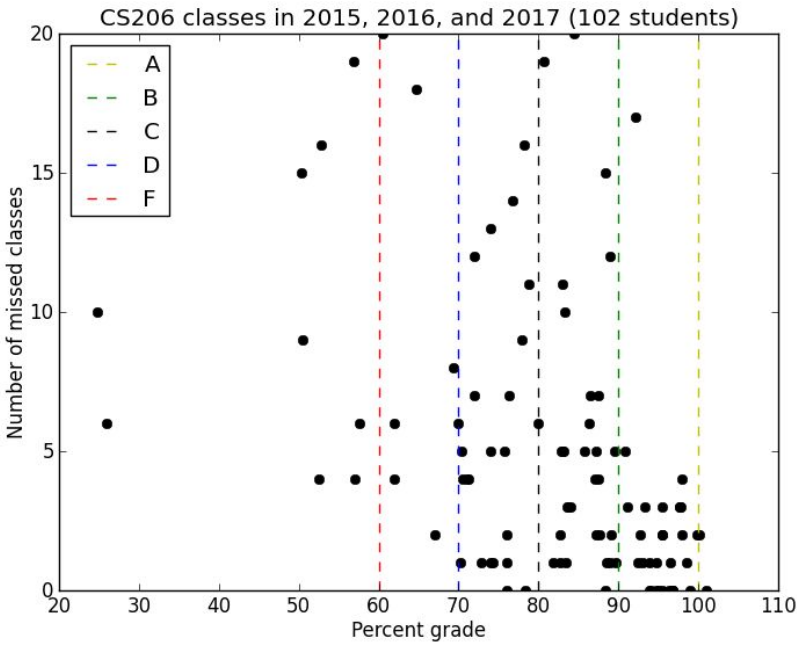
NASA’s Tensegrity Robot: The future of evolutionary robotics?

https://www.youtube.com/watch?v=wR0AIIwEgSE

The syllabus is available from BlackBoard.

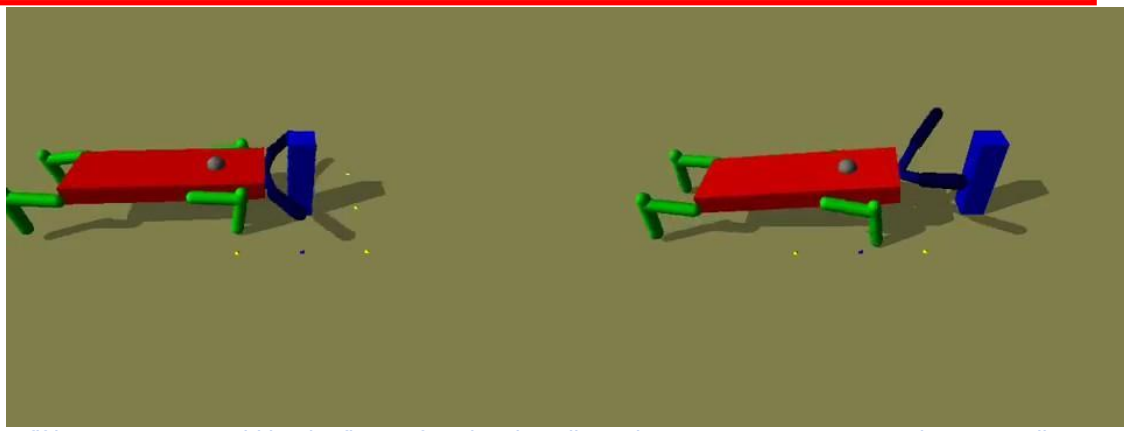
Here's why you should come to class.



3

Anatomy of an evolutionary robotics experiment:

1. Create a **task environment**.
2. Create the **robot**.
3. Create the robot's brain, or **Artificial Neural Network** (ANN).
4. Use an **evolutionary algorithm** to optimize the ANN so that the robot performs the desired task in its environment.



A fifth component would be the fitness function that allows the component to score a given controller.

Perversity instantiation:
- The robot does the thing that you asked it to do, but not in the same manner that you expected.

4

The essence of evolutionary robotics is that you want to tell the robot to solve a task, but you do not want to tell him how to solve it.

Who I am:

1993 – 1997: Undergraduate in Computer Science from McMaster University, Hamilton, Canada

1997 – 1998: Software Engineer, Computing Devices Canada, Calgary, Canada

1998 – 1999: MSc in Evolutionary and Adaptive Systems from Sussex University, Brighton, England

1999 – 2003: PhD in Mathematical and Physical Sciences from University of Zurich, Switzerland

2003 – 2006: Postdoctoral work at Cornell University, Ithaca, NY

Present: Associate Professor, UVM

What I expect from you:

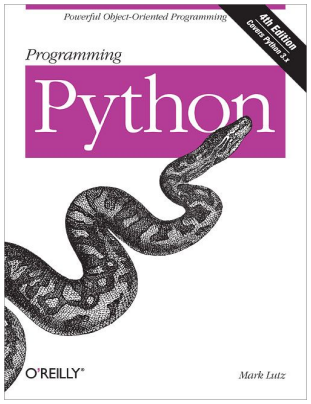
- Feedback
- Common sense
- Regular, not necessarily perfect attendance
- Hard work
- Memorization of key concepts (job interviews are closed-book)
- Creativity
- Self - learning
- A positive attitude when working with me, the TA, or fellow students.

What you _cannot_ expect from me:

- Help with buggy code
  (eg. “Why is my program crashing?”)

- Help with learning a programming language, installing software, etc.

What you _can_ expect from me:



- Help with _specific_ programming questions…
  (eg. “Why do I need to deallocate memory?”)

- … but only after you’ve consulted online resources
  and your peers.
  (Google “Python tutorial”)
  www.codecademy.com

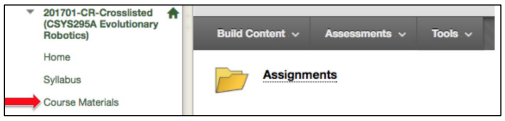
- Help with conceptual issues
  (“Can you go over the genetic algorithm again?”)

- Clarification about the assignments / project

- An emphasis on concepts, rather than specific tools,
  because tools change, but concepts change more slowly.

Weeks 1 to 10: Complete 10 cumulative programming assignments:

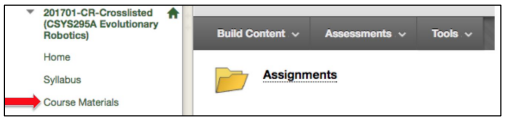
Blackboard



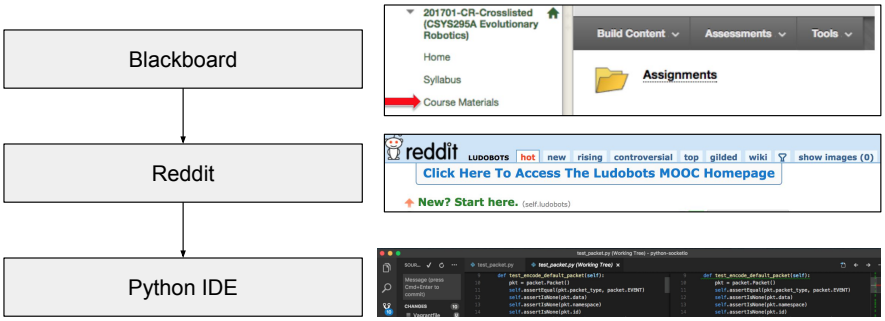
Weeks 1 to 10: Complete 10 cumulative programming assignments:

Blackboard

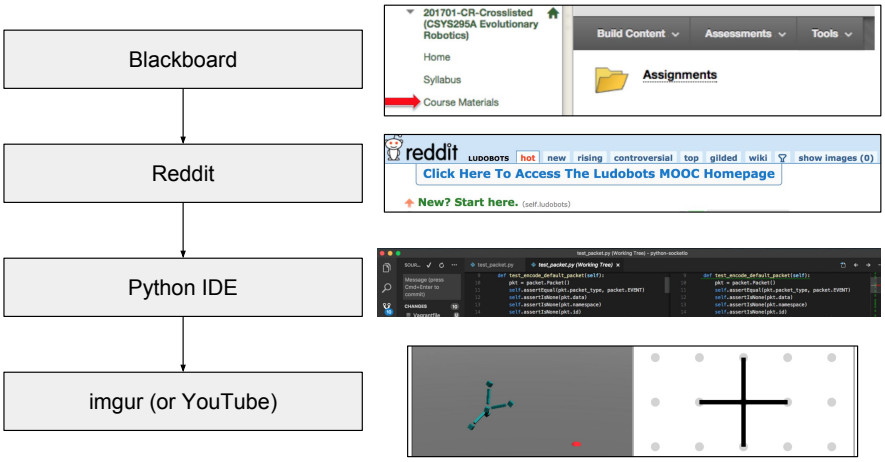
Reddit



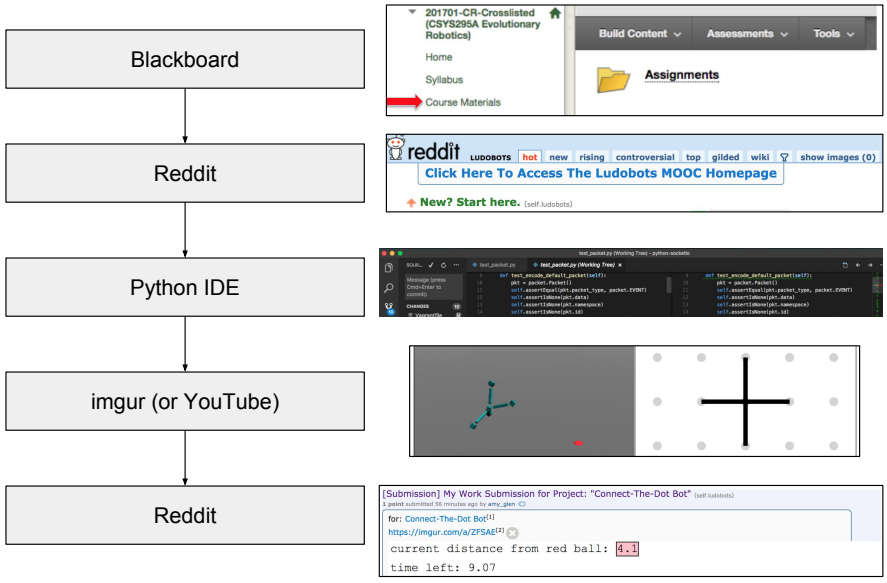
Weeks 1 to 10: Complete 10 cumulative programming assignments:



Weeks 1 to 10: Complete 10 cumulative programming assignments:

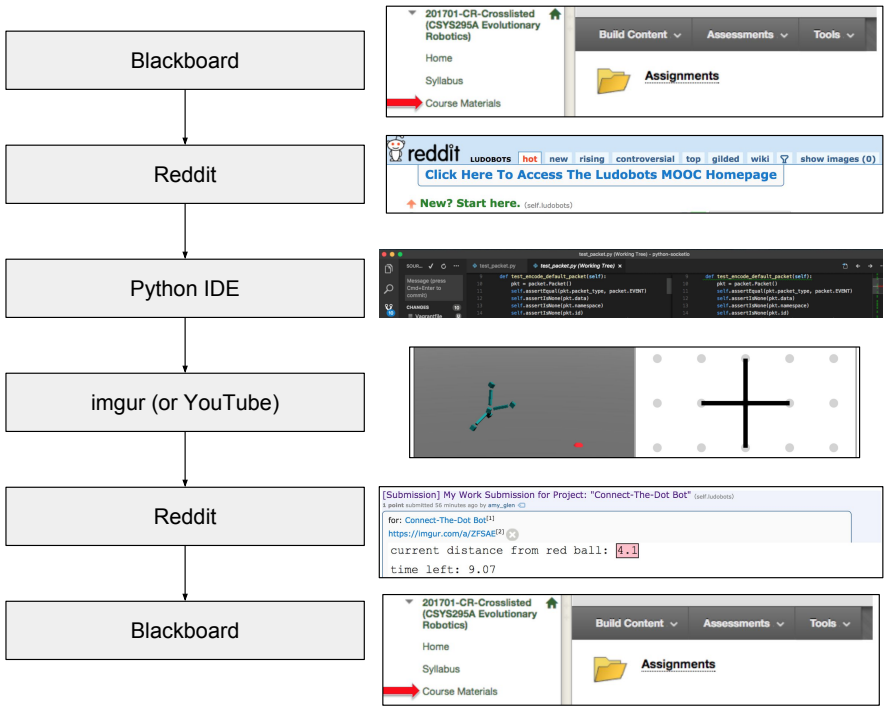


Weeks 1 to 10: Complete 10 cumulative programming assignments:



CS206 L01

Weeks 1 to 10: Complete 10 cumulative programming assignments:



Weeks 11 to 14: Use the system to conduct an evolutionary robotics experiment.

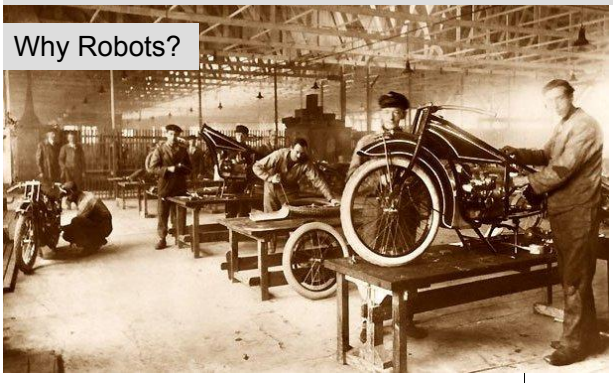
1. Given the same optimizer, neural network and eight hours of behavior optimization on the same computer, does a quadrupedal robot evolve to walk further or not as far as a hexapedal robot, or is there no significant difference?

2. Create five fitness functions that not only select for locomotion on the quadrupedal robot, but each also selects for a particular gait: walking, trotting, canter, galloping and pronking.

3. Create a fitness function that rewards NNs for locomotion, but penalizes them for requiring a lot of energy to realize the gait. This is difficult, as there are two solutions that are not desirable: evolution finds fast but inefficient gaits, or 'gaits' in which the robot does not move, and therefore does not consume energy.

4. Equip the robot with a simulated laser range finder, which tells the robot about objects in its environment. Evolve a robot that walks toward round objects, but walks away from rectangular objects.

Why Robots?



Why Robots?

It is one of the few 'open frontiers' in science:

There is no guiding theory about intelligence
Guiding theory in biology: evolution
Guiding theory in physics: Newtonian and quantum mechanics
Guiding theory in chemistry: …
…
There are several Nobel Prizes waiting to be won in this field.

It is one of the most interdisciplinary areas of study:

Evolutionary biology
Neuroscience
Cognitive science
Psychology
Biomechanics
Physics
Computer Science
Mechanical engineering, electrical engineering, bio-engineering
Chemistry
Mathematics
Philosophy (of mind)

Large-scale outdoor infrastructure:

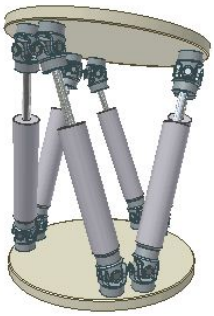
Wind farms,
Solar farms,
Wave farms,
Tide farms,
…

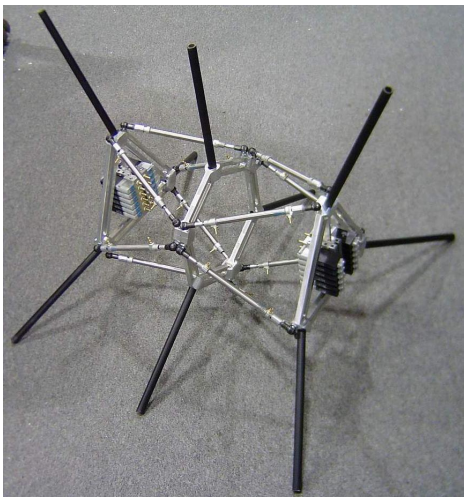
Why Robots Outside?

Why Evolutionary Robotics?

1. Creating a controller for a robot is non-intuitive: requires automation

	t1	t2	t3	t4	t5	t6	t7	t8	…
m1	+1	+1	+1	+1	+1	−1	+1	−1	…
m2	−1	+1	+1	+1	−1	+1	+1	+1	…
m3	+1	+1	+1	−1	+1	+1	−1	+1	…
m4	+1	−1	+1	+1	−1	+1	+1	−1	…
…									

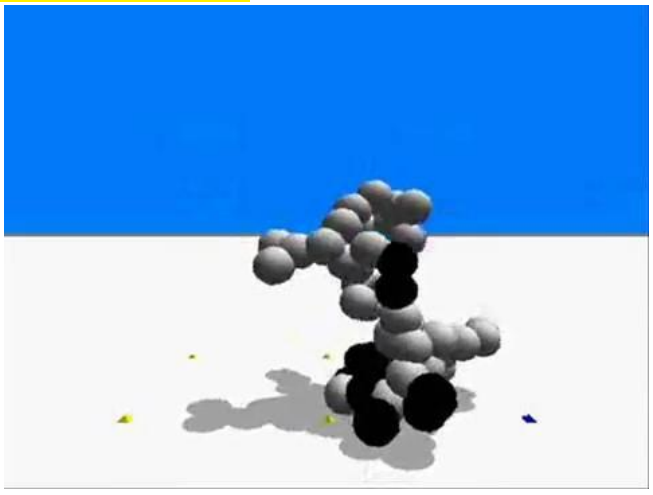




Why Evolutionary Robotics?

1. Creating a controller for a robot is non-intuitive: requires automation

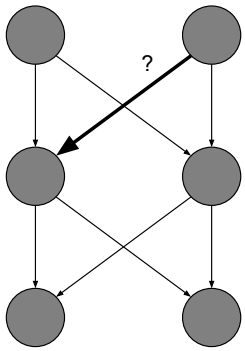
2. Learning algorithms only optimize controllers; evolution can optimize the whole robot.



Why Evolutionary Robotics?

1. Creating a controller for a robot is non-intuitive: requires automation

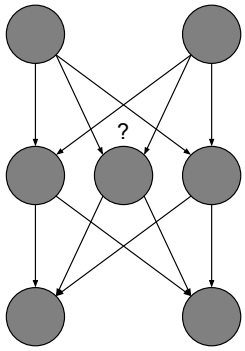
2. Learning algorithms only optimize controllers;
   evolution can optimize the whole robot.



Why Evolutionary Robotics?

1. Creating a controller for a robot is non-intuitive: requires automation

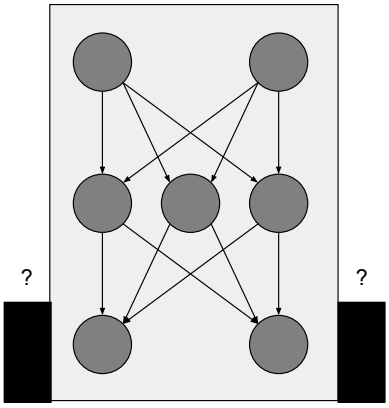
2. Learning algorithms only optimize controllers;
   evolution can optimize the whole robot.



Why Evolutionary Robotics?

1. Creating a controller for a robot is non-intuitive: requires automation

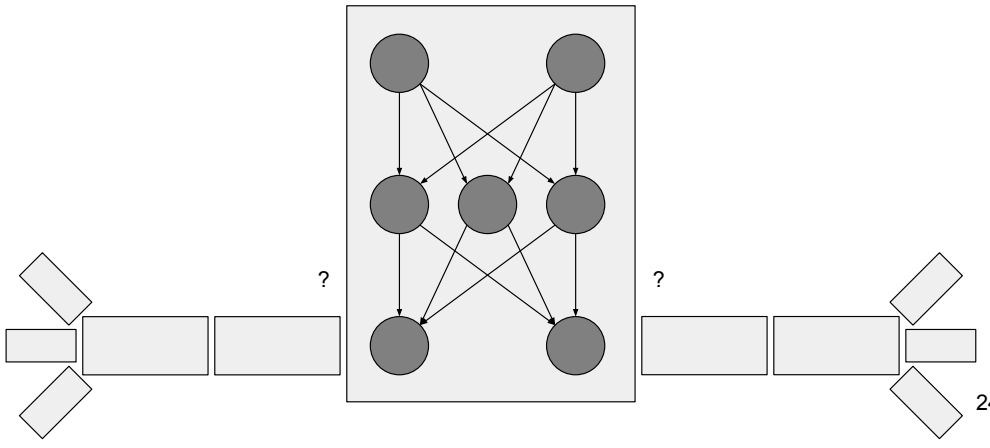
2. Learning algorithms only optimize controllers;
   evolution can optimize the whole robot.



Why Evolutionary Robotics?

1. Creating a controller for a robot is non-intuitive: requires automation

2. Learning algorithms only optimize controllers;
   evolution can optimize the whole robot.

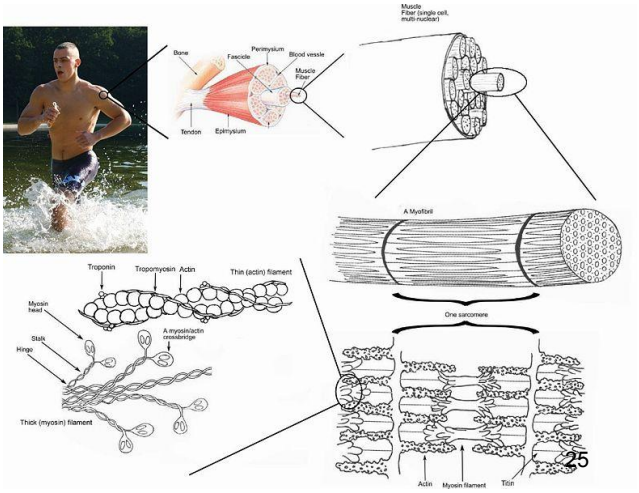


Why Evolutionary Robotics?

1. Creating a controller for a robot is non-intuitive: requires automation

2. Learning algorithms only optimize controllers;
   evolution can optimize the whole robot.

3. Biological evolution produced adaptive agents of unparalleled complexity ***with no supervision***.



CS206 L01

Why Evolutionary Robotics?

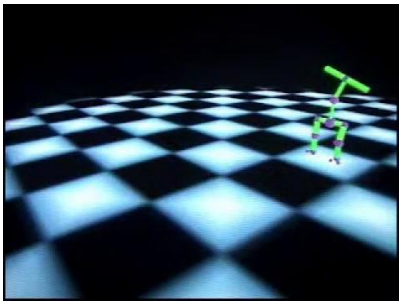
1. Creating a controller for a robot is non-intuitive: requires automation

2. Learning algorithms only optimize controllers;
   evolution can optimize the whole robot.

3. Biological evolution produced adaptive agents of unparalleled complexity ***with no supervision***.

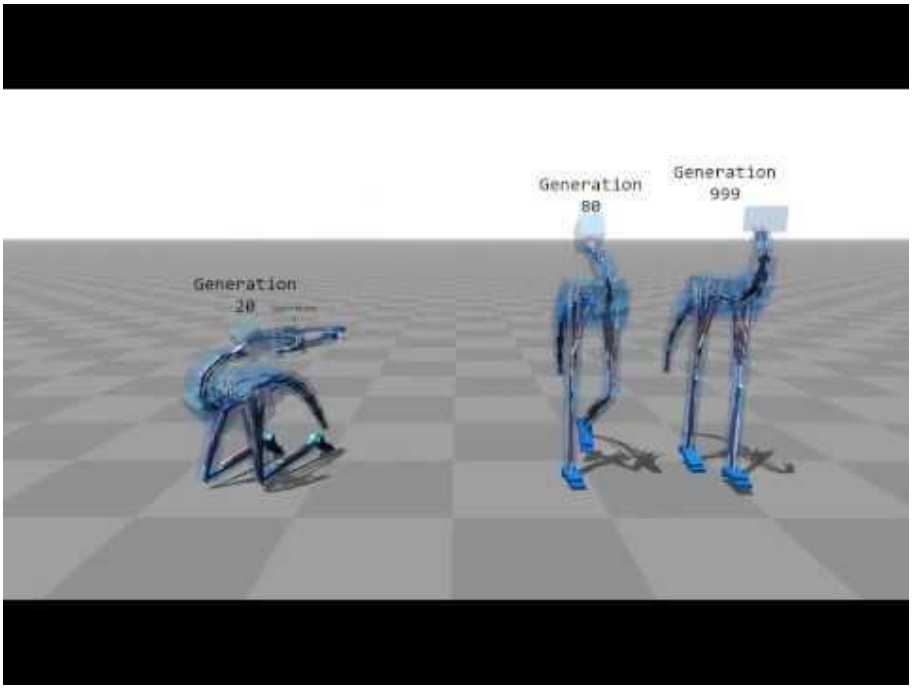
4. Most learning algorithms require detailed supervision:

	t1	t2	t3	t4	t5	t6	t7	t8	…
m1	+1	+1	+1	+1	+1	-1	+1	-1	…
m2	-1	+1	+1	+1	-1	+1	+1	+1	…
m3	+1	+1	+1	-1	+1	+1	-1	+1	…
m4	+1	-1	+1	+1	-1	+1	+1	-1	…
…					Should have been '+1'				



Evolutionary algorithm is often known as semi-supervised search method:
- Semi-supervision: The fitness function supervised the performance of the robot

Why Evolutionary Robotics?



27

“Cogito ergo sum.”

"Cogito ergo sum."

"I think, therefore I am."

↓

"Do I exist?"

↓

"There is something that is asking the question,
Because there is an "I" in the sentence.
Whatever that "I" thing is, it exists."

↓

The soul(/mind) surely exists;
the body, I'm not so sure about.

↓

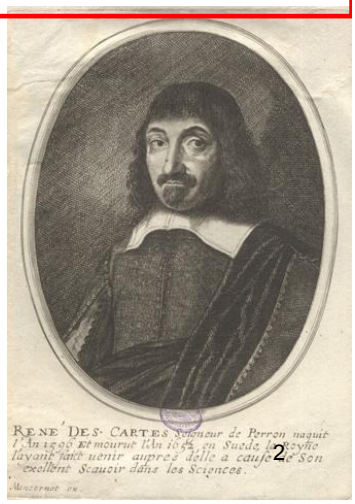
So the soul(/mind) and body are different.

↓

Cartesian Dualism; "The Mind/Body Problem"

Information in red boxes is not on the uploaded slides.

