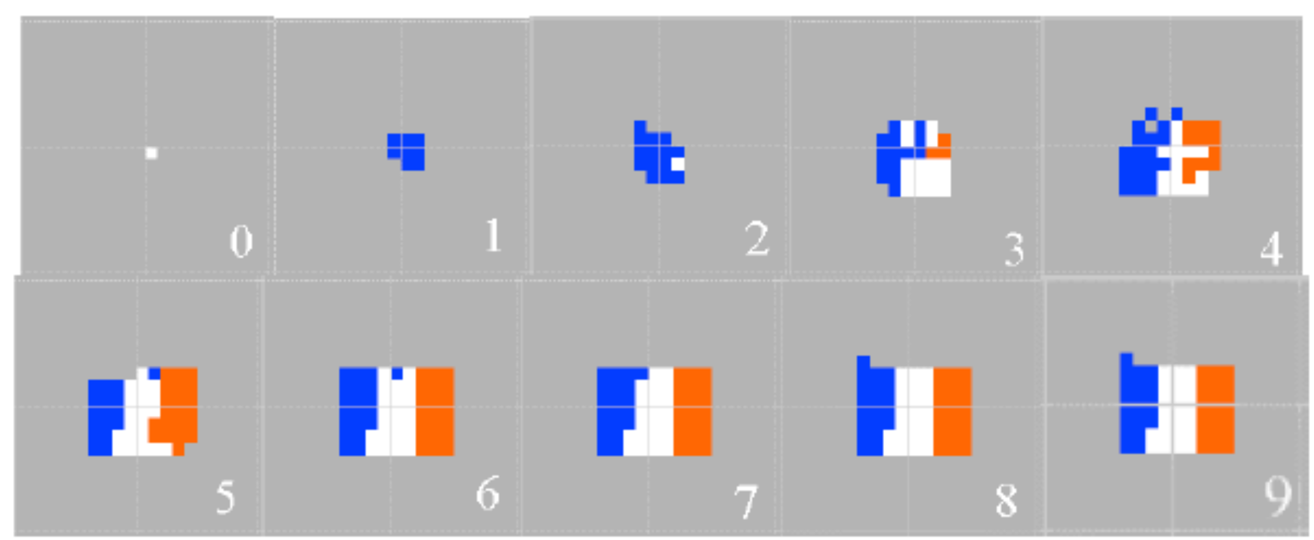


Fig. 4. Growth of fittest cell program from a white seed cell to a mature French flag (two chemicals)

"French Flag" Organism II

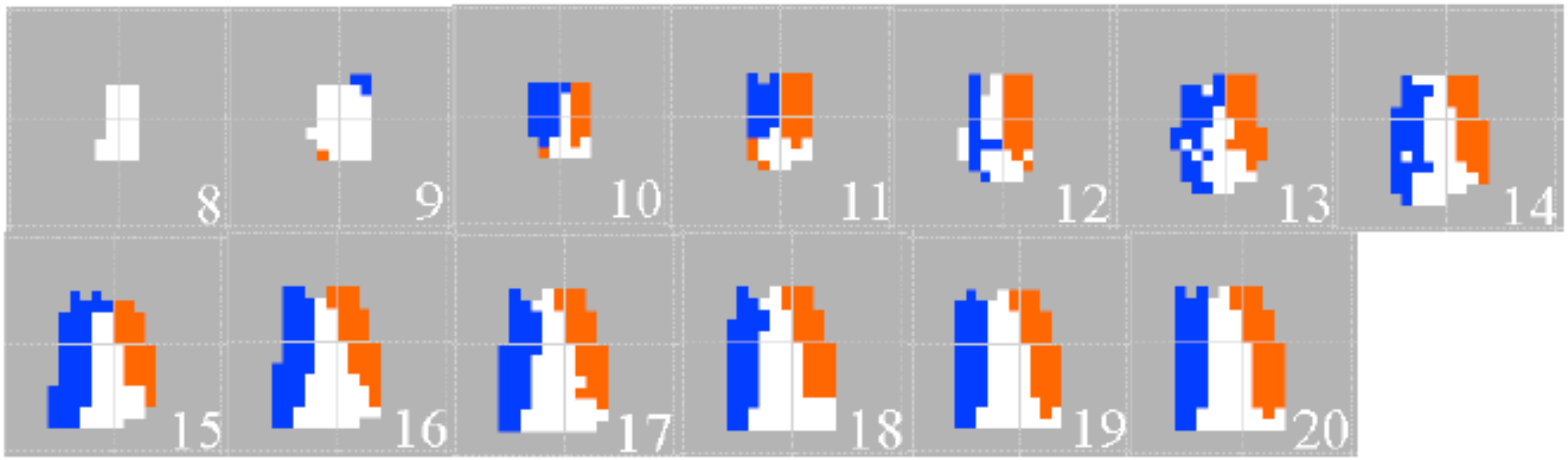


Fig. 7. Autonomous recovery of badly damaged French flag organism conditions (blue and red regions killed at iteration 8 - see Fig. 4). There is no further change after iteration 20

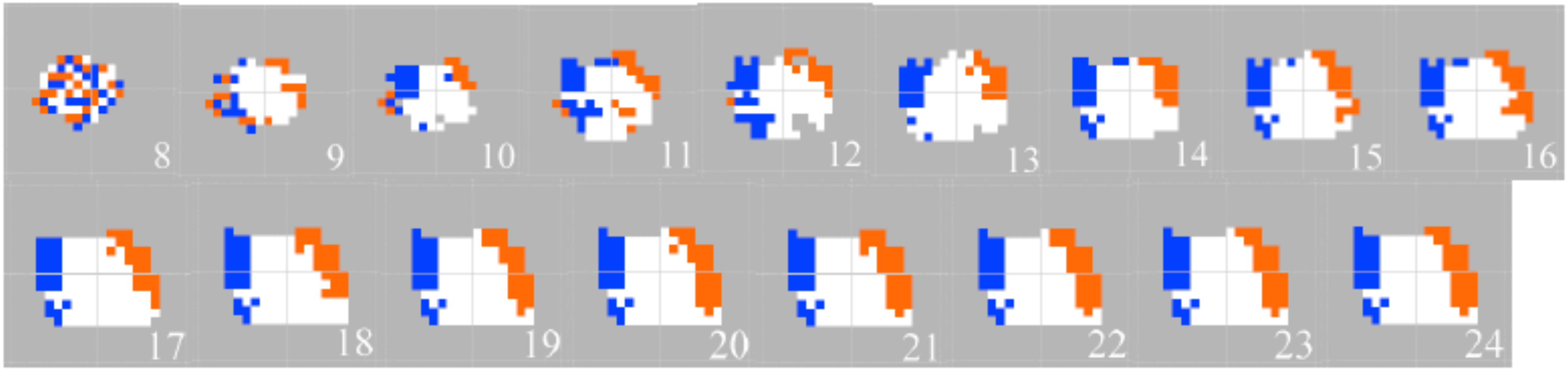
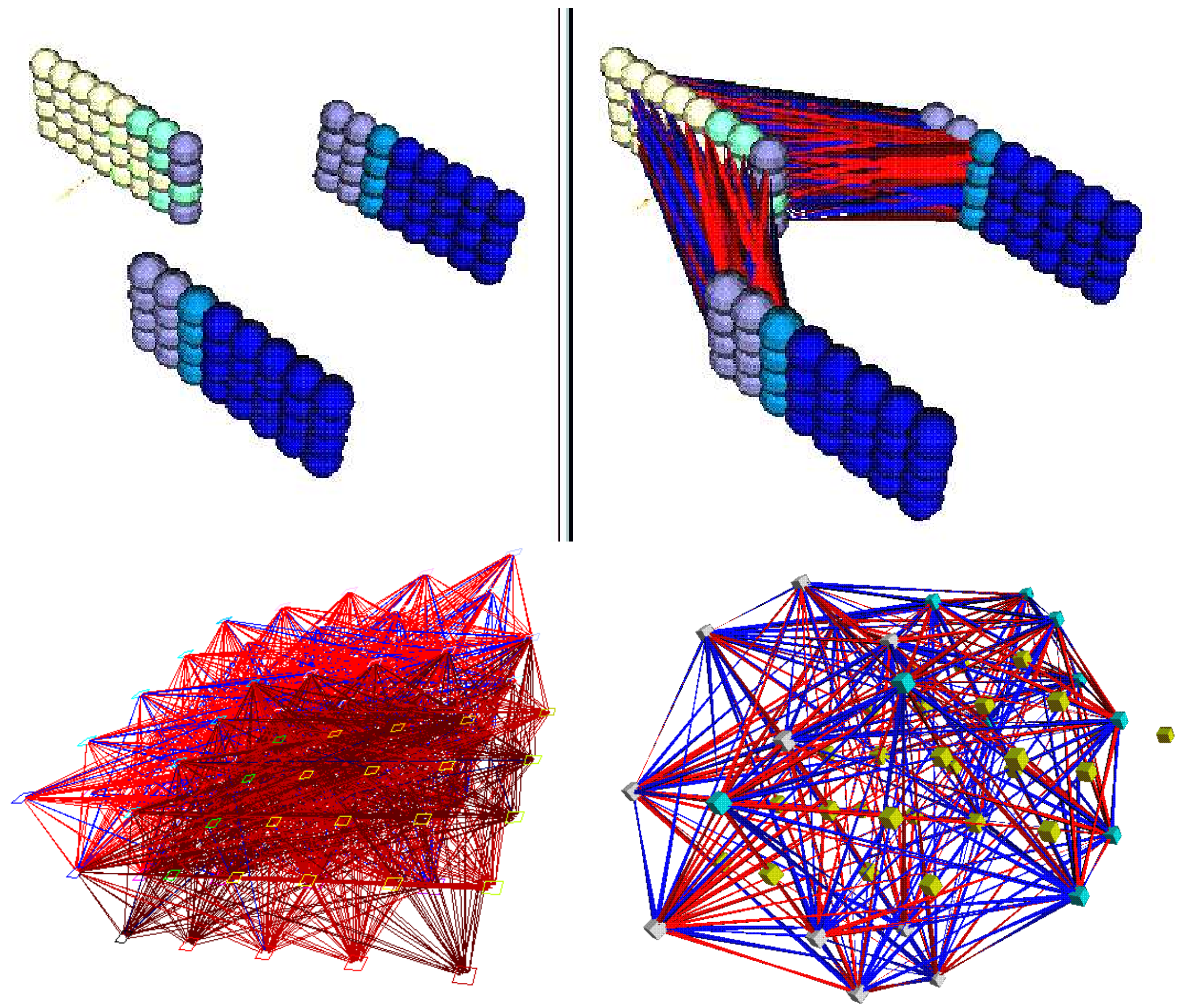


Fig. 8. Autonomous recovery of French flag from randomly rearranged cells (French flag at iteration 8 - see Fig. 4). There is no further change after iteration 24

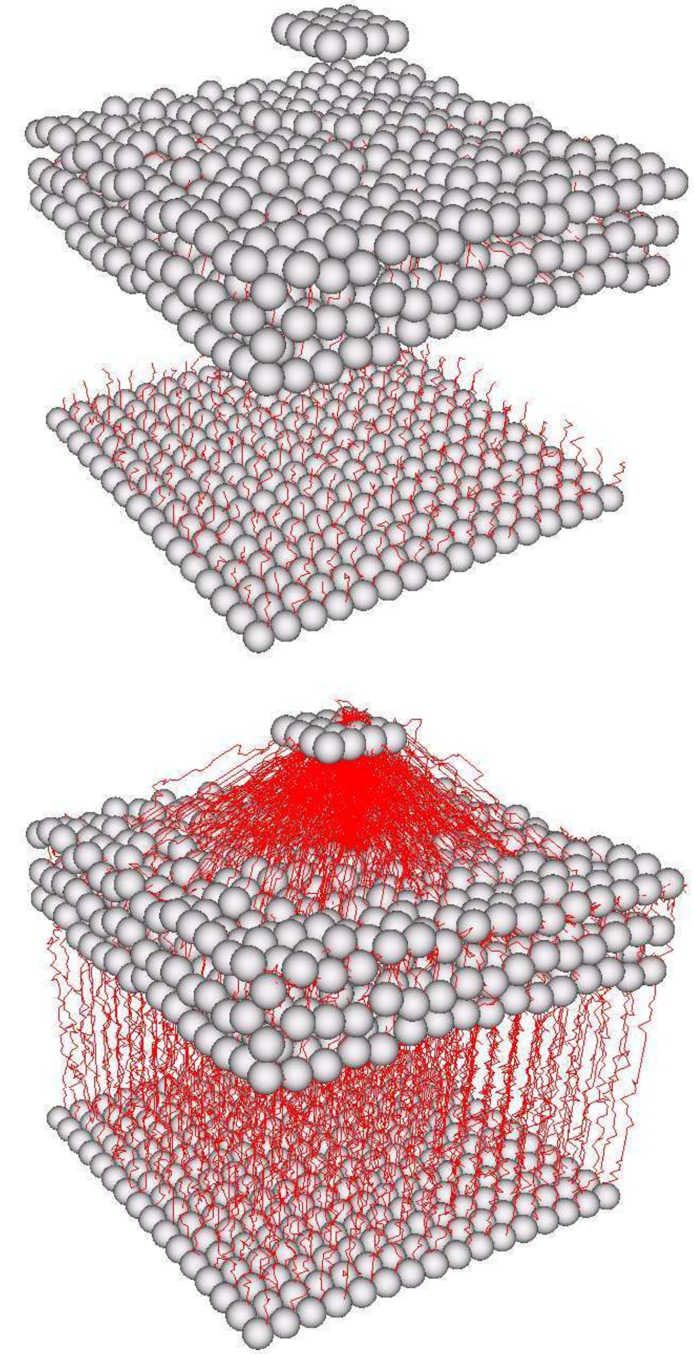
Indirect encodings of ANNs

- GRN-based
- Cellular Encoding
- Hypercube-based
- Other: rewriting rules, L-systems, ...

GRN-based



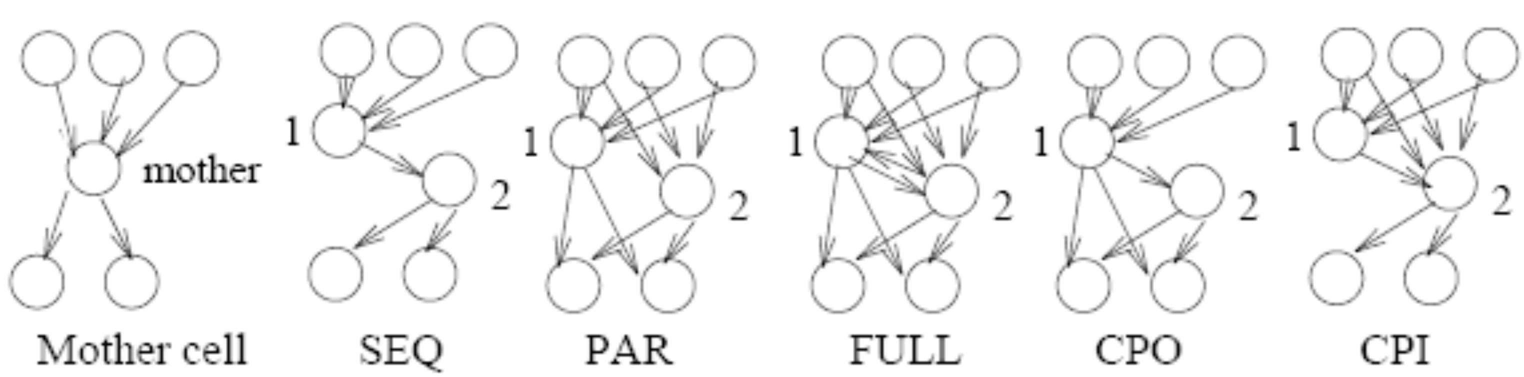
Peter Eggenberger-Hotz (1997):
Creation of Neural Networks Based on Developmental and Evolutionary Principles

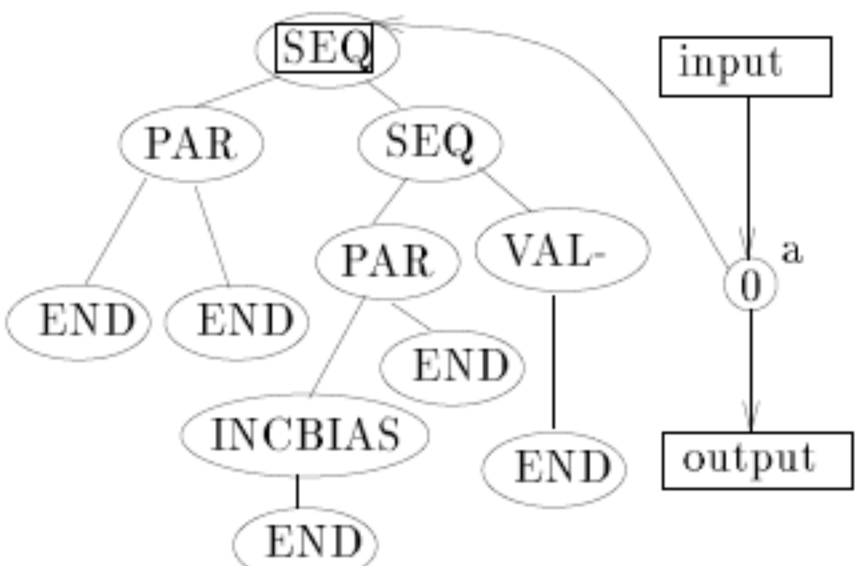


Peter Eggenberger-Hotz (2003):
Evolving the Morphology of a Neural Network for Controlling a Foveating Retina and its Test on a Real Robot

