Bio-Inspired Artificial Intelligence

Lecture 8: Evolutionary Robotics

http://www.informatik.uni-hamburg.de/WTM/

Motivation

- Artificial neural networks (ANN) are a good candidate for controllers of autonomous agents
  - Ability to ***learn*** and ***adapt*** to dynamic environment
  - Tolerance to noise, ***robustness***
  - Can represent complex functions through recurrent/lateral connections and non-linear transfer functions

But: Complex networks are difficult to design. How to select necessary topology, weights, and efficient learning parameters?

Slides partially based on: *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies* by Floreano & Mattiussi, MIT 2008.

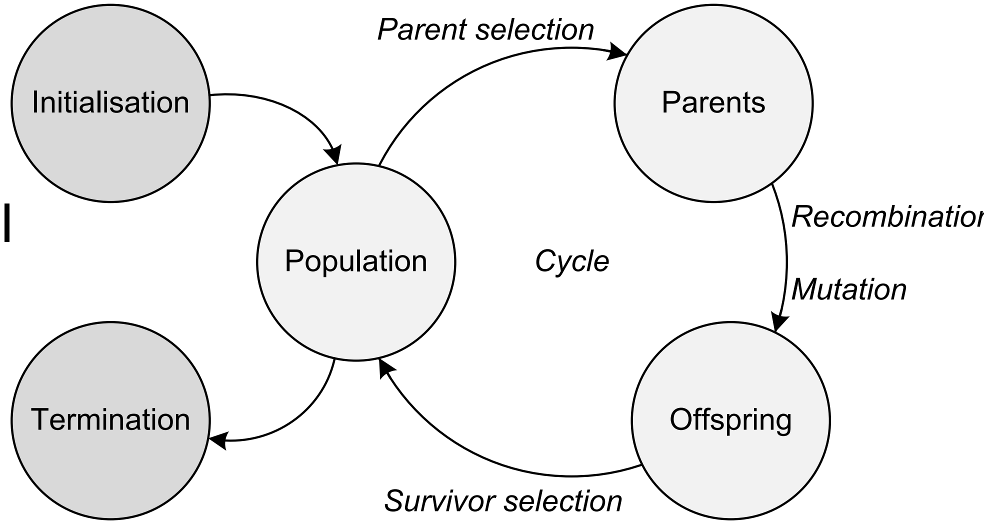
Motivation

- Creating neural network controllers by hand requires
  - ***Expert knowledge*** about the dynamics of the neural network
  - Sufficient knowledge about the ***problem*** to be solved

- ***Evolution*** can find complex solutions by sampling possible candidates
  - Gradual changes necessary, which ANN provide
  - Topology, weights and other parameters can be evolved at the same time
  - Fitness can be determined by testing and does not necessarily require detailed knowledge about possible solution ("What" often more interesting than "How")

Evolution Recap

- Evolution happens over generations, not within an agent's lifetime (≠ Learning)
- Genotype-Phenotype mapping
- Several operators
  - Selection
    - Works on population level
    - Uses fitness values
  - Mutation
    - Introduces small gradual change
  - Recombination
    - Allows for "jumps" on the fitness landscape



Questions to answer

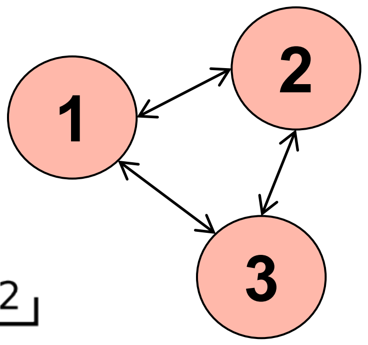
- What is necessary to evolve neural networks?
  - Mutation operators
    - Which parameters of the network can mutate and how?
    - How to mutate the gene to ensure gradual change?
  - Recombination
    - What does a cross-over operator do to a neural net?
    - How can “good” parts be conserved
  - Genetic encoding
    - What to encode, and how?

Genetic Encoding

- What do we have to encode?
  - Connection Weights: real value for each edge
  - Parameters of transfer function: n values for each node
  - Topology: fixed or under evolutionary control

- Fixed Topology
  - Nodes + Weight Table

$$\underbrace{x_1, y_1,}_{\text{Node 1}}\underbrace{x_2, y_2,}_{\text{Node 2}}\underbrace{x_3, y_3,}_{\text{Node 3}}\underbrace{\omega_{12}, \omega_{21}, \omega_{13}, \omega_{31}, \omega_{23}, \omega_{32}}_{\text{Weight Table}}$$

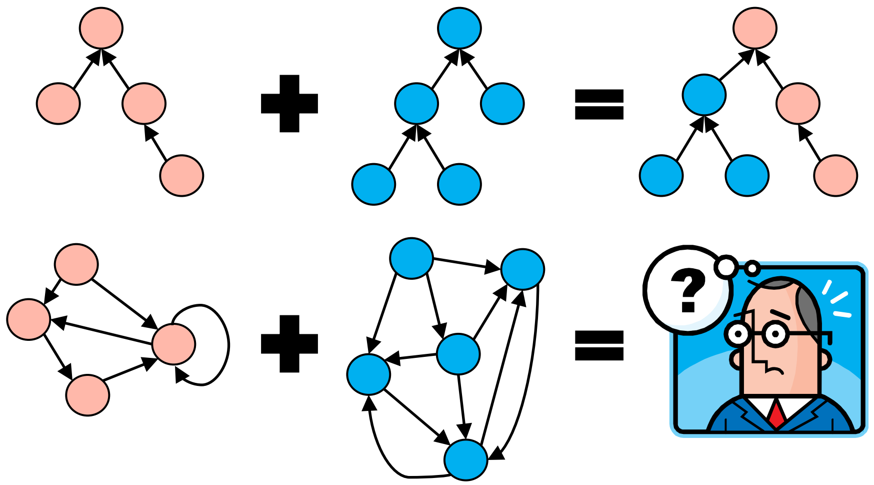


  - Nodes and incoming weights combined

$$\underbrace{x_1, y_1, \omega_{21}, \omega_{31},}_{\text{Node 1}}\underbrace{x_2, y_2, \omega_{12}, \omega_{32},}_{\text{Node 2}}\underbrace{x_3, y_3, \omega_{13}, \omega_{23}}_{\text{Node 3}}$$

Recombination

- What does recombination mean for the two solutions?



- Recombination is often destructive; favourable functions are not transferred to offspring, leading to low fitness
- Has to be carefully designed, depending on topology and representation/encoding using domain knowledge

Mutation Operators

- Using only standard mutation (small changes to values in the gene) without recombination, evolution becomes a parallel gradient search
- Usually several mutation operators applied for different types of mutation

- General operators
  - Change parameter values
  - Add/delete nodes
  - Add/delete connections

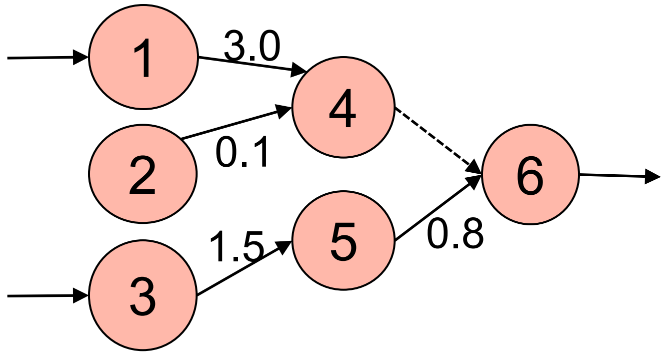
Basic Mutation

- Two possible variants
  1. With ***mutation rate (chance of mutation)***:
     1. go through all loci
     2. check if locus should be mutated
     3. add value from normal distribution centred on old value or use other mutation if not real value
  2. Mutate ***everything*** using ***normal distribution***
     1. go through all loci
     2. add value from normal distribution centred on old value

- Variance of distribution should be smaller in second case

Mutating Connections

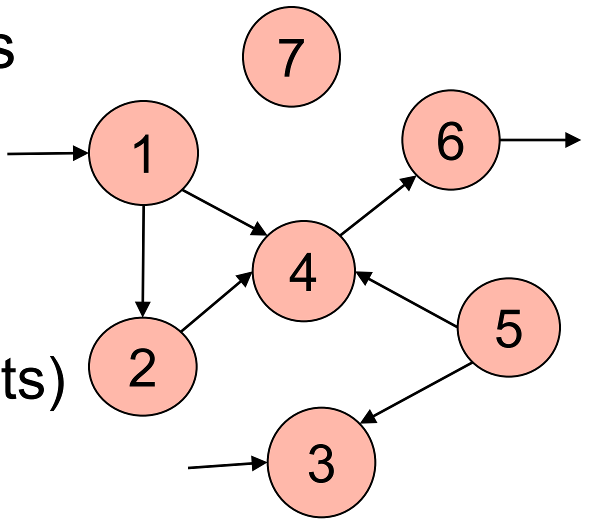
- Naïve random Method
  - Addition: select 2 random nodes without connection, add connection with random weight
  - Removal: remove random connection