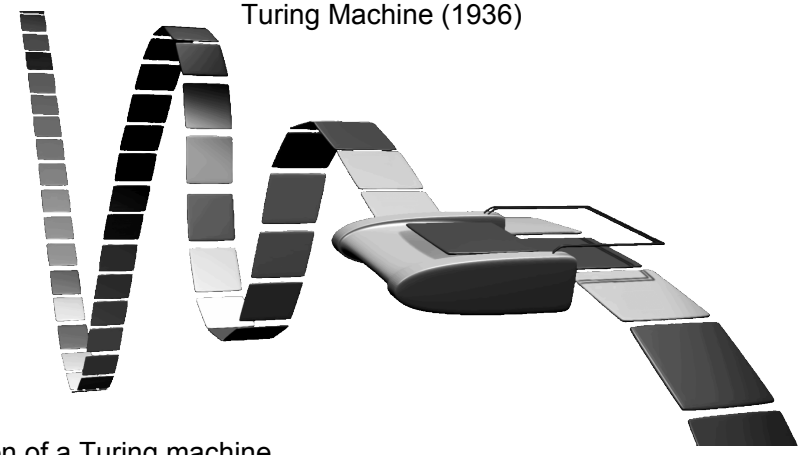
Discourse on Method (1637)



We tend to think input first, then information processing, then output.

The very definition of a computer relies on this notion of input first, then processing, then output.

Turing Machine (1936)



Q: Give a brief definition of a Turing machine.

A: Machine reads symbol at the current location on the tape (input);
Depending on machine's current state, and the symbol read,
it updates its own state (internal processing); writes a new symbol (output); or
moves the tape forward or backward(output).

The Dartmouth Summer Research Conference on Artificial Intelligence (1956)
[Origin of the word "**artificial intelligence**"]

Proposal: "We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that

every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to

(reproduce?) → simulate it. The weak AI stands.
Computer may be able to simulate human level intelligence.

An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and

improve themselves.

We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer."

The Dartmouth Summer Research Conference on Artificial Intelligence (1956)
[Origin of the word "**artificial intelligence**"]

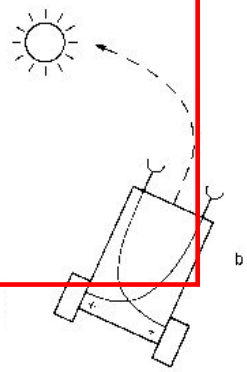
Attendees:
John McCarthy
Marvin Minsky
Nathaniel Rochester
Claude Shannon
Ray Solomonoff
Oliver Selfridge
Trenchard More
Arthur Samuel
Herbert Simon, and
Allen Newell

Goals of Artificial Intelligence:
(p. 17, *How The Body*…)

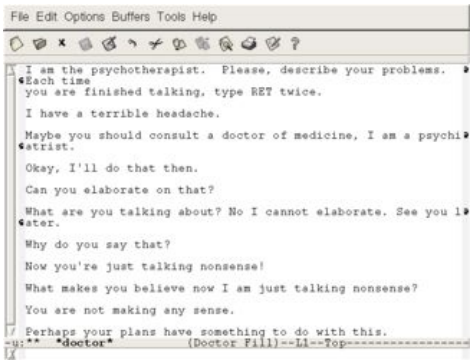
(1) _ Understand biological systems
E.g., how do simple animals translate
the detection of light
into phototaxis
(moving toward light sources)?

(2) _ Abstraction of the general principles of intelligent behavior

(3) _ Apply these principles to create intelligent artifacts



An example of classical Artificial Intelligence: ELIZA



Created by Joseph Weizenbaum between 1964 to 1966

↓

Internet chatbots

Sample ELIZA Rule: User: “I [verb] [proper noun]” →
ELIZA: “Why do you [verb] [proper noun]?”

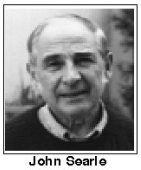
Q: Other candidate rules?

Turing Machine: Abstract description of a minimal computer.

Turing test

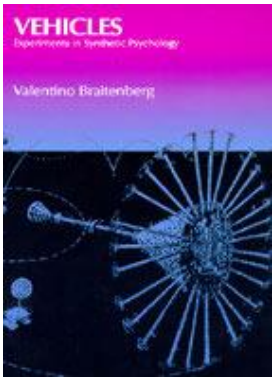
If a computer can make a human think it is intelligent, it is intelligent.

3 John Searle, 1980a, 1980b, 1990b
The Chinese Room argument. Imagine that a man who does not speak Chinese sits in a room and is passed Chinese symbols through a slot in the door. To him, the symbols are just so many squiggles and squoggles. But he reads an English-language rule book that tells him how to manipulate the symbols and which ones to send back out. To the Chinese speakers outside, whoever (or whatever) is in the room is carrying on an intelligent conversation. But the man in the Chinese Room does not understand Chinese; he is merely manipulating symbols according to a rule book. He is instantiating a formal program, which passes the Turing test for intelligence, but nevertheless he does not understand Chinese. This shows that instantiation of a formal program is not enough to produce semantic understanding or intentionality. **Note:** For more on Turing tests, see Map 2. For more on formal programs and instantiation, see the "Is the brain a computer?" arguments on Map 1, the "Can functional states generate consciousness?" arguments on Map 6, and sidebar, "Formal Systems: An Overview," on Map 7.



John Searle

in · ten · tion · al · it · y: The property (in reference to a mental state) of being directed at a state of affairs in the world. For example, the belief that Sally is in front of me is directed at a person, Sally, in the world. Intentionality is sometimes taken to be synonymous with representation, understanding, consciousness, meaning, and semantics. Although there are important and subtle distinctions in the definitions of "intentionality," "understanding," "semantics," and "meaning," in this debate they are sometimes used synonymously.

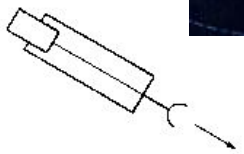


Valentino Braitenberg, *Vehicles*
Source Material: *http://www.bcp.psych.ualberta.ca/~mike/Pearl_Street/Margin/Vehicles/index.html*

Introduces a series of (hypothetical) simple robots that seem, to the outside observer, to exhibit complex behavior.

The complex behavior does not come from a complex brain, but from a simple agent ***interacting*** with a rich environment.

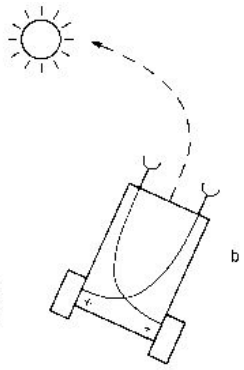
Vehicle 1: Getting around
A single sensor is attached to a single motor.
Propulsion of the motor is proportional to the signal detected by the sensor.
The vehicle will always move in a straight line, slowing down in the cold, speeding up in the warm.



Braitenberg: “Imagine, now, what you would think if you saw such a vehicle swimming around in a pond. It is restless, you would say, and does not like warm water. But it is quite stupid, since it is not able to turn back to the nice cold spot it overshot in its restless ness. Anyway, you would say, it is ALIVE, since you have never seen a particle of dead matter move around quite like that.”

Valentino Braitenberg, *Vehicles*

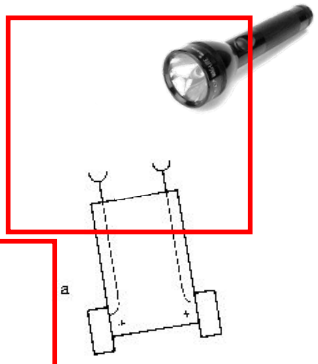
Vehicle series 2: Fear and aggression
Each vehicle has two sensors and two motors.
Still a directly proportionate relationship betwee the signal strength and force of the motor.



The Aggressor
Left sensor is attached to the right motor;
right sensor is attached to the left motor;
Will charge the light if it’s dead-ahead;
Will charge the light if it’s off to the side;
Will eventually hit it, as long as the light stays in the vicinity.

Valentino Braitenberg, *Vehicles*

Vehicle series 2: Fear and aggression
Each vehicle has two sensors and two motors.
Still a directly proportionate relationship between the signal strength and force of the motor.

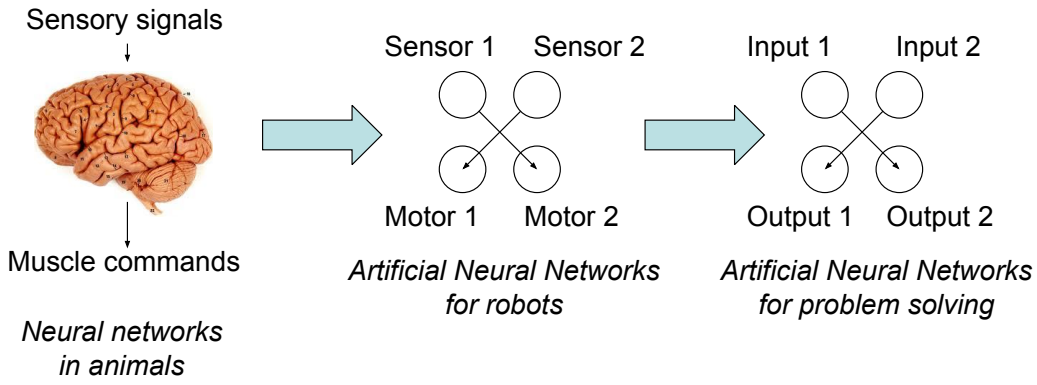


The
Left sensor is attached to the ***left*** motor;
right sensor is attached to the ***right*** motor…

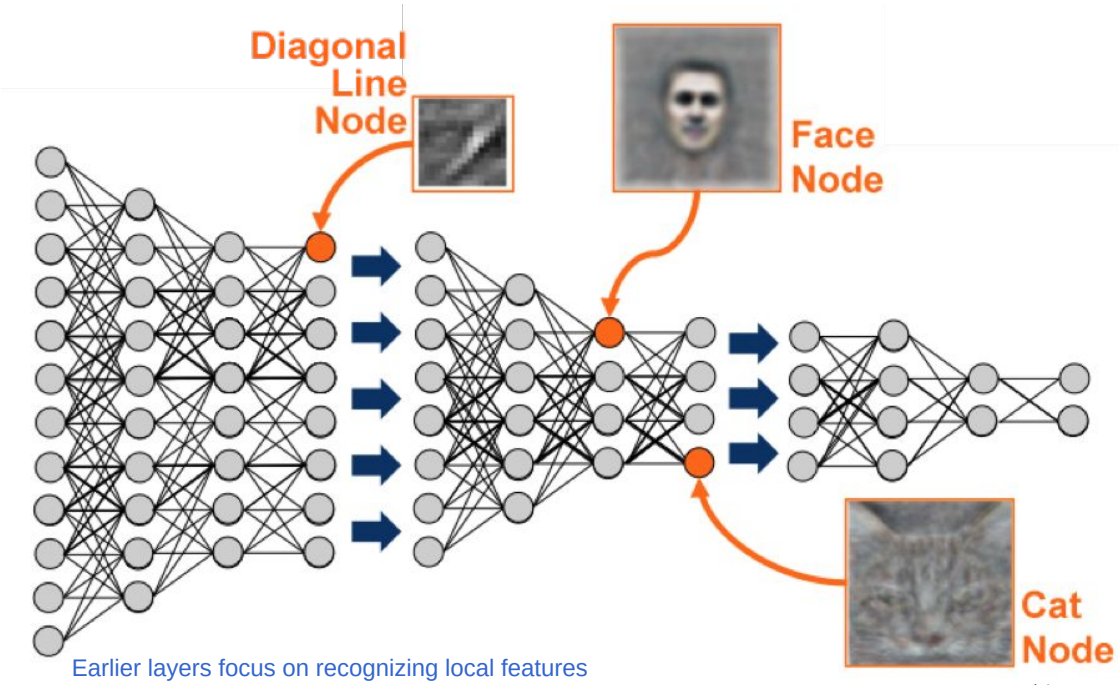
Q: How will this vehicle behave?

Connectionism (1980s): Networks, rather than IF/ELSE statements

Sample ELIZA Rule: **IF** User: "I [verb] [proper noun]",
THEN: ELIZA: "Why do you [verb] [proper noun]?"



"Deep" Learning (now): Networks with lots of layers



Earlier layers focus on recognizing local features

The popularity of AI research waxes and wanes.

AI has suffered two "winters" so far: one in the 1970s, and another in the 1990s:

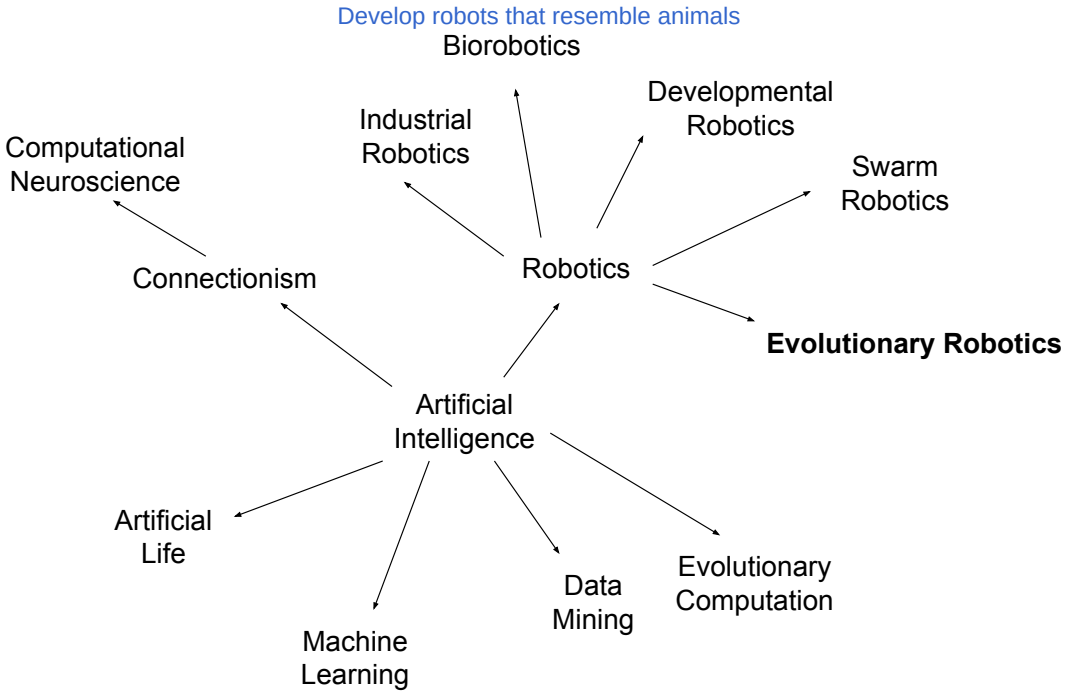
> en.wikipedia.org/wiki/AI_winter

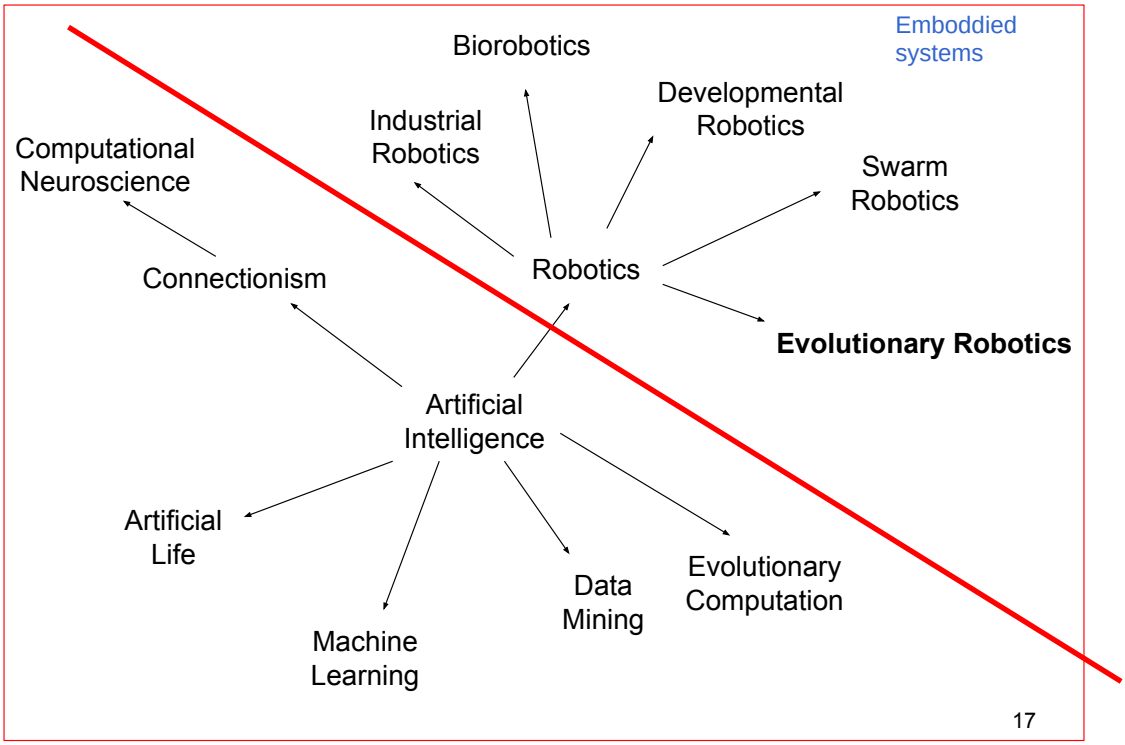
We are currently enjoying an AI "High Summer".

…is a third "winter" on the way?

Hans Moravec:

> "Many researchers were caught up in a web of increasing exaggeration. Their initial promises to DARPA had been much too optimistic. Of course, what they delivered stopped considerably short of that. But they felt they couldn't in their next proposal promise less than in the first one, so they promised more."
>
> - p. 115, Crevier, Daniel (1993),
> *AI: The Tumultuous Search for Artificial Intelligence*,
> New York, NY: BasicBooks, ISBN 0-465-02997-3


Develop robots that resemble animals
Biorobotics
Industrial Robotics
Developmental Robotics
Computational Neuroscience
Swarm Robotics
Connectionism
Robotics
Evolutionary Robotics
Artificial Intelligence
Artificial Life
Data Mining
Evolutionary Computation
Machine Learning


Emboddied systems
Biorobotics
Industrial Robotics
Developmental Robotics
Computational Neuroscience
Swarm Robotics
Connectionism
Robotics
Evolutionary Robotics
Artificial Intelligence
Artificial Life
Data Mining
Evolutionary Computation
Machine Learning

Not embodied approach

Cognitive ability: Pattern recognition

Challenge: How to recognize objects in a scene; where are the object's boundaries? This problem is known as 'image segmentation' in the field of computer vision.

It was one of the most difficult problems in computer science.

Image segmentation

Cognitive ability: Pattern recognition

Challenge: How to recognize objects in a scene; where are the object's boundaries? This problem is known as 'image segmentation' in the field of computer vision.

Solution:
Cog: ***interacts*** with the scene,
rather than passively observing it.

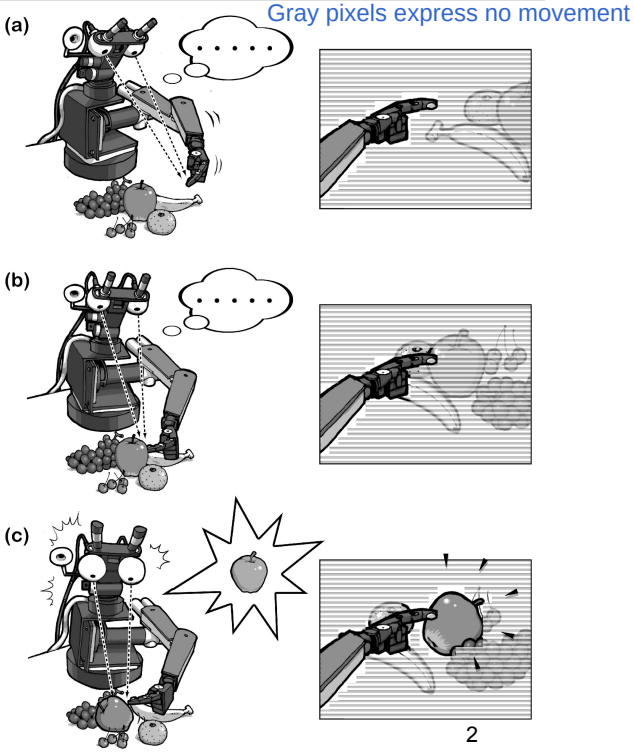
Q: What else can Cog infer about the objects it interacts with?

Early integration of vision and manipulation.

G Metta, P Fitzpatrick

Adaptive Behavior, 2003

Gray pixels express no movement



The idea behind an embodied approach is that you learn by interacting with the environment rather than through a huge dataset.

Embodied approach can also be done through a simulated body.

Cognitive ability: Planning

Not embodied planner

Kasparov vs. Deep Blue:
Won its first game against world champion: February, 1996
Won its first match against world champion: November, 1997

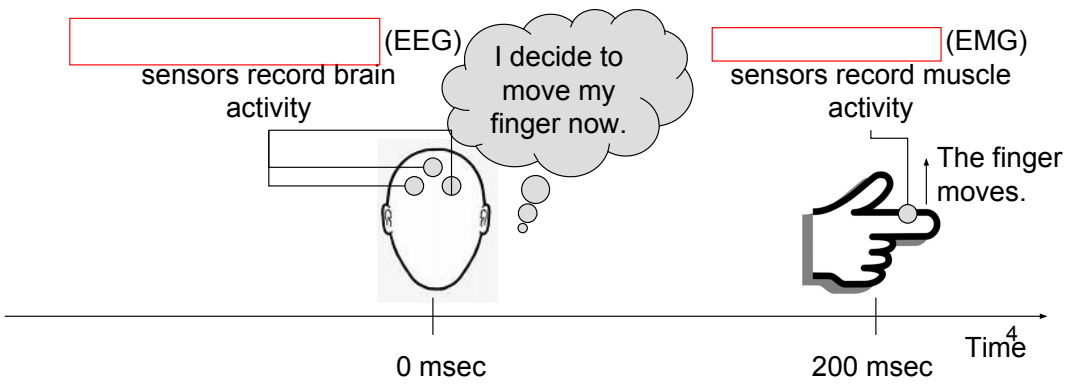
Q: How does Deep Blue work?

Building a tree of the possible moves that are generated depending on the actions taken by Deep Blue

Cognitive ability: Free will

Example: Another aspect of human cognition: free will.
Most people believe that they have free will:
"I decide in my mind that I want to do something, and then I do it."

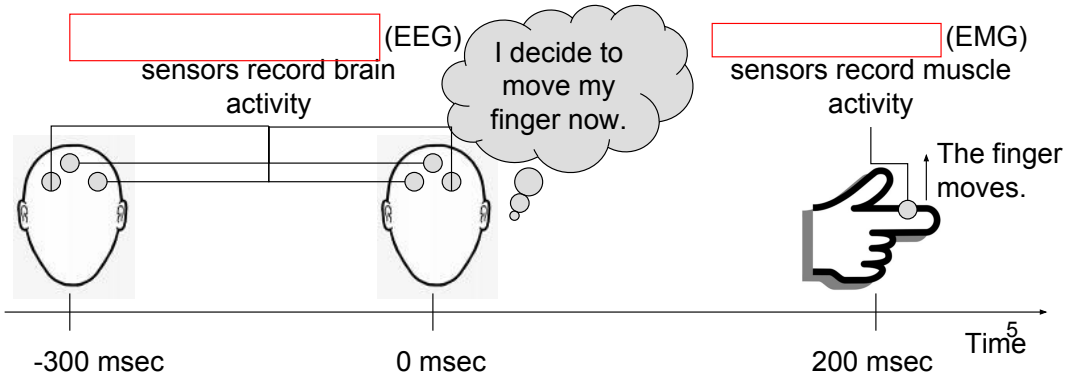
Libet, B., Gleason, C. A., Write, E. W., and Pearl, D. K. (1983). Time of conscious intention to act in relation to onset of cerebral activity (readiness-potential): The unconscious initiation of a freely voluntary act. *Brain*, **106**: 623-642.



Warning: Thinking about thinking is misleading; <u>introspection is dangerous</u>

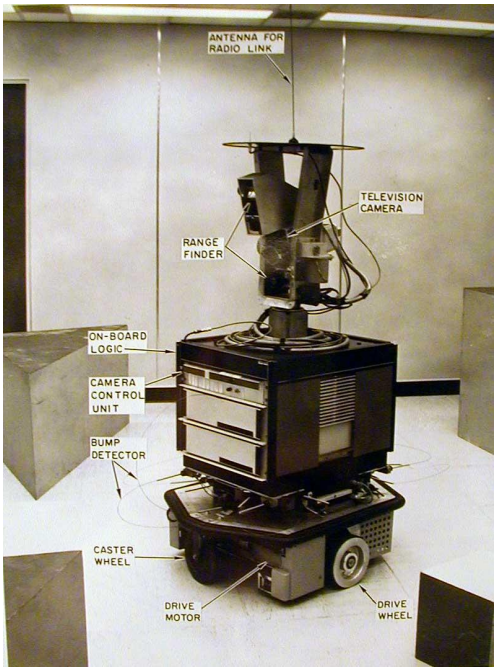
Example: Another aspect of human cognition: free will.
Most people believe that they have free will:
"I decide in my mind that I want to do something, and then I do it."
→ Thinking about free will seems to imply that free will exists … but does it?

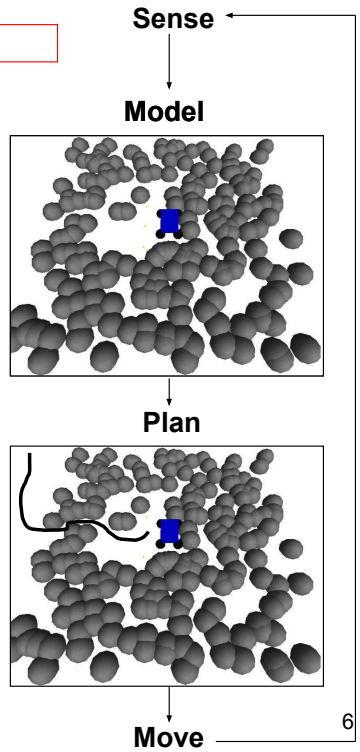
Observation: Brain activity associated with the finger movement begins 300 msec before the subject experiences the conscious will to move the finger.
Conclusion: If free will is unconscious, and our consciousness has no control over it, then it can't be "free" will.