Cellular Encoding (CE)

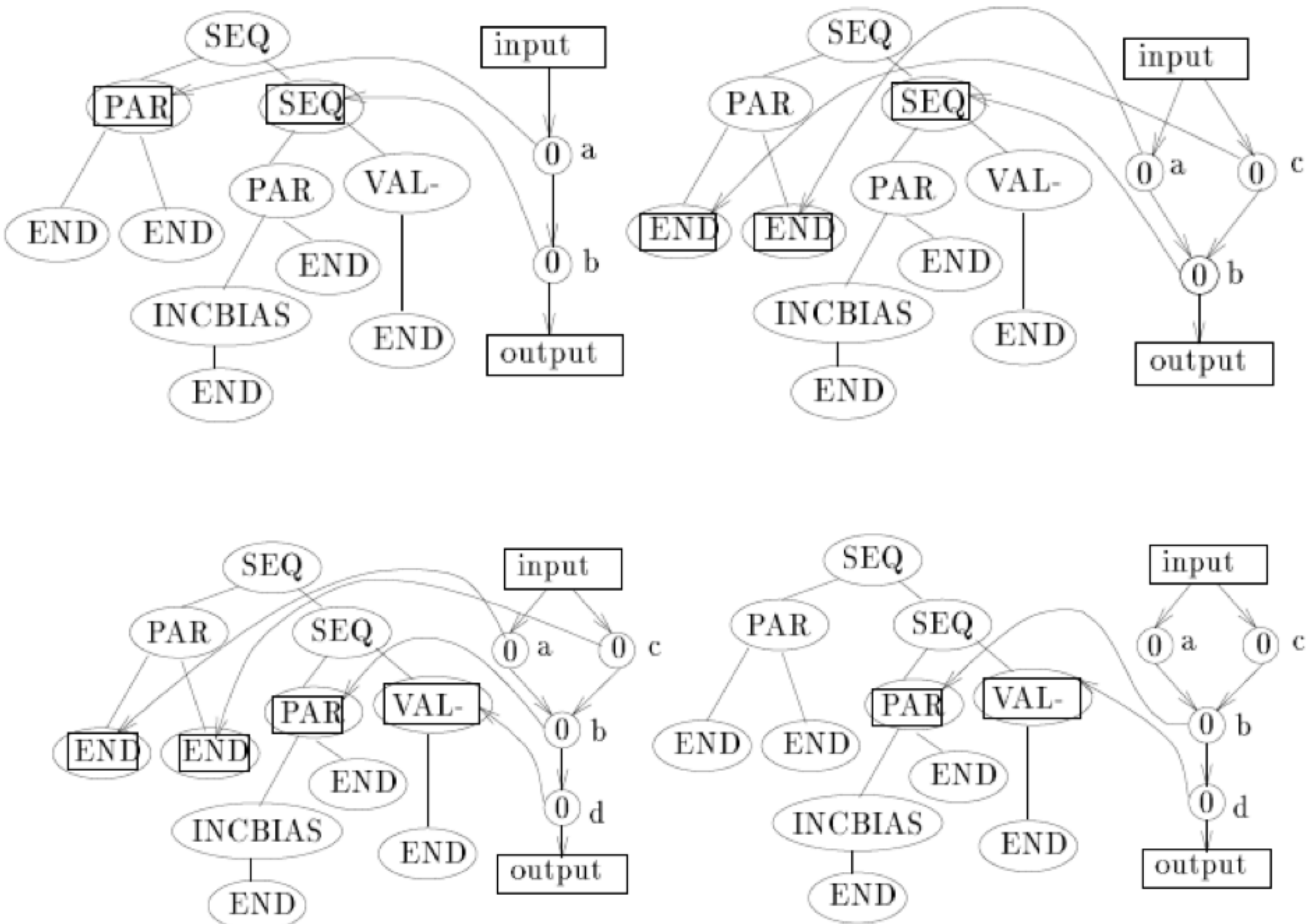
- 1993, Fréderic Gruau: indirect encoding example.
- Inspiration in embryo-genesis (cell division and differentiation). Cells → neurons.
- Program to “grow” ANN is represented by a tree (Genetic Programming).
- Operations: parallel/sequential divisions, connections change, change of weights/bias...





Frédéric Gruau (2004):
Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm

Cellular Encoding II



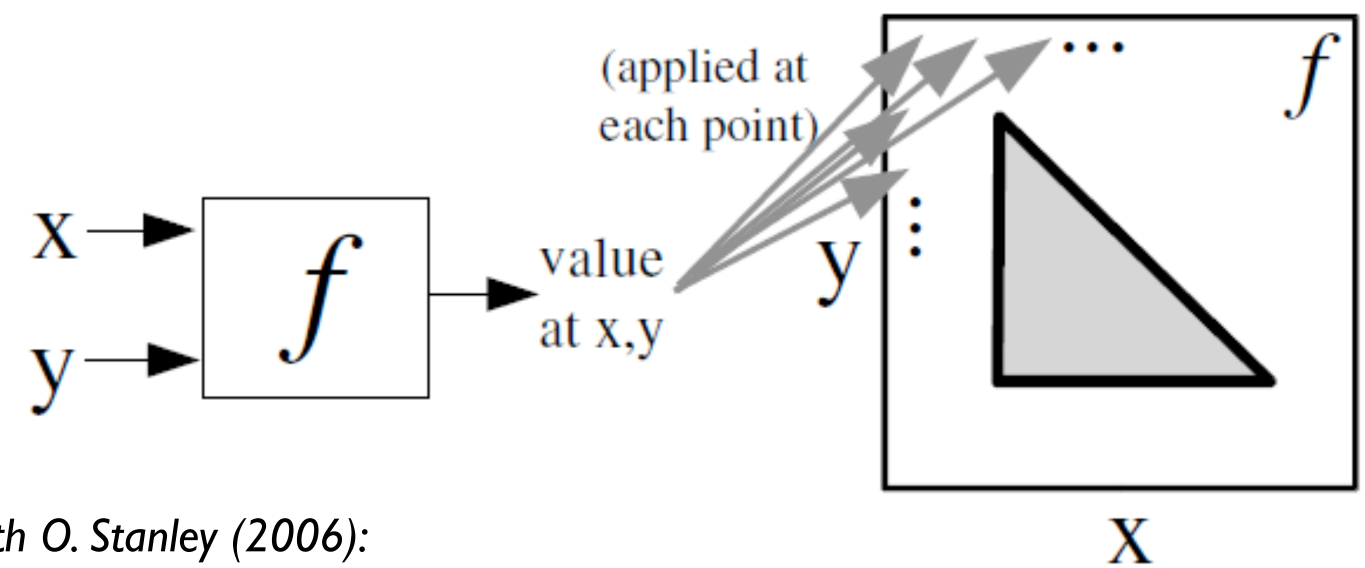
COMPUTATIONAL INTELLIGENCE GROUP

Cellular Encoding III

- 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*.

Compositional Pattern Producing Networks (CPPNs)

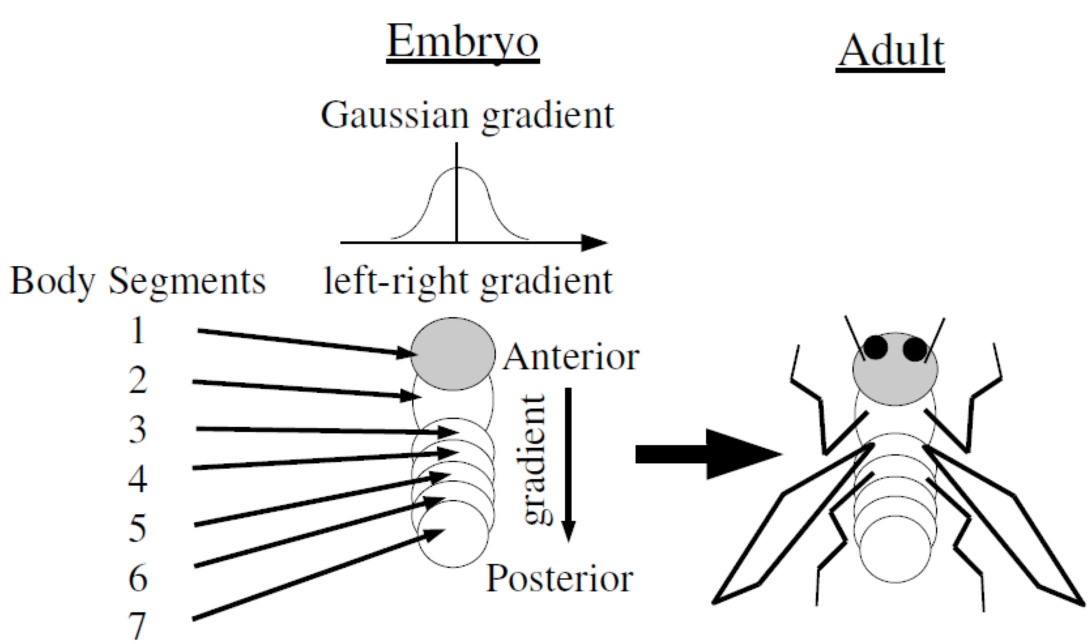
- Stanley 2006.
- **Can we create such regular patterns without development in time?**
- We can ask a special function called CPPN, where the cells are, using absolute coordinates.

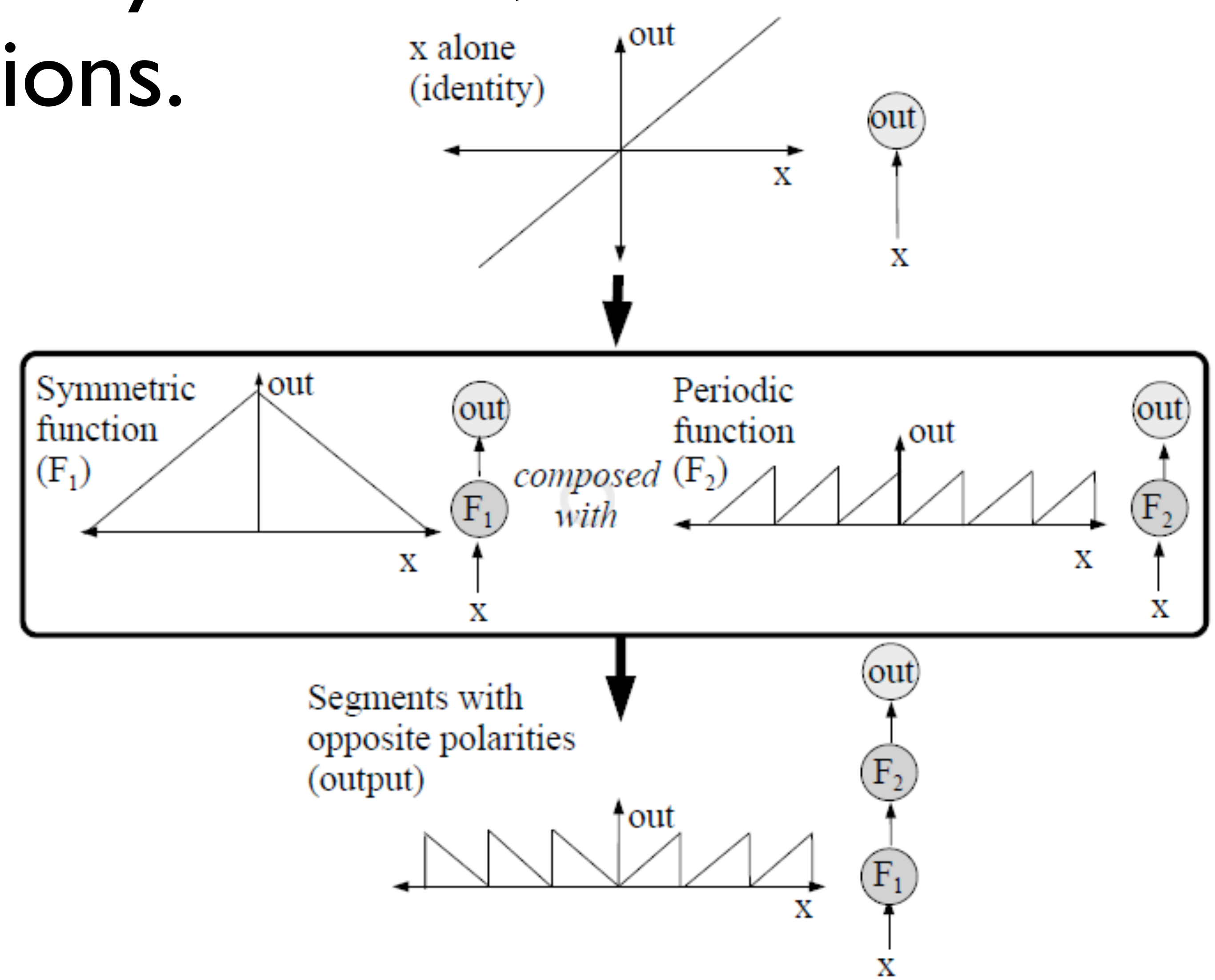


Kenneth O. Stanley (2006):
Compositional Pattern Producing Networks:
A Novel Abstraction of Development

Regularities by CPPN

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

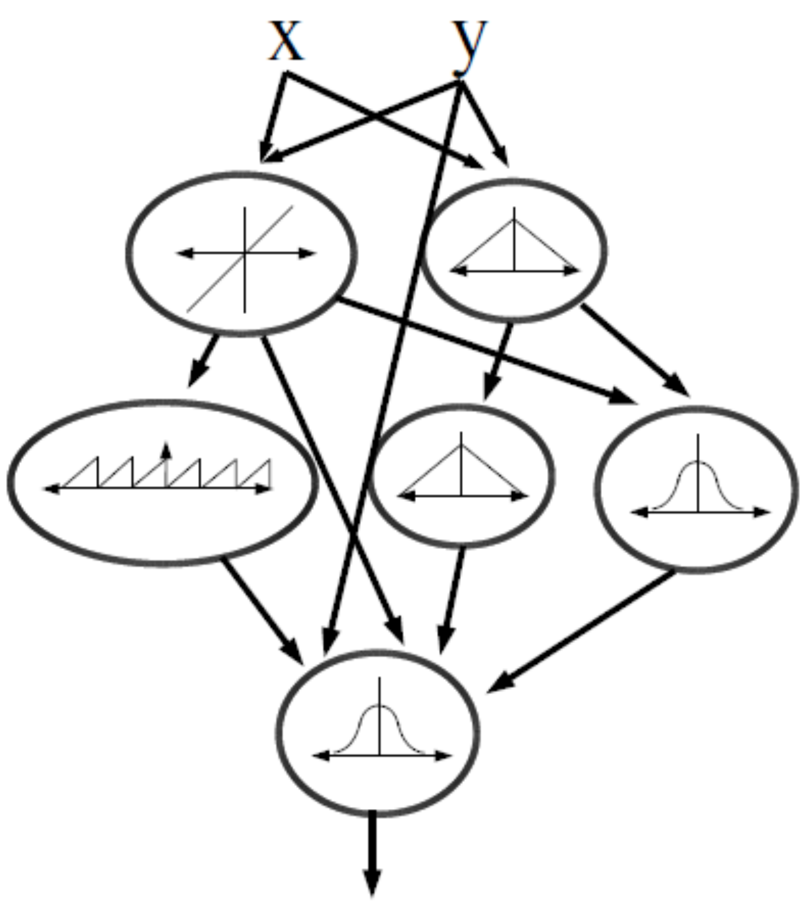




COMPUTATIONAL
INTELLIGENCE
GROUP

Regularities by CPPN II

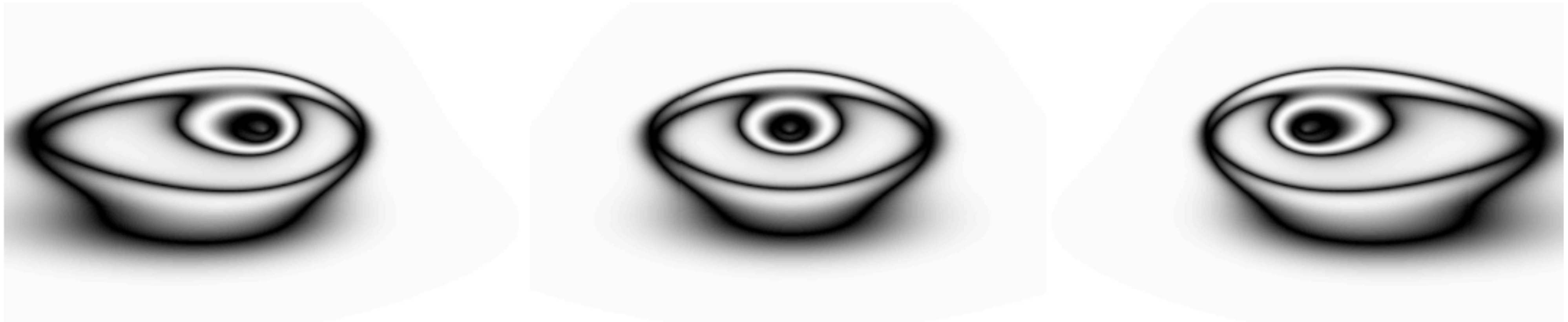
- **CPPN is a composition** of symmetric, periodic and other functions.