- ***Better methods***
  - Removal:
    - Probability inversely ***proportional*** to weight of connection
    - Remove random connection with weight below threshold
  - Addition:
    - Add connection with weight zero or low weight
    - Select nodes by taking into account current connection scheme

Mutating Number of Nodes

- naïvely adding/deleting random nodes leads to hugely varying results

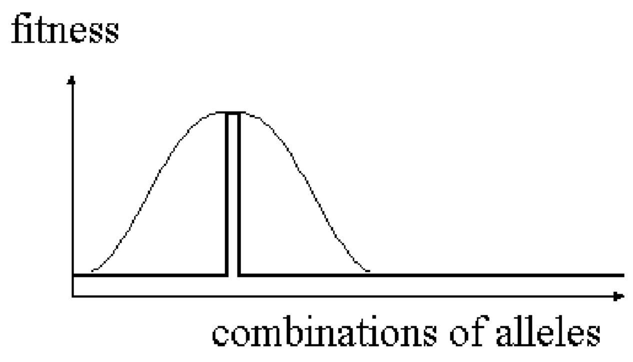


- Things to consider for removal:
  - Outgoing connections (with low weights)
  - Number of incoming connections
  - Node parameters

- General suggestions
  - Calculate probability depending on criteria
  - Chance of removal higher than addition to avoid unnecessary growth

Evolution and Learning

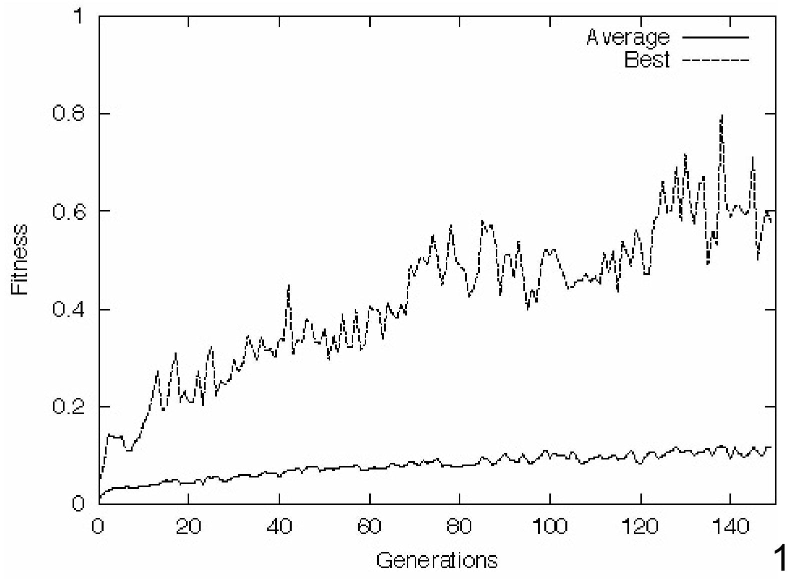
- Evolution can be combined with neural network learning
  - Evolve parameters of learning rules (e.g. Learning rate)
  - What learning rules to use on node to node basis
  - Learning as local search in hybrid approach
  - Lamarkian Evolution – Feedback of adaptations into genes



- Interaction between learning and evolution
  - Learning can smoothen fitness landscape
  - Chance to improve fitness in changing environments

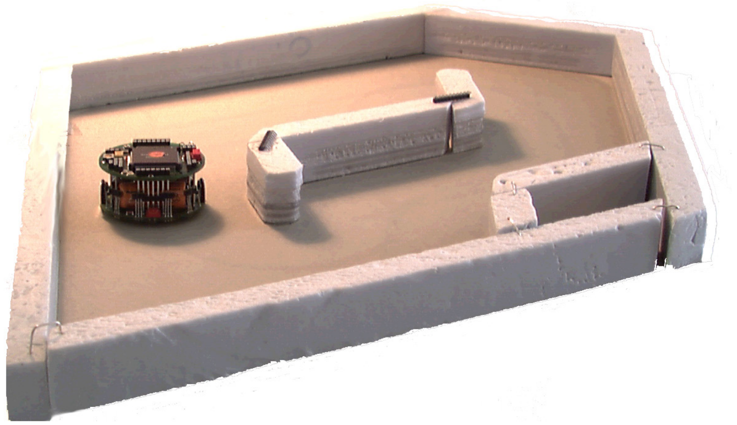
Many options…..

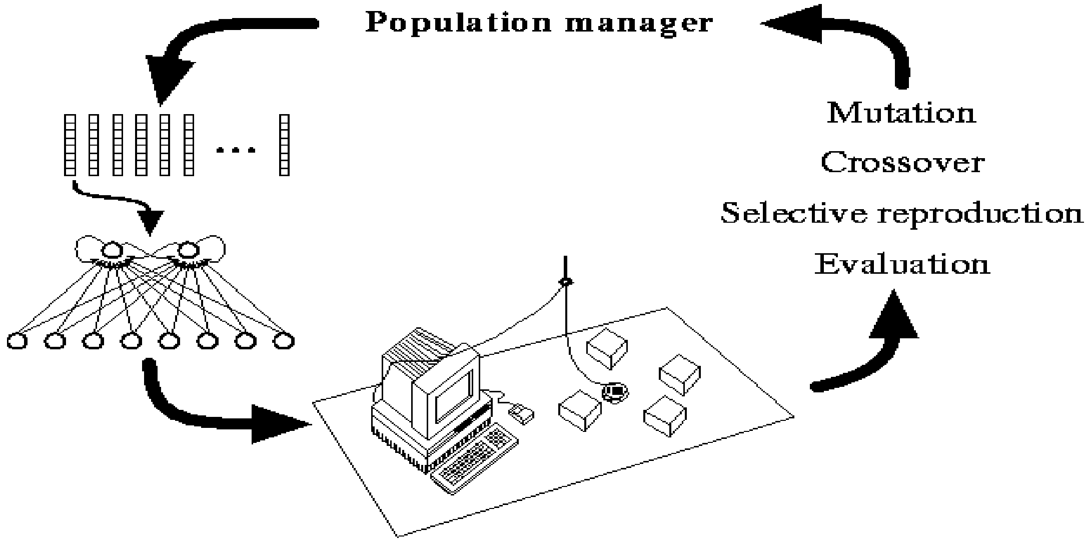
- ***..so which one is the best combination?***
  - Depends on context and network type
- ***Can I do something wrong?***
  - Not really, because:
    - Destructive mutation/recombination gets removed quickly
    - But wrong choice usually leads to slower evolution, i.e. higher number of generations
    - Problems can occur when observing best individual and average fitness of population
    - Good fitness function important



Example 1: Collision-free Navigation

- Khepera robot had to drive as fast as possible round a track
- Evolution directly on real robot





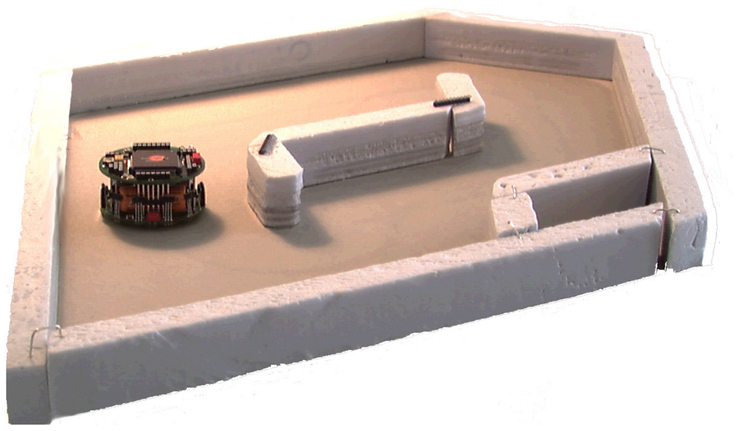
Reference: Floreano, D., Mondada, F., & others. (1994). Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. From animals to animats, 3, 421–430.