Embodied Planning

Shakey the robot (Stanford Research Institute, 1966-1972)

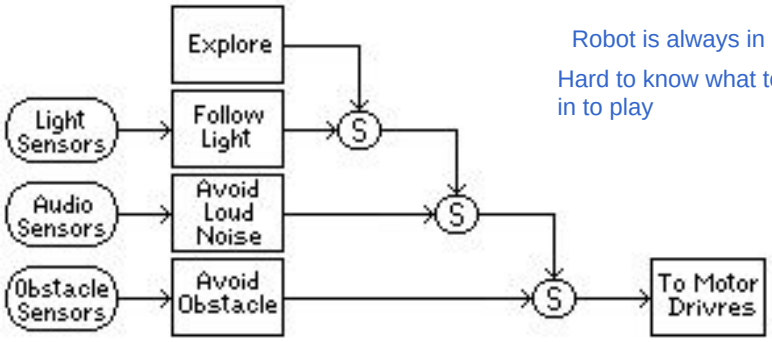




This model does not hold for dynamic environments

Roomba has this architecture in its software

Subsumption Architecture



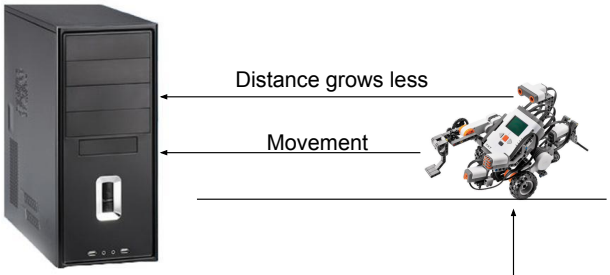
Robot is always in motion

Hard to know what to do when multiple sources come in to play

1. Behaviors are arranged in parallel, not serial.
2. One behavior is always in control at any one time.
3. When stimuli are not present, higher levels *subsume* control of the robot.

Embodied cognition

(This does not count as a body→)



Why not?
A body is a tool for affecting the world,
and for being affected by it.

Embodied cognition: the way you process information is affected by the fact that you have a body.

Example: If you have a body that can move, and you can see, then moving will cause *immediate* feedback.

Non-embodied technologies (such as computers) must wait for feedback from another computer (via packets) or a person (via an input device).

Situated cognition

(**Embodied cognition**: the way you process information is affected by the fact that you have a body.)

Situated cognition: the way you process information is affected by the fact that you are physically situated in the world.

Example: Embedded devices

Changes in sensor readings are often the result of physical processes:
1. they occur in real-time,
2. they are not under the control of the device or agent,
3. and they do not wait for a signal from the device to change.

Specific example: A wireless sensor senses changes in light levels; light levels change regardless of whether the device records the data or not.

A computer: Data within computer memory does not change, unless it is explicitly changed by the computer itself.

Embodied and Situated cognition

(**Embodied cognition**: the way you process information is affected by the fact that you have a body.)

(**Situated cognition:** the way you process information is affected by the fact that you are physically situated in the world.

A **complete agent** is an agent that is both situated and embodied.

Complete agents have three important properties that distinguish them from other kinds of agents:

1. They are subject to the laws of physics (by being in the world).
   Q: Examples?

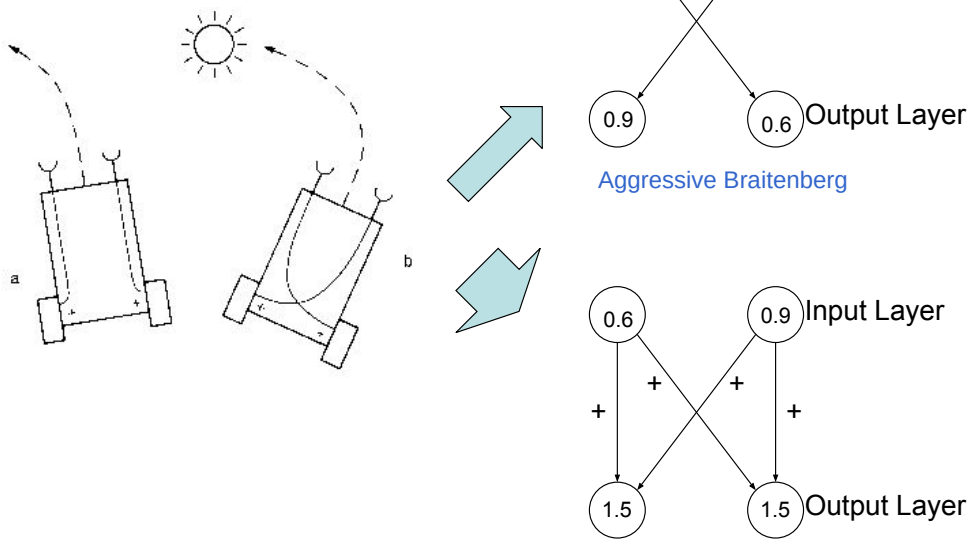
2. They generate sensory stimulation (through behavior).
   Q: Examples?

3. They affect the environment (through behavior).
   Q: Examples?

Classification of technologies:

	Disembodied	Embodied
Not situated	Computer	RC Car
Situated	Sensors Embedded device	Humans Living organisms Autonomous robots

Artificial Neural Networks

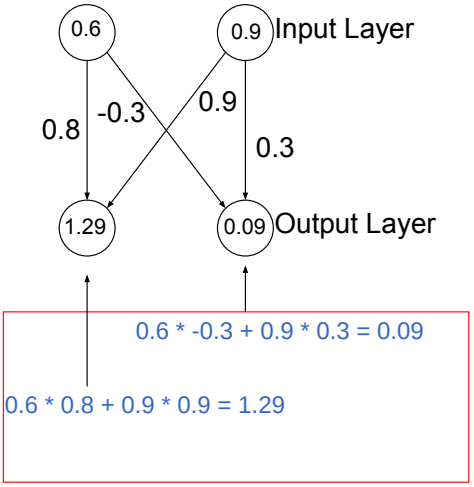


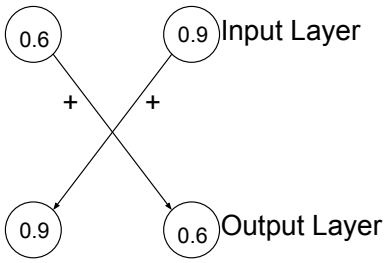
Aggressive Braitenberg

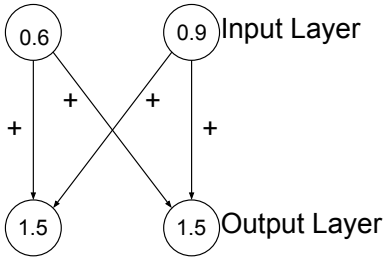
Robot can only go in straight line
It can go either faster or slower

Synaptic Weights

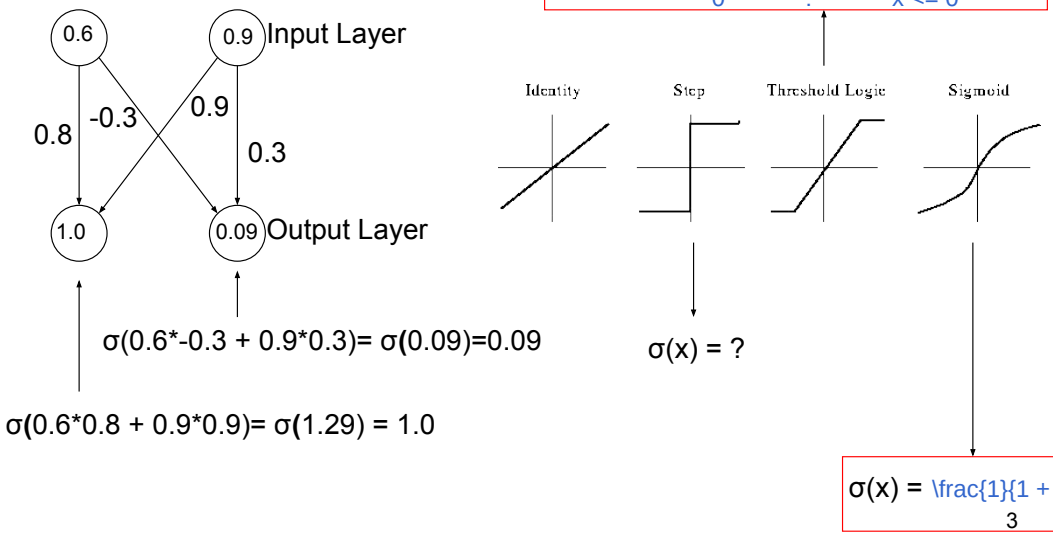
Weights represents the level of influence of the neurons



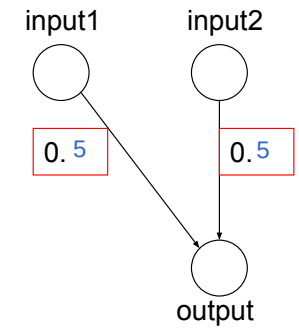




Activation functions: keeping neuron values within limits



Neural networks as functions



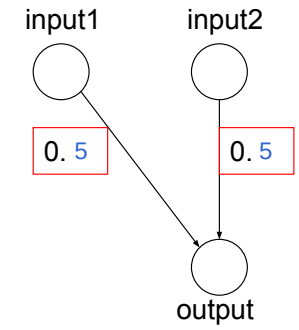
$$\sigma(x) = \begin{cases} 1 & : \quad x > \\ 0 & : \quad x <= \end{cases} 0.8$$

Truth Table: AND

input1	input2	output
0	0	0
0	1	0
1	0	0
1	1	1

σ(0* + 0*) = σ( ) = 0
σ(0* + 1*) = σ( ) = 0
σ(1* + 0*) = σ( ) = 0
σ(1* + 1*) = σ( ) = 1

Neural networks as functions



$$\sigma(x) = \begin{cases} 1 & : \quad x > \\ 0 & : \quad x <= \end{cases} 0.4$$

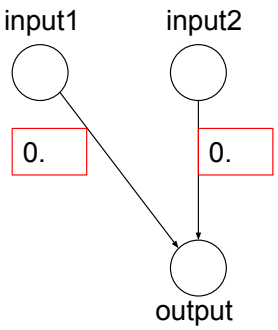
Truth Table: or

input1	input2	output
0	0	0
0	1	1
1	0	1
1	1	1

σ(0* + 0*) = σ( ) = 0
σ(0* + 1*) = σ( ) = 1
σ(1* + 0*) = σ( ) = 1
σ(1* + 1*) = σ( ) = 1

Neural networks as functions

Cannot be done with this structure



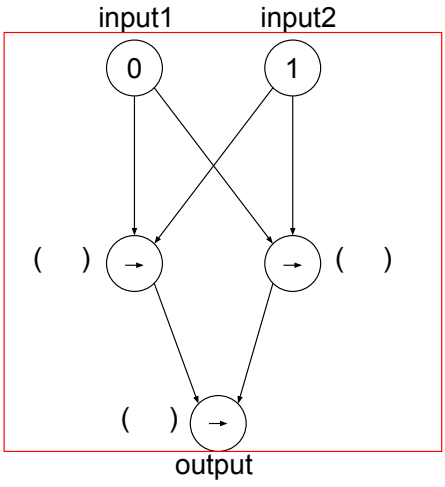
$$\sigma(x) = \begin{cases} 1 & : \quad x > \\ 0 & : \quad x <= \end{cases}$$

Truth Table: xor

input1	input2	output
0	0	0
0	1	1
1	0	1
1	1	0

σ(0* + 0*) = σ( ) = 0
σ(0* + 1*) = σ( ) = 1
σ(1* + 0*) = σ( ) = 1
σ(1* + 1*) = σ( ) = 0

Hidden neurons allow for nonlinear transformations

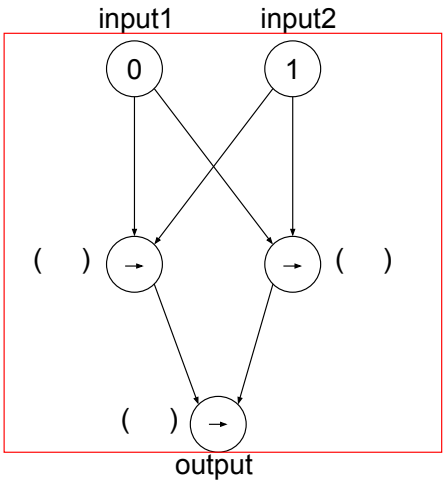


$$\sigma(x) = \begin{cases} 1 & : \quad x > \ (\quad) \\ 0 & : \quad x <= \ (\quad) \end{cases}$$

Truth Table: xor

input1	input2	output
0	0	0
0	1	1
1	0	1
1	1	0

Hidden neurons allow for nonlinear transformations

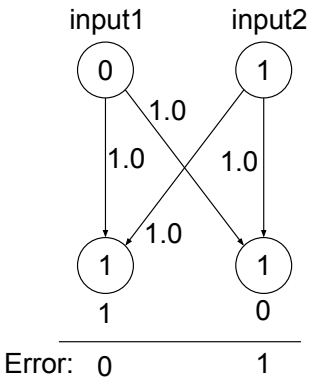


Truth Table: xor

input1	input2	output
0	0	0
0	1	1
1	0	1
1	1	0

$$\sigma(x) = \begin{cases} 1 & : \quad x > \ (\quad) \\ 0 & : \quad x <= \ (\quad) \end{cases}$$

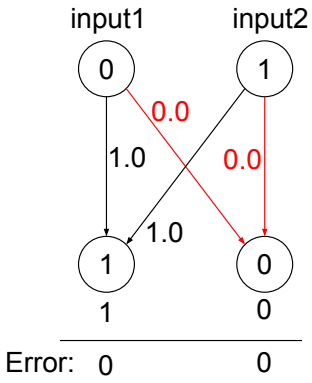
Backpropagation: Learning synaptic weights through training



Truth Table:

Input1	input2	out1	out2
0	0	…	…
0	1	1	0
1	0	…	…
1	1	…	…

Backpropagation: Learning synaptic weights through training



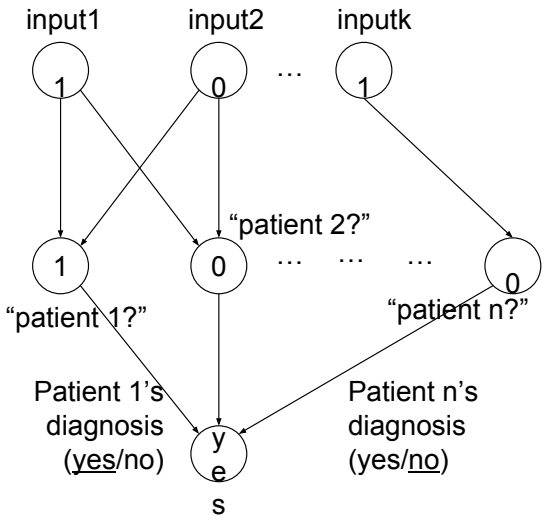
Truth Table:

Input1	input2	out1	out2
0	0	…	…
0	1	1	0
1	0	…	…
1	1	…	…

localized change to the synaptic weights →
low chance of increasing error for another input pattern →

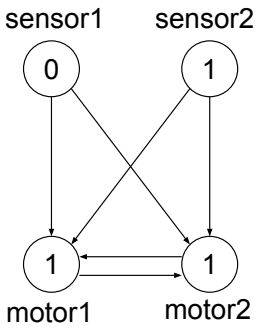
"using **tweezers** on the neural network"

Overfitting: Failing to learn the proper relationship between input and output

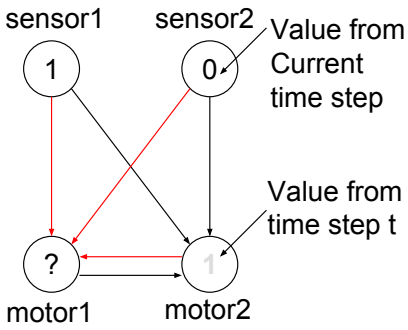


	symptom1	…	symptomk	Disease?
patient1	1	0	1	yes
patient2	0	0	1	no
….				
patient n	1	0	0	no

Recurrent connections: Adding memory to a neural network



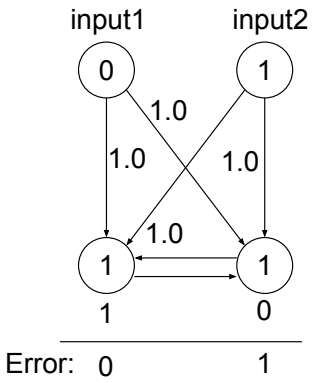
Robot at time step t



Robot at time step t+1

Value of motor 1 is a function of the current sensor values, and the value of motor 2 from the previous time step.

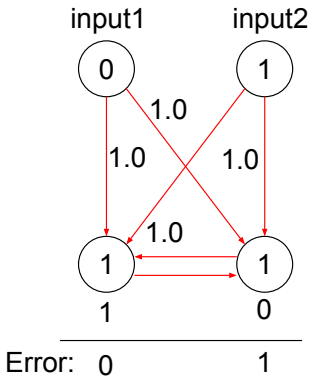
Recurrent connections: Adding memory to a neural network



Truth Table:

Input1	input2	out1	out2
0	0	…	…
0	1	1	0
1	0	…	…
1	1	…	…

Backpropagation: Does not work for recurrent connections



Truth Table:

Input1	input2	out1	out2
0	0	…	…
0	1	1	0
1	0	…	…
1	1	…	…

global change to the synaptic weights →
good chance of increasing error for another input pattern →

"using a **sledge hammer** on the neural network"

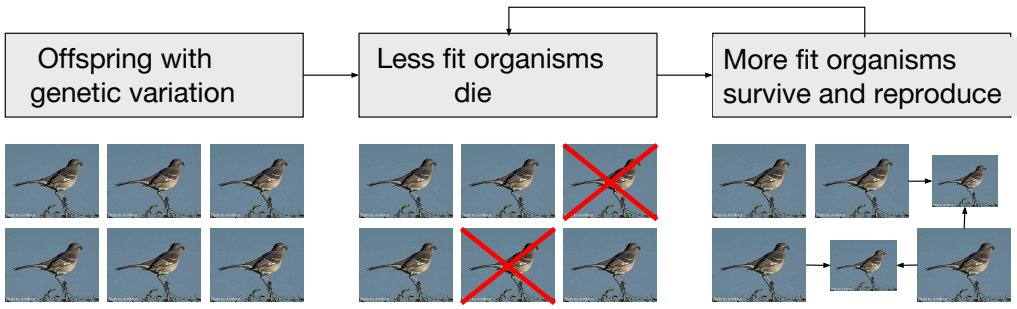
Supervised Learning: Correct output for each input pattern is known.

Input1	…	Inputn	Output1	…	Outputm
1	…	0	1	…	1
0	…	1	0	…	1
1	…	1	0	…	0

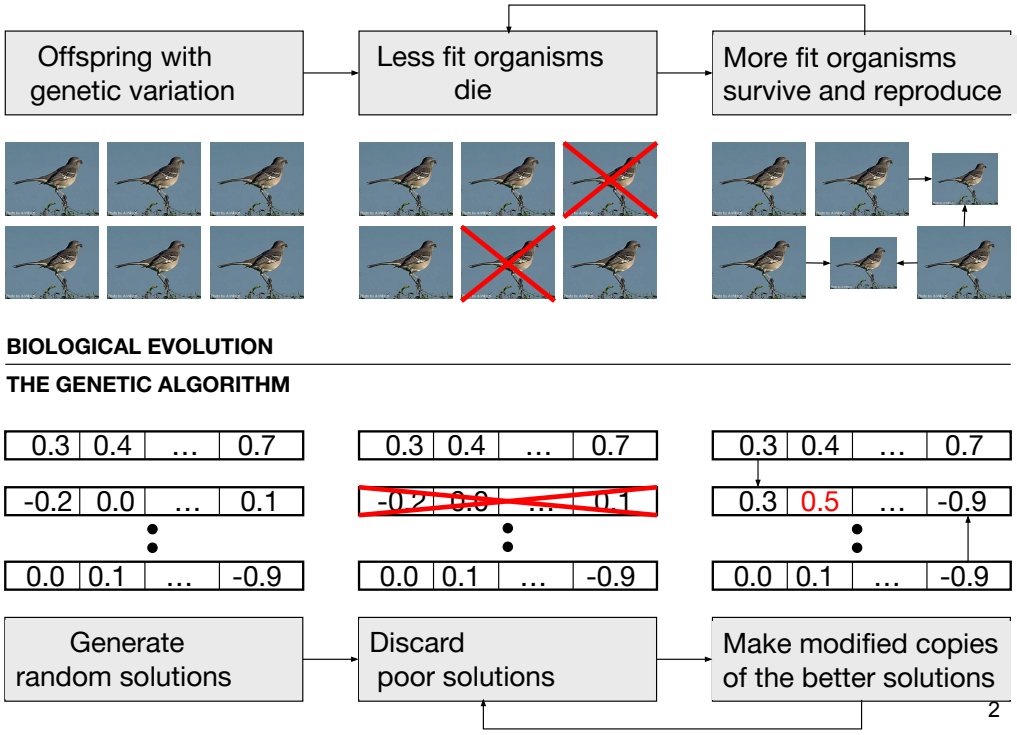
Evolutionary robotics falls into the semi-supervised learning area.
More similar to reinforcement learning.

Semi supervised Learning: Reward signal returned for a series of input patterns.

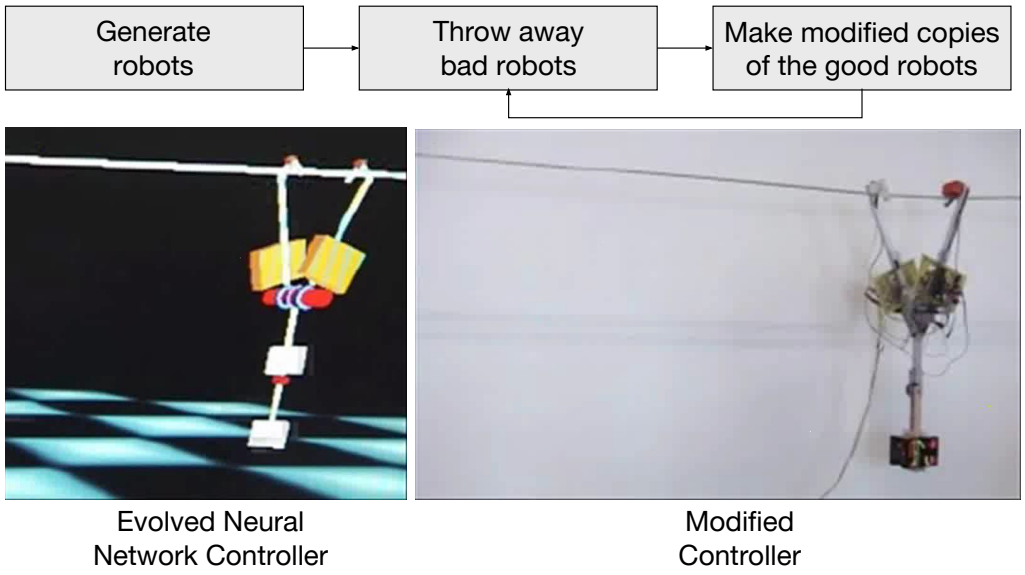
Sensor1	…	Sensorn	Motor1	…	Motorm
1	…	0		…	
0	…	1		…	
1	…	1		…	
Distance:			6.3 meters		



BIOLOGICAL EVOLUTION

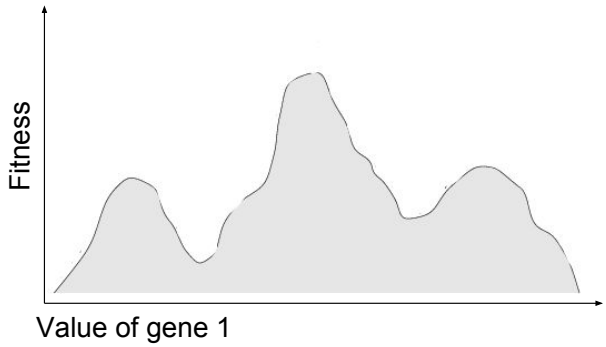


Genetic algorithm + robot simulator = evolutionary robotics



Frutiger, D. R., J. C. Bongard and F. Iida (2002) "Iterative Product Engineering: Evolutionary Robot Design", in Bidaud, P. and F. B. Amar (eds.), *Proceedings of the Fifth International Conference on Climbing and Walking Robots*, Professional Engineering Publishing, pp. 619-629.

The Fitness Landscape (Sewall Wright, 1932)

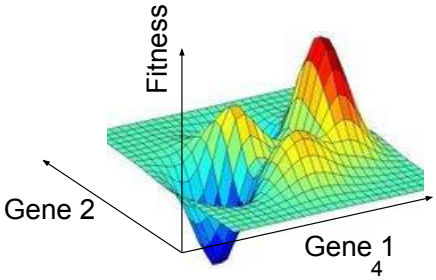


If a **genotype** has n genes, the fitness landscape exists in n+1 dimensions:

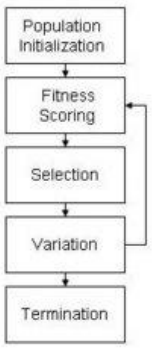
the fitness of the **phenotype** produced by that genotype is the extra dimension.