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

Picbreeder

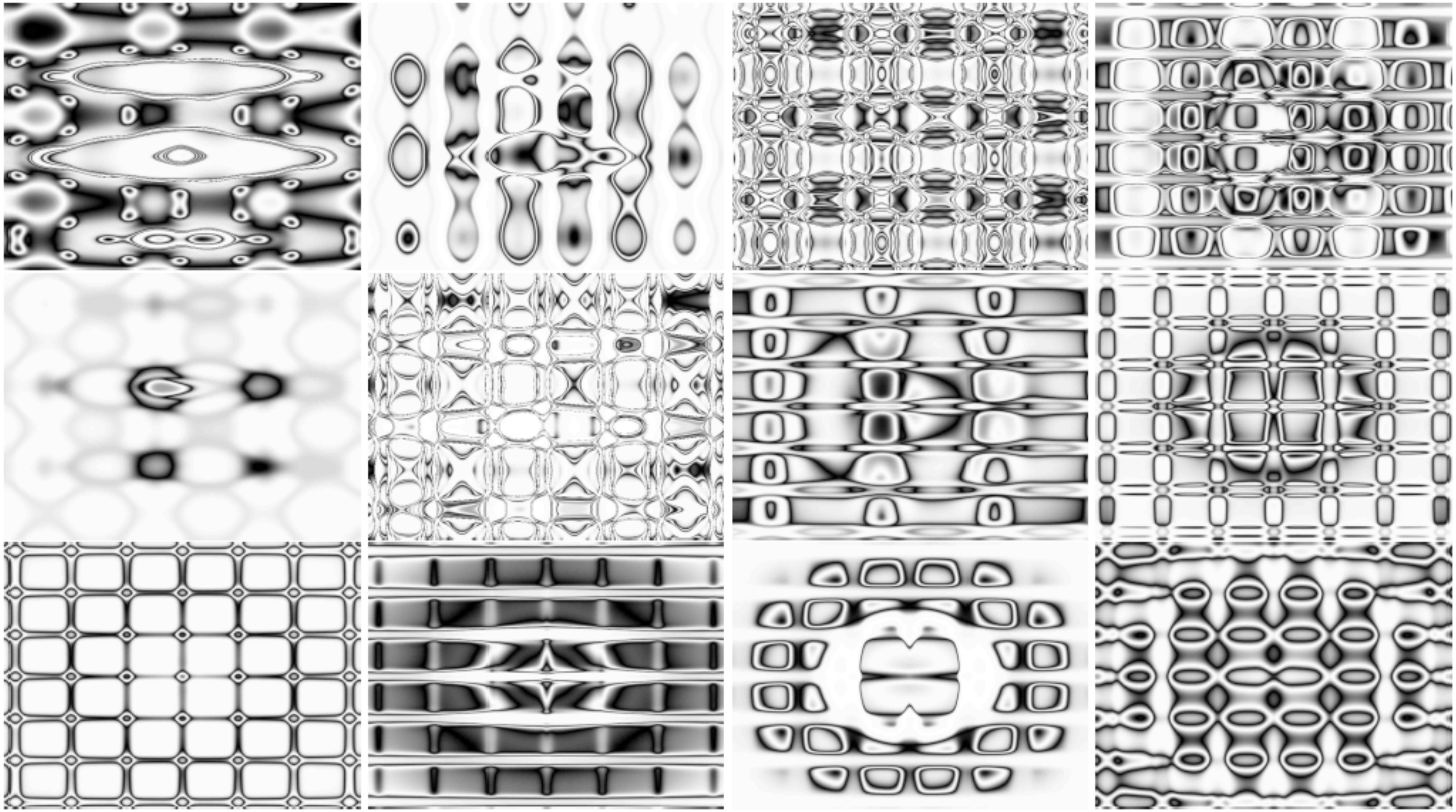
- **Interactive evolution** of images.
- CPPN output: level of grey.
- CPPNs evolved using NEAT.
- http://picbreeder.org/



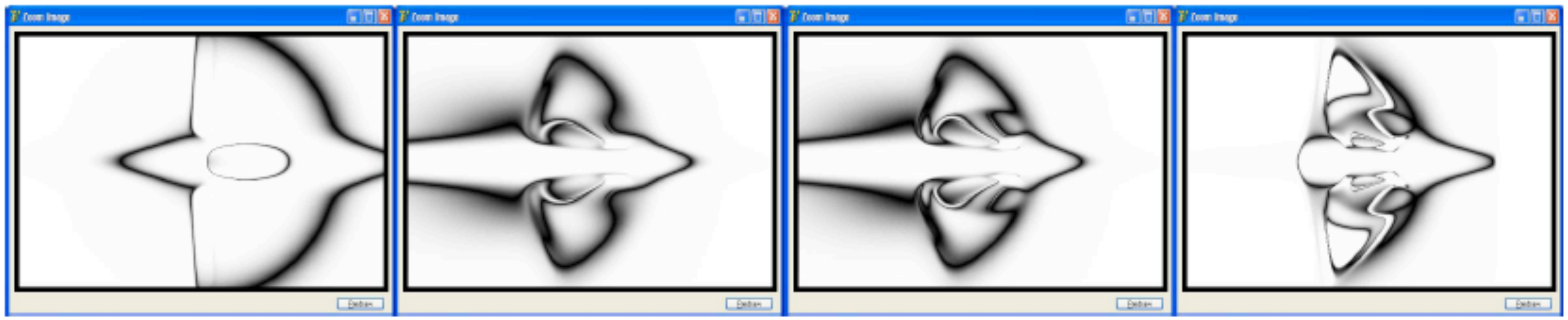
(a) Eye warped left (b) Symmetric eye (c) Eye warped right

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

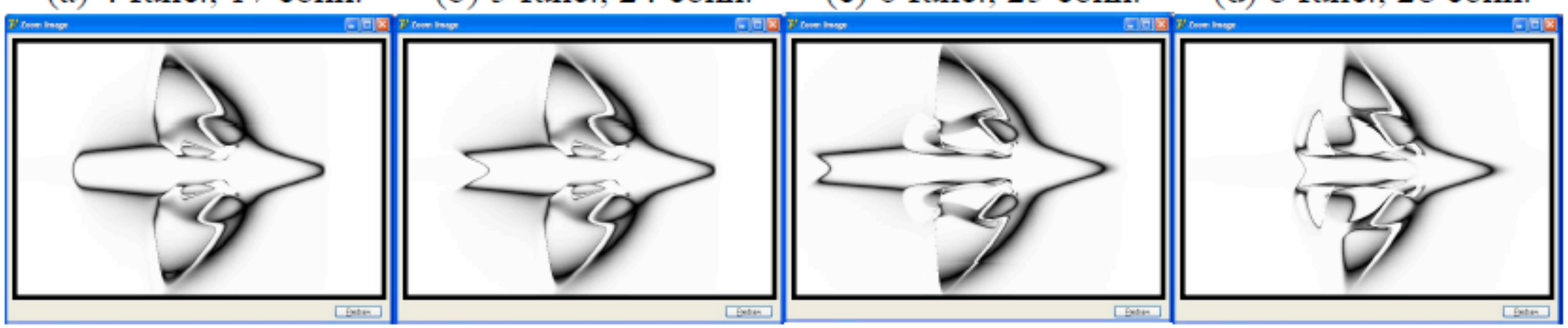
Picbreeder II



Picbreeder: Space Ship



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

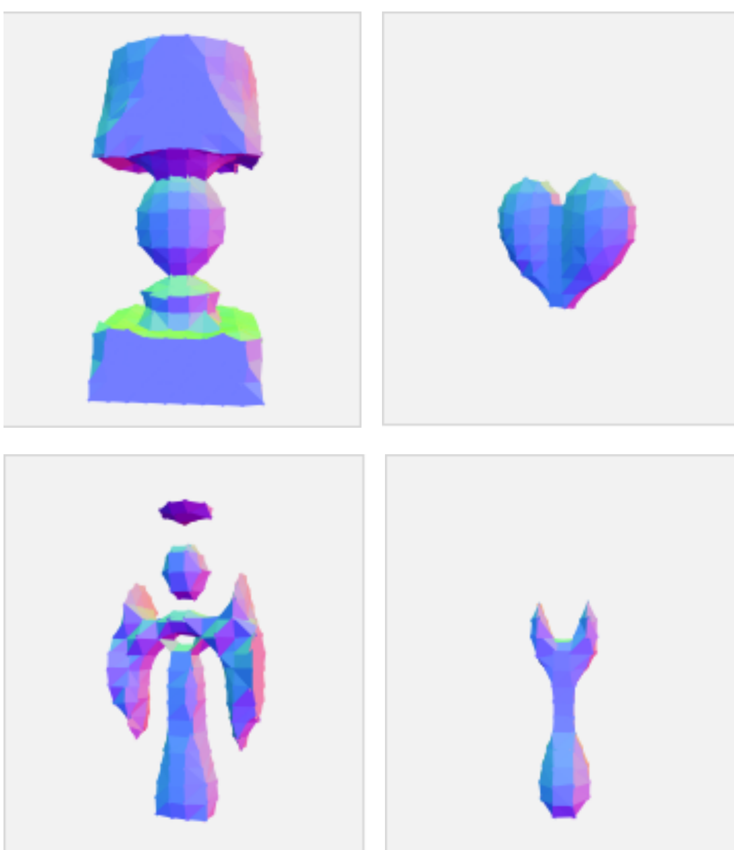


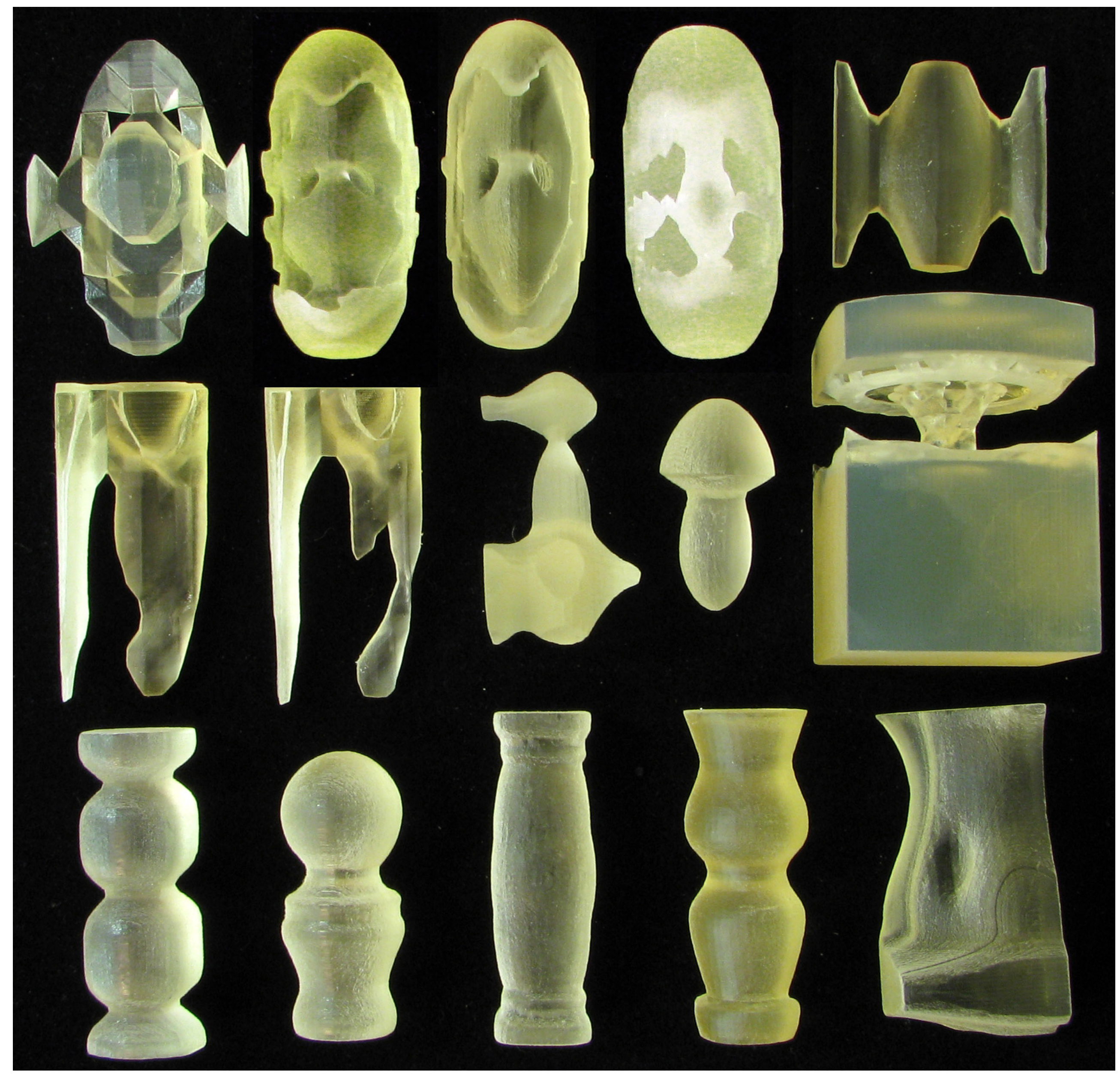
(e) 8 func., 30 conn. (f) 8 func., 31 conn. (g) 8 func., 32 conn. (h) 8 func., 34 conn.

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

Endless Forms

- Similar approach in 3D.
- http://endlessforms.com





COMPUTATIONAL INTELLIGENCE GROUP

Jeff Clune, Hod Lipson (2011):
Evolving Three-Dimensional Objects with a Generative Encoding Inspired by Developmental Biology

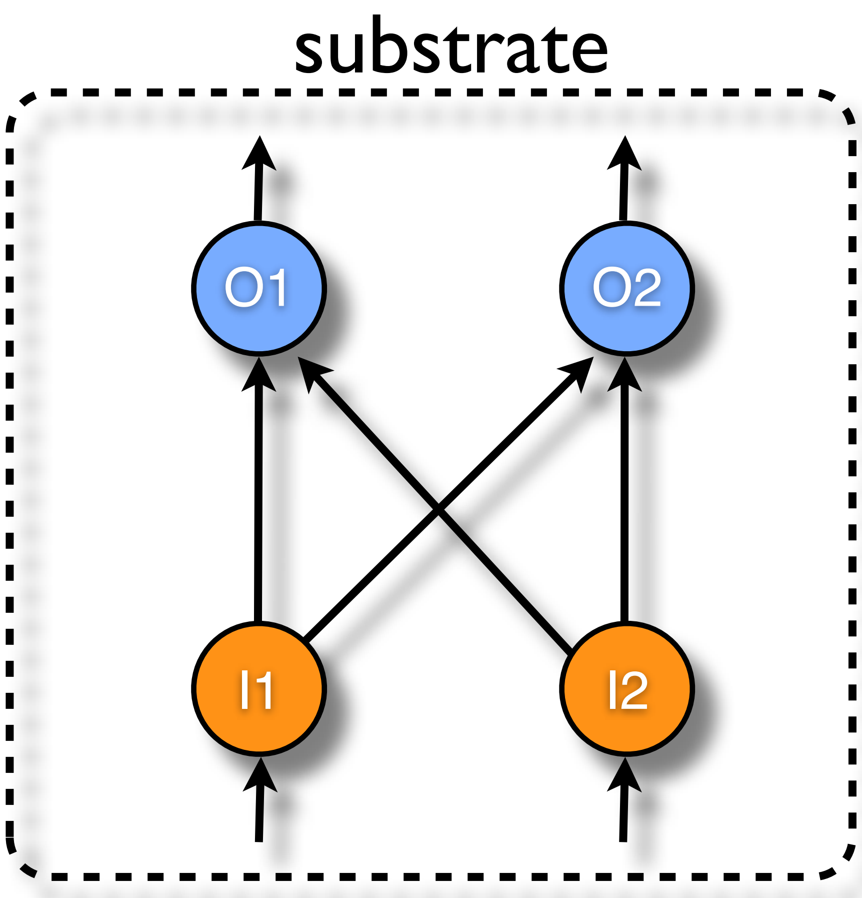
Hypercube-based Encoding

- Stanley 2007.
- Uses CPPNs in a similar way to Picbreeder: evolves **connectivity patterns**.
- Best known for **HyperNEAT** algorithm which evolves ANNs.