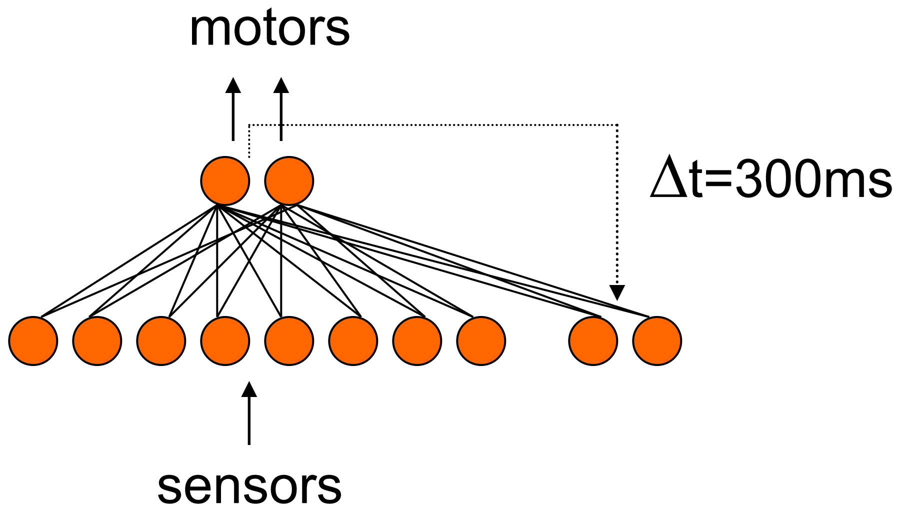
Collision-free Navigation

Fitness = $V(1-\sqrt{\Delta v})(1-s)$

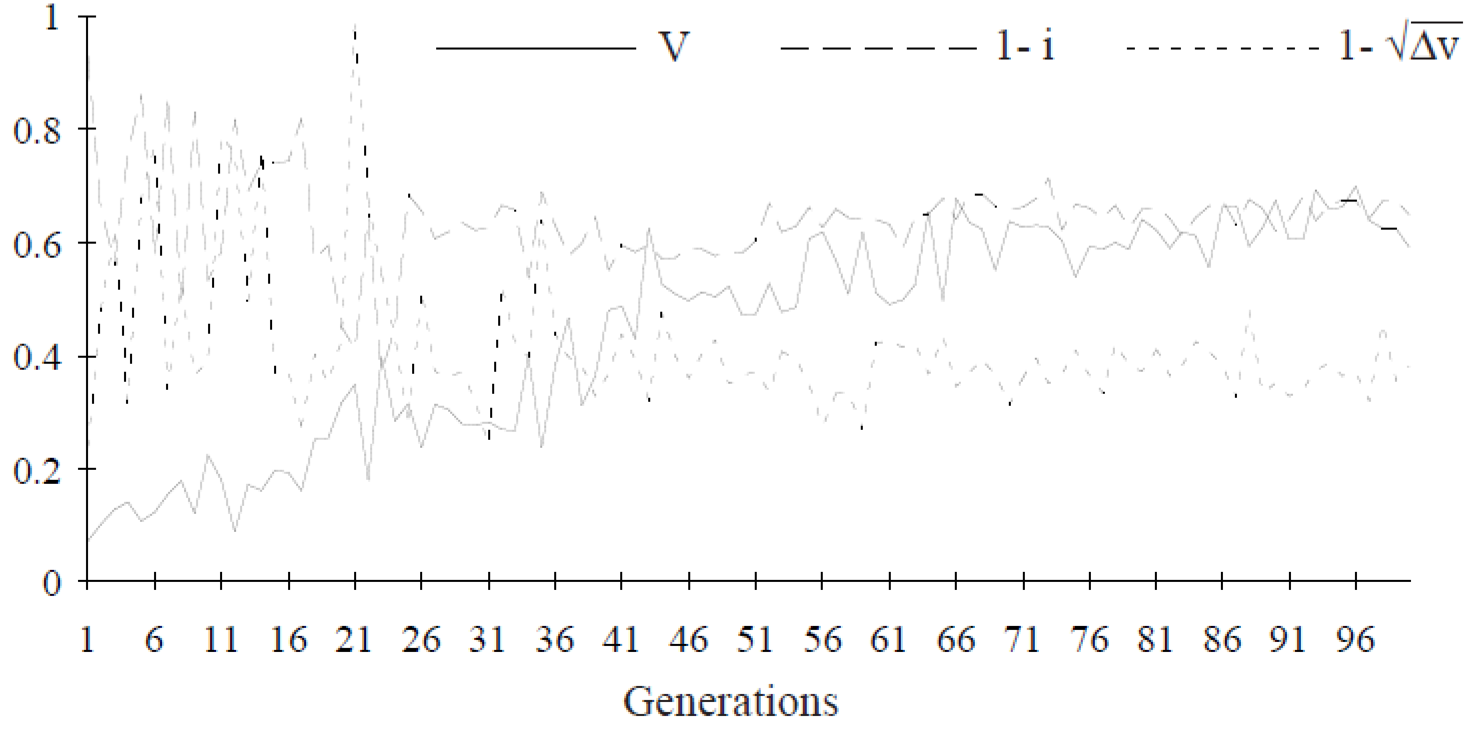
- V: Average unsigned speed of both wheels
- Δv: Difference of signed speeds of each wheel
- s: highest activation of IR sensor

- Population of about 80 individuals





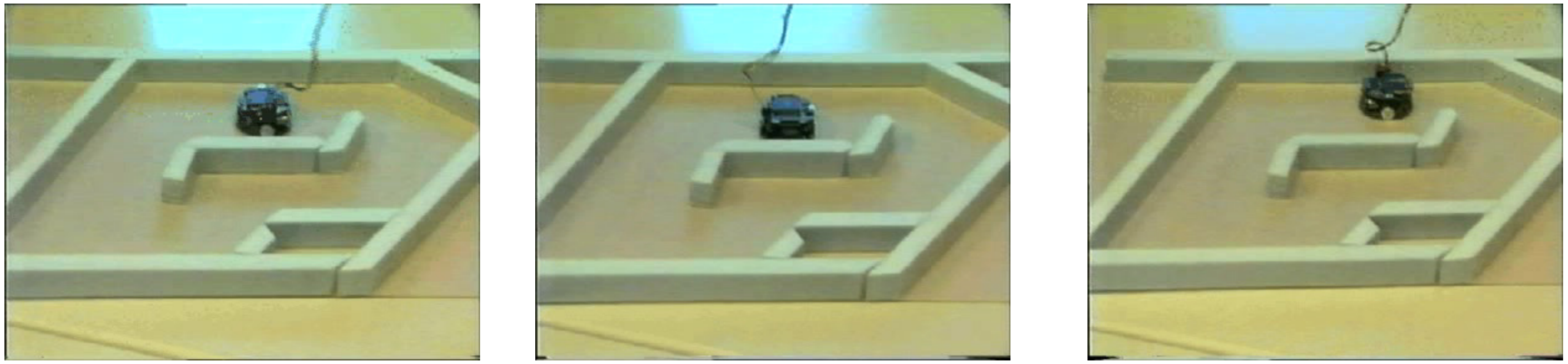
Fitness components for the best individual of each generation