Genotype: Storage of genetic information

Phenotype: System produced by that information



The Hill Climber



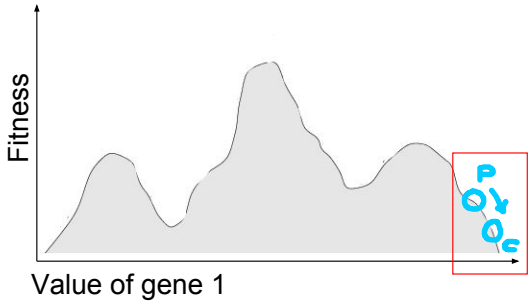
Population size: 1

0.3	0.4	…	0.7

"parent"

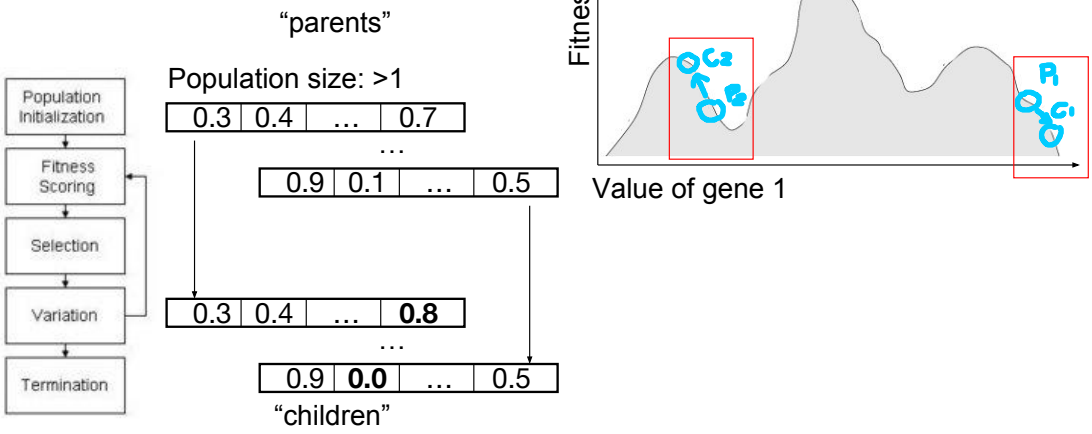
0.3	0.4	…	**0.8**

"child"



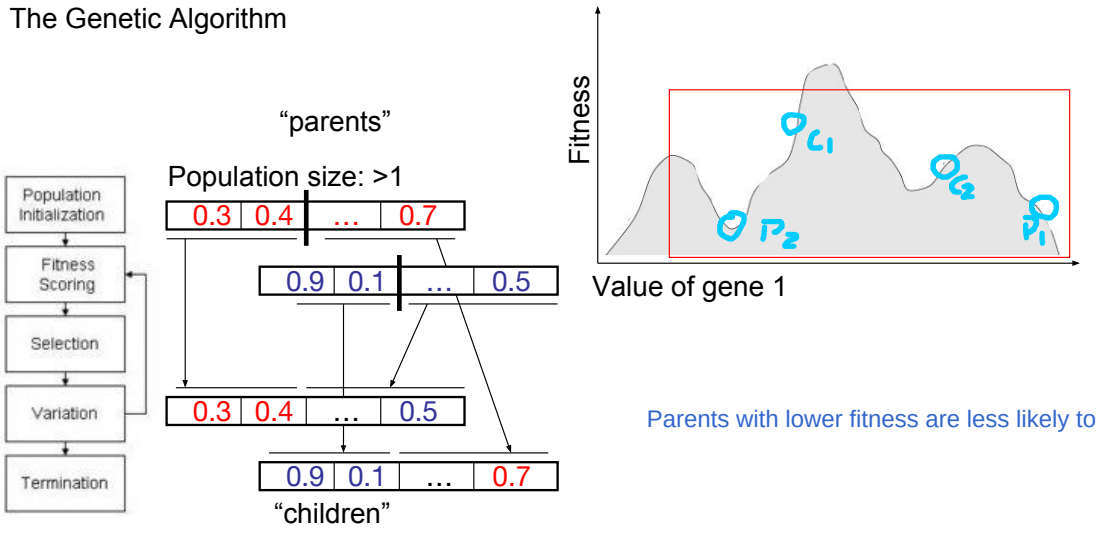
It has issues with local optimas

The Parallel Hill Climber



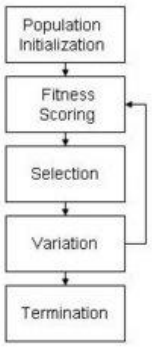
Two separate parent-children cases. They do not mix.

The Genetic Algorithm



Parents with lower fitness are less likely to produce offsprings

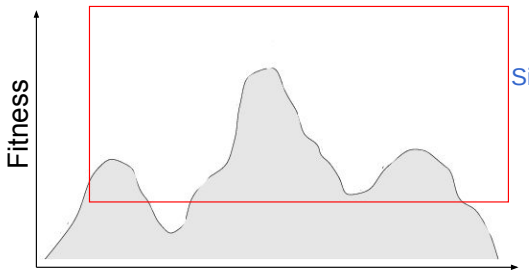
The Evolution Strategy



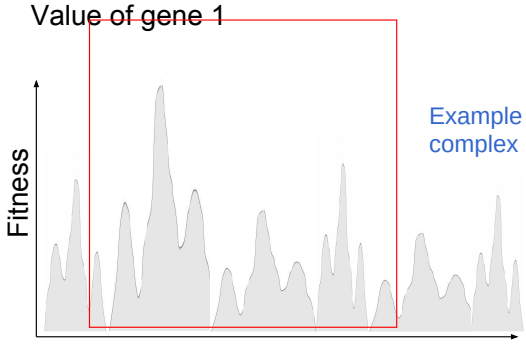
"Genes"

0.3	0.4	…	0.7
0.1	0.2	…	0.05

Step sizes, or strategy parameters



Simpler landascape



Example of a more complex landscape

You associate the step size additionally to the synaptic weights

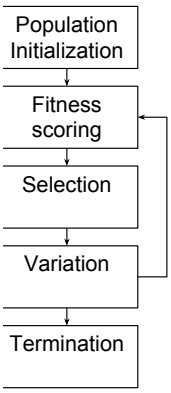
You associate each gene with a value to express how much you are going to change

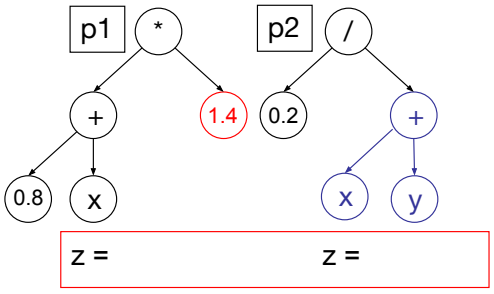
* You can dynamically adjust the step sizes over time (simulated annealing)
* Based on the relationship between the parent and the children in terms of their fitnesses.

One of the best ways is to use CMA-ES to adapt the step sizes in a smarter way

Symbolic regression

Genetic Programming





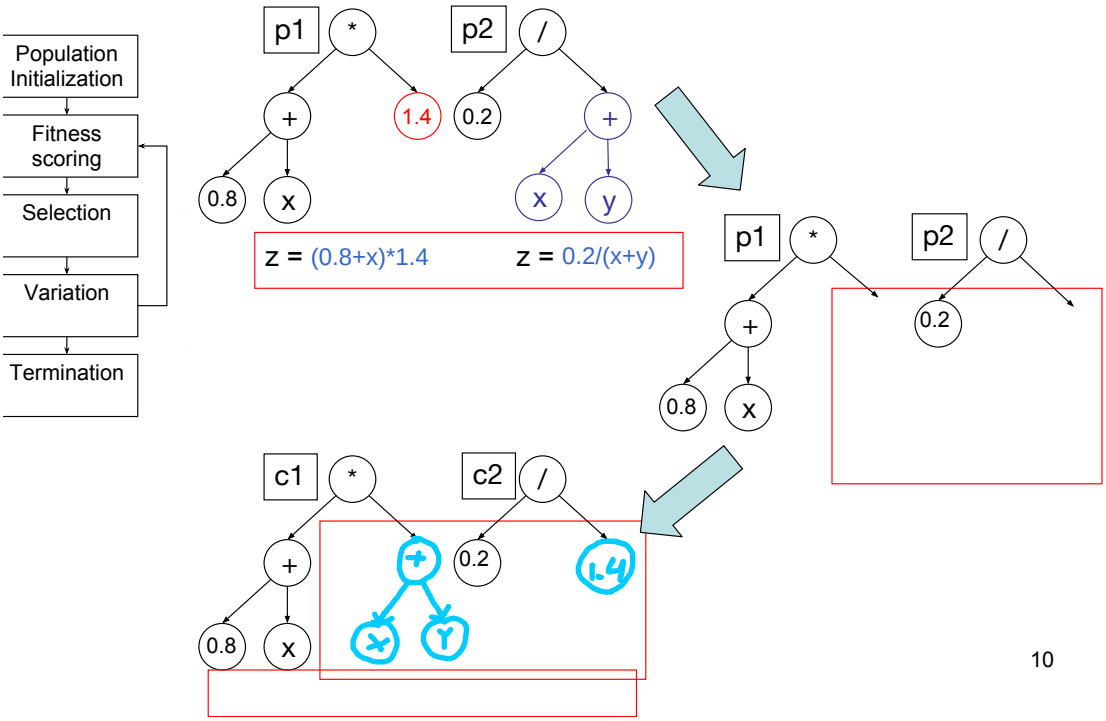
Possible branch nodes:	sin(a) cos(a) plus(a,b) minus(a,b) mult(a,b) div(a,b) pow(a,b)
Possible terminal nodes	x,y,… 0.1,-3.0,…

Genotype is the blueprint, the thing that evolution mutate.
Phenotype is the thing that the genotype produces.

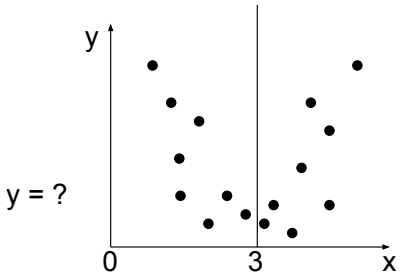
In genetic algorithms you deal with genes that are vectors.
The genomes are trees or forests.

Genotype are trees
Phenotype are the equations encoded by these trees.

Symbolic regression

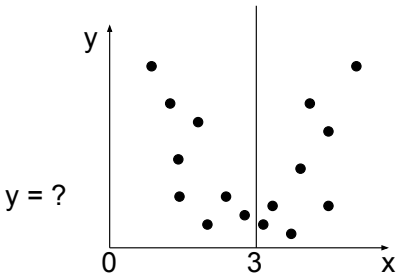


Genetic Programming: Symbolic Regression



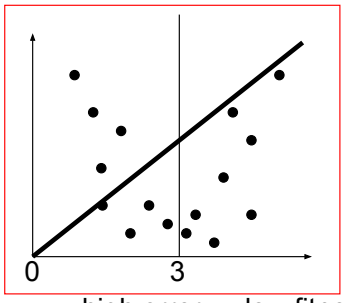
x	y
-0.4	1.4
0.0	0.6
…	…
0.5	2.1

Genetic Programming: Symbolic Regression



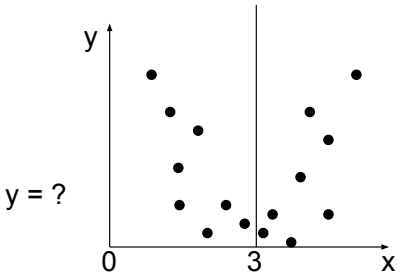
x	y
-0.4	1.4
0.0	0.6
…	…
0.5	2.1

$$\text{Error} = \sum_{i=1}^{n} \frac{\sqrt{(x_i - x_i')^2 + (y_i - y_i')^2}}{n}$$



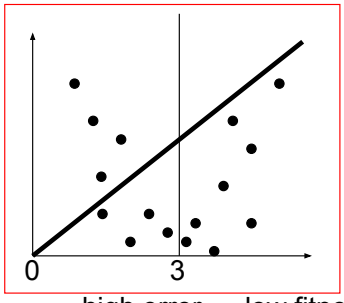
y = x → high error → low fitness

Genetic Programming: Symbolic Regression

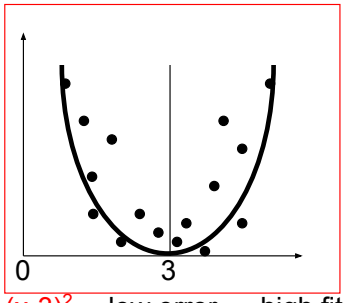


x	y
-0.4	1.4
0.0	0.6
…	…
0.5	2.1

$$\text{Error} = \sum_{i=1}^{n} \frac{\sqrt{(x_i - x_i')^2 + (y_i - y_i')^2}}{n}$$



y = x → high error → low fitness



y = $(x-3)^2$ → low error → high fitness

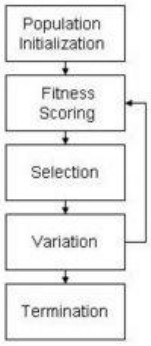
Genetic Programming: Symbolic Regression

Question: What causes toxic algal blooms in Lake Champlain?

i.e., T = ?

	Nitrogen (N)	Phosphorus (P)	Water temp (W)	Turbidity (B)	Rainfall (R)	…	Toxic algae/ml (T)
Malletts Bay 08/01/99	0.4	0.6	0.2	0.1	0.4	…	0.5
Shelburne 08/02/99	0.1	0.3	0.6	0.4	0.1	…	0.05
Malletts Bay 07/05/00	0.9	0.1	0.9	0.4	0.6	…	0.8
…	…	…	…	…	…	…	…

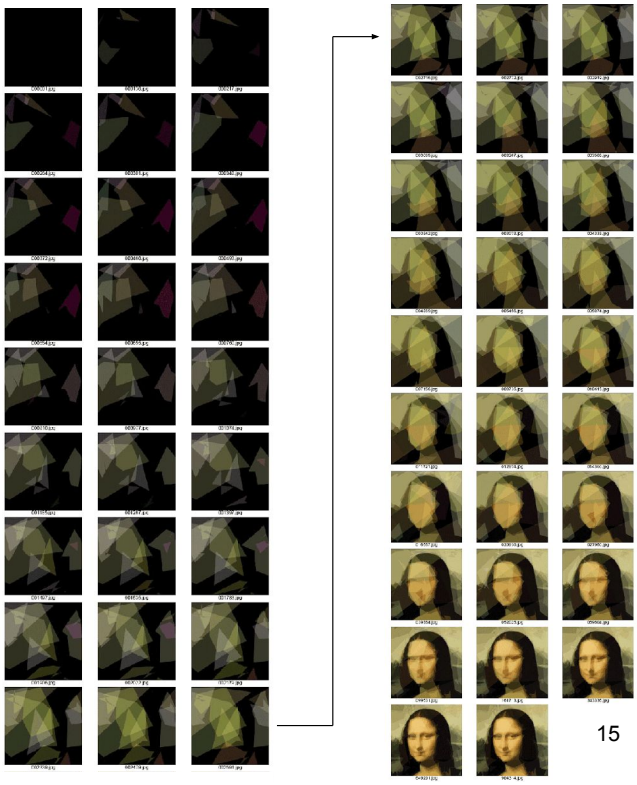
Genetic Programming



Q:
What is a good genotype?

What is the phenotype?

How to compute fitness?

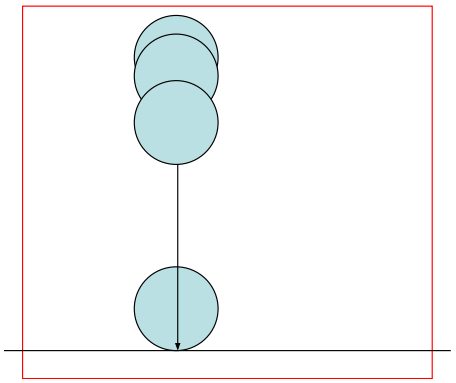


They are encoding colored polygons in the trees of the genetic programming.

Physical Simulation

1. Define the position, orientation, shape, mass and friction properties of each object.
2. Define how objects are attached together.
3. Start the simulation:

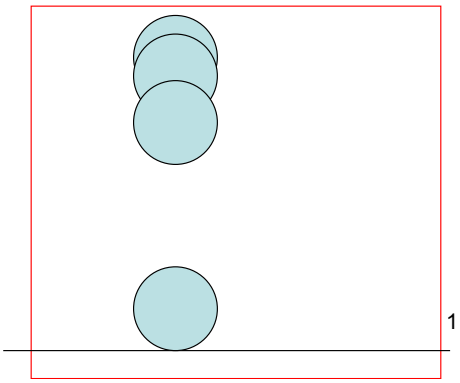
*Force = mass * acceleration (F=MA) → acceleration = Force / mass*



3D Computer Graphics

1. Define the position, orientation, and shape of each object at time t.
2. Render.
3. Define the position, orientation, and shape of each object at time t+1.

…



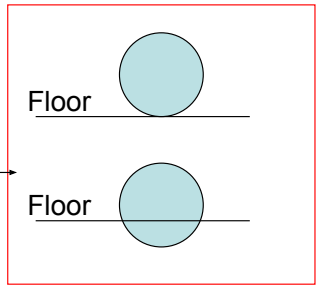
Preliminaries (From ode\demo\demo_boxstack.cpp)

dInitODE2(0); *Initialize Open Dynamics Engine*

world = dWorldCreate(); *contains all the objects*

space = dHashSpaceCreate (0); *contains all the objects that cannot interpenetrate*

contactgroup = dJointGroupCreate (0); *contains all the objects that are in contact*



dWorldSetGravity (world,0,0,-GRAVITY);

dWorldSetCFM (world,1e-5); *Has to do with friction; don't worry about for now.*

Bodies

from http://artis.imag.fr/Membres/Xavier.Decoret/resources/ode/tutorial1.html

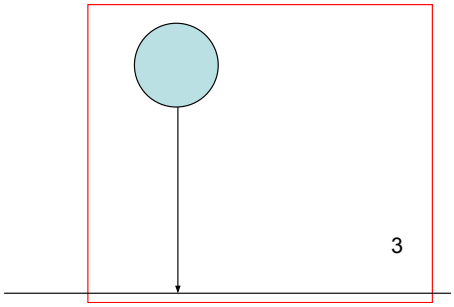
```
4   dBodyID body = dBodyCreate(world);
5
6   dMass mass;
7   dMassSetBox(&mass,1,1,1,1);
8   dMassAdjust(&mass,0.2f);
9
10  dBodySetMass(body,&mass);
11  dBodySetPosition(body,0,6,0);
```

Object has a density of '1'

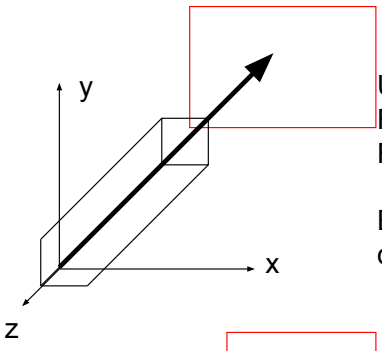
Object is a 1x1x1kg box

Object has a mass of 0.2kg.

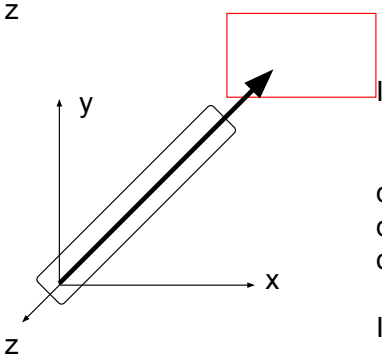
Object starts at position (x=0,y=6,z=0)



Bodies – specifying orientation in 3D



Usually requires four numbers:
First three define axis along which object lies;
Fourth specifies how much the object is rotated about that axis.
Encoded as a quaternion:
$q = \cos(\theta/2) + (x1,y1,z1)\sin(\theta/2)$



If object is a cylinder
we can specify orientation with 3 numbers:

```
dMatrix3 R;
dRFromZAxis(R,x1,y1,z1);
dBodySetRotation(body,R);
```

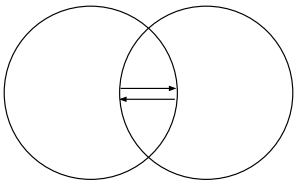
If object is a sphere… ?

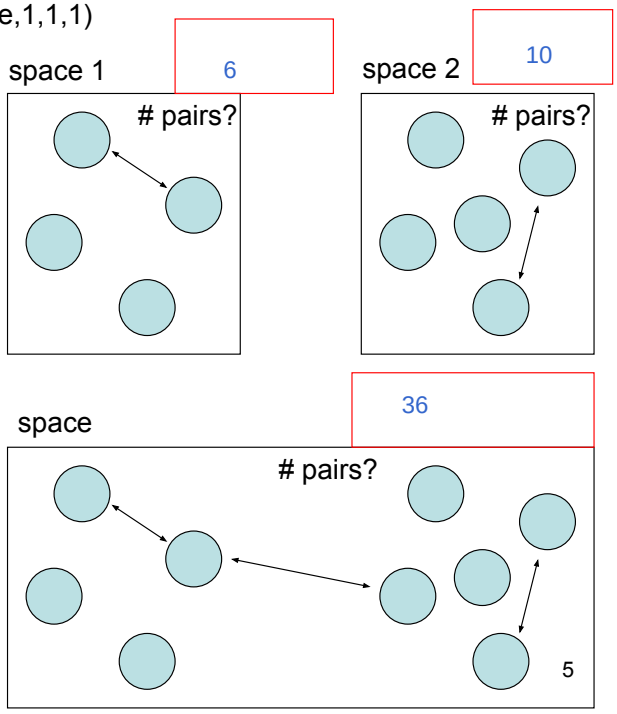
Geoms – specifying how bodies interact with other bodies

dGeomID geom = dCreateBox(space,1,1,1)
dGeomSetBody(geom,body);

To keep objects from colliding:

1. Compute the distance between each **geom** pair.
2. If distance<=0, push the **bodies** apart proportional to the interpenetration





Joints – specifying how bodies move relative to each other

dJointID hinge; *Define a hinge joint data structure.*

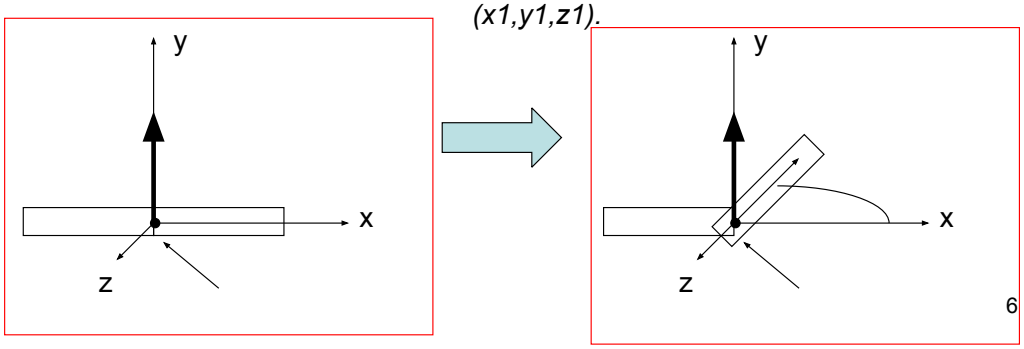
hinge = dJointCreateHinge (world,0); *Create a hinge joint data structure.*

dJointAttach (hinge,body[0],body[1]); *Which objects does the joint connect?*

dJointSetHingeAnchor (hinge,0,0,1); *Where is the joint's fulcrum?*

dJointSetHingeAxis (hinge,x1,y1,z1); *When the objects move relative to one another, what 2D plane do they define?*

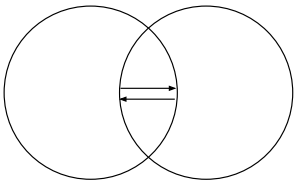
The plane that is perpendicular to vector (x1,y1,z1).

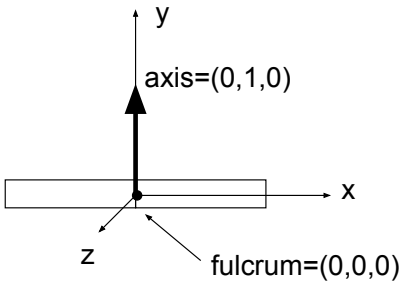


Contacts – what happens when two objects collide?

```
static void nearCallback (void *data, dGeomID o1, dGeomID o2)
{  int i;

// exit without doing anything if the two bodies are connected by a joint
dBodyID b1 = dGeomGetBody(o1);
dBodyID b2 = dGeomGetBody(o2);
if (b1 && b2 && dAreConnectedExcluding (b1,b2,dJointTypeContact))
      return;
                                . . .
```





Continue…

Return…

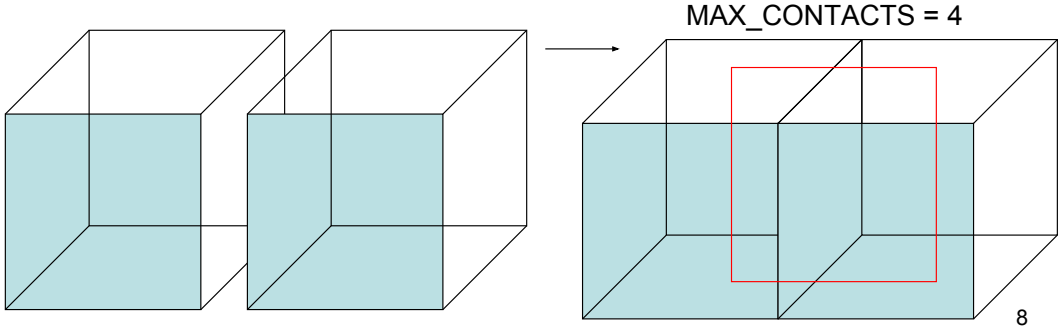
Contacts – what happens when two objects collide?

```
                                . . .
 dContact contact[MAX_CONTACTS];
 // up to MAX_CONTACTS contacts per box-box

for (i=0; i<MAX_CONTACTS; i++) {
      contact[i].surface.mode = dContactBounce | dContactSoftCFM;
contact[i].surface.mu = dInfinity;    contact[i].surface.mu2 = 0;
contact[i].surface.bounce = 0.1;    contact[i].surface.bounce_vel = 0.1;
contact[i].surface.soft_cfm = 0.01;
}                               . . .
```

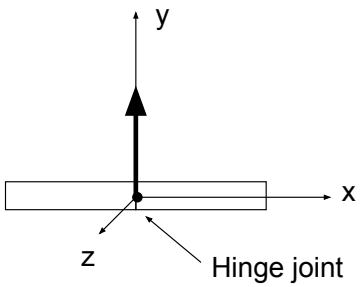
Describes the forces that arise at each contact point

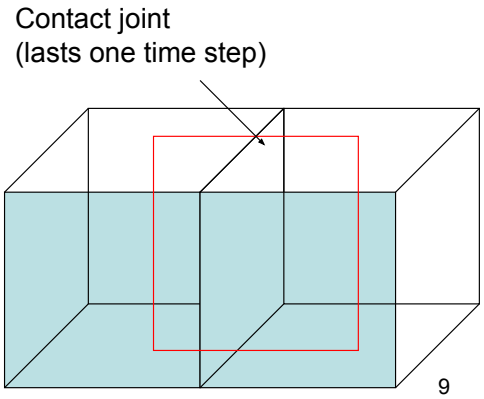
MAX_CONTACTS = 4



Contacts – what happens when two objects collide?

```
                                . . .
 if (int numc = dCollide (o1,o2,MAX_CONTACTS,&contact[0].geom,
sizeof(dContact))) {
                                . . .
      for (i=0; i<numc; i++) {
            dJointID c = dJointCreateContact (world,contactgroup,contact+i);
      dJointAttach (c,b1,b2);
      }
                                . . .
}
```





simLoop – updating the physics, and updating the graphics

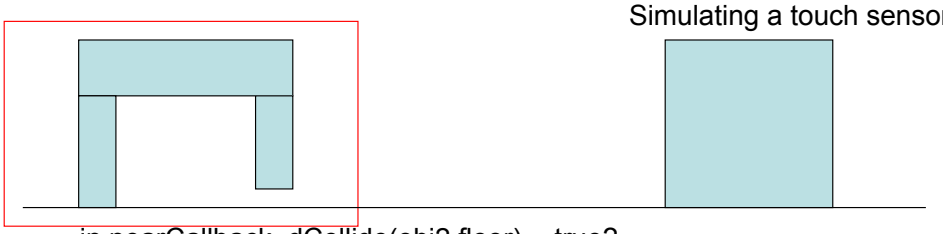
```
static void simLoop (int pause) {
                                . . .
      dSpaceCollide (space,0,&nearCallback);          Find out who collided
                                . . .
      if (!pause)
            dWorldStep(world,0.05);                   Update the positions
                                                      and orientations of all objects
                                . . .
      dJointGroupEmpty (contactgroup);                Delete all contact joints
                                . . .
      for (int object = 0; object < totalObjects ; object++)
                                                      Draw all objects
            dsSetColor (obj[object].r, obj[object].g, obj[object].b);
            Draw(obj[object]);
```

Timestep value is the one responsible of making the computations in the physics engine more accurate. The main problem is that it requires more computation

```
}
```

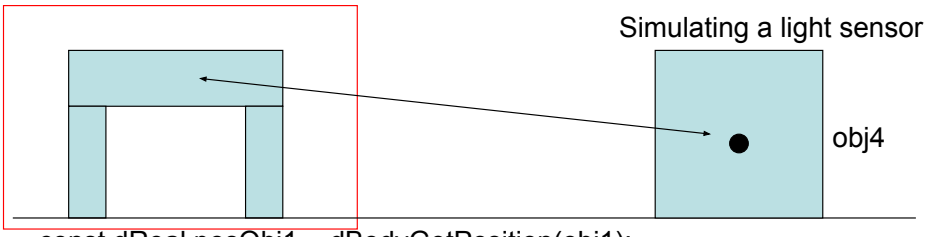
Sensors – Allowing the robot to sense in ODE…

Simulating a touch sensor



in nearCallback, dCollide(obj2,floor)==true?