HyperNEAT

- Stanley et al. 2007: Hypercube-based encoding.

Substrate is a template for a possibly large-scale neural network.

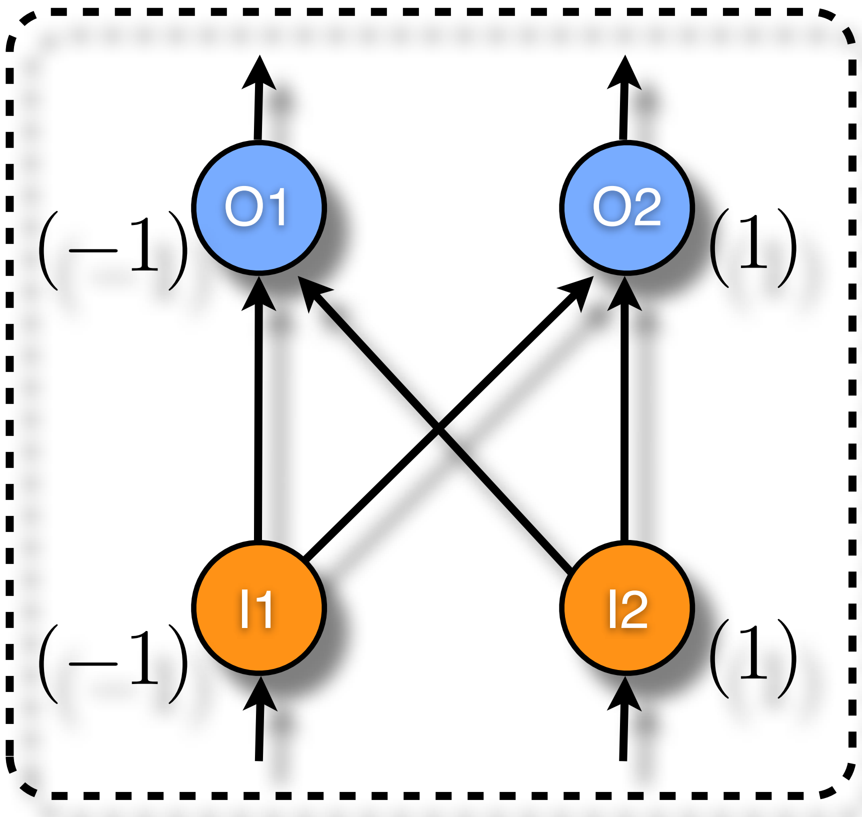


HyperNEAT

- Stanley et al. 2007: Hypercube-based encoding.

Each neuron is assigned coordinates. The weights of connections are unknown.

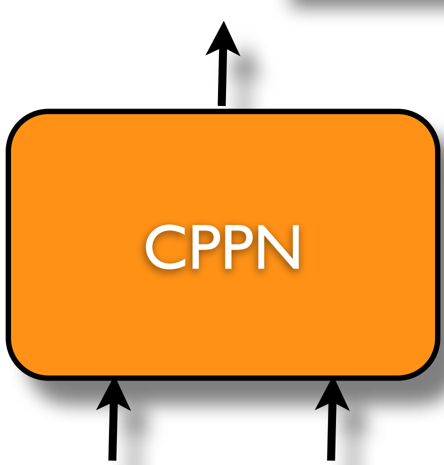
substrate

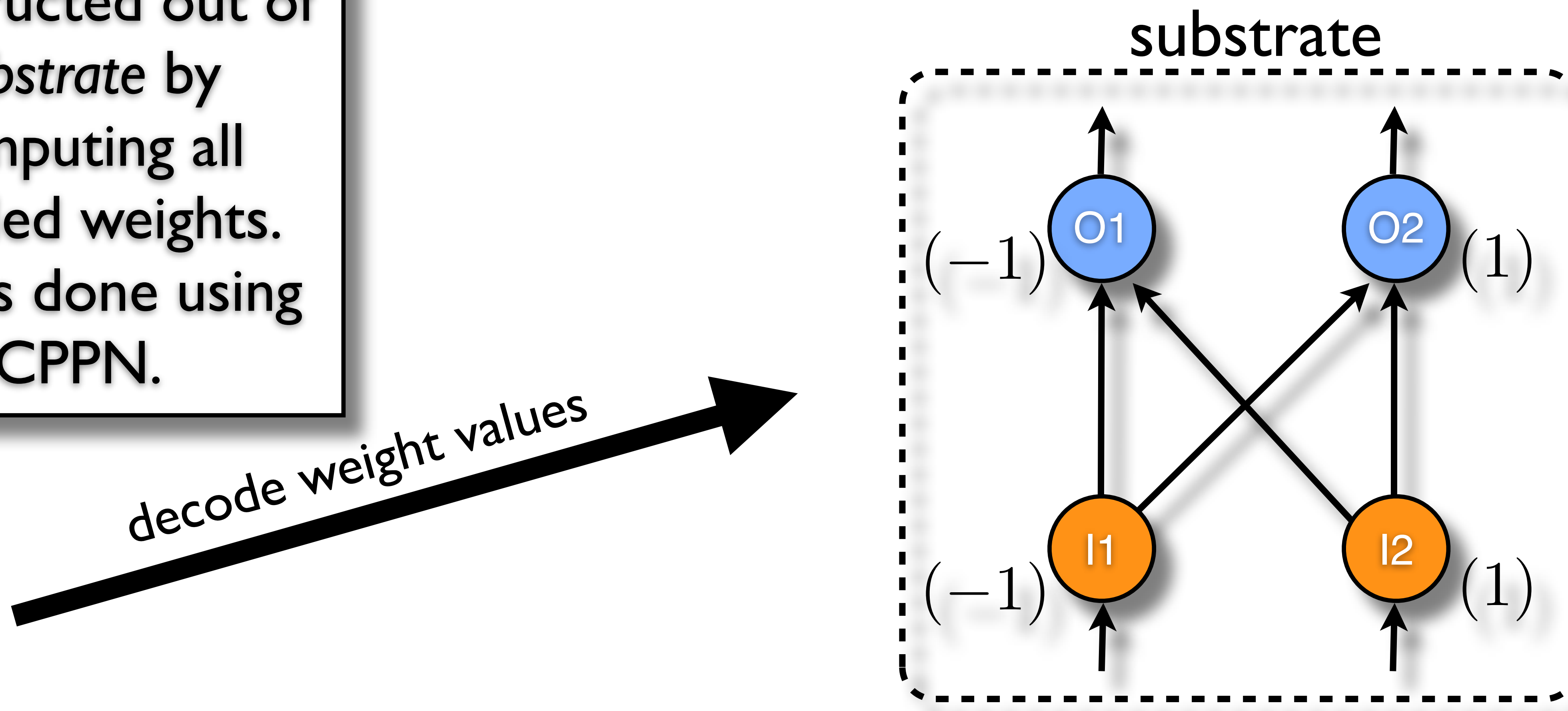


HyperNEAT

- Stanley et al. 2007: Hypercube-based encoding.

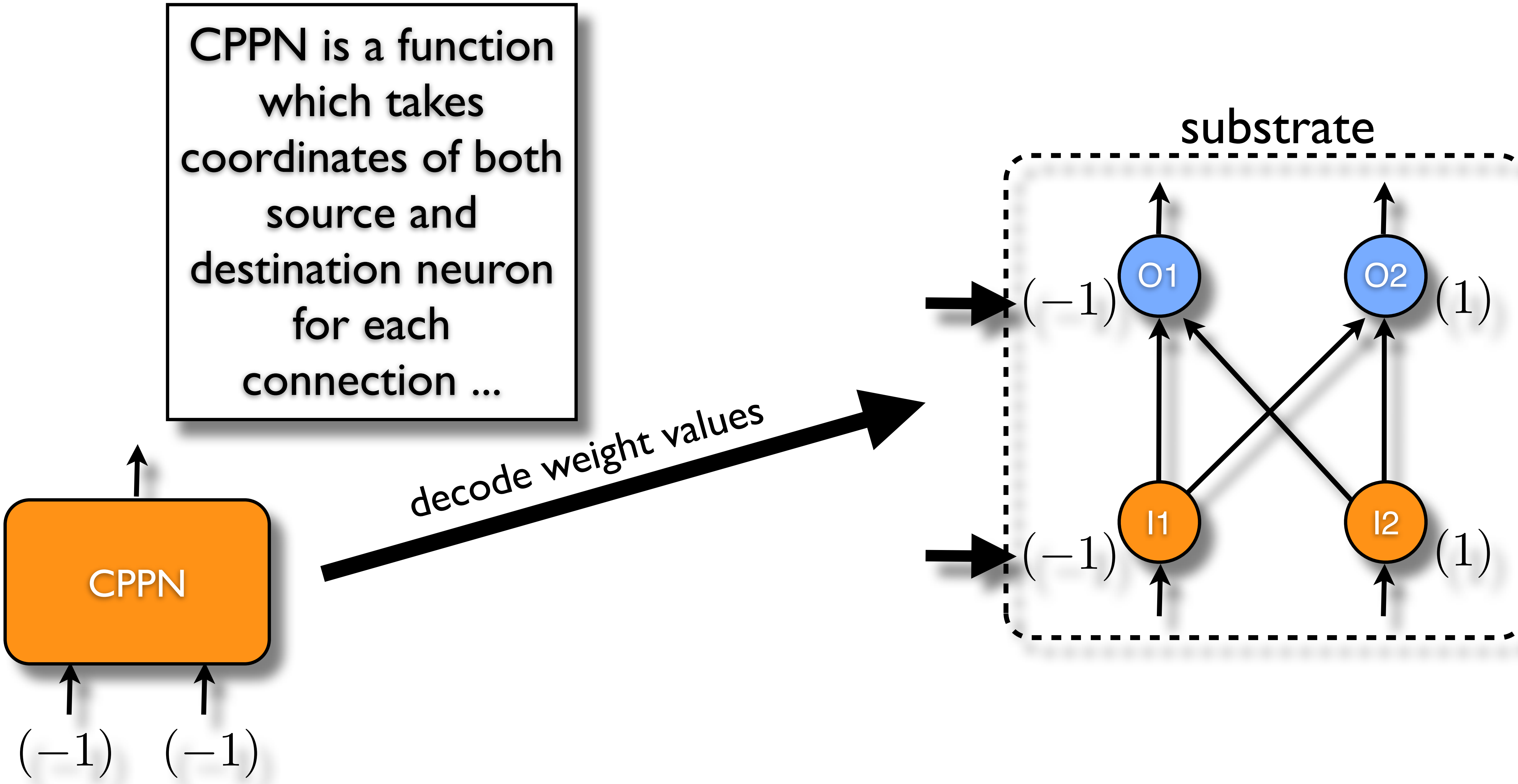
The *final network* is constructed out of *substrate* by computing all needed weights. This is done using CPPN.





HyperNEAT

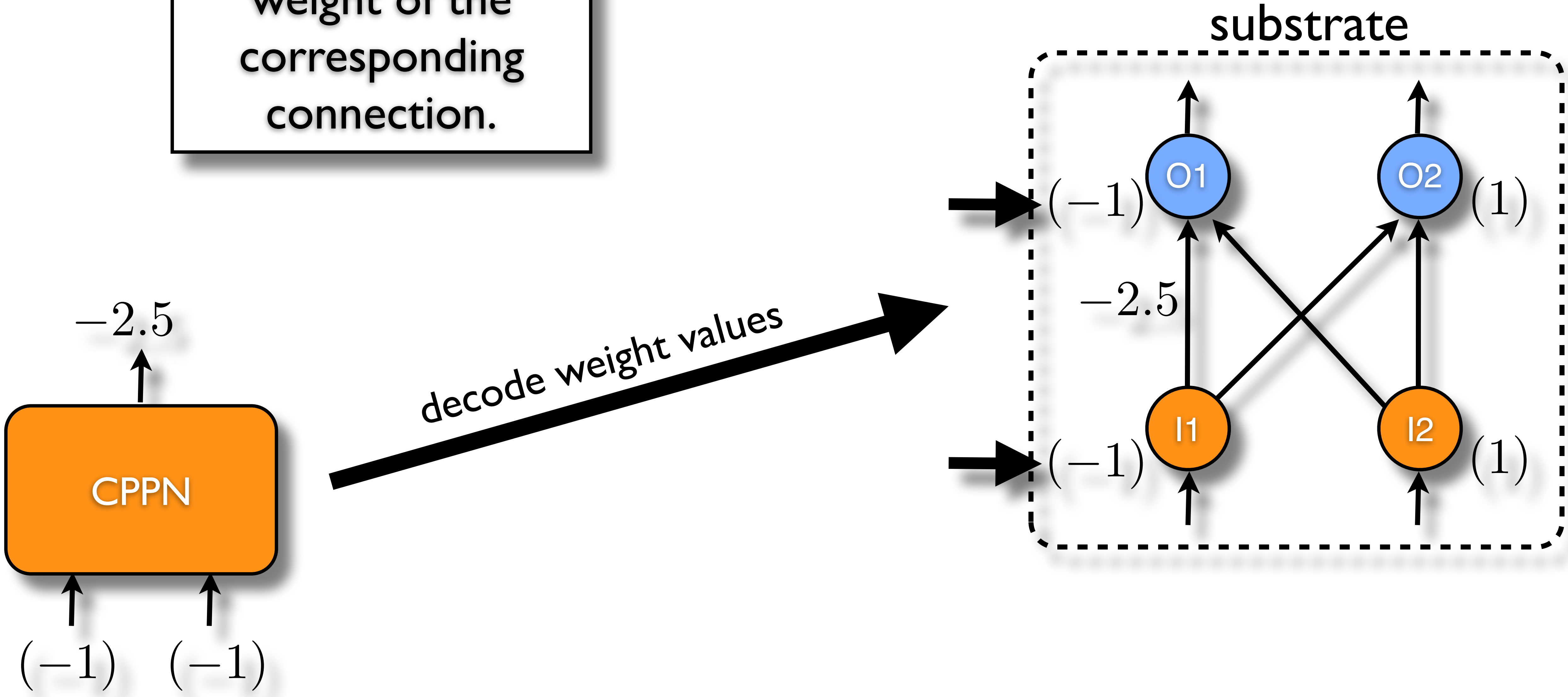
- Stanley et al. 2007: Hypercube-based encoding.



HyperNEAT

- Stanley et al. 2007: Hypercube-based encoding.

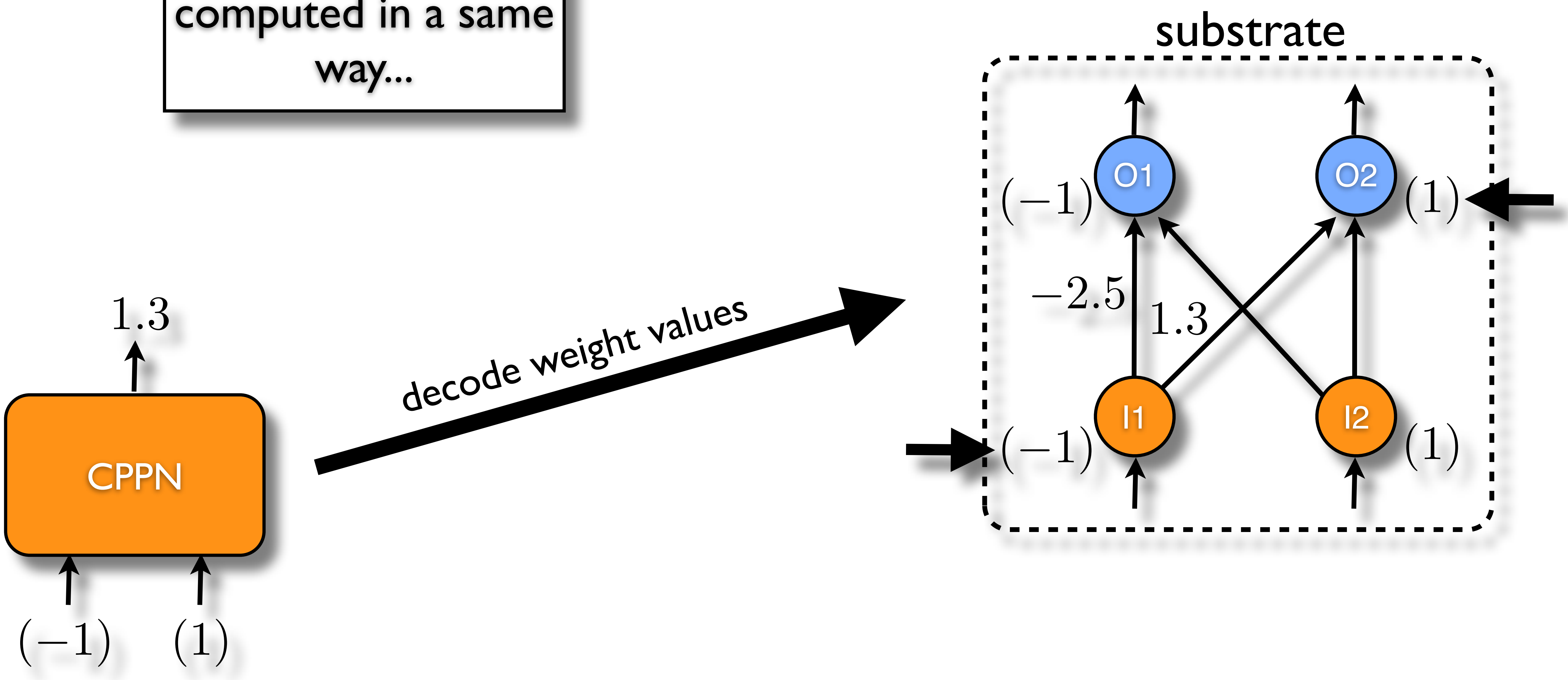
... and computes the weight of the corresponding connection.