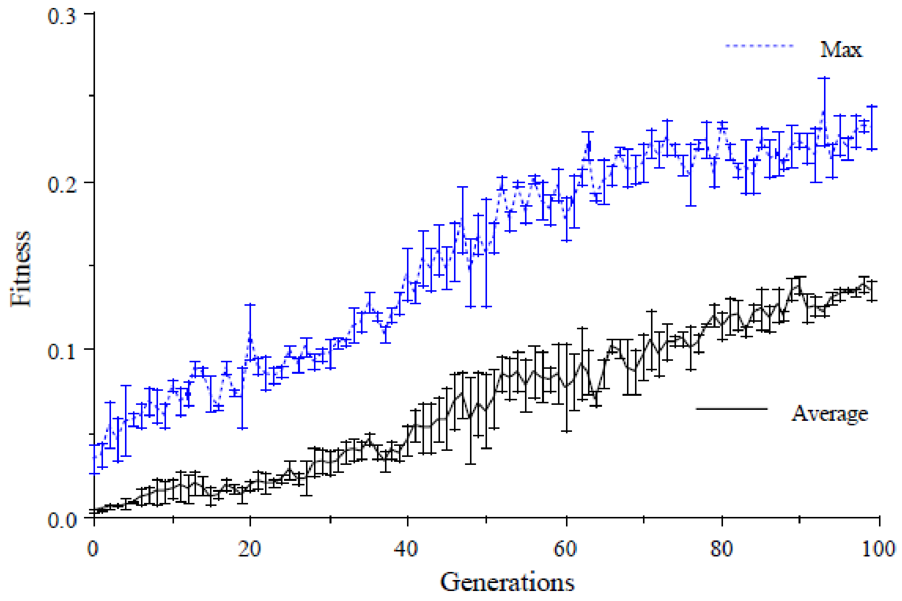


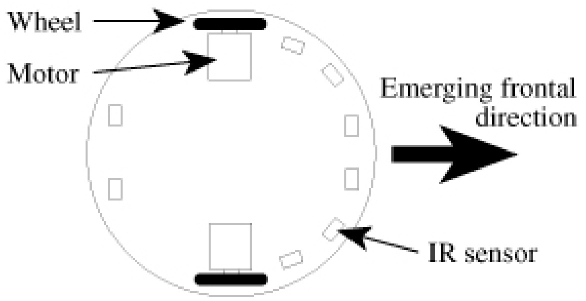
Example: Collision-free Navigation

Initial generation

After 45 generations

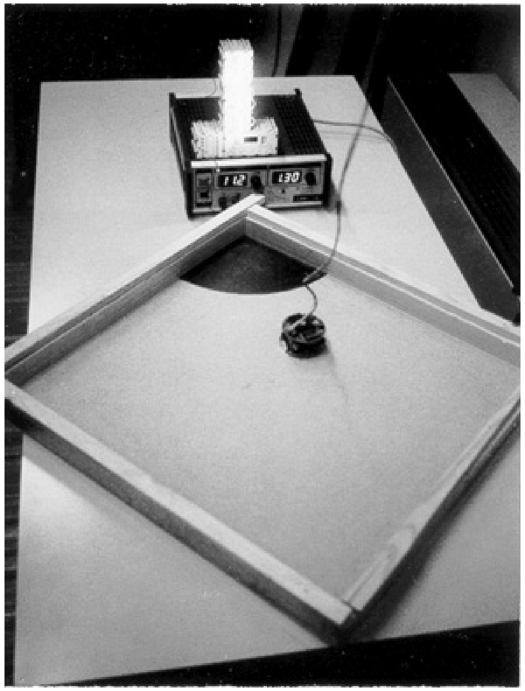




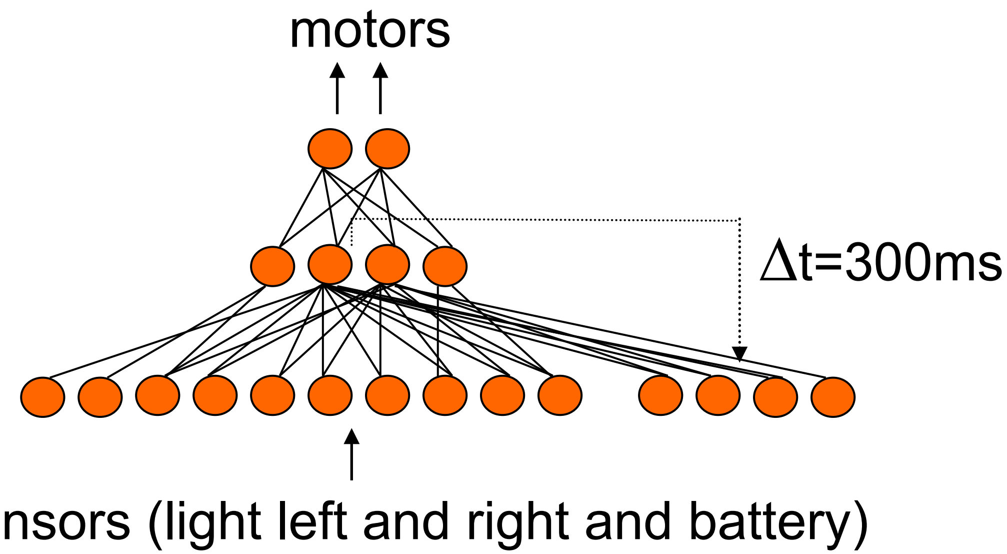


Example 2: Homing for Battery Charge

- Now the robot is in a more complex environment with simpler fitness function.
- The robot is equipped with a battery that lasts only 20 s and there is a battery charger in the arena.

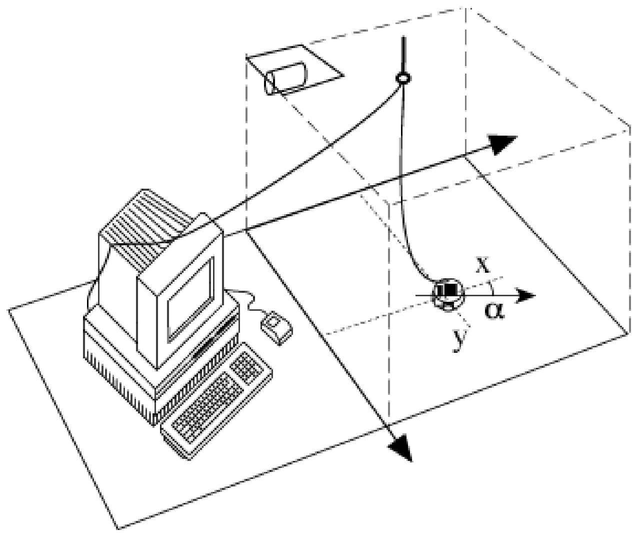


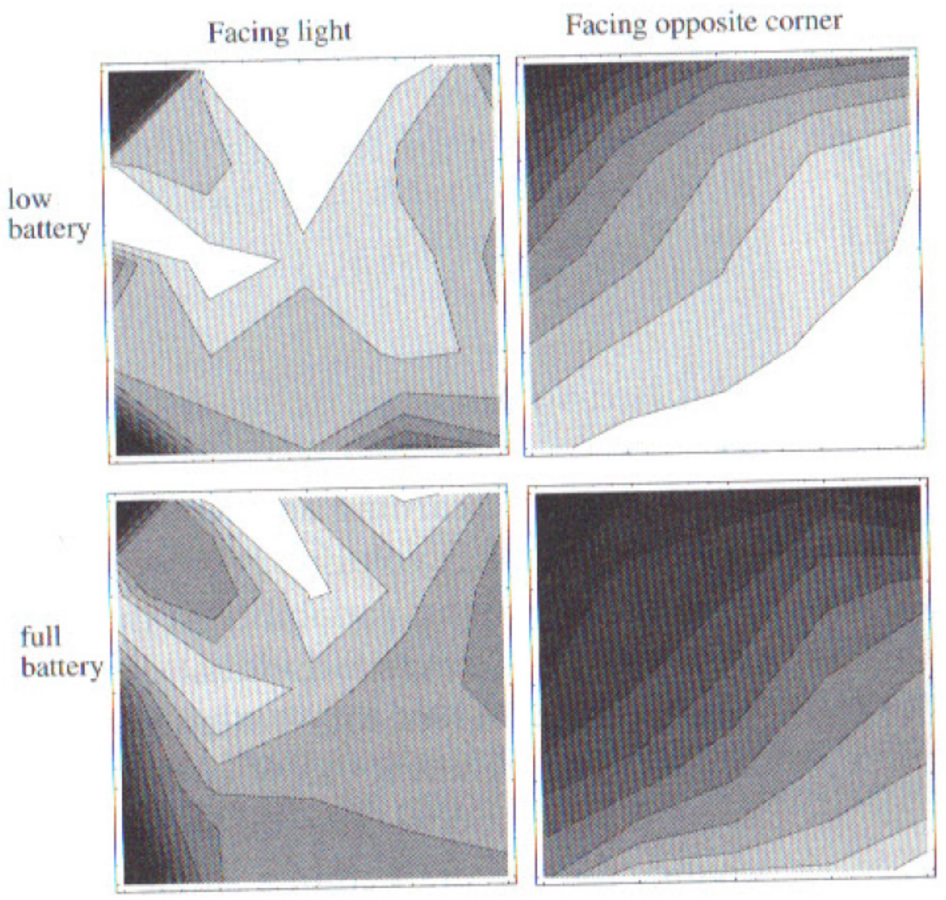
Fitness = $V(1-s)$



Reference: Floreano, D. and Mondada, F. (1996) Evolution of Homing Navigation in a Real Mobile Robot. IEEE Transactions on Systems, Man and Cybernetics Part B : Cybernetics, 26(3) pp. 396-407.