Simulating a light sensor

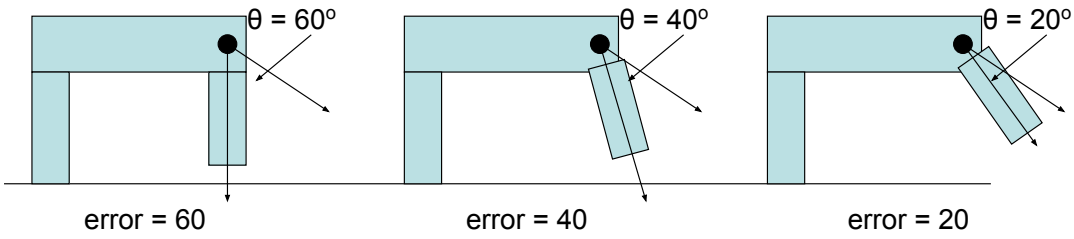


const dReal posObj1 = dBodyGetPosition(obj1);

const dReal posObj4 = dBodyGetPosition(obj4);

double lightLevel = 1 / (1 + EuclideanDistance(posObj1,posObj4))

Motors – Allowing the robot to move in ODE…



dJointSetHingeParam(joint,dParamFMax,motorForce); *The maximum force the joint can handle.*

double desiredAngle = 60; *Apply force to rotate the leg to 60 degrees forward.*

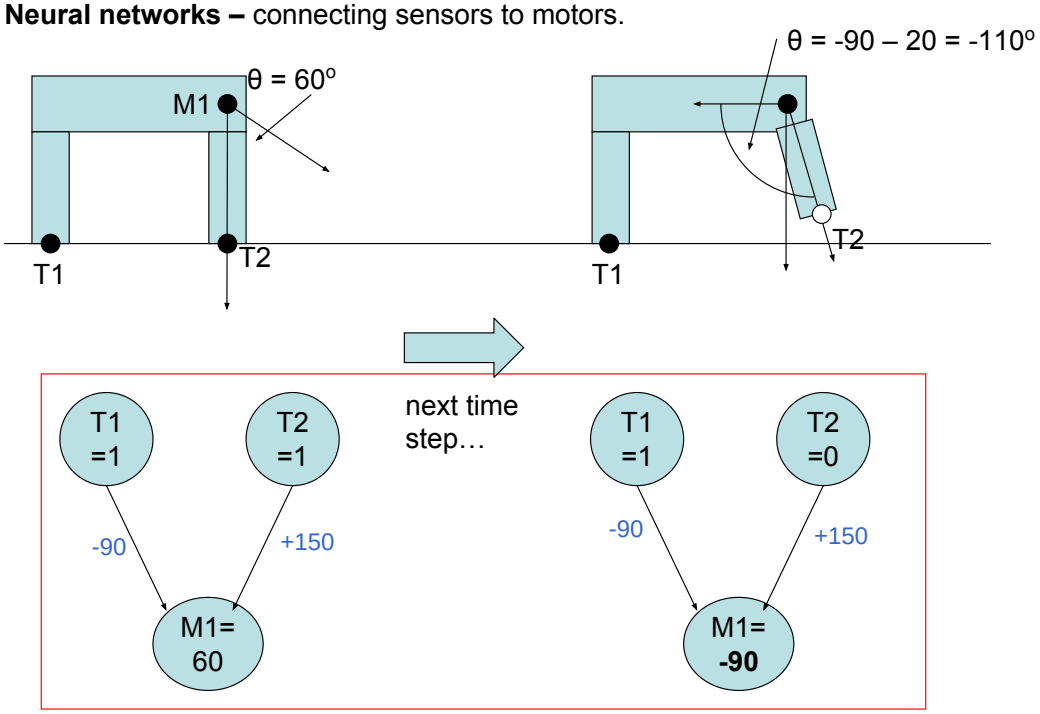
double actualAngle = dJointGetHingeAngle(joint); *What angle is the joint at now?*

double diff = desiredAngle – actualAngle; *How much difference is there?*

dJointSetHingeParam(joint,dParamVel, diff); *Apply force proportional to how far the joint is from where it should be.*

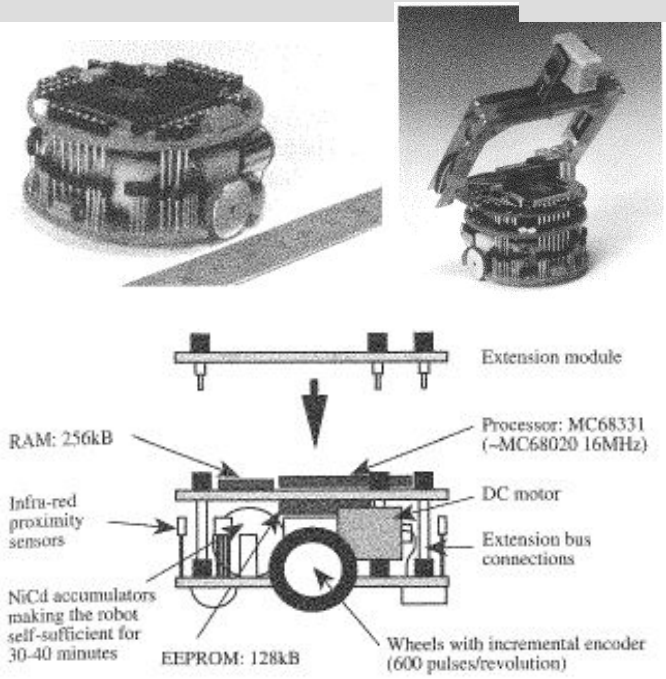
Positive numbers represent extension, negative numbers represent flexion

Neural networks – connecting sensors to motors.



13

The changes in the motor values generates changes in the inputs of the neurons as they are related to the sensors in the legs of the robot



The Italian Approach

The Khepera robot (1996)

Developed at EPFL
Lausanne, Switzerland(!)
by
Francesco Mondada
Diameter: 55 mm

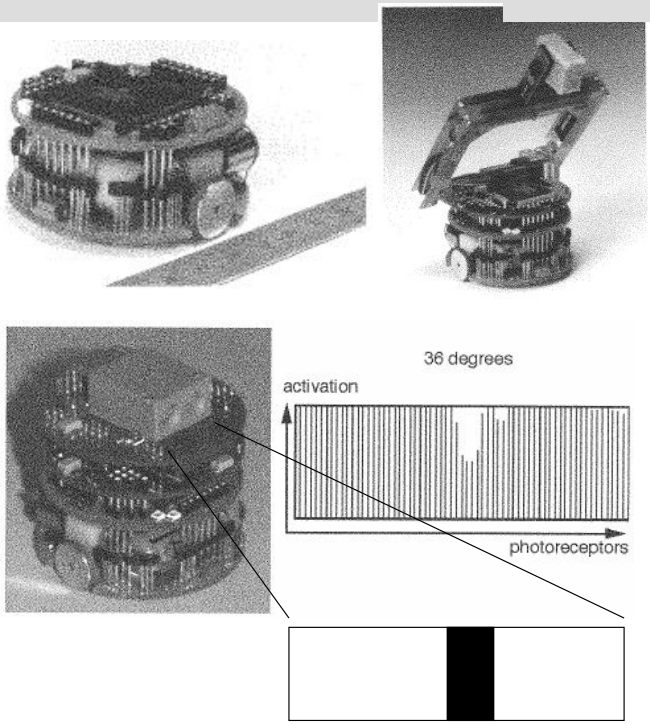
Could perform experiments on a table, rather than in a large arena.

Modular architecture:

Extension modules could be stacked onto the base.

Extensions: gripper.

1



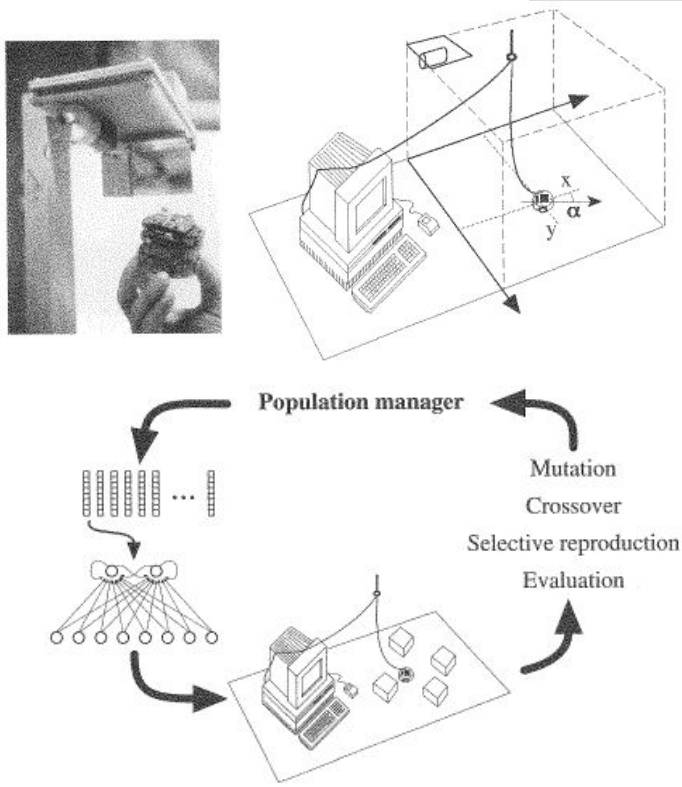
The Italian Approach

The Khepera robot (1996)

Developed at EPFL
Lausanne, Switzerland(!)
by
Francesco Mondada
Diameter: 55 mm

Could perform experiments on a table, rather than in a large arena.

Modular architecture:



The Italian Approach

Doing evolutionary robotics on a physical robot.

Robot tethered to computer:

computer does the evolution;

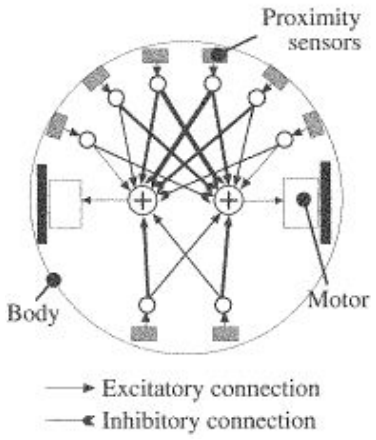
robot does the behaving.

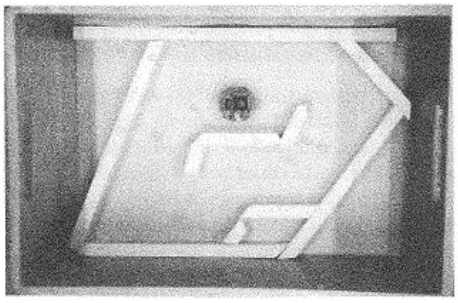
Laser-emitting device
Emits lasers into the workspace

Khepera has a positioning turret attached:

can detect the lasers and thus compute its position (x,y) and heading (α).

What behaviors could evolve?





The Italian Approach

Evolution of simple navigation:

Evolve a neural network such that the robot…

1. Circles through maze as fast as possible.
2. Does not hit the walls.

Robot is given the NN architecture shown.

Robot's wheels can…
- rotate backwards quickly (-0.5)
- stay still (0.0)
- rotate forward quickly (0.5)
- or anything in this range (-0.5,0.5)

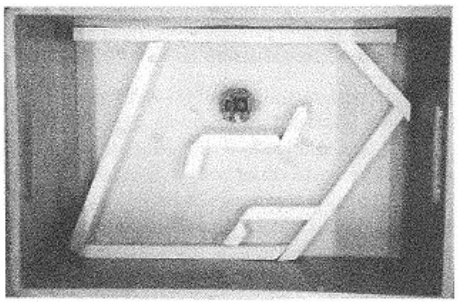
Robot's eight proximity sensors return…
- 0 if obstacle is 5cm or further (or absent)
- 1 if proximity sensor is touching object.

Create a fitness function to select for this behavior:

φ=

Hints:

φ=0 for worst performance
φ=1 for best possible performance (may not be reachable)
v_L, v_R = speed of left/right wheel
$i_1, \ldots i_8$=value of proximity sensor



The Italian Approach

Evolution of simple navigation:

Evolve a neural network such that the robot…

1. Circles through maze as fast as possible.
2. Does not hit the walls.

Robot is given the NN architecture shown.

Robot's wheels can…
- rotate backwards quickly (-0.5)
- stay still (0.0)
- rotate forward quickly (0.5)
- or anything in this range (-0.5,0.5)

Robot's eight proximity sensors return…
- 0 if obstacle is 5cm or further (or absent)
- 1 if proximity sensor is touching object.

Create a fitness function
to select for this behavior:

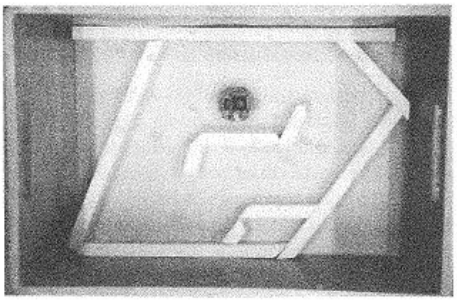
φ= V (1 - \sqrt(\Delta v)) (1-i)

V = |vL| + |vR| (0 -> 1)

\Delta v = |(vL + 0.5) - (vR + 0.5)|

\sqrt(\Delta v) = string weight to small differences

i = maximum firing proximity sensor



The Italian Approach

Evolution of simple navigation:

Evolve a neural network such that
the robot…

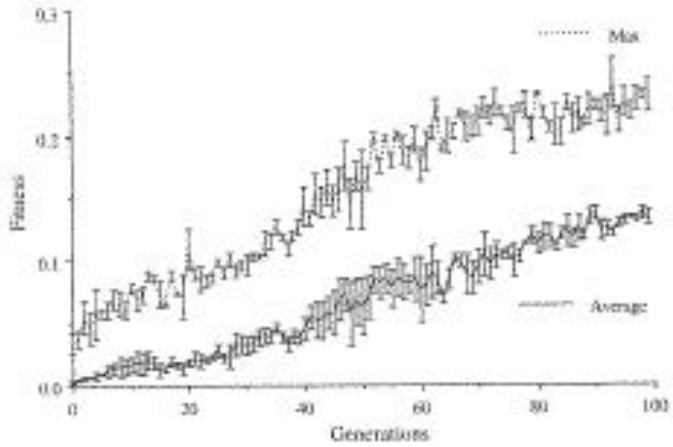
1. Circles through maze as fast as possible.
2. Does not hit the walls.

Robot is given the NN architecture shown.

Robot's wheels can…
rotate backwards quickly (-0.5)
stay still (0.0)
rotate forward quickly (0.5)
or anything in this range (-0.5,0.5)

Robot's eight proximity sensors return…
0 if obstacle is 5cm or further (or absent)
1 if proximity sensor is touching object.

The fitness function in here combines rewards and penalizing terms. The reward term is the V while the 1 - i works as a penalization term.

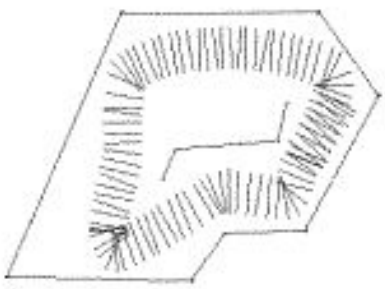


The Italian Approach

A sample run:

One generation:
40 minutes

100 generations:
66 hours
2.77 days

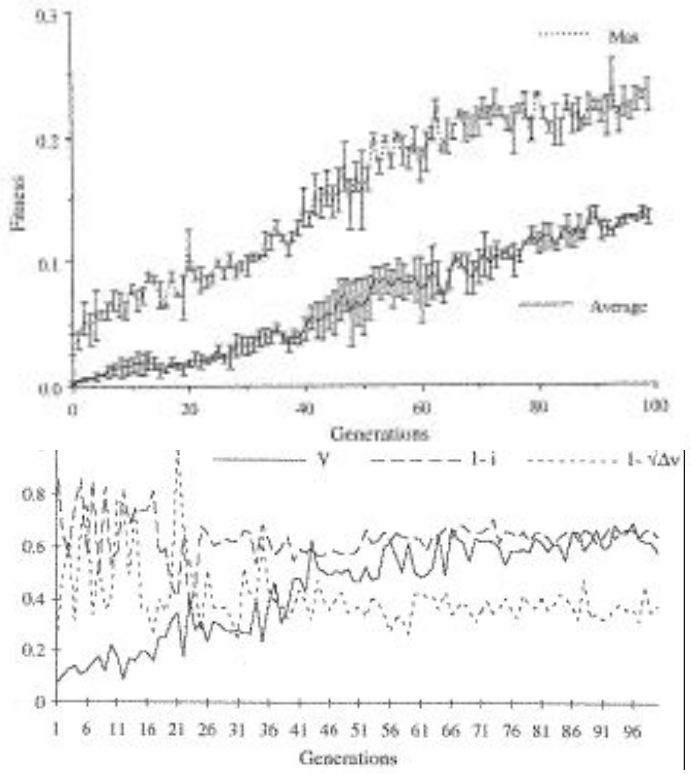


Line segment:

Center of segment
Indicates robot's position

Line passes through the
robot's left and right wheels.

The Italian Approach

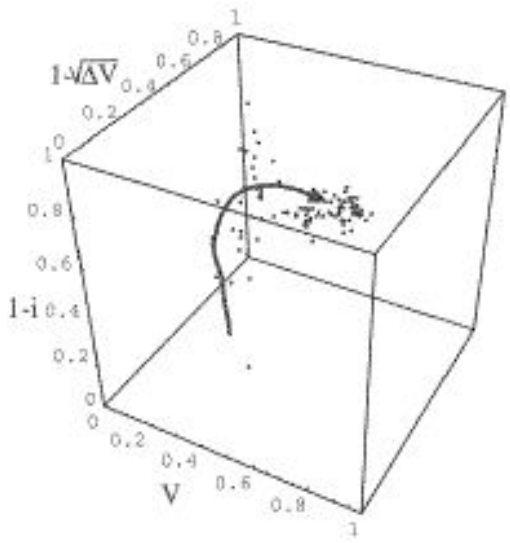


←Fitness components for the best controller at each generation.

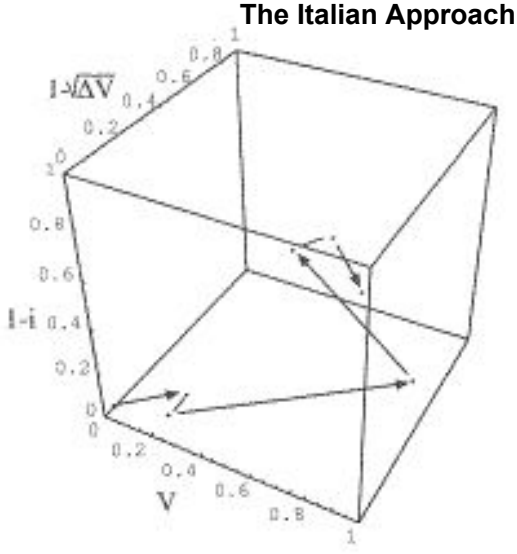
Q: How did the robot's behavior change over time?

Evolution is not an optimizer, but a satisfyser. It tries to find the balance between the desired aspects implicit in its fitness function

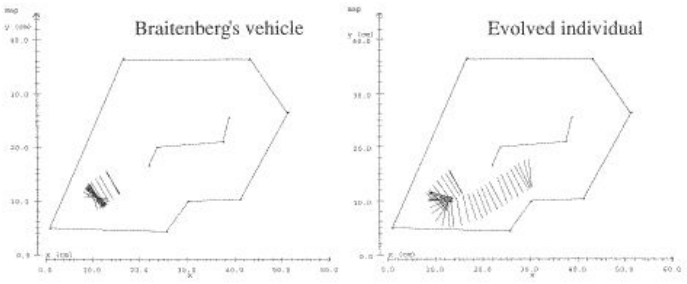
The Italian Approach

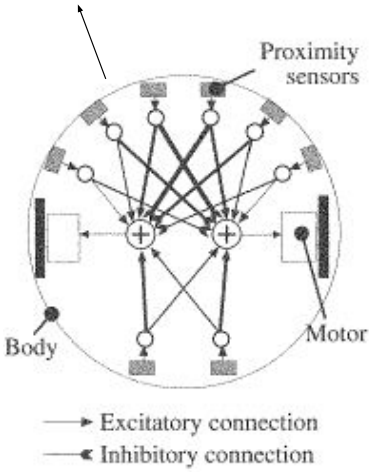


How the fitness components changed over evolutionary time (over 2.77 days)



How the fitness components changed for the best evolved controller, as it controlled the robot (~1 minute)





The Italian Approach

If synaptic weights (thickness of lines) are bilaterally symmetric,

robot gets stuck in corners.

Why?

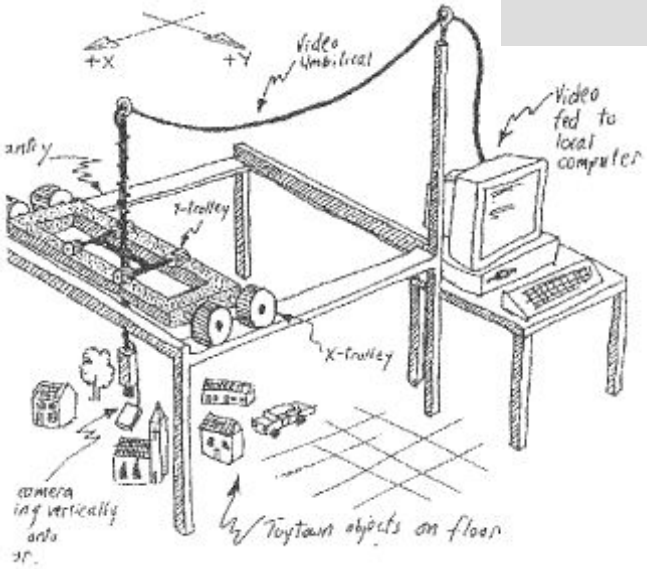
Evolved robot that approaches corners does not get stuck.

Why do you think this is so?

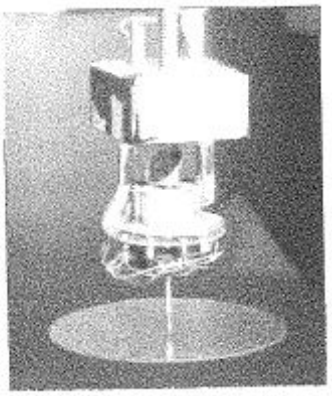
Evolved controllers only ever drove the robot at a Maximum speed of 48 mm/s.

Actual top speed is 80 mm/s.

Why not drive at top speed?



The English Approach

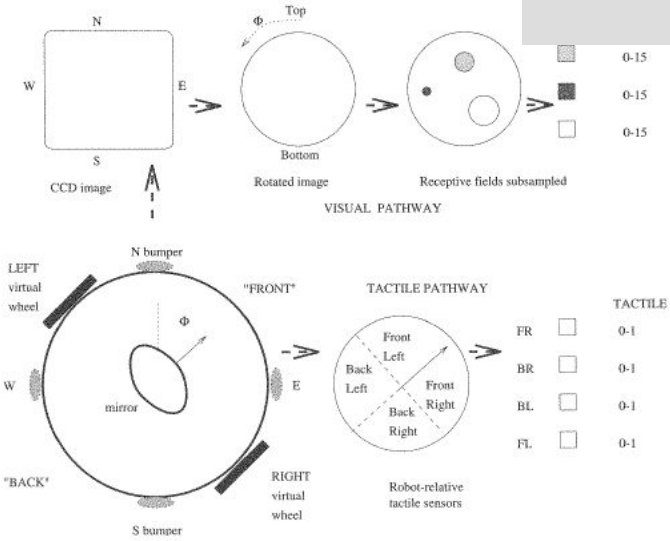


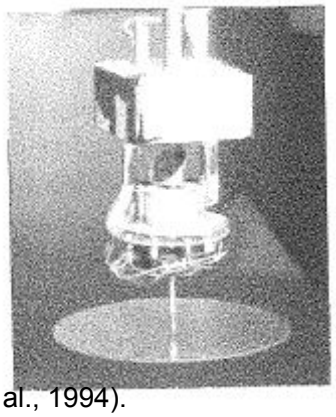
Gantry Robot (University of Sussex, Inman Harvey et al., 1994).

Three ways to move (degrees of freedom); three motors.

Back and forth on x-trolley; back and forth on y-trolley; tilted mirror can be rotated.