HyperNEAT

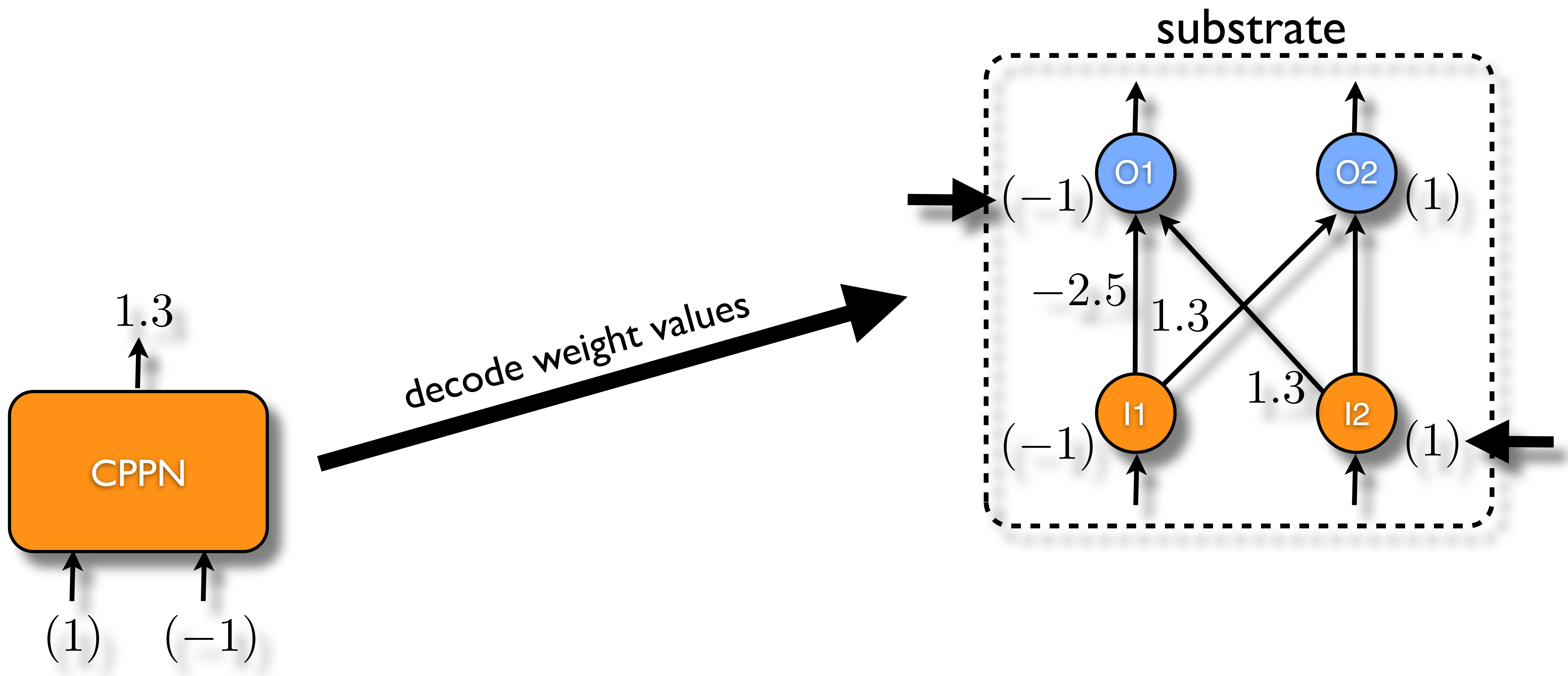
- Stanley et al. 2007: Hypercube-based encoding.

All weights are computed in a same way...



HyperNEAT

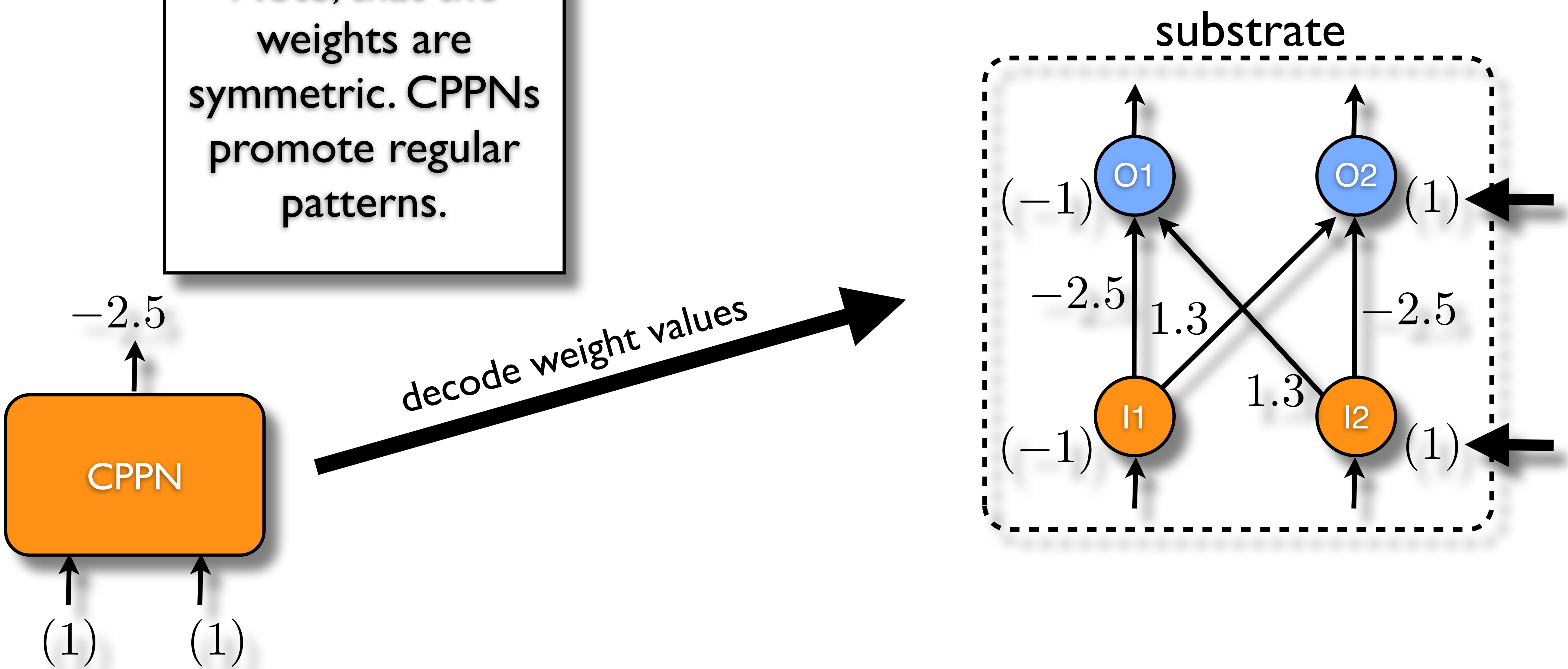
- Stanley et al. 2007: Hypercube-based encoding.



HyperNEAT

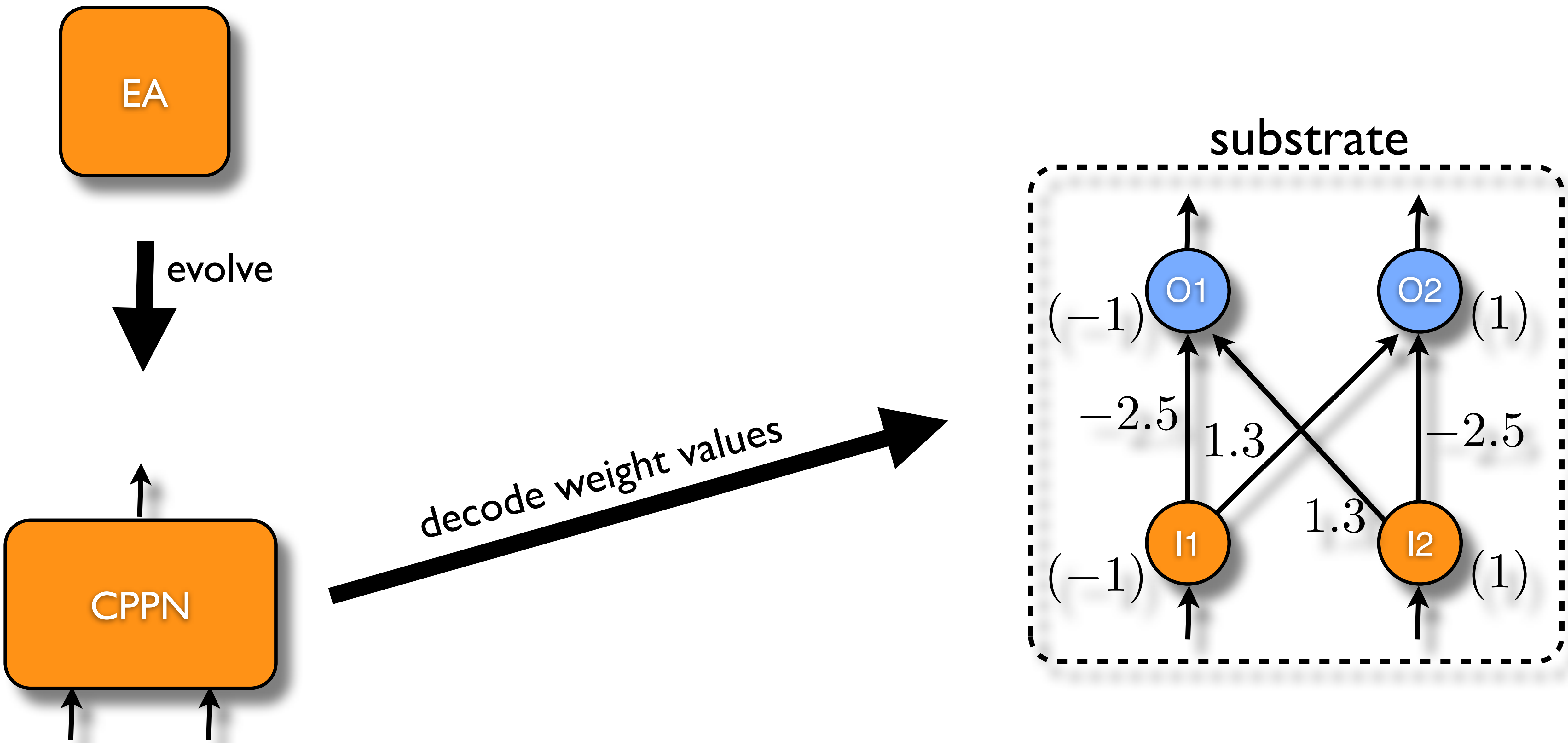
- Stanley et al. 2007: Hypercube-based encoding.

Note, that the weights are symmetric. CPPNs promote regular patterns.



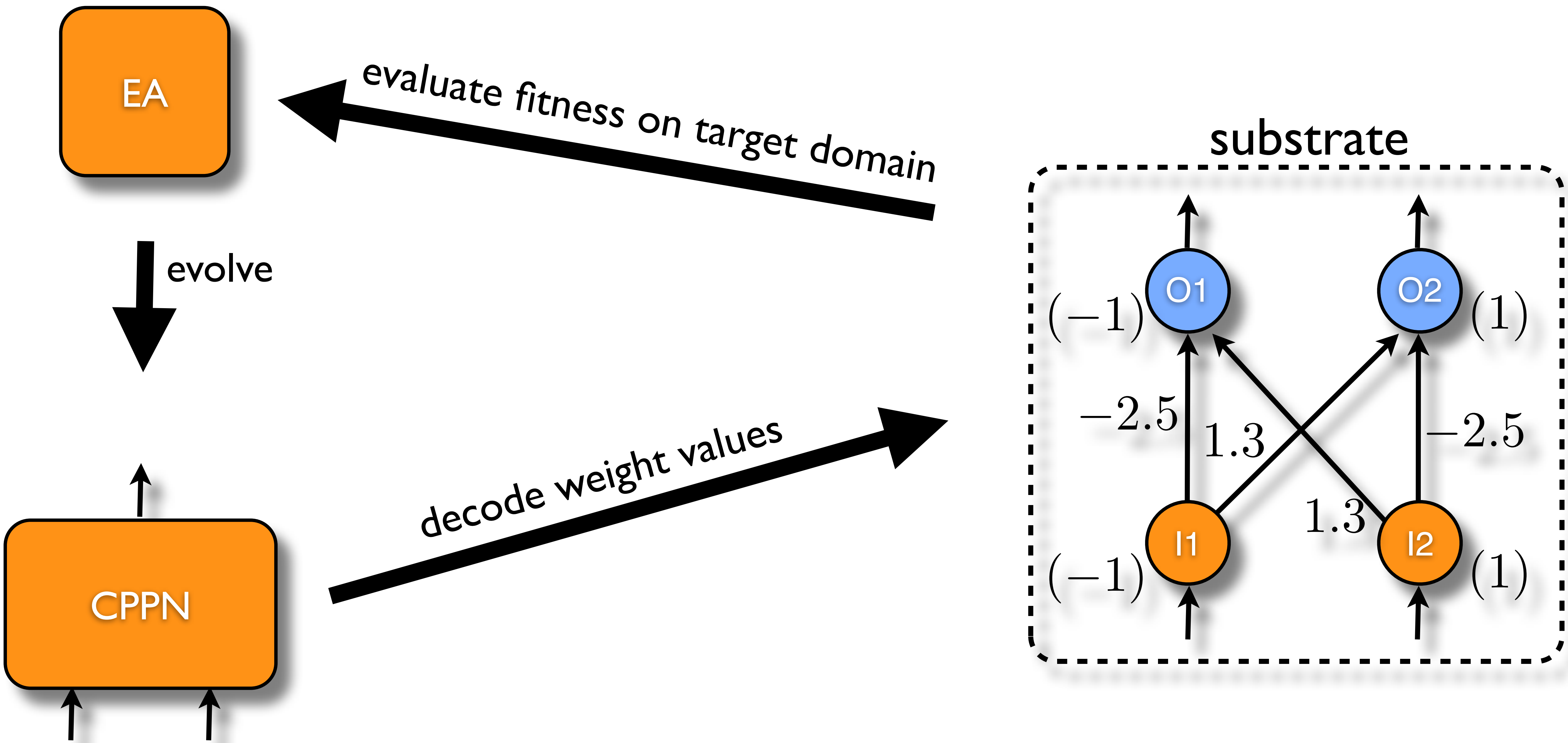
HyperNEAT

- Stanley et al. 2007: Hypercube-based encoding.