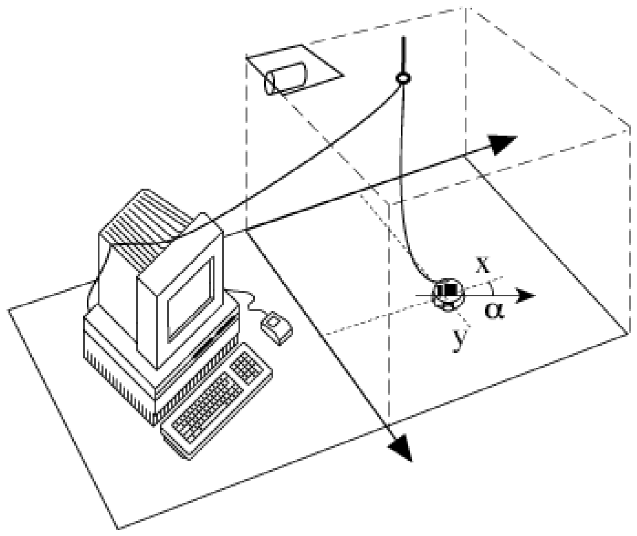
Activation levels of an internal neuron plotted over environment





- Brightness proportional to activation

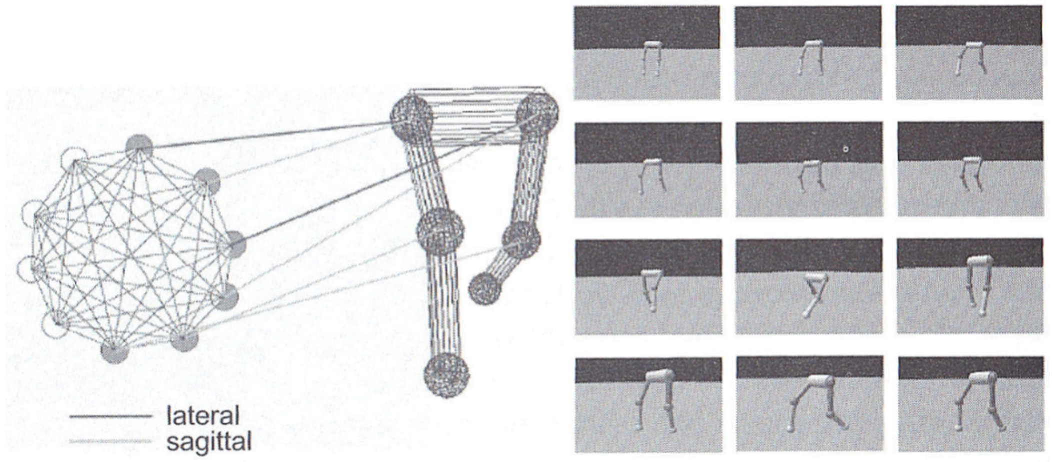
Example: Homing for Battery Charge



- After 240 generations, a robot was capable of moving around and going to recharge 2 seconds before the batteries are completely discharged.
- Network had to encode spatial information to correctly calculate return times

Example3: Bi-Pedal Walking

- Evolving walking gait of bipedal walker (Reil & Husbands)



Transfer Function:

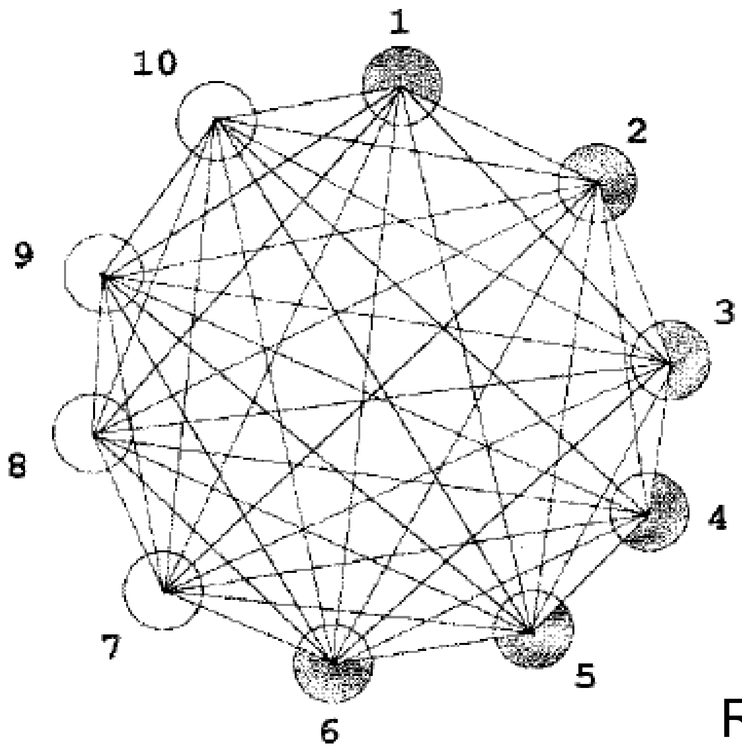
$$\tau_i \dot{A}_i = -A_i \sum \omega_{ji} O_j$$

$$O_j = (1 + e^{(a_j - A_j)})^{-1}$$

- Encoded as fixed string with Gaussian mutation
- Rank-based selection which kept 50% of population
- No recombination

Reference: Reil, T., & Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. IEEE Transactions on Evolutionary Computation, 6(2), 159-168. doi:10.1109/4235.996015

Bi-Pedal Walking Neural Network



Transfer Function:

$$\tau_i \dot{A}_i = -A_i \sum \omega_{ji} O_j$$

$$O_j = (1 + e^{(a_j - A_j)})^{-1}$$

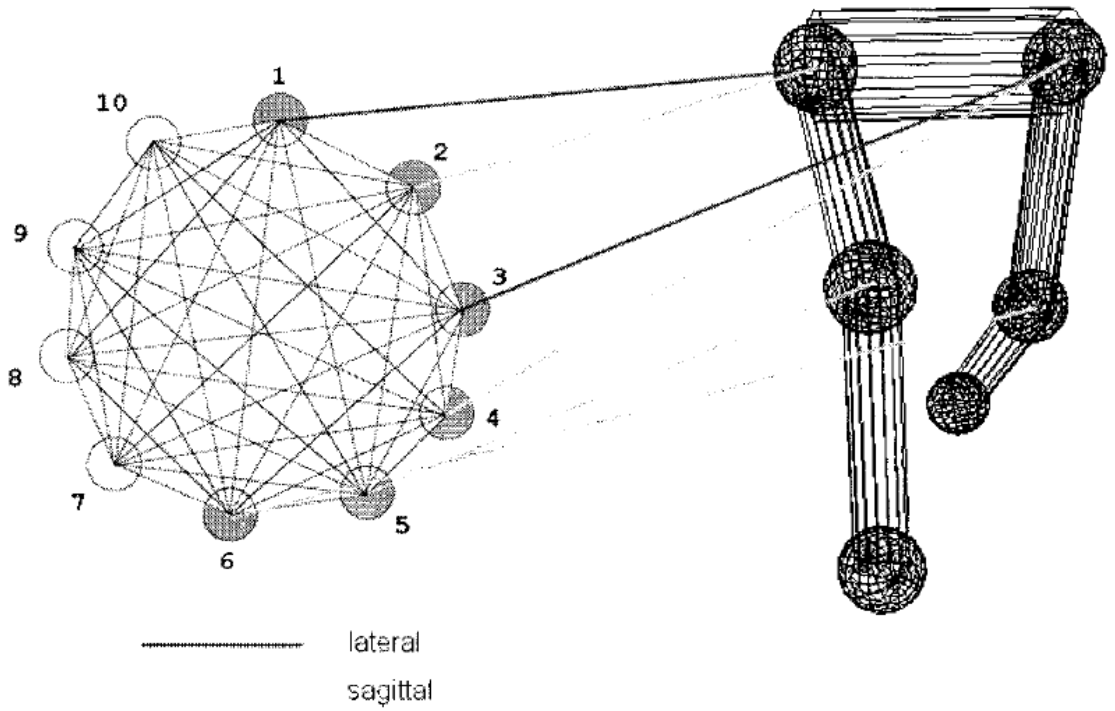
Recurrent continuous time NN

Shaded neurons are motor neurons:

Reference: Reil, T., & Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. IEEE Transactions on Evolutionary Computation, 6(2), 159-168. doi:10.1109/4235.996015

Example: Bi-Pedal Walking

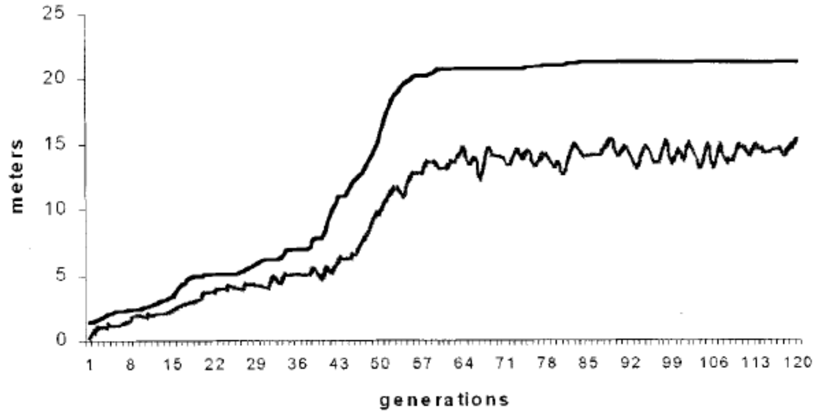
- Motor connections between controller and walker: Hips have two DOFs each (sagittal, i.e., front to back, and lateral)); knees have one DOF each (sagittal).