The English Approach





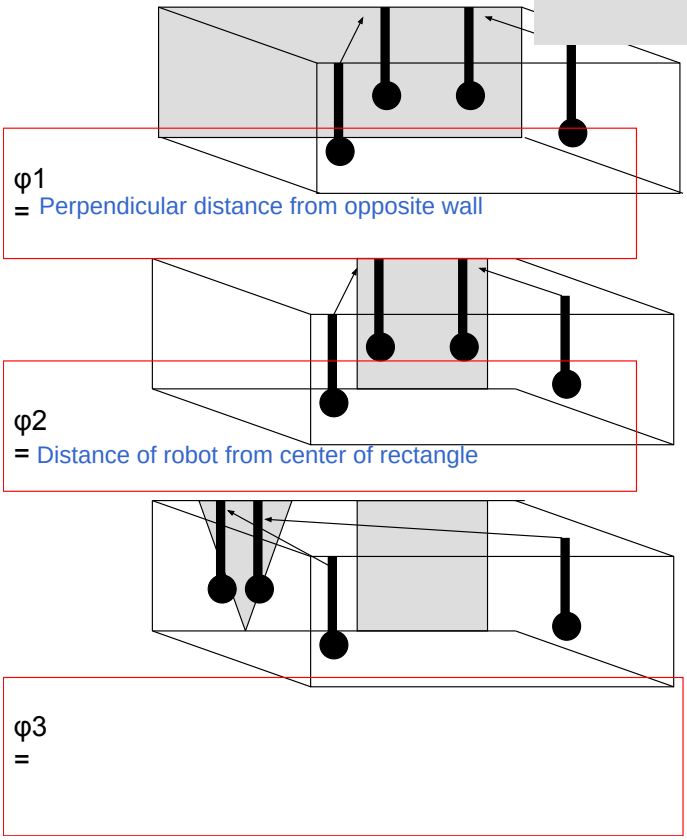
Gantry Robot (University of Sussex, Inman Harvey et al., 1994).

Seven ways to sense:

4: Bumped from the front right, back right, back left, or front left

3: Pixel matrix reduced to three receptive fields; fields averaged to 0-15 brightness.
Position and radius of each receptive field is evolved.

The English Approach



φ1
= Perpendicular distance from opposite wall

φ2
= Distance of robot from center of rectangle

φ3
=

Incremental evolution:

←1. approach back wall
from 4 different initial positions

… evolve …

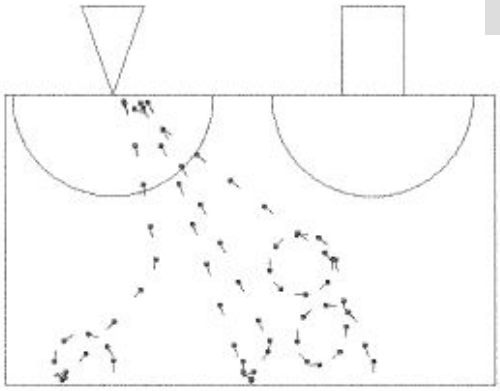
… success!

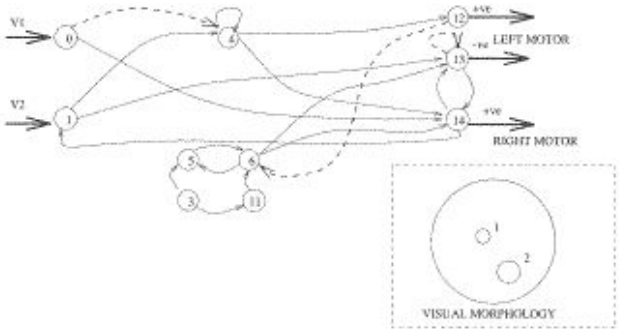
←2. approach back rectangle
from 4 different initial positions

…evolve …

… success!

←3. ***avoid*** back rectangle,
approach triangle.
from 4 different initial positions





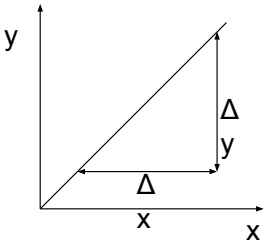
The English Approach

An evolved solution:

Robot does not use bump sensors;

Only uses 2 of the 3 visual fields.

Q: How could it discriminate between the two shapes without a ‘recognize triangle’ or ‘recognize rectangle’ algorithm?



y = mx,
m = Δy/Δx

What happens when Δx and Δy approach 0?

$x_{t+1} = x_t + 1$ ← Difference equation
$x_{t+0.5} = x_t + 0.5$
…
$x_{t+0} = x_t + 0$?

$dx/dt = x' = 1$ ← Differential equation

“The rate of change in x (dx) is equal to the rate of change of t (dt).”

Continuous-Time Recurrent Neural Networks

The derivative

Continuous-Time Recurrent Neural Networks

Value of neuron 1 at time step t+1

$x^{(1)}_{t+1} = \ldots x^{(1)}_t \ldots$

$x_{(1)}^{t+1} = \sigma(w_{1,1}x^{(1)}_t + w_{2,1}x^{(2)}_t + \ldots)$

Neuron values change over time.

Updating neuron values using a **difference equation**

$T_i y'_i = -y_i + \sum^{N}_{j=1} w_{ji} \sigma(g_j (y_j+\theta_j)) + I_i$

Updating neuron values using a **differential equation**

y_i = value of neuron i

T_i = time constant

w_{ji} = synaptic weight from neuron j to i

$\sigma(x)$ = activation function = tanh(x)

g_i = gain of neuron j ; It is used by evolution to kill neurons

θ_j = bias of a neuron j

I_i = input of a sensor to sensor neuron i
(if neuron i is hidden or motor, I_i = 0)

Why $-y_i$? Value approaches 0 in a slightly curved manner. This resembles the biologically neurons.

What does a neuron with…

…low/high time constant do?
high -> slow decay; low -> fast decay, might oscilate

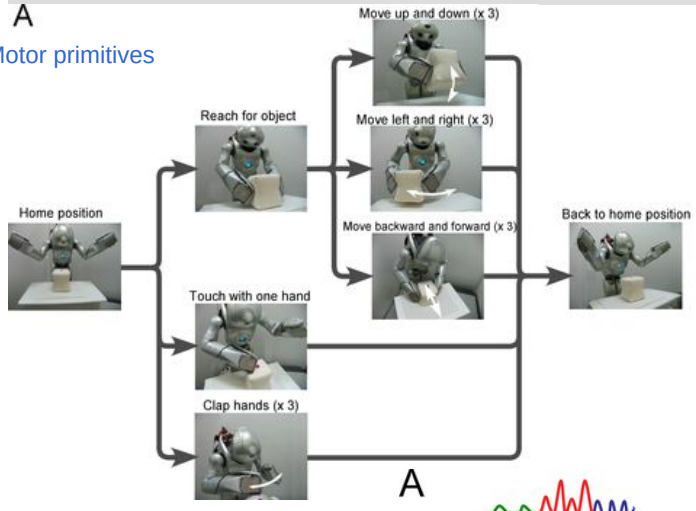
…low/high gain do?
Regulates the influence of neuron j in neuron i

…low/high bias do?
Modifies the default value of the neuron. Displace the oscilations.

Continuous-Time Recurrent Neural Networks

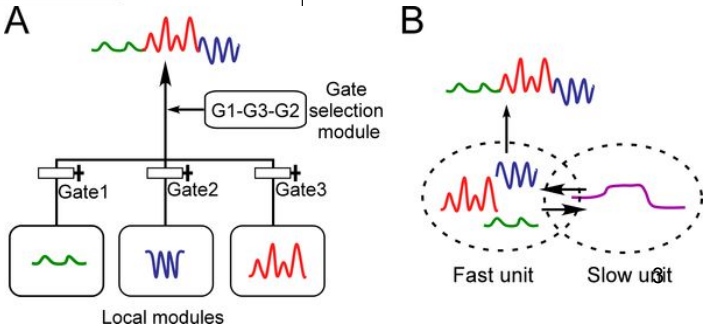
A

Motor primitives

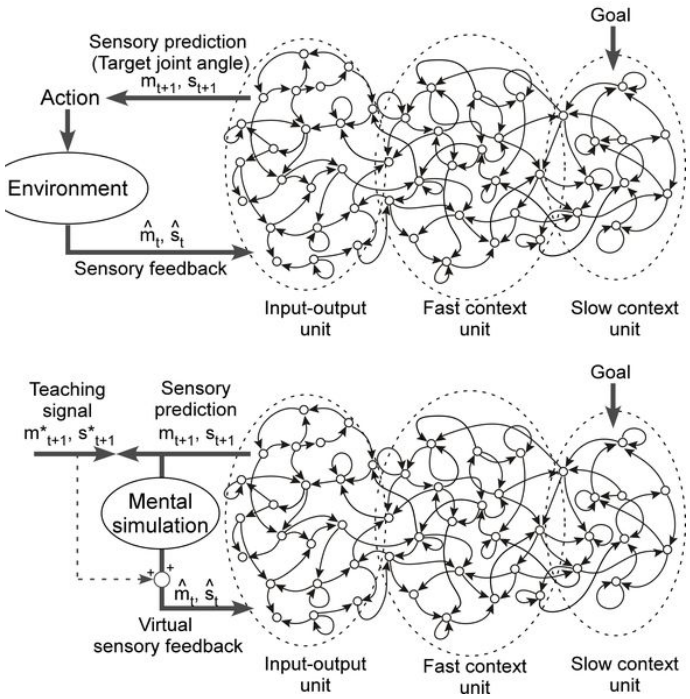


$$\tau_i y'_i = -y_i + \sum_{j=1}^{N} w_{ji}\, \sigma(g_j(y_j+\theta_j)) + I_i$$

Yamashita, Tani (2008)
Emergence of functional hierarchy in a multiple timescale neural network model:
a humanoid robot experiment



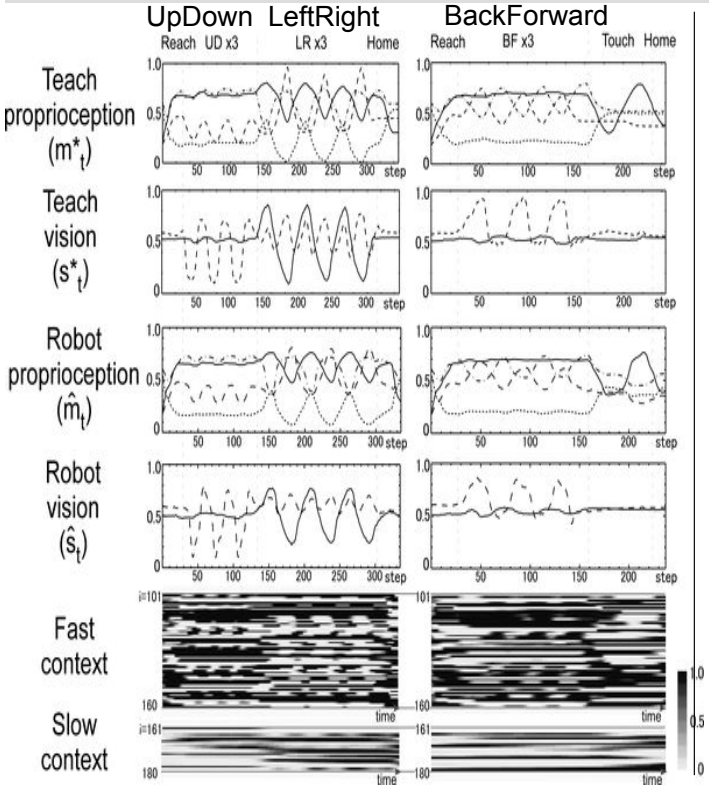
It uses a subsumption architecture for the selection gate



Continuous-Time Recurrent Neural Networks

$$\tau_i \, y'_i = -y_i + \sum_{j=1}^{N} w_{ji} \, \sigma(g_j(y_j+\theta_j)) + I_i$$

Yamashita, Tani (2008) Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Computational Biology*, 4:e1000220



Continuous-Time Recurrent Neural Networks

$$\tau_i \, y'_i = -y_i + \sum_{j=1}^{N} w_{ji} \, \sigma(g_j(y_j+\theta_j)) + I_i$$

Yamashita, Tani (2008) Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Computational Biology*, 4:e1000220





What do these objects have in common?

What is common about them?

What things should a visual pattern recognizer look for?

Minimal Cognition

Sect. 3: Perceiving Affordances

(from wikipedia):

An affordance is a quality of an object, or an environment, that allows an individual to perform an action.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

L09




Minimal Cognition

Sect. 3: Perceiving Affordances

(from wikipedia):

An affordance is a quality of an object, or an environment, that allows an individual to perform an action.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

Minimal Cognition

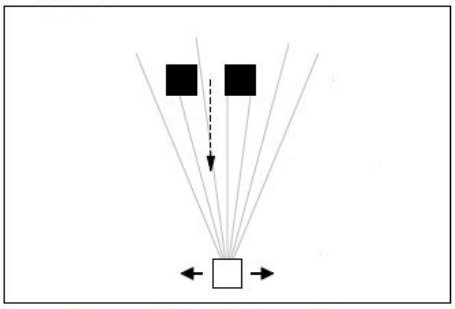


Figure 1: Experimental setup for the passability experiments. The agent moves horizontally while a wall with an adjustable aperture falls from above. The rays of the agent's proximity sensors are shown in gray.

The performance measure to be maximized was:

$$\sum_{i=1}^{NumTrials} p_i \Big/ NumTrials$$

where $p_i = \begin{cases} 2|d_i| & \text{if agent collides with wall} \\ 100 & \text{otherwise} \end{cases}$

for an opening too narrow for the agent to pass through and

$$p_i = \begin{cases} \max(0, 80 - 4|d_i|) & \text{if agent collides with wall} \\ 100 & \text{if between blocks} \\ 0 & \text{if beyond blocks} \end{cases}$$

for an aperture wide enough for the agent to pass through, and d_i is the final horizontal separation between the center of the agent and the center of the aperture at the end of the i^{th} trial. This fitness measure assigns near-zero fitness to incor-

The robot sense affordance in the sense whether the object is passable or not.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

L09

Minimal Cognition

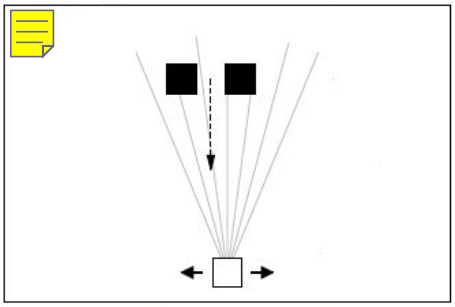


Figure 1: Experimental setup for the passability experiments. The agent moves horizontally while a wall with an adjustable aperture falls from above. The rays of the agent's proximity sensors are shown in gray.

The performance measure to be maximized was:

$$\sum_{i=1}^{NumTrials} p_i \Big/ NumTrials$$

where $p_i = \begin{cases} 2|d_i| & \text{if agent collides with wall} \\ 100 & \text{otherwise} \end{cases}$

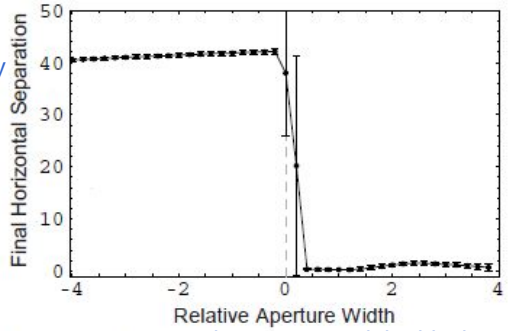
for an opening too narrow for the agent to pass through and

$$p_i = \begin{cases} \max(0, 80 - 4|d_i|) & \text{if agent collides with wall} \\ 100 & \text{if between blocks} \\ 0 & \text{if beyond blocks} \end{cases}$$

for an aperture wide enough for the agent to pass through, and d_i is the final horizontal separation between the center of the agent and the center of the aperture at the end of the i^{th} trial. This fitness measure assigns near-zero fitness to incor-

Ran away

d_i



how separated the blocks are

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

Minimal Cognition

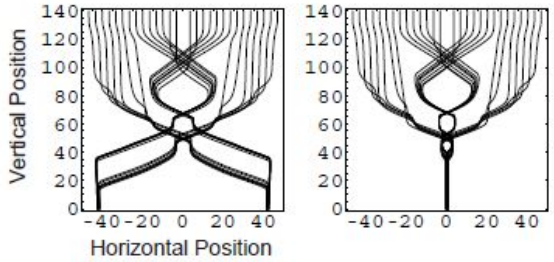


Figure 4: Behavior of the best passability agent. The wall's horizontal and vertical position over time relative to the agent is plotted for an aperture 1 unit smaller than the agent (left) and 1 unit larger than the agent (right). Trials begin at top and time increases from top to bottom.

The performance measure to be maximized was:

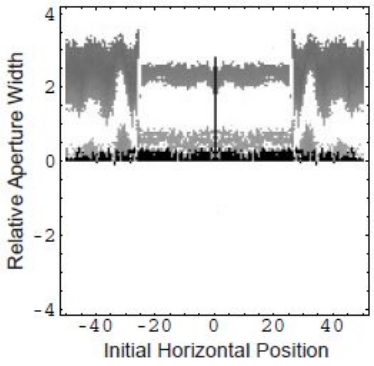
$$\sum_{i=1}^{NumTrials} p_i \Big/ NumTrials$$

where $p_i = \begin{cases} 2|d_i| & \text{if agent collides with wall} \\ 100 & \text{otherwise} \end{cases}$

for an opening too narrow for the agent to pass through and

$$p_i = \begin{cases} \max(0, 80 - 4|d_i|) & \text{if agent collides with wall} \\ 100 & \text{if between blocks} \\ 0 & \text{if beyond blocks} \end{cases}$$

for an aperture wide enough for the agent to pass through, and d_i is the final horizontal separation between the center of the agent and the center of the aperture at the end of the i^{th} trial. This fitness measure assigns near-zero fitness to incor-



White = good
Gray = mediocre
Black = poor

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

Minimal Cognition

Sect. 4: Self/NonSelf Discrimination

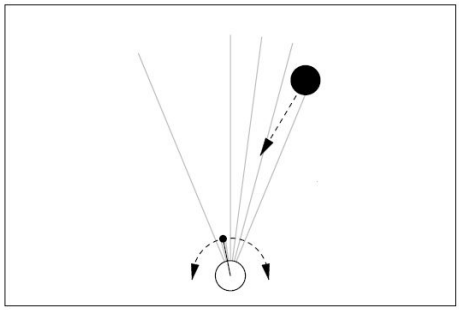
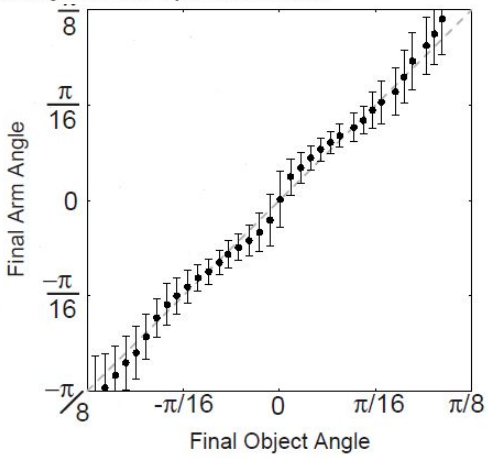


Figure 5: Experimental setup for self/nonself discrimination experiments. The agent is stationary, but can swing an arm with an opaque hand along an arc while objects fall from above.



The performance measure to be maximized was:

$$\sum_{i=1}^{NumTrials} p_i \Big/ NumTrials$$

p_i = 1 - \frac{min(\frac{\pi}{4}|\theta_i|)}{\frac{\pi}{4}}

where

and θ_i is the angular error at the end of the i^{th} trial.

Theta is the penalty factor

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

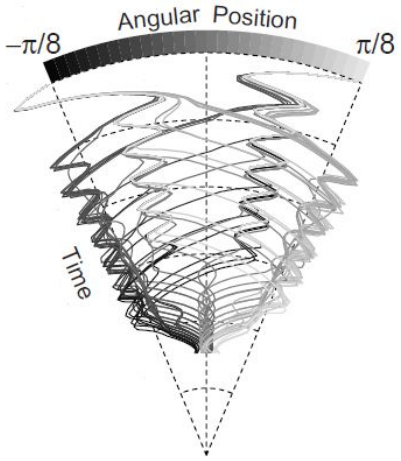


Figure 7: Arm angle trajectories over time of the best self/nonself discrimination agent catching objects at the midline from initial hand positions at either the left or right edge of the visual field. The trajectories are shaded according to the initial angular position of the object as indicated at the top of the plot.

Minimal Cognition

Sect. 4: Self/NonSelf Discrimination

The performance measure to be maximized was:

$$\sum_{i=1}^{NumTrials} p_i \Big/ NumTrials$$

where

and θ_i is the angular error at the end of the i^{th} trial.

It could be possible to add a penalty term in terms of the amount of energy being used to achieved the task

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

Force a robot to have memory

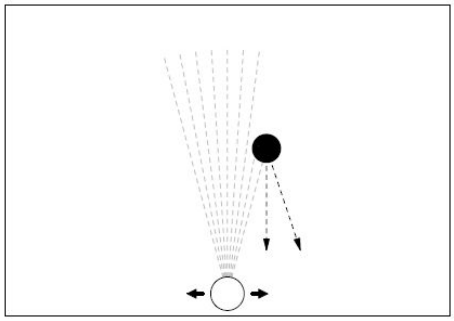
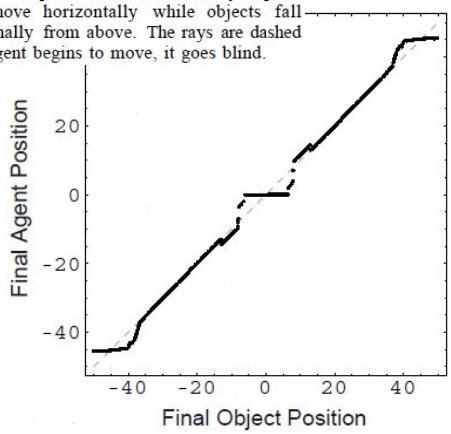


Figure 8: Experimental setup for short-term memory experiments. The agent can move horizontally while objects fall either vertically or diagonally from above. The rays are dashed because, as soon as the agent begins to move, it goes blind.



Minimal Cognition

Sect. 5: Short-term memory

When initially still,robot can sense.

Once it moves, sensors turned off

$$200 - \sum_{i=1}^{NumTrials} |d_i| \Big/ NumTrials$$

d_i = horizontal distance between object's final position and agent's final position.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

Robot moves at the same constant velocity

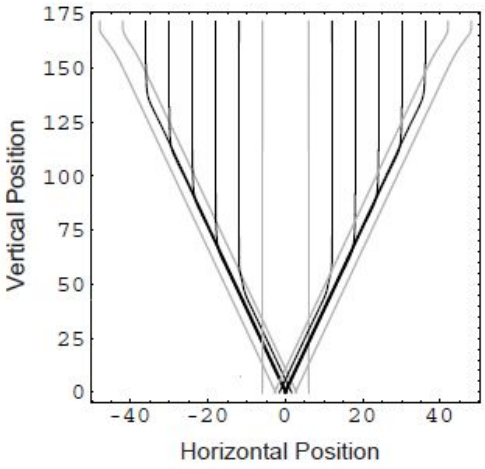


Figure 10: Behavior of the best short-term memory agent for vertically falling objects. Trajectories of motion relative to the agent of objects falling vertically from several different initial horizontal offsets are shown. Gray trajectories are those for which the agent's strategy begins to break down.

Minimal Cognition

Sect. 5: Short-term memory

When initially still,robot can sense.

Once it moves, sensors turned off

$$200 - \sum_{i=1}^{NumTrials} |d_i| \Big/ NumTrials$$

d_i = horizontal distance between object's final position and agent's final position.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

L09

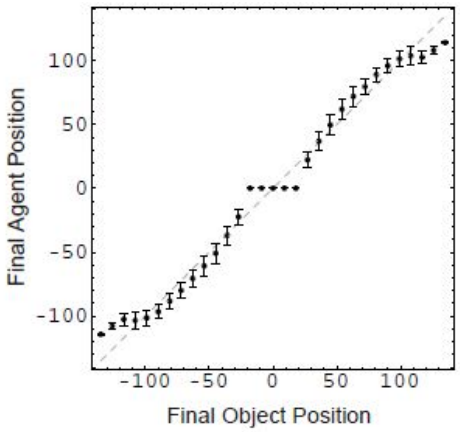


Figure 11: Mean accuracy of the best short-term memory agent for diagonally falling objects. The final horizontal position of the agent is plotted against the final horizontal position of the object (mean ± s.d., $N = 200$ trials). Note that the average performance closely approximates the ideal (dashed gray line).

Minimal Cognition

Sect. 5: Short-term memory

When initially still,robot can sense.

Once it moves, sensors turned off

$$200 - \sum_{i=1}^{NumTrials} |d_i| \Big/ NumTrials$$

d_i = horizontal distance between object's final position and agent's final position.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

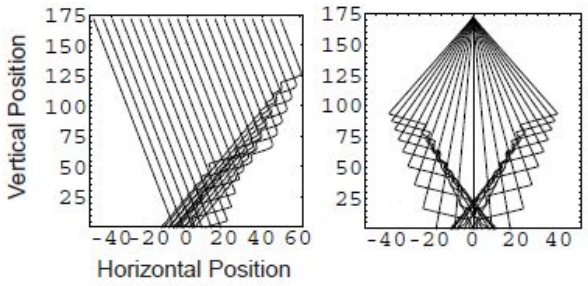


Figure 12: Behavior of the best short-term memory agent for diagonally falling objects. (Left) Trajectories of motion relative to the agent of objects falling diagonally with a horizontal velocity of 0.5 from several different initial horizontal positions are shown. (Right) Trajectories of motion relative to the agent of objects falling diagonally from the midline with several different horizontal velocities are shown.

Minimal Cognition

Sect. 5: Short-term memory

When initially still,robot can sense.

Once it moves, sensors turned off

$$200 - \sum_{i=1}^{NumTrials} |d_i| \Big/ NumTrials$$

d_i = horizontal distance between object's final position and agent's final position.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

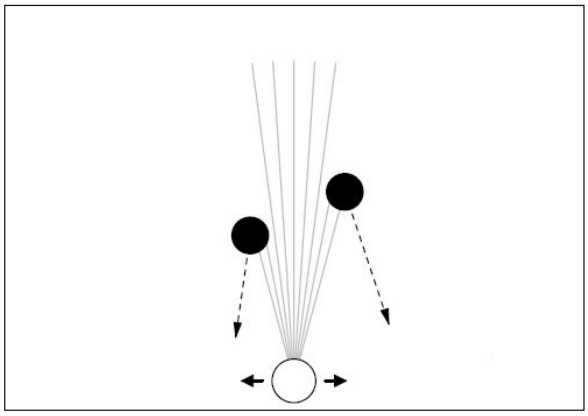


Figure 13: Experimental setup for selective attention experiments. The agent can move horizontally while 2 objects fall from above. As described in the text, the initial positions and velocities of the two objects are constrained so that the agent has a reasonable chance of catching both.