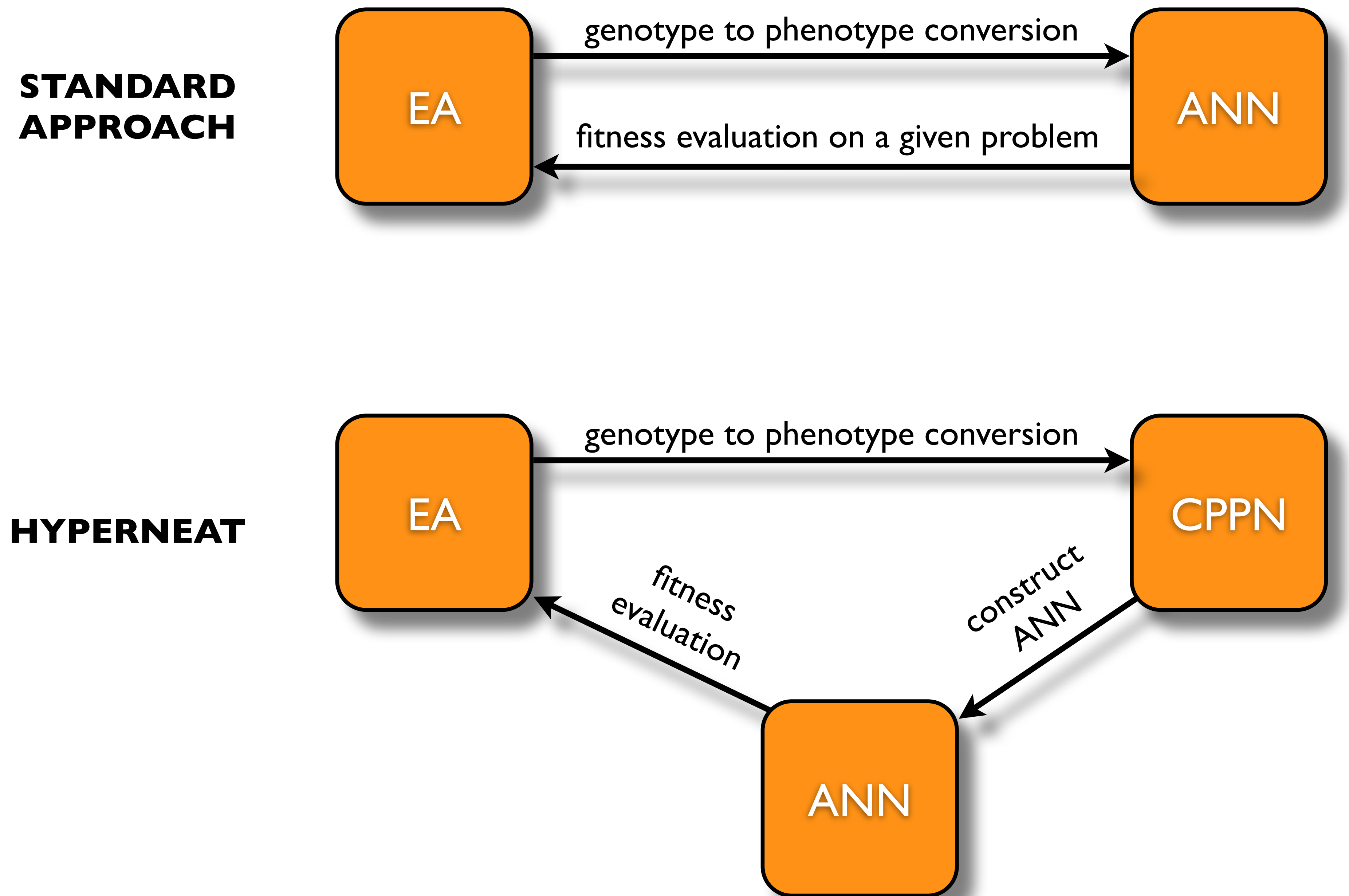


HyperNEAT

- Stanley et al. 2007: Hypercube-based encoding.

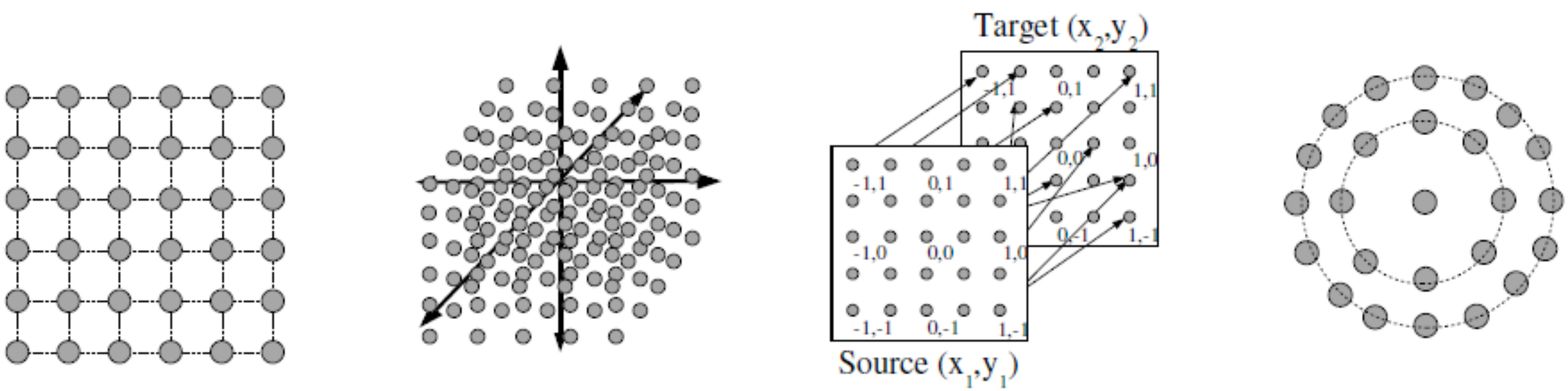


HyperNEAT vs. Standard Approaches



Types of Substrate?

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



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

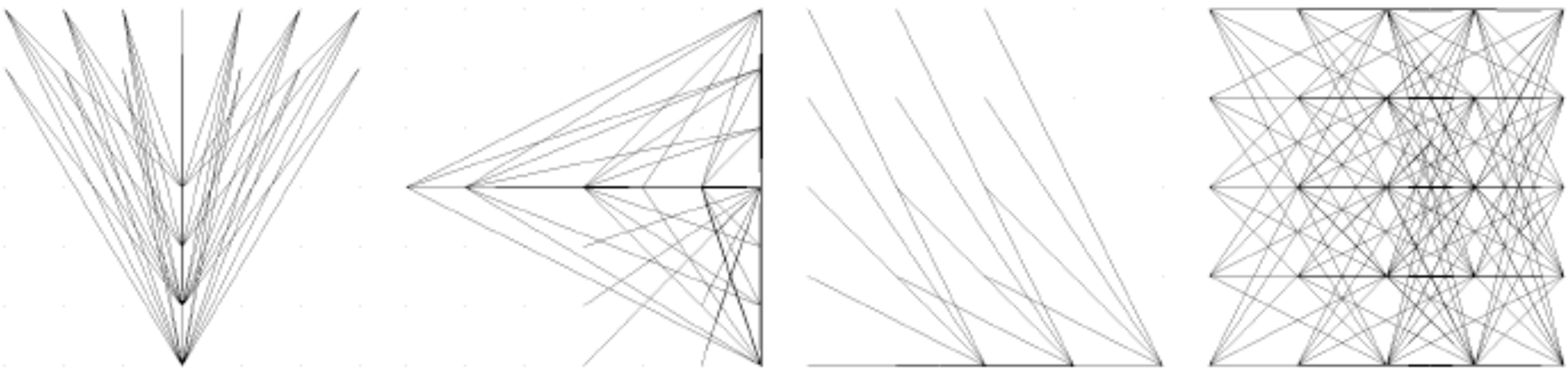
Create or not Create a Link?

- 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

- Patterns evolved using interactive evolution:



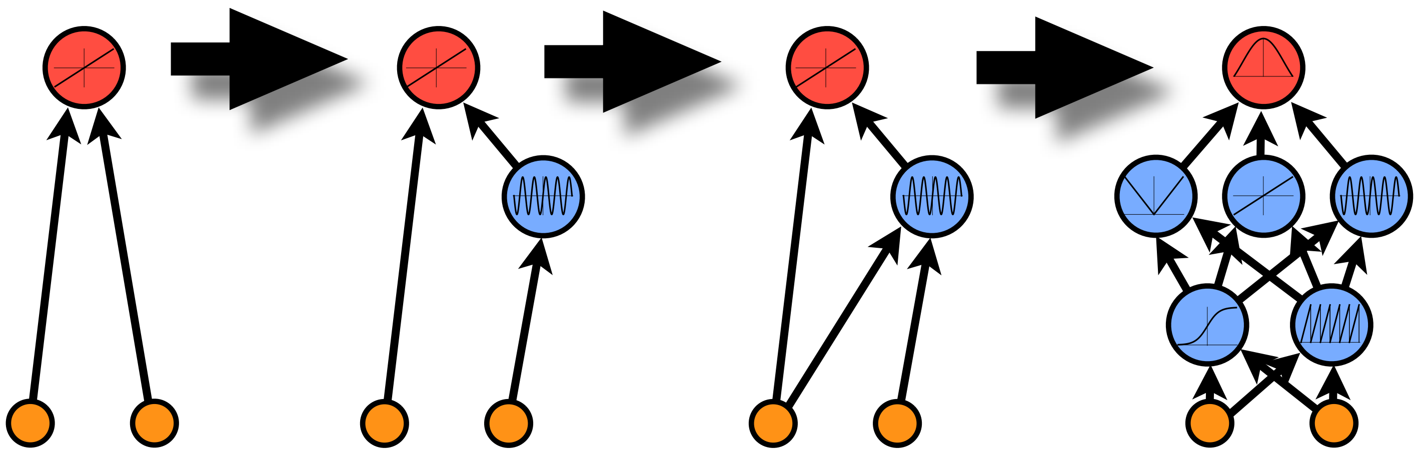
(a) Sym. (b) Imperf. (c) Repet. (d) Var.

Spatial Representation

- HyperNEAT **exploits spatial representation of a problem**. The same happens in 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.

NEAT in HyperNEAT

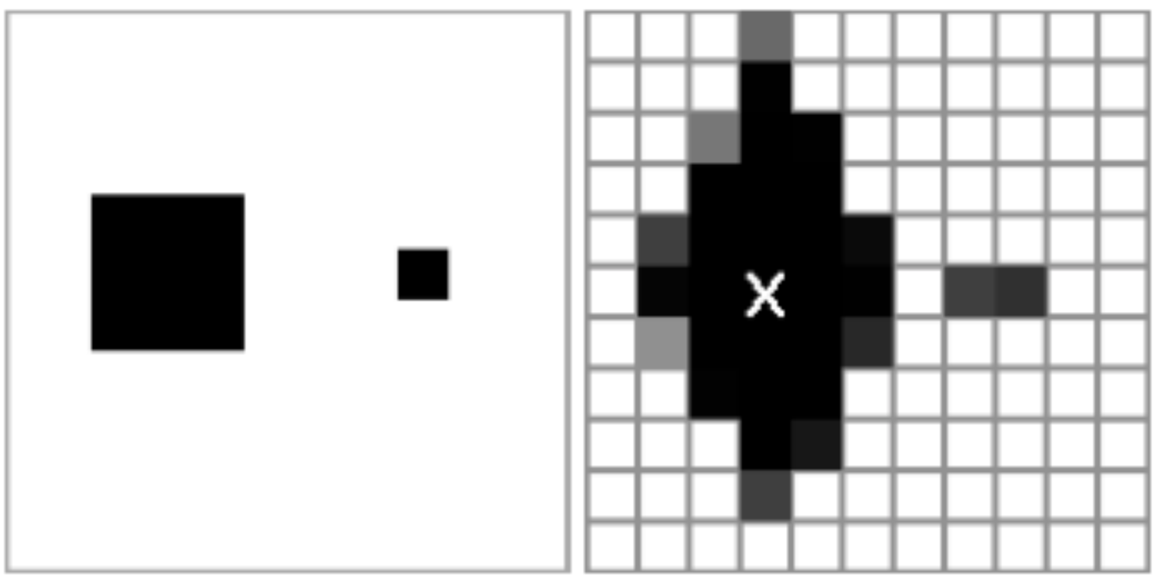
- HyperNEAT uses a slightly modified NEAT (Stanley 2001) as a **base algorithm** to evolve CPPNs.
- NEAT is neuro-evolutionary algorithm able to evolve ANNs of arbitrary topologies.



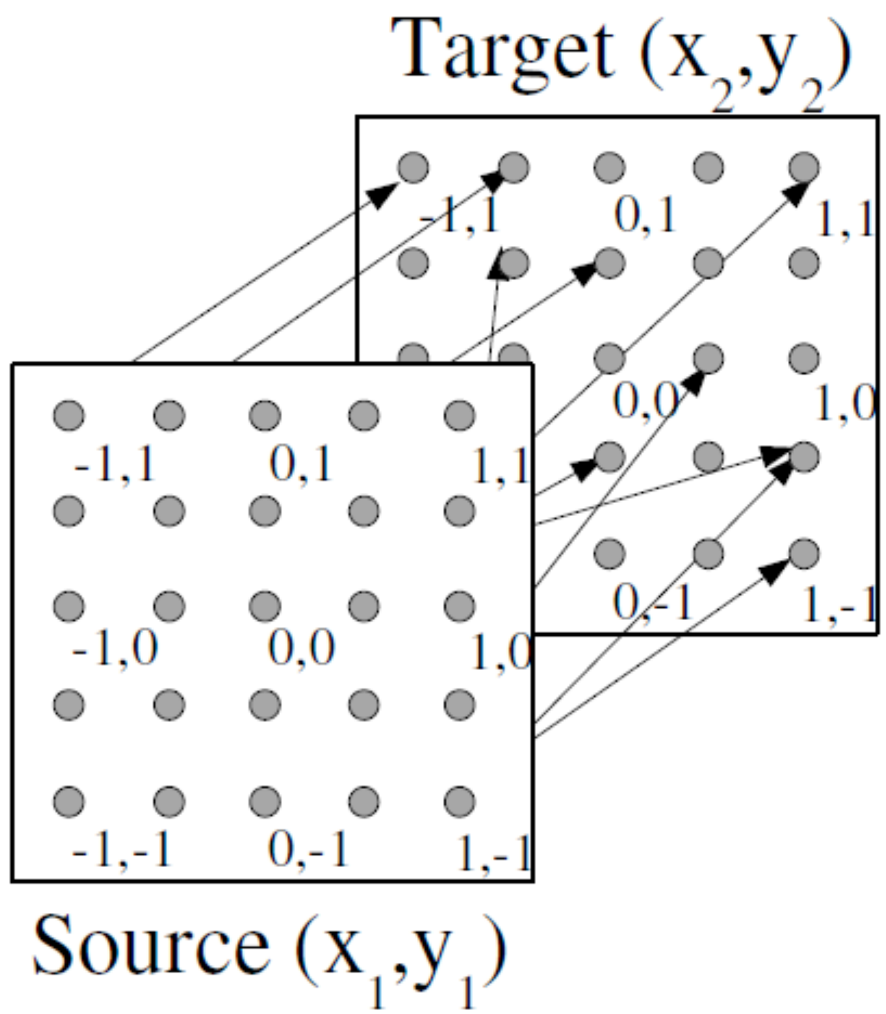
- It is based on:
  - **complexification** → evolving gradually more complex ANNs,
  - **innovation numbers** → track structural innovations,
  - **niching** → allows simultaneous evolution of small and large ANNs in one population. **Requires to define a distance measure for ANNs.**

Visual Discrimination

- Visual targeting: distinguish the larger object.
- "Sandwich substrate".