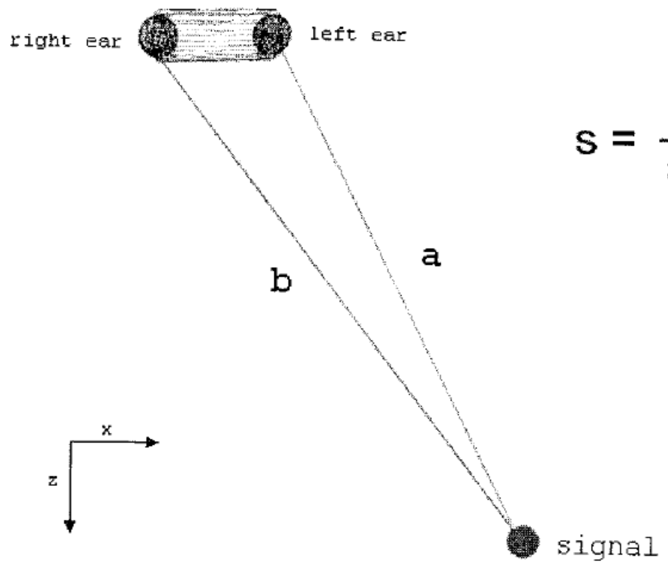
Example: Bi-Pedal Walking

- Fitness evaluation:
  1. Distance from origin
  2. Centre of gravity not lowered below threshold

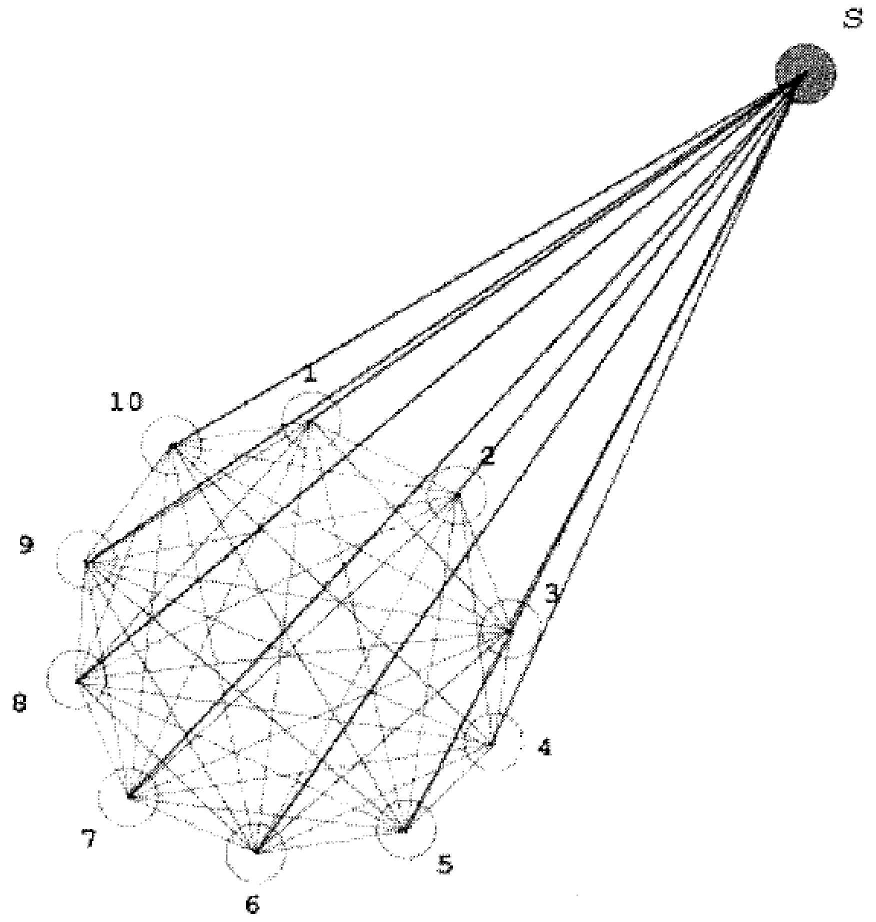
- Stable walk evolved in 10% of runs
- 80%, if fitness included reward for cyclic activity



Bi-Pedal Walking extension to sound localization



$$s = \frac{a - b}{a + b}$$

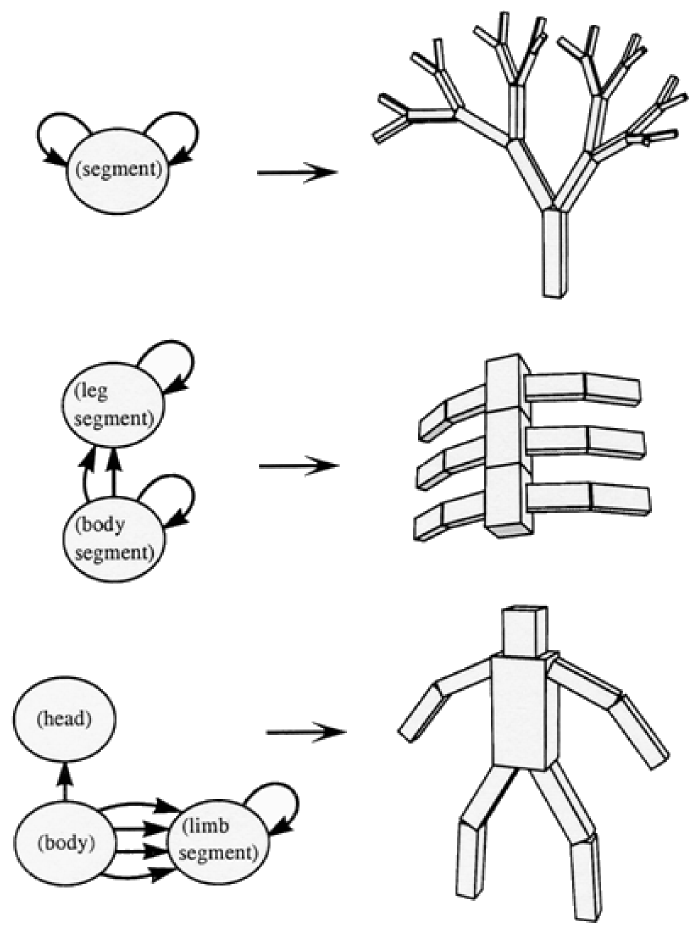


Example: Bi-Pedal Walking

Generation 0

(c) NaturalMotion Ltd. 2002

Example 4: Evolving Body and Brain



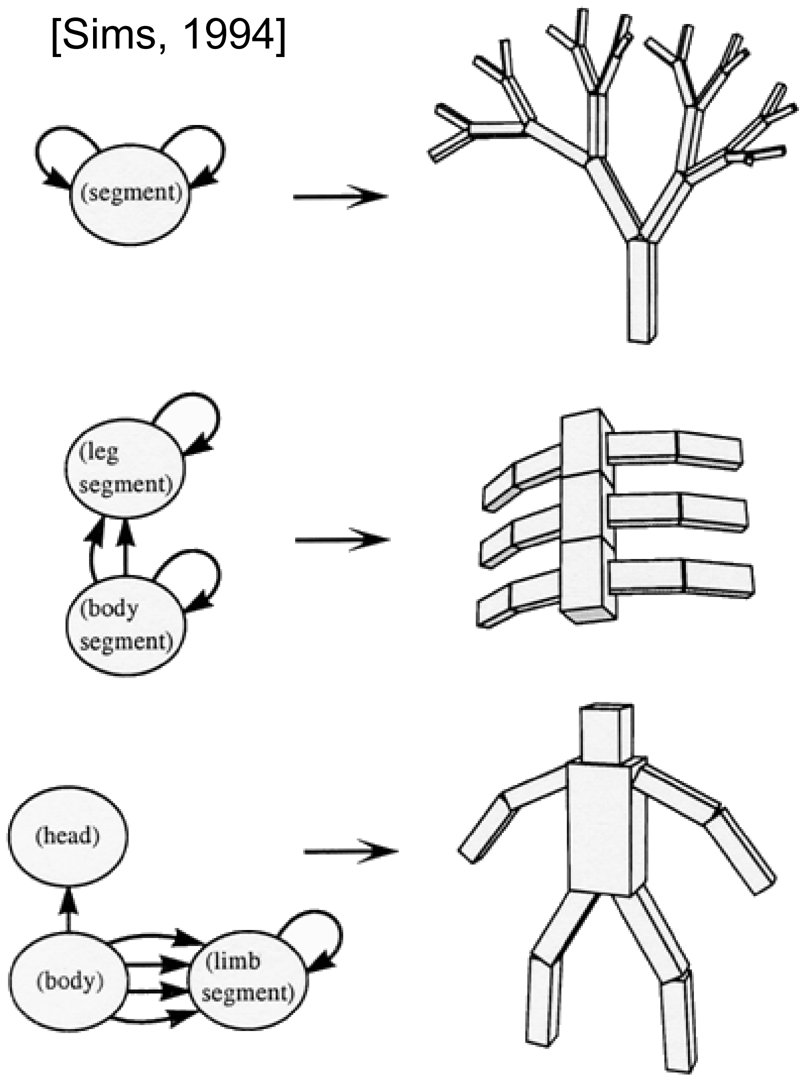
genotype phenotype

- Indirect encoding used for body and neural circuit
- Genotype is represented by directed graph
- Phenotype is a hierarchy of 3D parts