PO = Passing objects problem:
obj further away lands first

OP = object performance problem:
chasing one object causes the other to pass out of view

Minimal Cognition

Sect. 6: Selective Attention

$$200 - \sum_{i=1}^{NumTrials} p_i \Big/ NumTrials$$

$p_i = |d_{i,1}| + |d_{i,2}|$
$d_{i,j}$ = horizontal distance between object j's final position and agent's final position.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

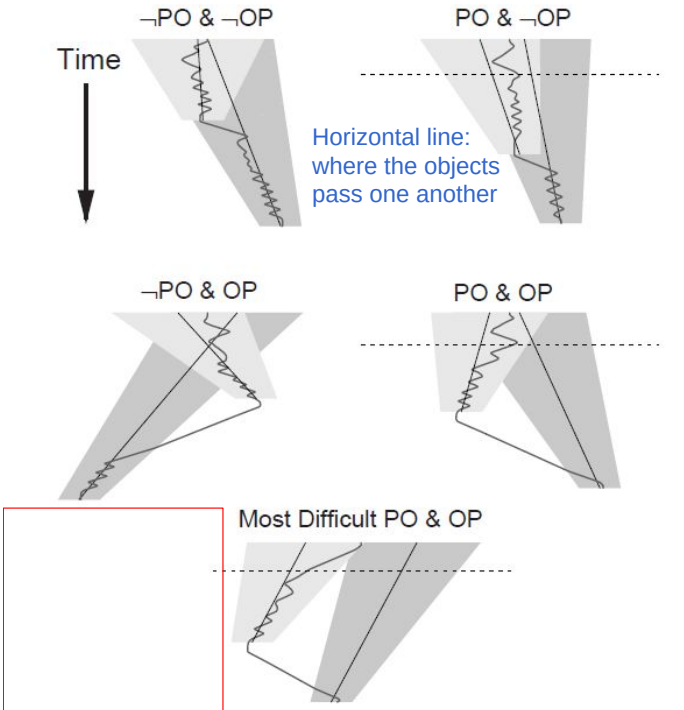


Figure 14: Behavior of the best selective attention agent.

Minimal Cognition

Sect. 6: Selective Attention

$$200 - \sum_{i=1}^{NumTrials} p_i \Big/ NumTrials$$

$p_i = |d_{i,1}| + |d_{i,2}|$
$d_{i,j}$ = horizontal distance between object j's final position and agent's final position.

Slocum, Downey, Beer (2000). Further Experiments in the Evolution of Minimally Cognitive Behavior: From Perceiving Affordances to Selective Attention. In *Animals to Animats 6*, pp. 430—439.

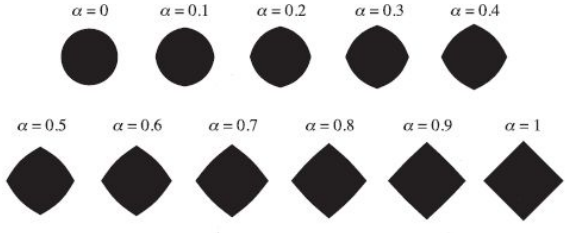
Active Categorical Perception

Task:

Several objects are placed in front of you.

You must separate the round from edged objects while wearing a blindfold.

How would you interact with the objects?



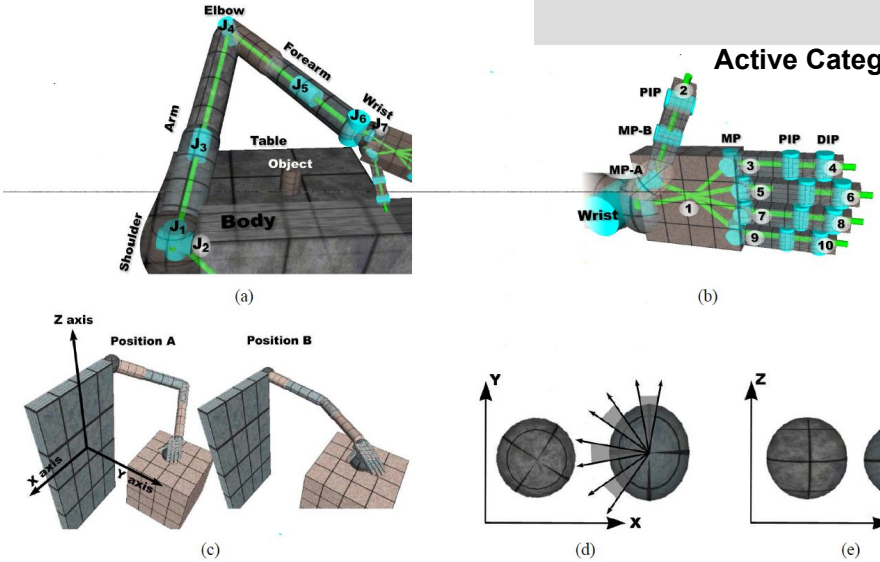
Active categorical perception:

The ability of an animal (or robot) to **interact** with objects of interest to

(1) reduce within-category differences

(2) magnifiy between-category differences

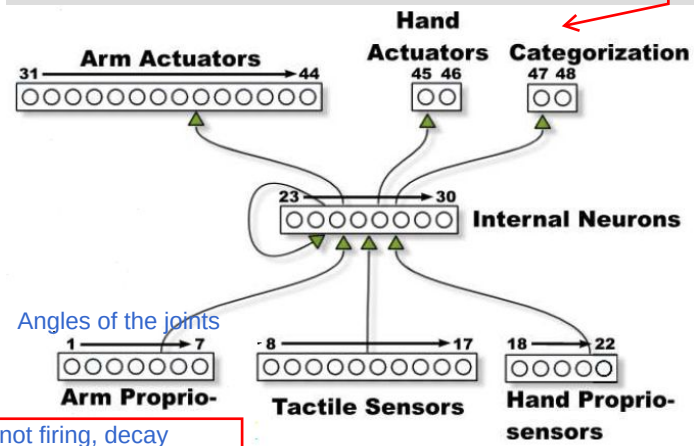
Active Categorical Perception



Tuci, E. and Massera, G. and Nolfi, S. (2009). Active categorical perception in an evolved anthropomorphic robotic arm. In IEEE Congress on Evolutionary Computation, pp. 31-38.

The robot discovers the categories in its own, one for each category

Active Categorical Perception



If not firing, decay towards 0

Sensor neurons

ODEs

Motor neurons

$$\tau_i \dot{y}_i = \begin{cases} -y_i + gI_i; & i = 1,..,22 \\ -y_i + \sum_{j=n}^{m} \omega_{ji}\sigma(y_j + \beta_j); & i = 23,..,48; \end{cases}$$

$$n = 1,\ m = 30 \text{ for } i = 23,..,30;$$

$$n = 23,\ m = 30 \text{ for } i = 31,..,48;$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- It has 48 neurons:
- 48 ODEs for the CTRNN

Cognitive architecture:

Why 14 (31→44) motor neurons, but only 7 joints?

Why two different forms for $\tau_i y'_i$?

What does the

n=1, m=30 for i=23,…,30;
n=23,m=30 for i=31,..,48

mean?

Tuci, E. and Massera, G. and Nolfi, S. (2009). Active categorical perception in an evolved anthropomorphic robotic arm. In IEEE Congress on Evolutionary Computation, pp. 31-38.

Active Categorical Perception

- in each trial e, the agent represents the experienced object (i.e., the sphere S or the ellipsoid D) by associating to it a rectangle R_{Se} or R_{De} whose vertices are:

the bottom left vertex:

$$(\min_{0.95T<t<T} \sigma(y_{47}(t) + \beta_{47}), \min_{0.95T<t<T} \sigma(y_{48}(t) + \beta_{48}))$$

the top right vertex:

$$(\max_{0.95T<t<T} \sigma(y_{47}(t) + \beta_{47}), \max_{0.95T<t<T} \sigma(y_{48}(t) + \beta_{48}))$$

- the sphere category, referred to as C_S, corresponds to the minimum bounding box of all R_{Se}; the ellipsoid category, referred to as C_D, corresponds to the minimum bounding box of all R_{De}.

C_S = (C)lassified as a (S)phere
C_D = (C)lassified as an ellipsoi(D)

Mental representation of spheres and ellipsoids:

$\sigma(y_{47}(t) + \beta_{47})$ =

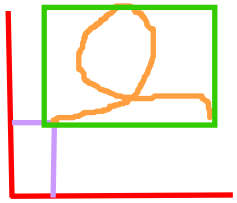
Value of neuron 47 at time step t.

0.95T<t<T =

The last 5% of the time steps during the behavior.

Tuci, E. and Massera, G. and Nolfi, S. (2009). Active categorical perception in an evolved anthropomorphic robotic arm. In IEEE Congress on Evolutionary Computation, pp. 31-38.

48 neuron



47 neuron

Changes of the values over time of the neurons 48 and 47

Green is the bounding box of the values

There are 16 trials, so 16 bounding boxes are generated

L10

Active Categorical Perception

with d_e the euclidean distance between the object and the centre of the palm at the end of the trial e; d_{max} the maximum distance between the palm and the object when located on the table.

C_S = (C)lassified as a (S)phere
C_D = (C)lassified as an ellipsoi(D)

(C_S, C_D are bounding boxes)

The Fitness Function (FF)

Why two fitness terms F_1 and F_2?

What does F1 select for?
What does F2 select for?

Is the term d_e/d_{max} a reward or punishment term?

What does F2=1 indicate about C_S and C_D?

F1 = tell the robot to contact the object. Ranges between 0 and 1

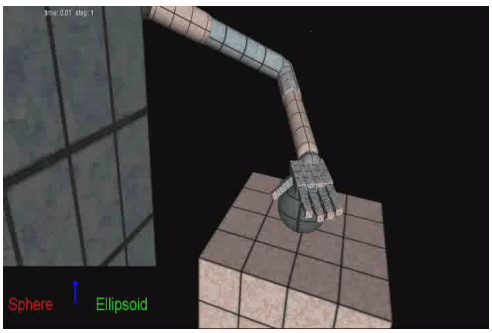
If F1 is less than 1, F2 is zero.

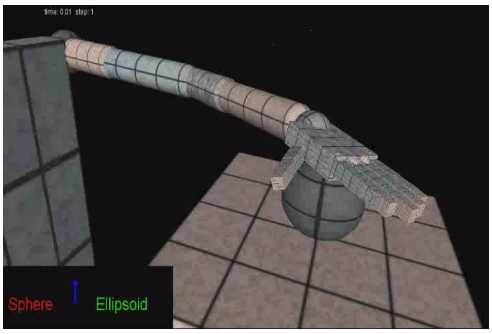
The fitness function measure the amount of overlapping between the bounding boxes that reference to the different classifications for spheres and Ellipsois

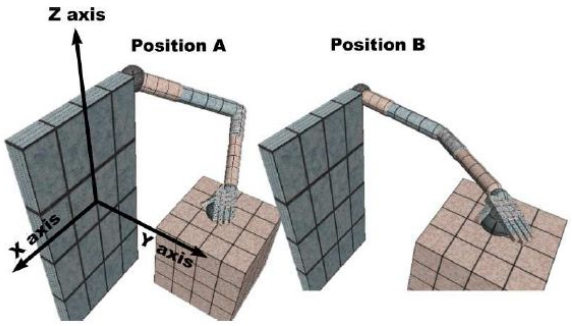
The fitness function just guide the robot to touch the object to be able to categorize them

Tuci, E. and Massera, G. and Nolfi, S. (2009). Active categorical perception in an evolved anthropomorphic robotic arm. In IEEE Congress on Evolutionary Computation, pp. 31-38.

Active Categorical Perception



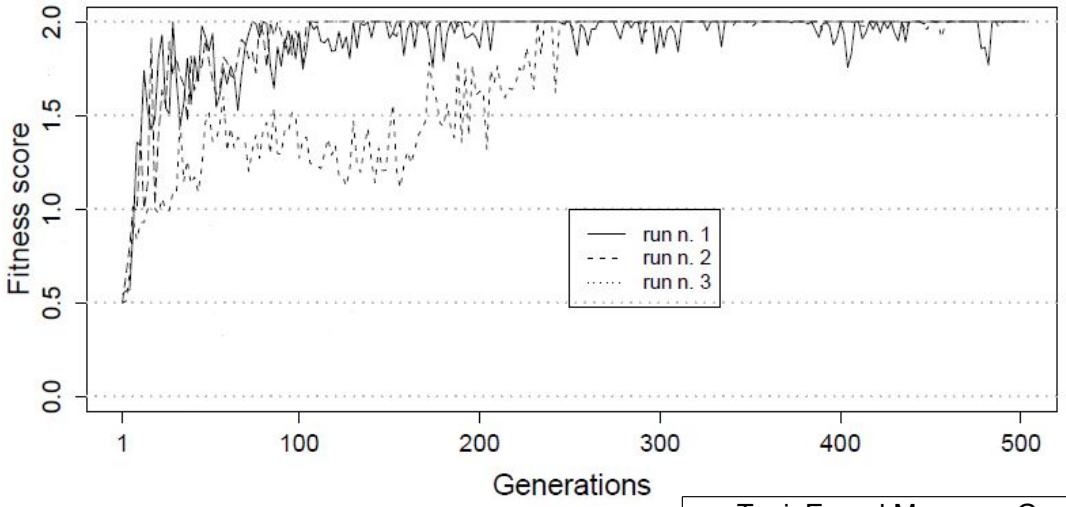




Tuci, E. and Massera, G. and Nolfi, S. (2009). Active categorical perception in an evolved anthropomorphic robotic arm. In IEEE Congress on Evolutionary Computation, pp. 31-38.

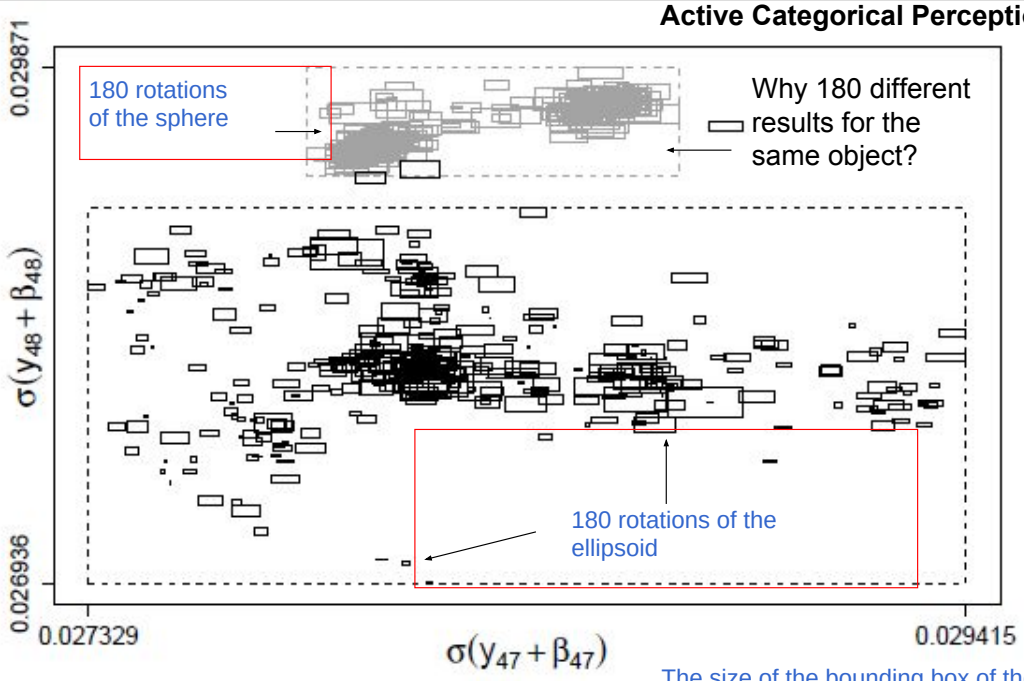
Active Categorical Perception

Performed three separate evolutionary runs:



Tuci, E. and Massera, G. and Nolfi, S. (2009). Active categorical perception in an evolved anthropomorphic robotic arm. In IEEE Congress on Evolutionary Computation, pp. 31-38.

Active Categorical Perception

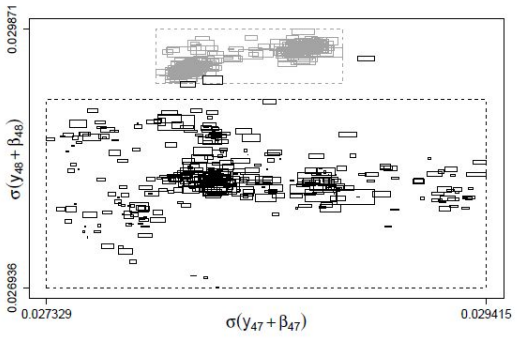


Results from the best evolved controller.

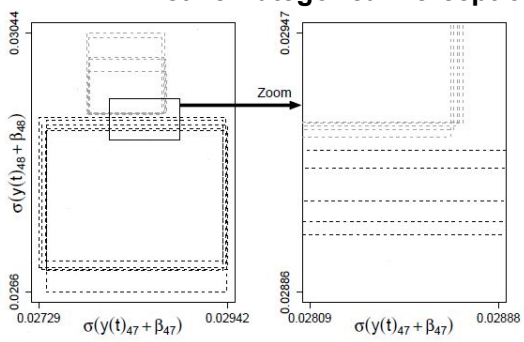
The size of the bounding box of the ellipsoid means of the different kinds of sensory experiences

There was noise added to the sensors

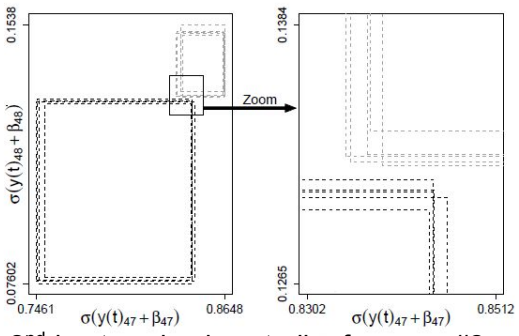
Active Categorical Perception



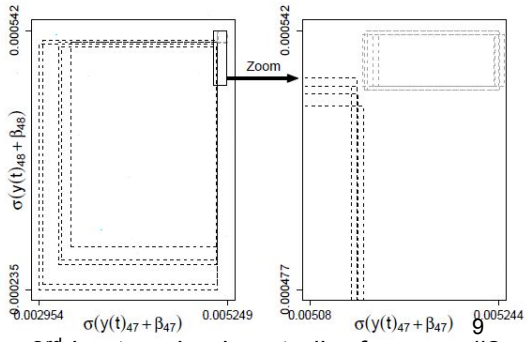
(a)



Best evolved controller from run #2.



2nd-best evolved controller from run #2.



3rd-best evolved controller from run #2.

We are energy efficient

Terrestrial Locomotion

Movement Requires a balance between

(1) displacement,
(2) robustness,
(3) energy and
(4) stability.

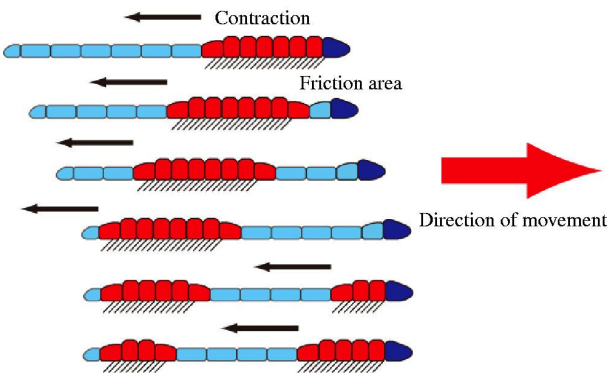
All four are usually in opposition:

Improving one degrades the others.

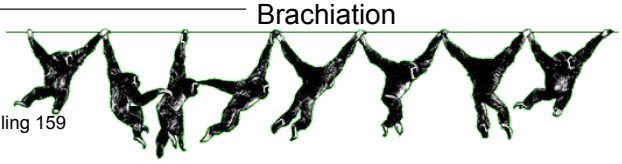
Q: Examples?

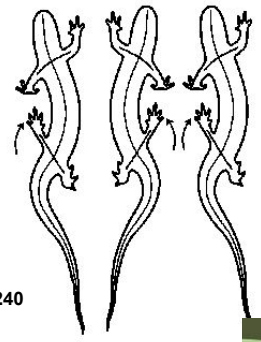
Alexander, R. M. (2002) *Principles of Animal Locomotion.* Princeton University Press.

Locomotion

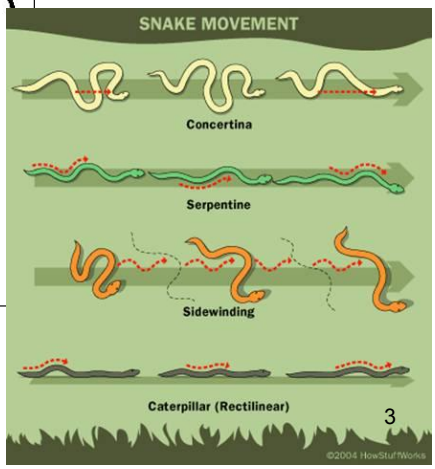


Brachiation





Q: Why evolve away
From snake-like or reptilian
locomotion?



L11
Locomotion

Legged Locomotion

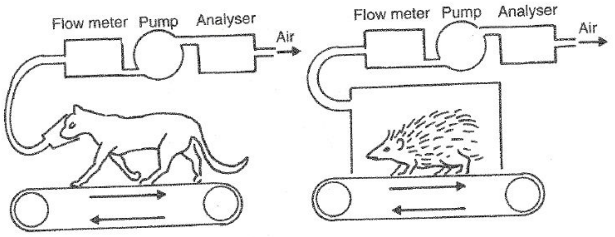


Fig. 4.1 Measuring the oxygen consumption of animals running on treadmills. In (a) the animal wears a mask from which air is sucked for analysis; in (b) the whole animal is enclosed by a hood from which air is drawn.

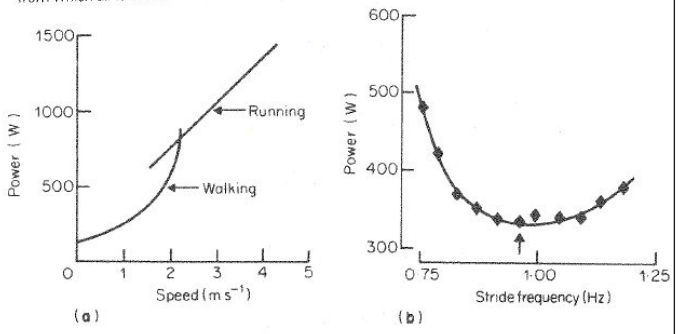


Figure 3-5. The metabolic power consumed by men walking and running, calculated from measurements of oxygen consumption. In (a) power is plotted against speed. The subjects used whatever stride frequency they preferred, at each speed. In (b) power is plotted against stride frequency for walking at 1.5 ms^{-1}. The arrow shows the preferred frequency. Data from (a) Margaria, 1976, (b) Zarrugh and Radcliffe (1978).

Question 1:

For a given gait,
at a given speed,
how much energy is consumed?

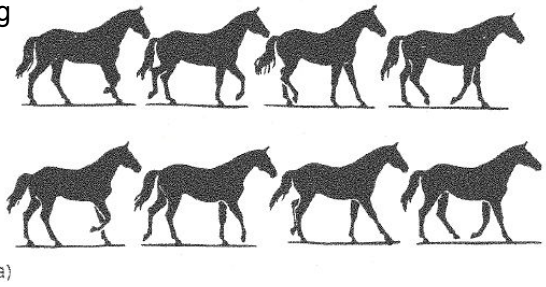
Question 2:

For walking,
How does stride length change the metabolic cost?

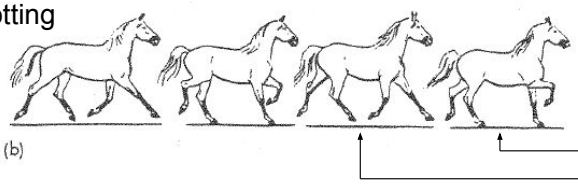
Legged Locomotion

Gaits

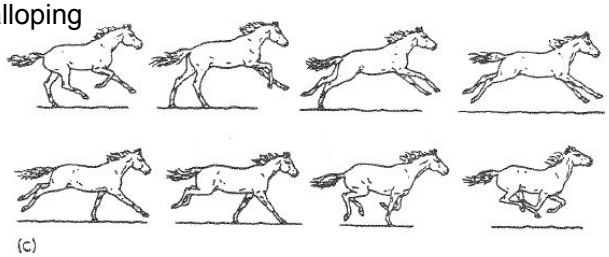
Walking



Trotting



Galloping

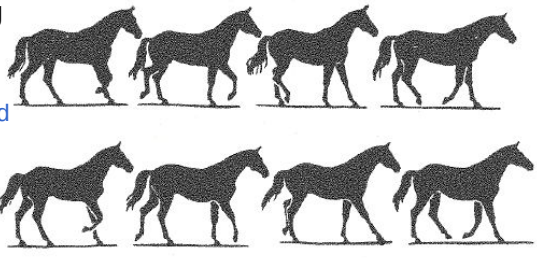


Gait cycle of a human is when the same foot is repeated.

Stance phase
(at least one foot on the ground)
Flight phase
(no feet on the ground)

Legged Locomotion

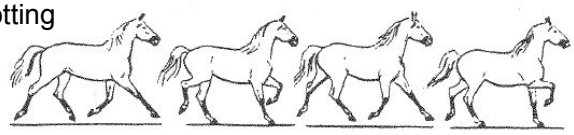
Walking



(a)

Static stability can be determined based on the center of mass and the polygon of support. The center of mass is vertically centered whitin the polygon of support the animal has static stability

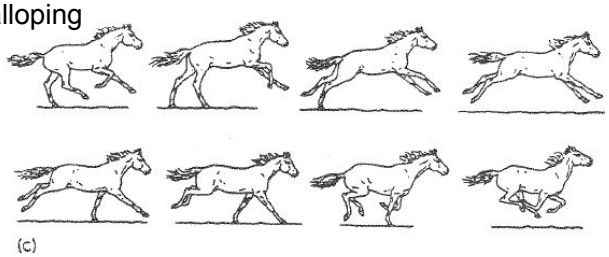
Trotting



(b)

Position sensors and touch sensors in the feet can be used to compute the polygon of support in simmulation.

Galloping



(c)

It is possible to encourange a particular gate by assigning a better reward based on the number of feet in the ground

Static stability (if movement stops, animal will not fall over)

Stay stills once stops moving

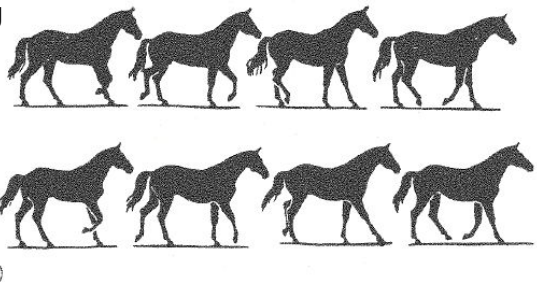
Examples?