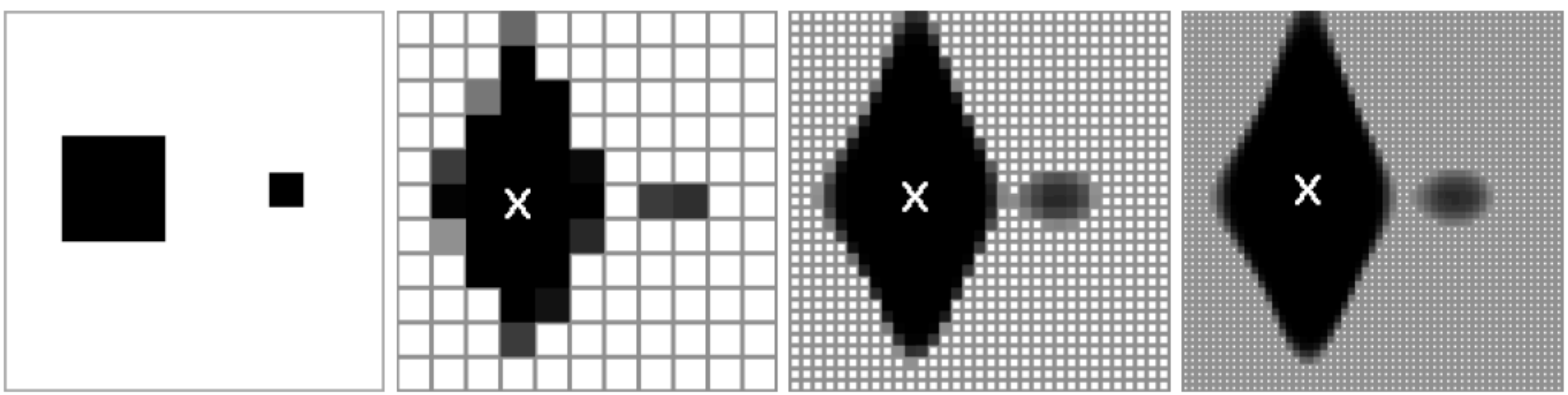
(a) Sensor Field (object placement)   (b) 11x11 Target Field (12,827 connections)



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

Visual Discrimination II: Scaling the Substrate

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



(a) Sensor Field (object placement)

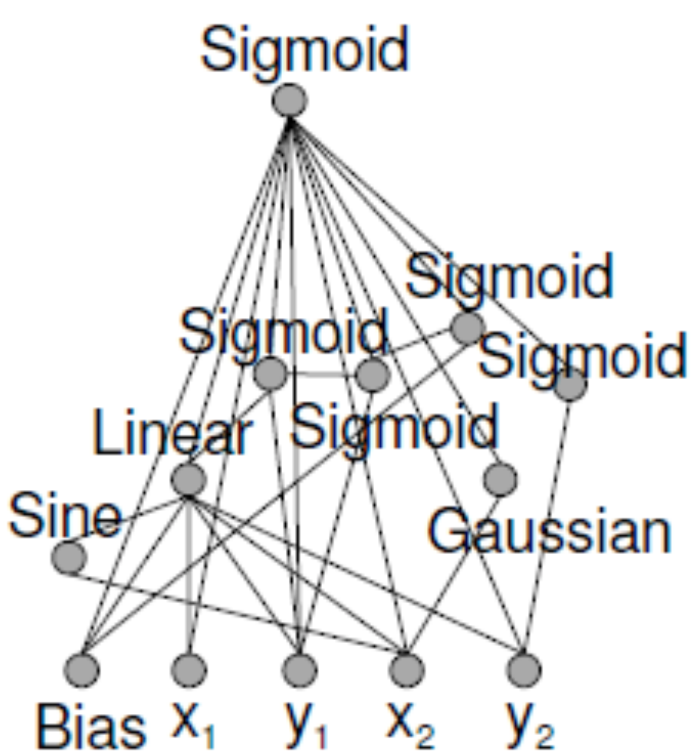
(b) 11x11 Target Field (12,827 connections)

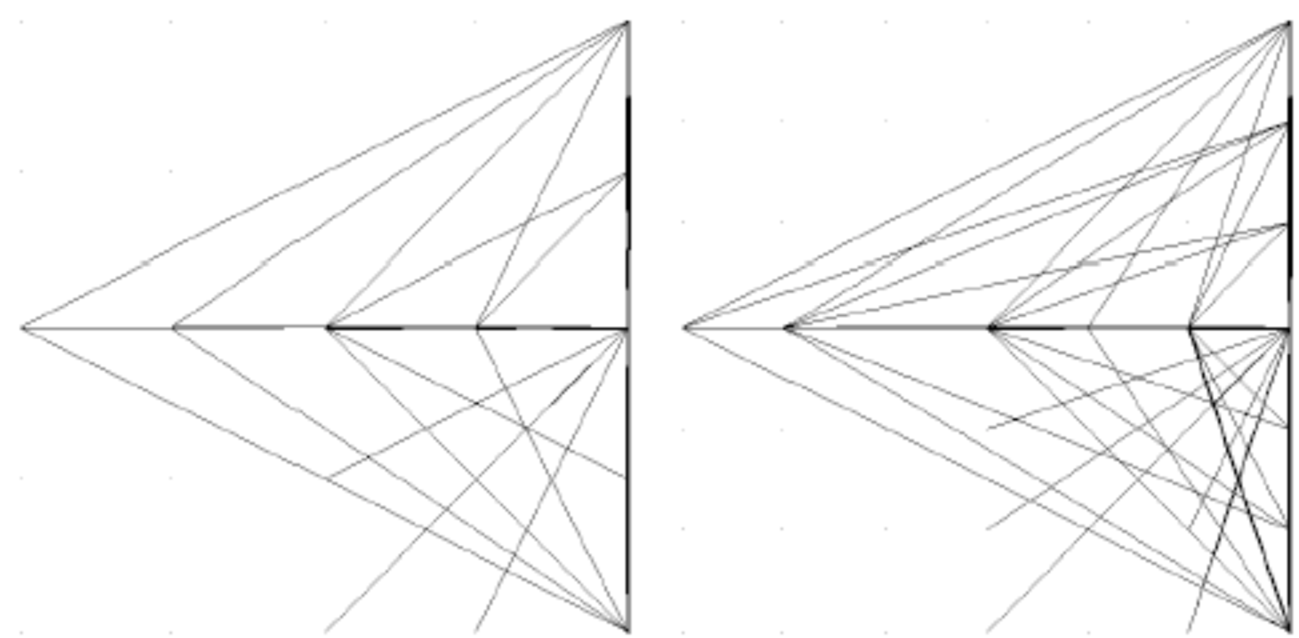
(c) 33x33 Target Field (1,033,603 connections)

(d) 55x55 Target Field (7,970,395 connections)

Visual Discrimination III: Scaling the Substrate

- An equivalent connectivity concept at different
- substrate resolutions.

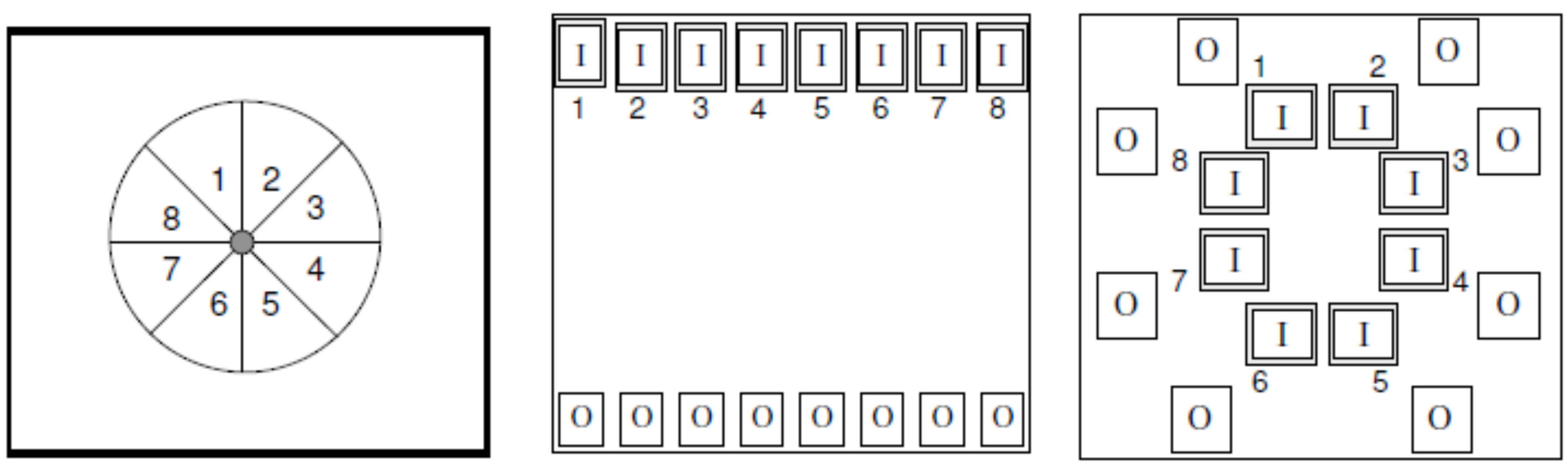




(a)5 × 5 Concept (b) 7 × 7 Concept

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

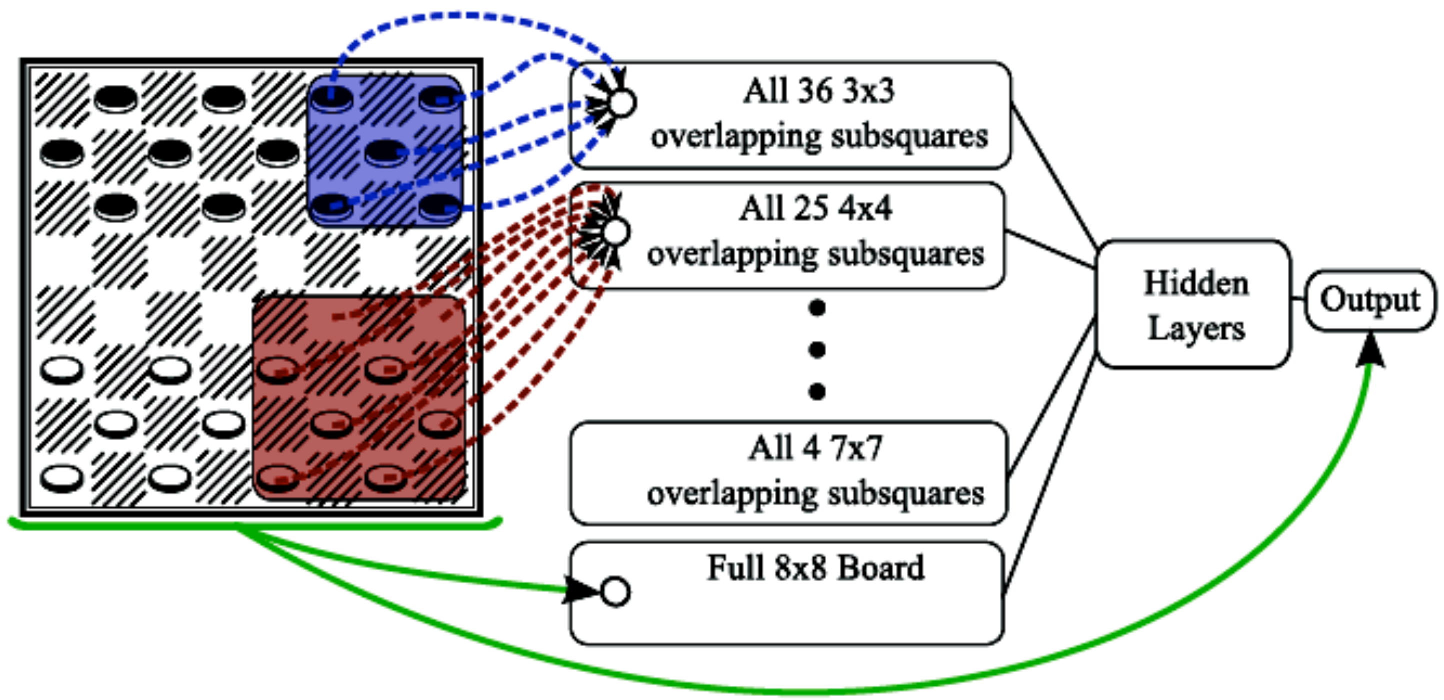
David B. D'Ambrosio and Kenneth O. Stanley (2007)
A Novel Generative Encoding for Exploiting Neural Network Sensor and Output Geometry

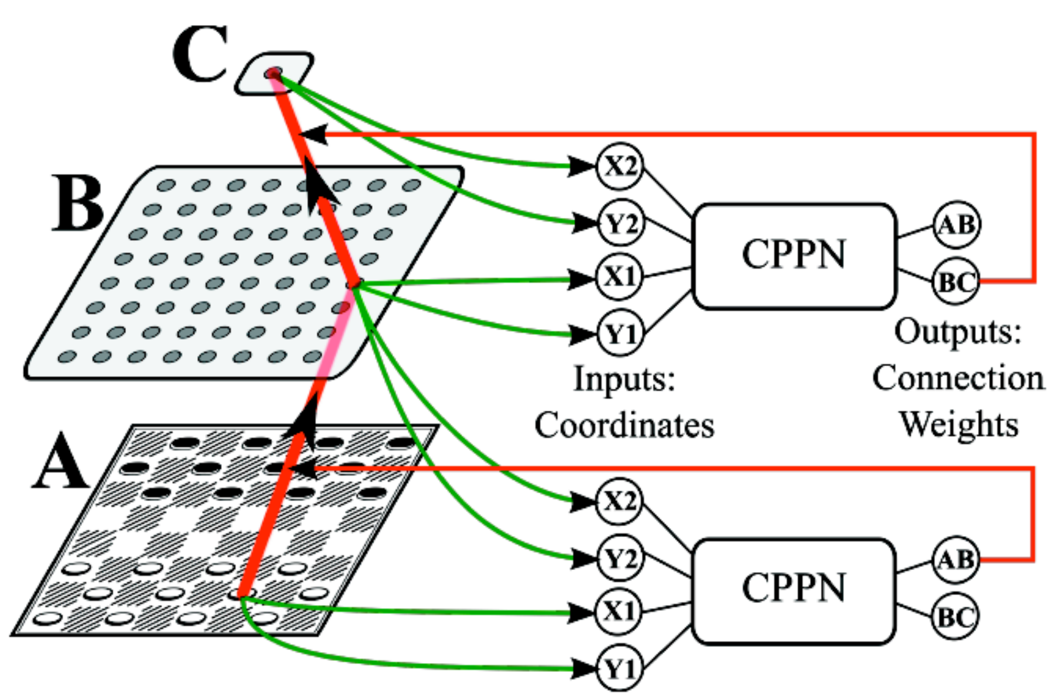
Food Gathering Problem II

- Parallel worked better than Concentric because less computation is needed for CPPN.
- New CPPN inputs added: the *distances*
- *(x1-x2)* and *(y1-y2)*
- When CPPN is provided the distances, both work the same.

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.

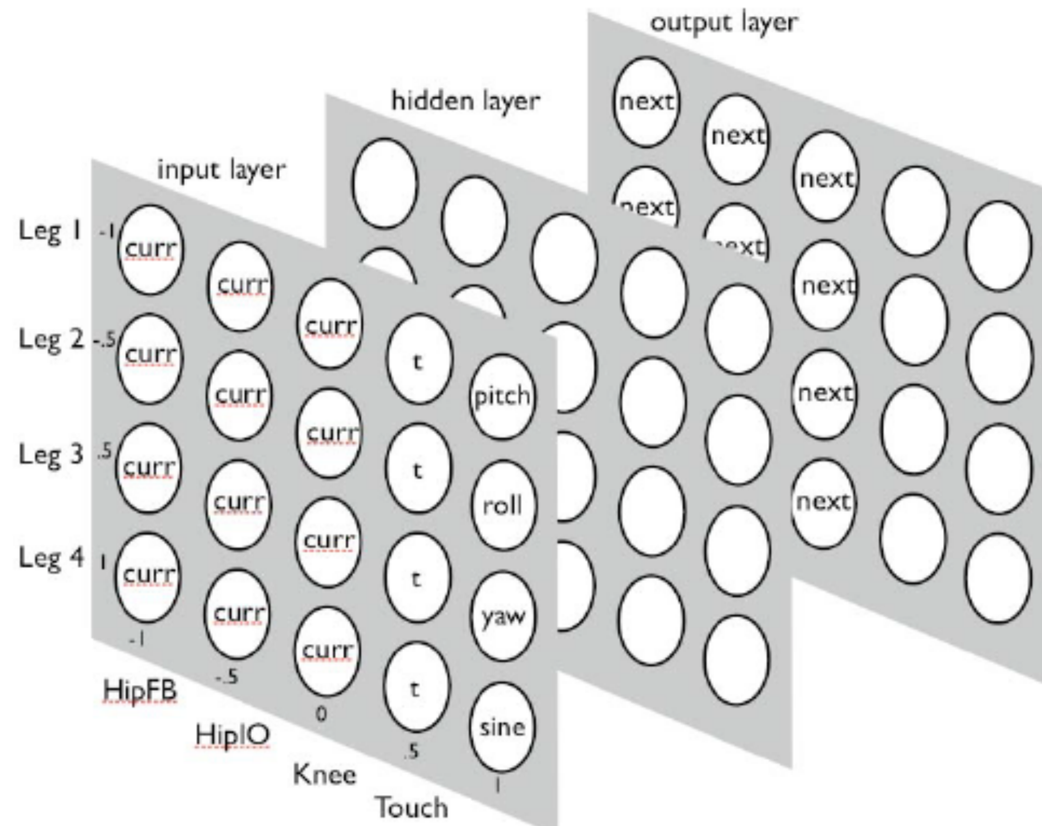




Jason Gauci and Kenneth O. Stanley (2008):
A Case Study on the Critical Role of Geometric Regularity in Machine Learning