Reference:Sims, K. (1994). Evolving virtual creatures. Proceedings of the 21st annual conference on Computer graphics and interactive techniques SIGGRAPH 94, 28(July), 15-22. ACM Press. doi:10.1145/192161.192167

Evolving Body and Brain

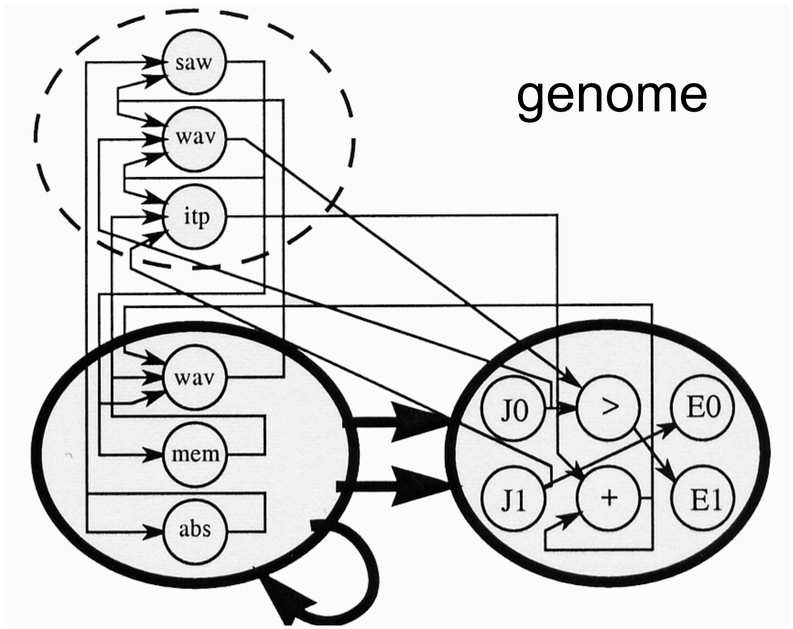
[Sims, 1994]



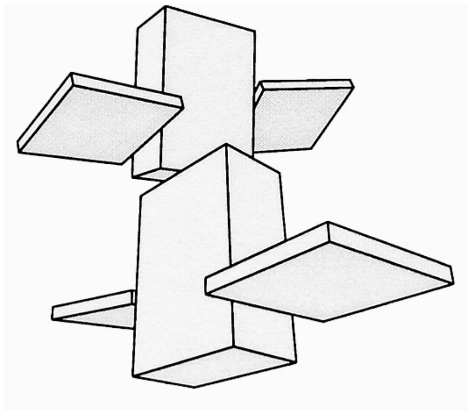
genotype phenotype

- Neural network also represented as directed graph
- Node types:
  - sensor
    - joint sensor
    - contact sensor
    - photosensor
  - neuron (math type)
    - sum
    - memory
    - oscillator
    - max, etc.
  - effector (force on muscle)
    - positive/negative (push/pull)

Evolved nested graph genotype

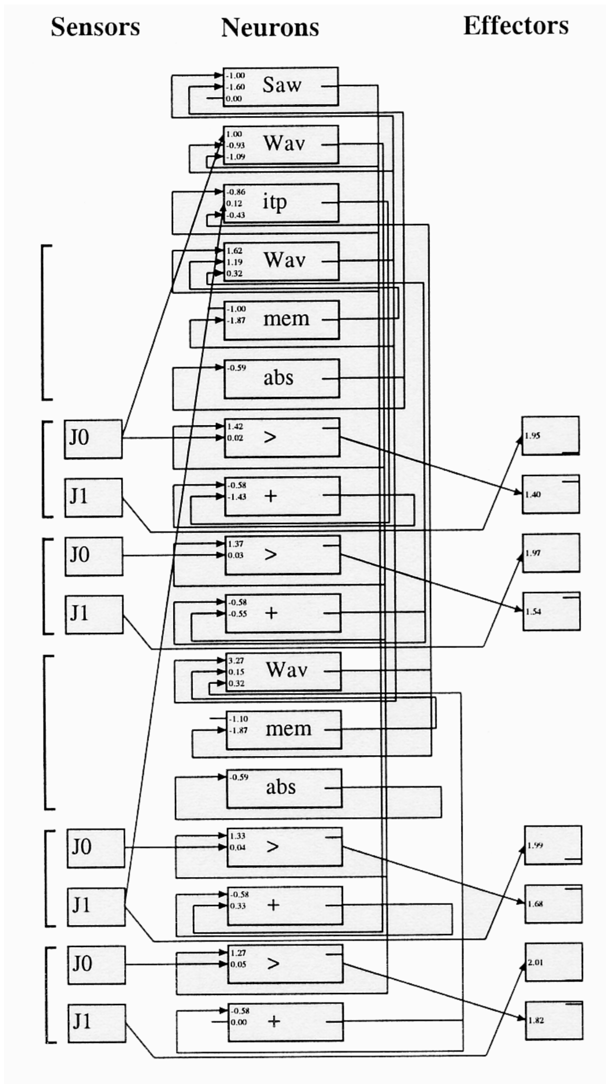


Genotype: The outer graph in bold describes a creature's morphology. The inner graph describes its neural circuitry.
J0 and J1 are joint angle sensors, and E0 and E1 are effector outputs.
The dashed node contains centralized neurons that are not associated with any part.



The **phenotype** morphology generated from the evolved genotype

Phenotype “brain”



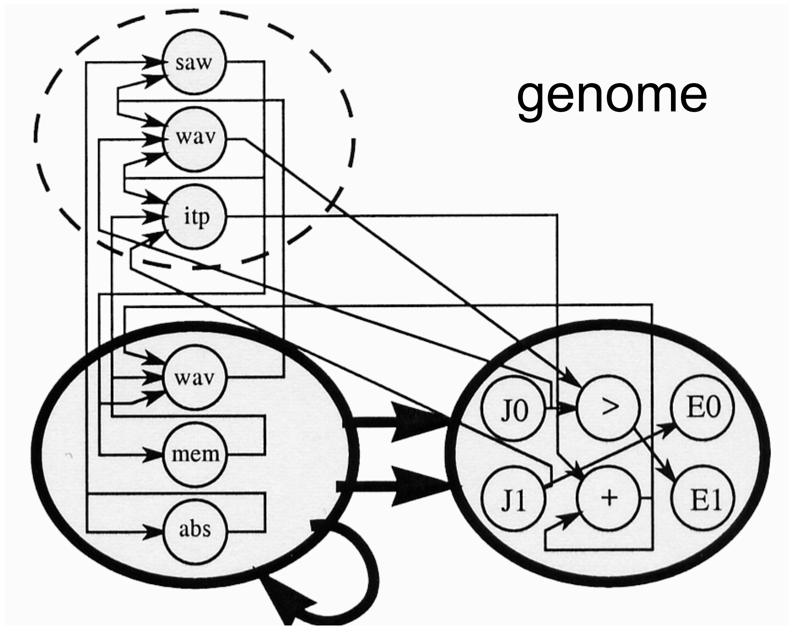
The phenotype “brain” generated from the evolved genotype shown from previous figure.

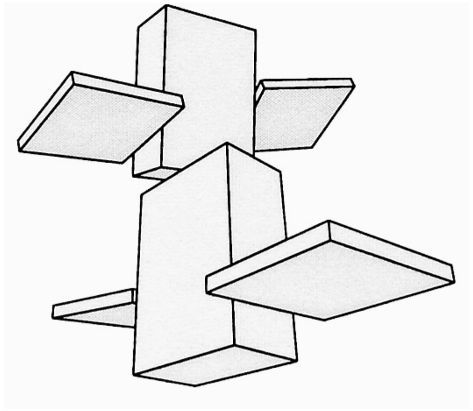
The effector outputs of this control system cause paddling motions in the four flippers of the morphology above.