HyperNEAT Coordinated Quadruped Gaits

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

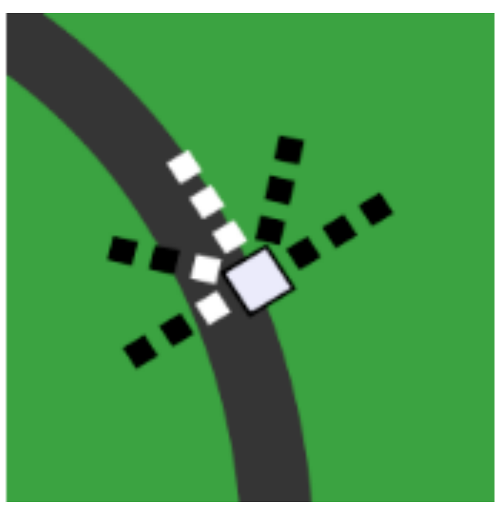


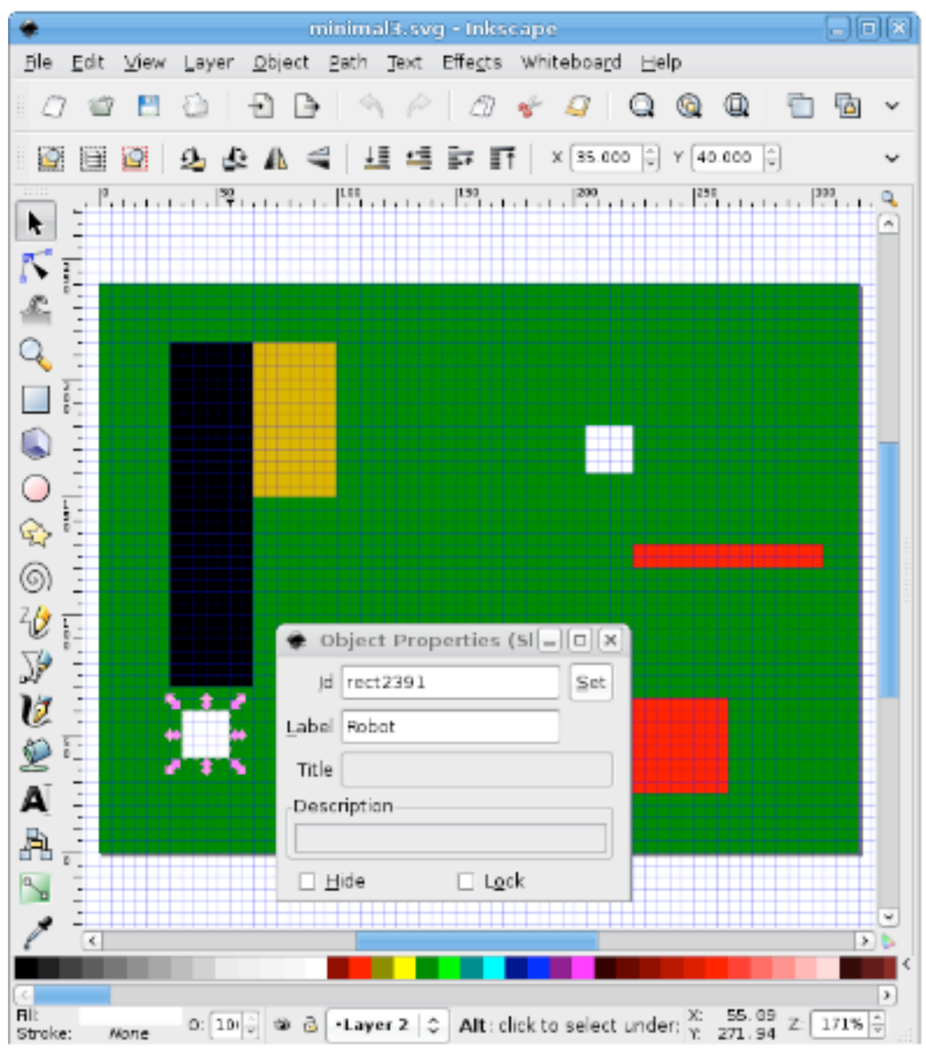
COMPUTATIONAL INTELLIGENCE GROUP

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

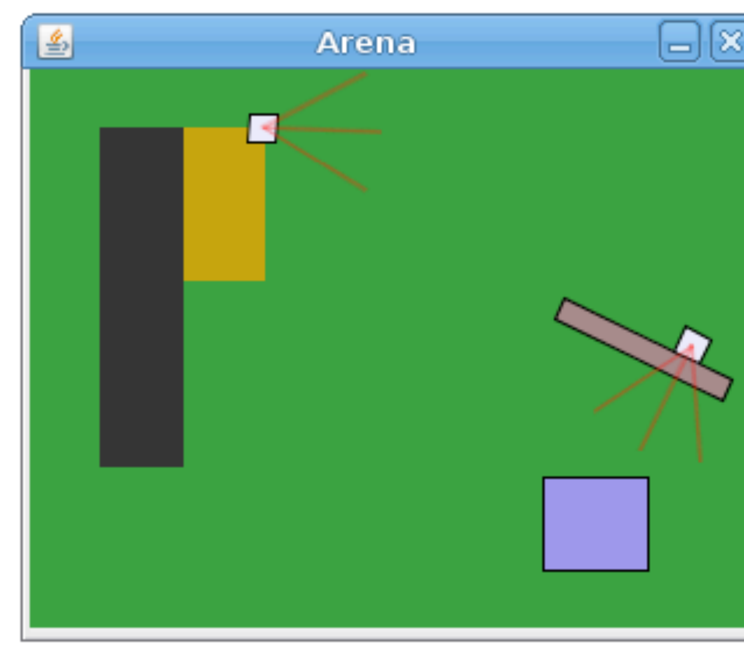
Mobile Robot Navigation

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





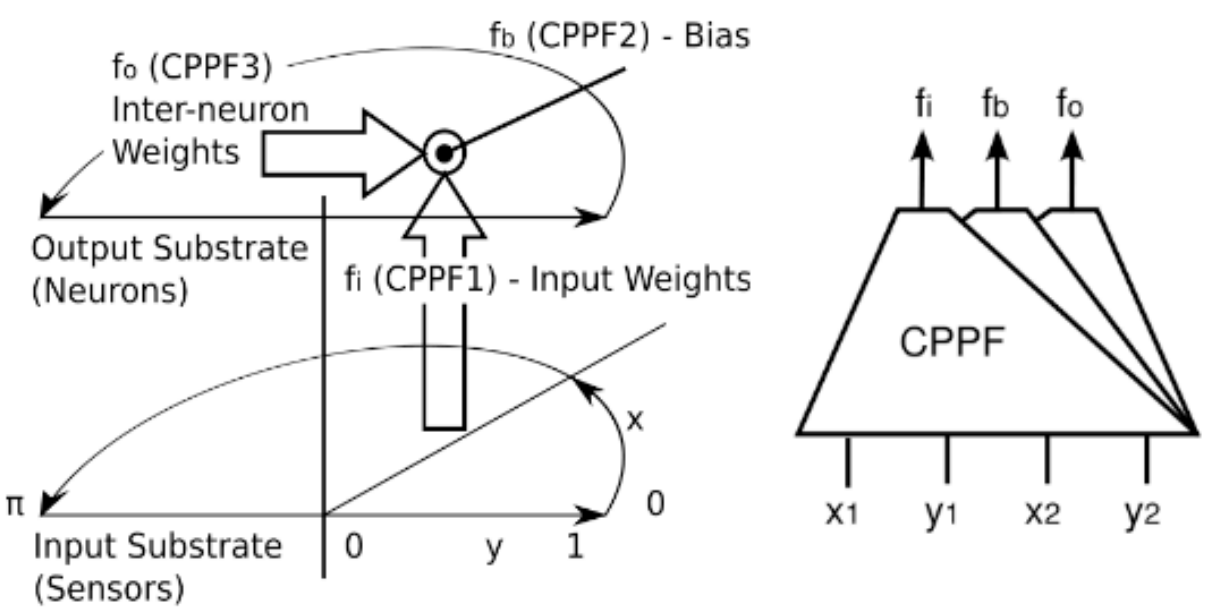
→

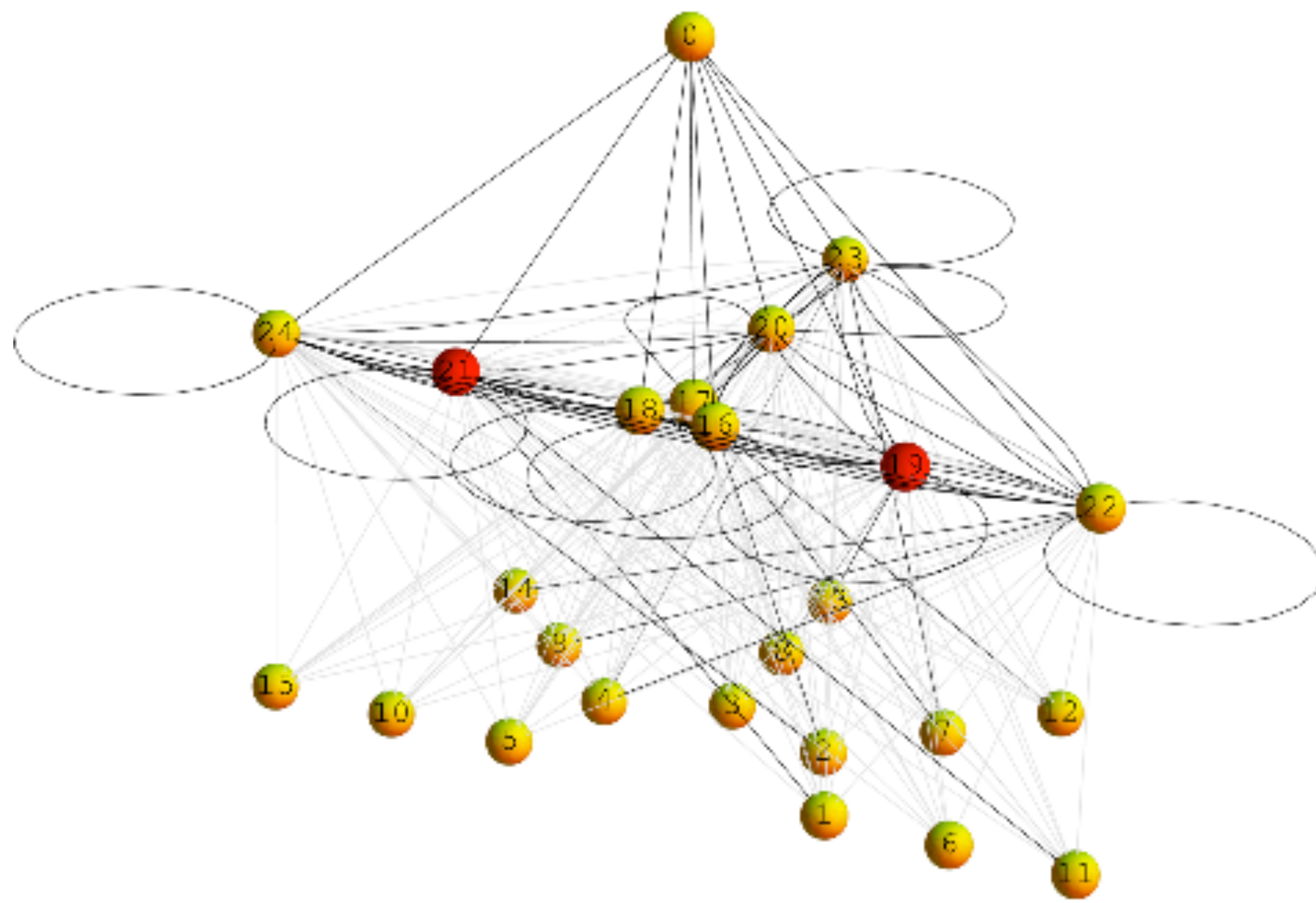


COMPUTATIONAL INTELLIGENCE GROUP

Mobile Robot Navigation II

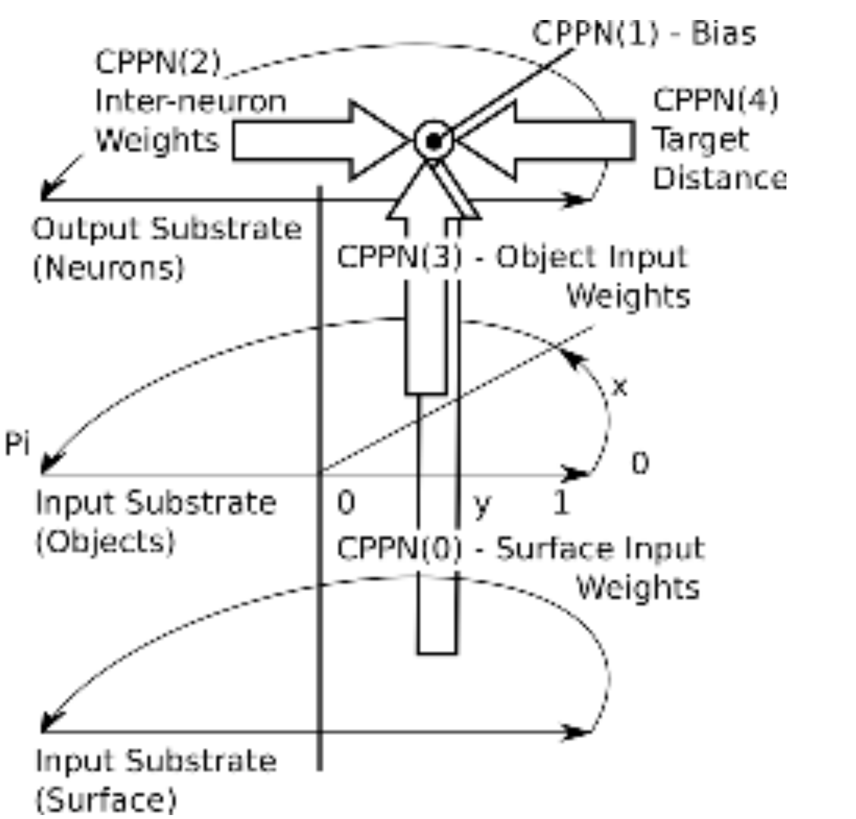
- Substrate uses polar coordinates.
- Input + 1 fully recurrent layer
- See VIDEO...

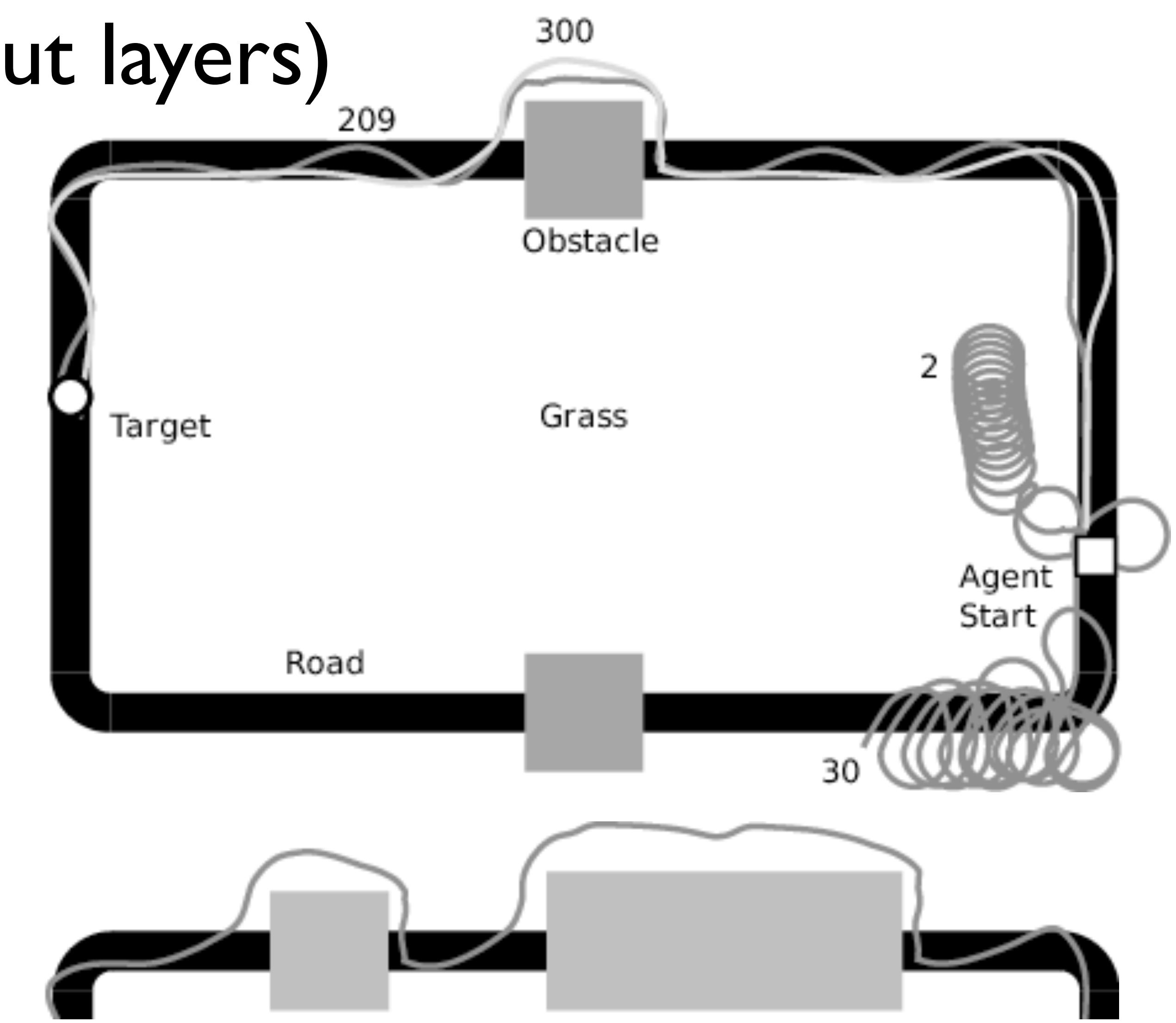




Mobile Robot Navigation III

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





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

Q&A