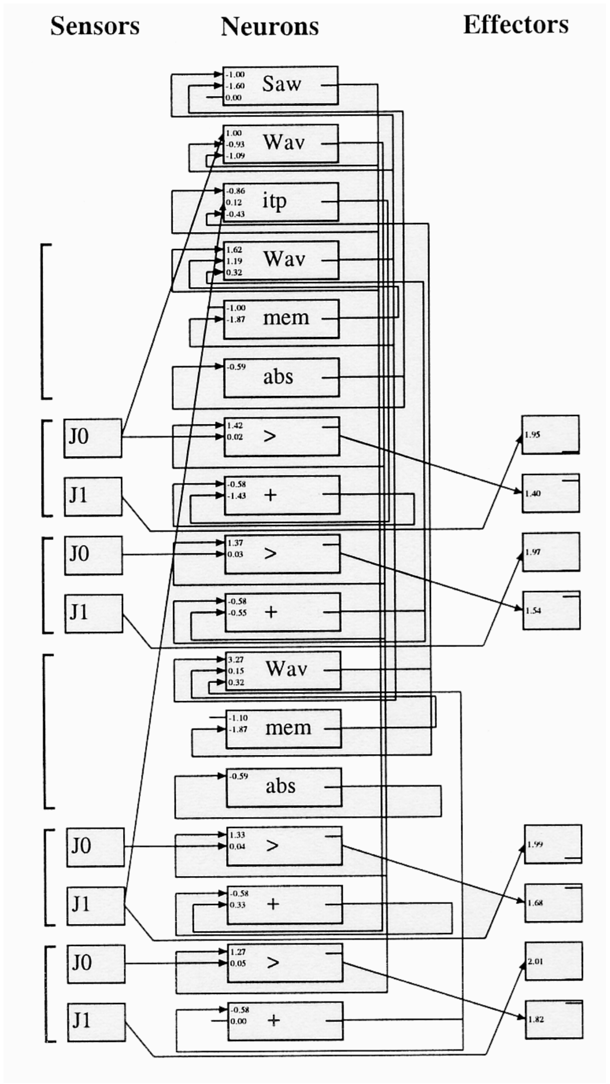
Example

Body

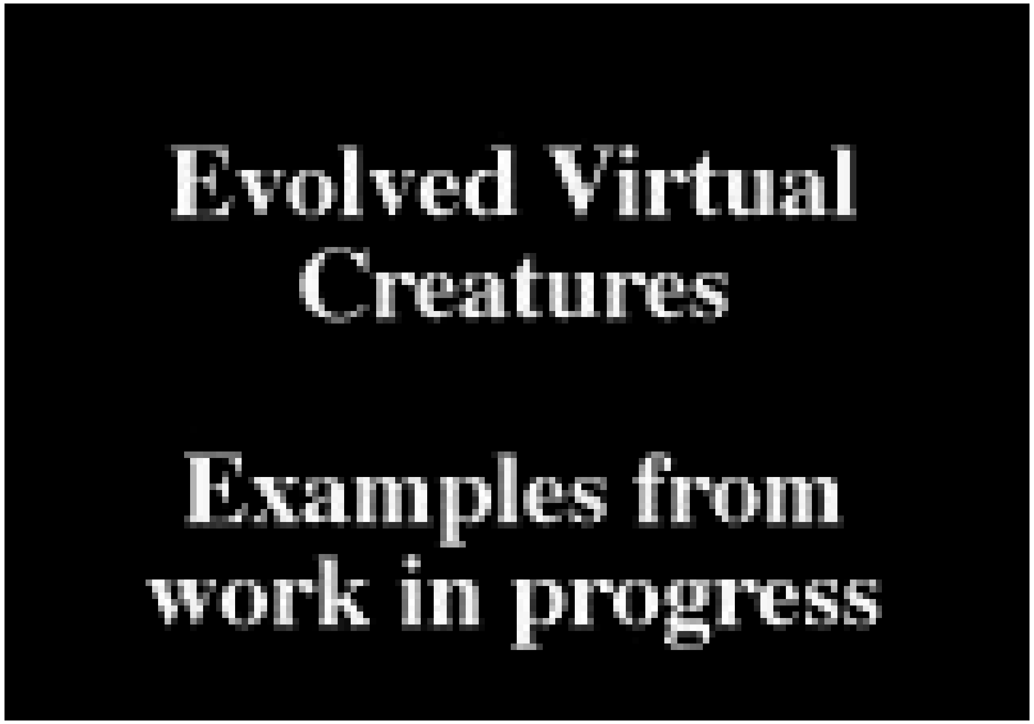
Brain





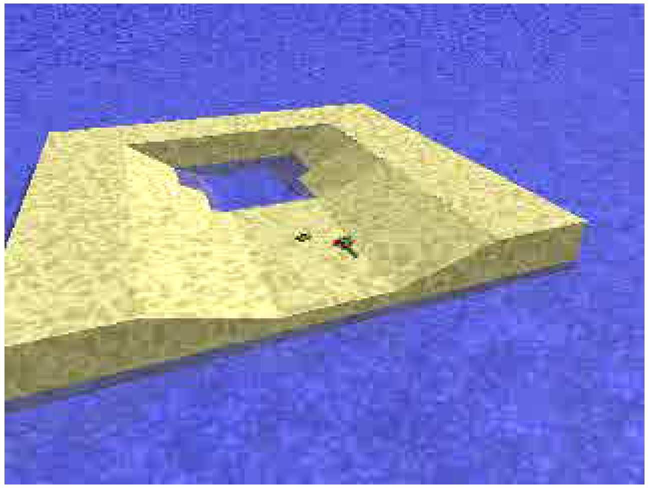


Sims' Evolved Creatures

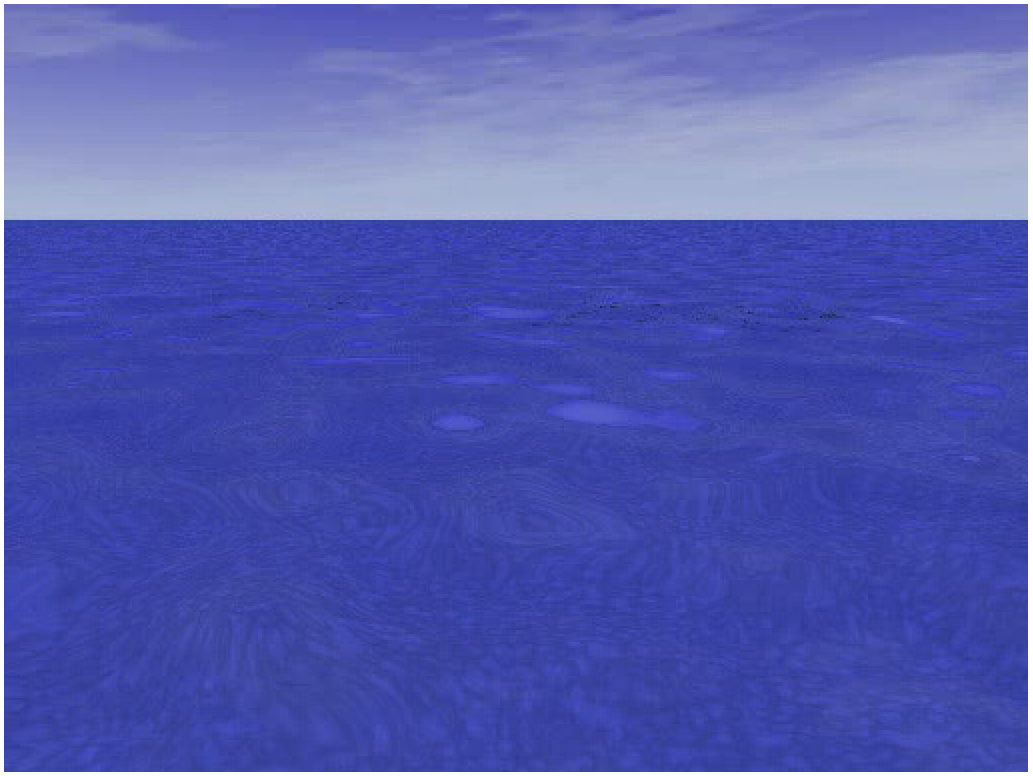

Evolved Virtual Creatures

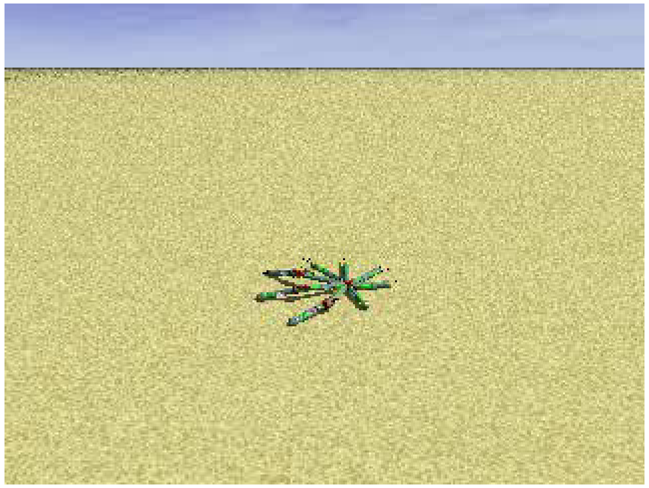
Examples from work in progress

Evolved for Speed



Examples of evolved individuals with fitness proportional to speed (from frams.alife.pl)





Summary

- Evolution of neural networks
- Evolution of body structure
- Evolution of controllers for neural networks governing robot

- Open Areas of Research
  - Hardware Self-Replication
  - Open-ended Evolution
  - Evolution, development, learning