Dynamic stability (an observed pattern over time continues, despite external perturbations)

Once you start moving and there is a perturbation you will return to your gate

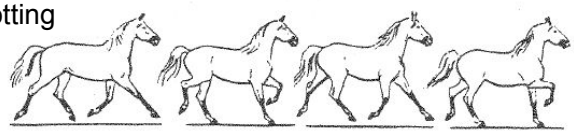
Examples?

Walking



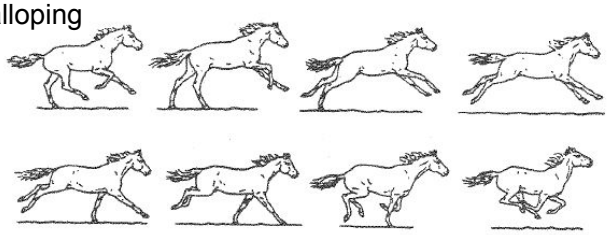
(a)

Trotting



(b)

Galloping



(c)

Static stability
(if movement stops, animal will not fall over)

Examples?

Dynamic stability
(an observed pattern over time continues, despite external perturbations)

Examples?

BigDog, Boston Dynamics

"BigDog", ?

Dynamic stability
(an observed pattern over time continues, despite external perturbations)

Examples?

*Ponies trained to exhibit a specific gait at different speeds.

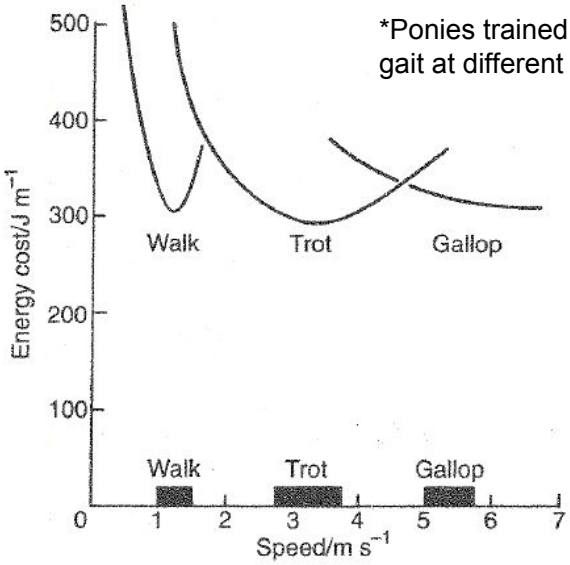


Fig. 4.3 Metabolic energy per unit distance used by 100 kg ponies walking, trotting, and galloping on a treadmill. Bars at the bottom of the graph show the ranges of speeds at which the ponies used each gait when moving freely in their paddock. (Redrawn from D. F. Hoyt and C. R. Taylor (1981) *Nature* **292**, 239–40.)

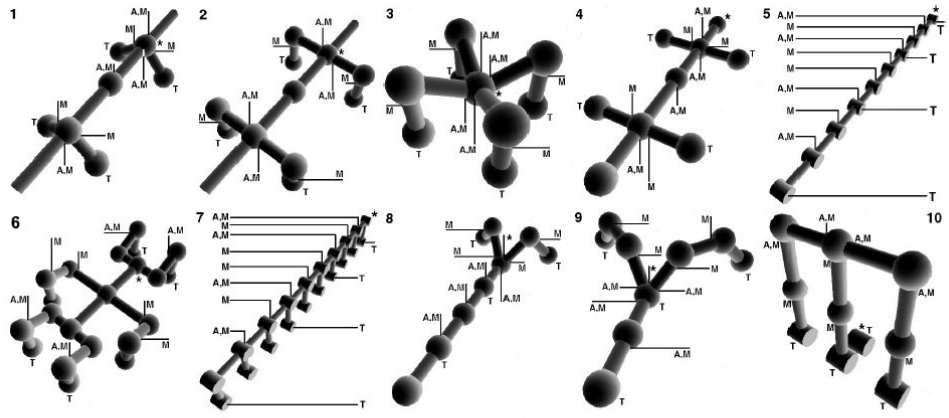


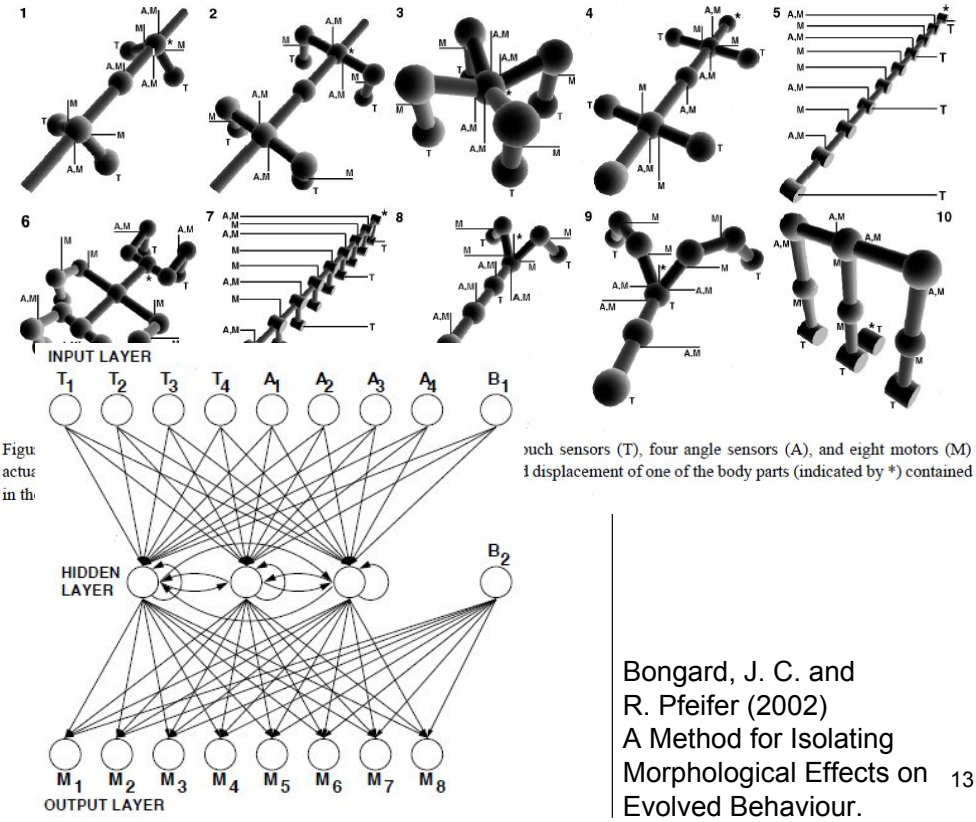
Figure 1: **The agents used for comparison.** Each agent contains four touch sensors (T), four angle sensors (A), and eight motors (M) actuating eight one degree-of-freedom joints. Fitness is based on the forward displacement of one of the body parts (indicated by *) contained in the agent over a fixed period of time.

Q: Given 10 robots with different bodies but the same…

- number of sensors and motors,
- neural network,
- fitness function,
- and optimizer,…

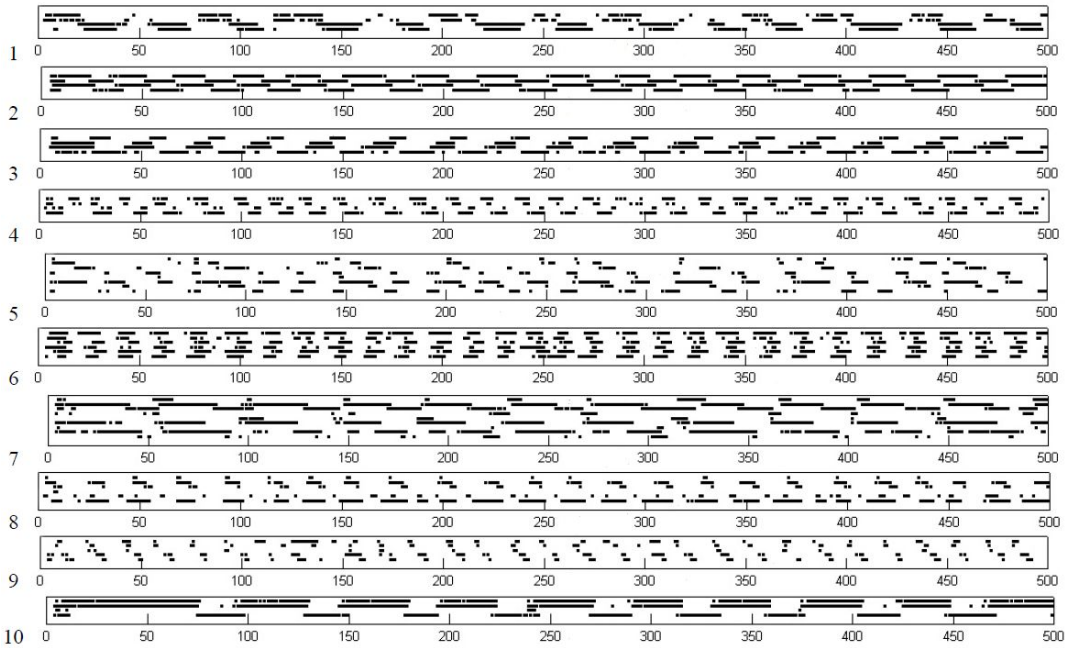
Which one will evolve the fastest gaits?

Bongard, J. C. and R. Pfeifer (2002) A Method for Isolating Morphological Effects on Evolved Behaviour.



Figu… …uch sensors (T), four angle sensors (A), and eight motors (M) actu… …l displacement of one of the body parts (indicated by *) contained in th…

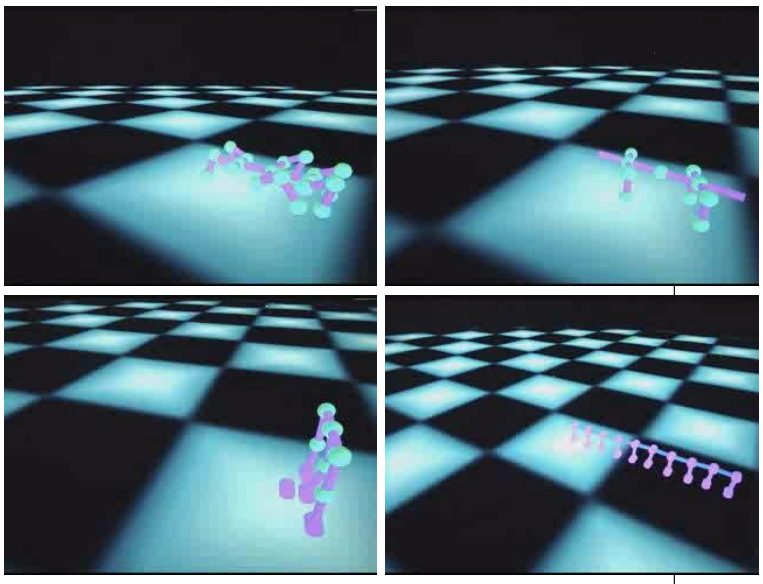
Bongard, J. C. and R. Pfeifer (2002) A Method for Isolating Morphological Effects on Evolved Behaviour. 13



Footprint graph

panel: results from one evolved controller
row: reports the touch patterns for one foot
black pixel: that foot touched the ground at that time
white pixel: that foot was in the air at that time

Bongard, J. C. and R. Pfeifer (2002) A Method for Isolating Morphological Effects on Evolved Behaviour. 14



Bongard, J. C. and R. Pfeifer (2002) A Method for Isolating Morphological Effects on Evolved Behaviour.

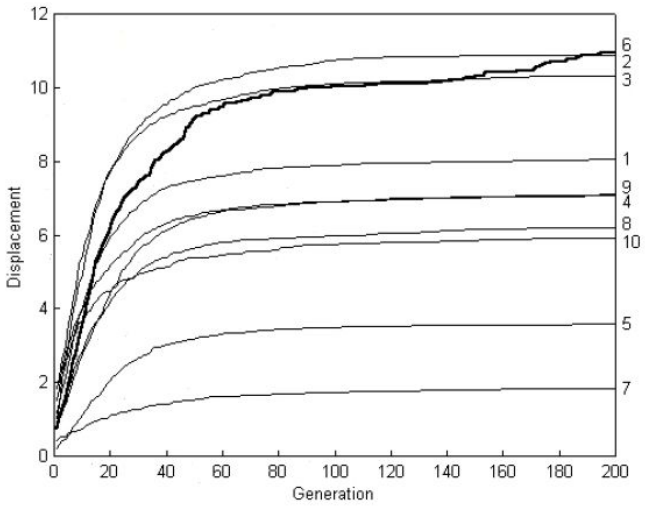


Figure 3: **Average evolutionary performance of the 10 agents.** The curves are averages of the best fitness curves taken over the 30 evolutionary runs for each agent. The numbers to the right indicate to which agent that curve belongs (ie., agents 6 and 2 performed the best). Displacement is in meters.

Q: What is it about the robot's morphology that predicts the ability to evolve gaits for it?

Given your experiences with Pyrosim, what physical aspects of the robot may make a difference?

Bongard, J. C. and R. Pfeifer (2002) A Method for Isolating Morphological Effects on Evolved Behaviour.

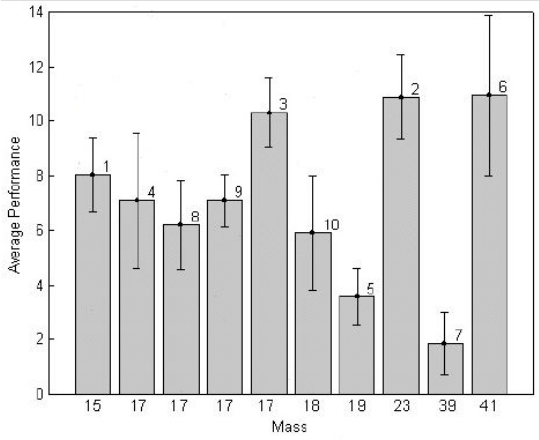


Figure 5: **Mass versus evolutionary performance.** The horizontal axis indicates the total mass of the agent, in kilograms. The vertical axis indicates the average displacement of the targetted body part for each agent type, in meters. The numbers above the bars indicate the agent index as given in Fig. 1. The error bars are two standard deviation units in length.

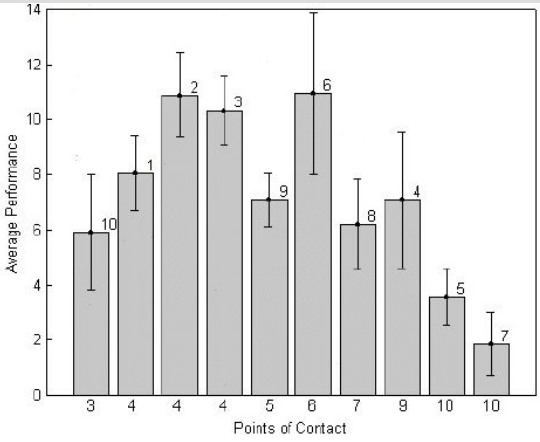


Figure 6: **Points of contact versus evolutionary performance.** The horizontal axis indicates how many body parts of the agent can contact the ground plane. The vertical axis indicates the average displacement of the targetted body part for each agent type, in meters. The numbers above the bars indicate the agent index as given in Fig. 1. The error bars are two standard deviation units in length.

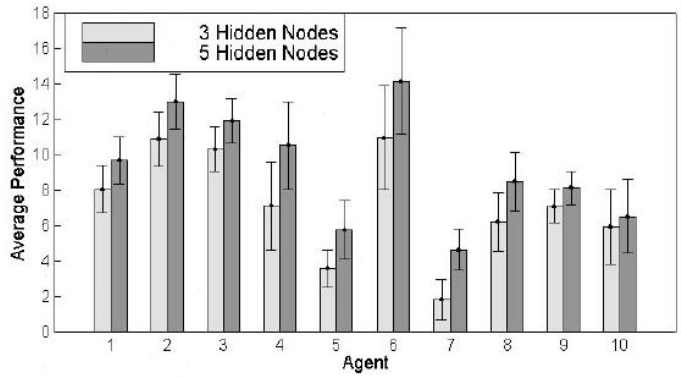
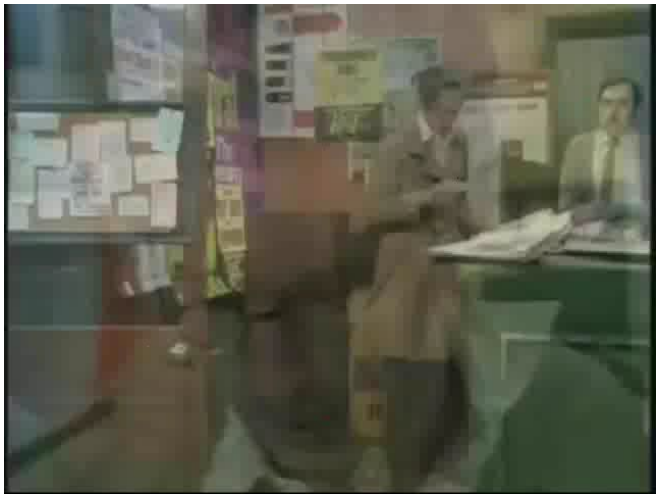


Figure 7: **Change in performance based on addition of hidden neurons.** The light coloured bars indicate the average evolutionary performance for that agent using three hidden neurons. The dark coloured bars indicate average performance for that agent using five hidden neurons. Numbers along the horizontal axis denote the agent's index number, as denoted in Fig. 1. The error bars are two standard deviation units in length.

Q1: Do changes to
The cognitive architecture
(going from 3 to 5 hidden nodes)
affect the evolution
of gaits?

Q2: Is this effect the same across different robots?



Bipedal Locomotion

Question 1:

Why only walking and running?

Question 2:

What (if any) part
does the upper body
play in bipedal locomotion?

Honda's ASIMO robot

Bipedal Locomotion

Question 1:

Why only walking and running?

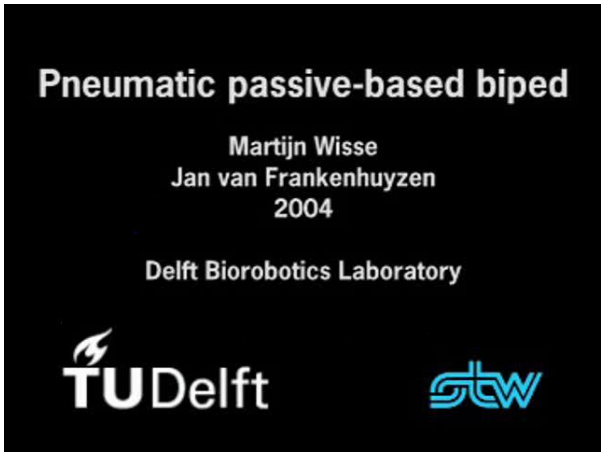
Question 2:

What (if any) part
does the upper body
play in bipedal locomotion?

Bipedal Locomotion

Passive Dynamic Locomotion

Legged locomotion is achieved down a declined plane without any motors (or sensors or neural networks)



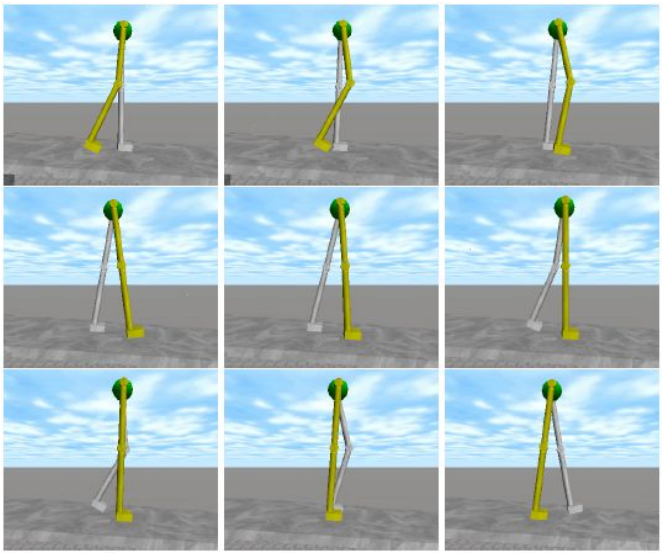
Steve Collins, Martijn Wisse, Andy Ruina, Cornell

Hybrid Dynamic Locomotion

During part of the gait cycle a joint is powered, and otherwise it swings freely

Martijn Wisse, TU Delft (2004)

Bipedal Locomotion



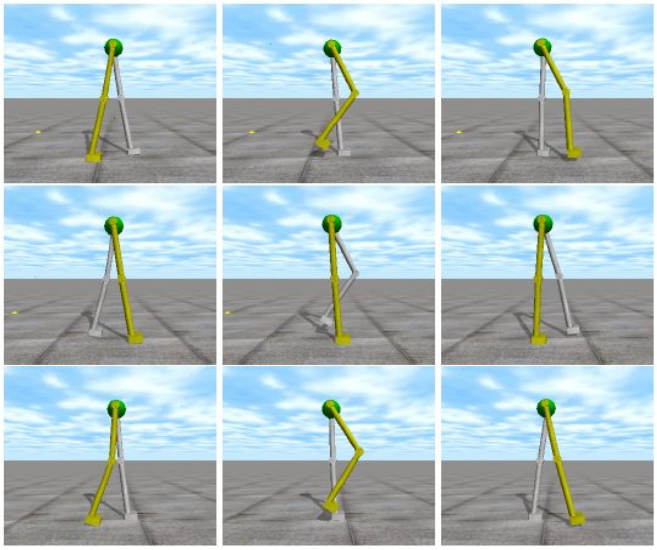
Research question:

Can passive dynamic walking be evolved? How?

Could evolved passive dynamic walking be evolved further into hybrid dynamic walking?

Vaughan, E., Di Paolo, E., Harvey, I. (2004)
The evolution of control and adaptation in a 3D powered passive dynamic walker.
In *Artificial Life IX,* pp. 139-145.

Bipedal Locomotion

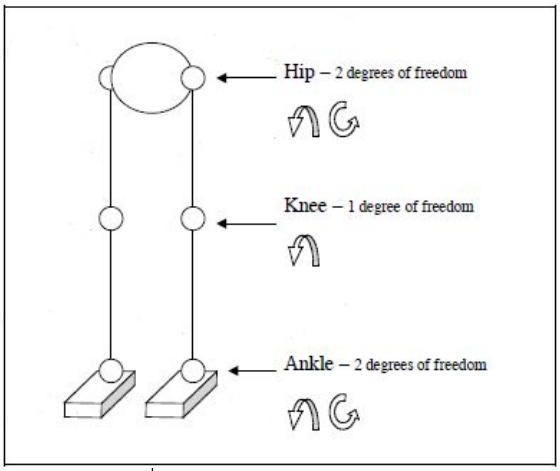


Research question:

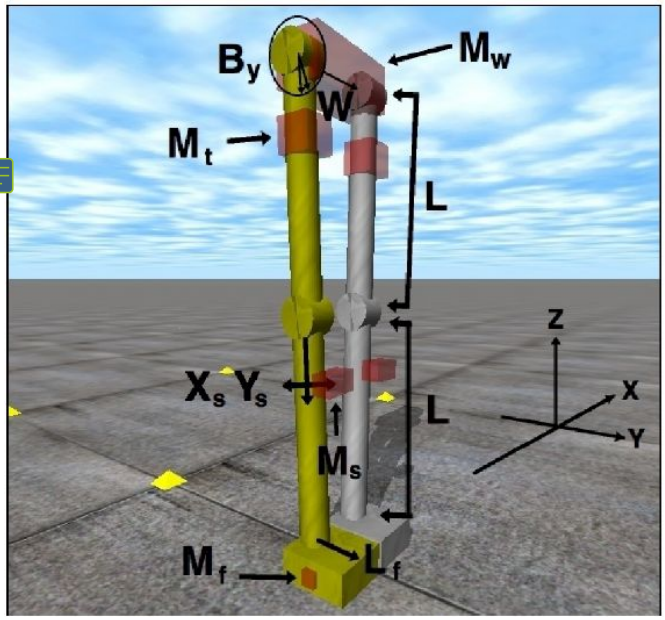
Can passive dynamic walking be evolved? How?

Could evolved passive dynamic walking be evolved further into hybrid dynamic walking?

Vaughan, E., Di Paolo, E., Harvey, I. (2004)
The evolution of control and adaptation in a 3D powered passive dynamic walker.
In *Artificial Life IX,* pp. 139-145.



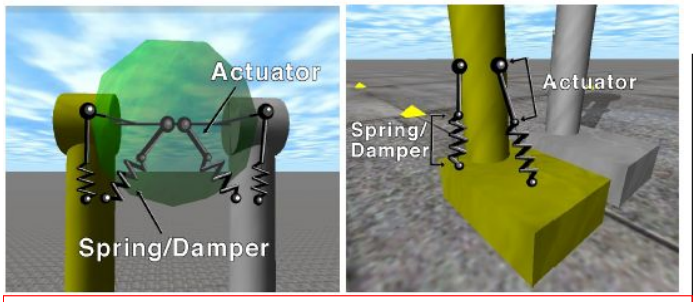
Vaughan, E., Di Paolo, E., Harvey, I. (2004)
The evolution of control and adaptation in a 3D powered passive dynamic walker.
In *Artificial Life IX,* pp. 139-145.



M_w = waist mass
M_t = thigh mass
M_s = shank mass
M_f = foot mass
L = leg segment length
X_t = thigh mass x-offset
Y_t = thigh mass y-offset
X_s = shank mass x-offset
Y_s = shank mass y-offset
L_f = foot length
W = waist radius
B_y = starting hip angle around y-axis

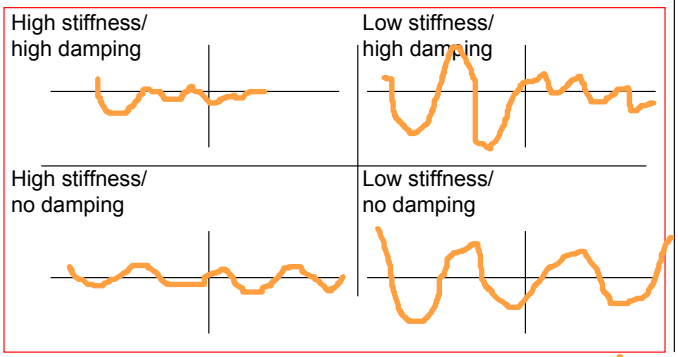
Placing the robot's body under evolutionary control: evolve not just NN parameters, but body parameters as well.

Vaughan, E., Di Paolo, E., Harvey, I. (2004) The evolution of control and adaptation in a 3D powered passive dynamic walker. In *Artificial Life IX,* pp. 139-145.



Spring stiffness: How much a spring resists pulling and pushing

Spring damping: how long a spring takes to stop moving once it reaches its rest length

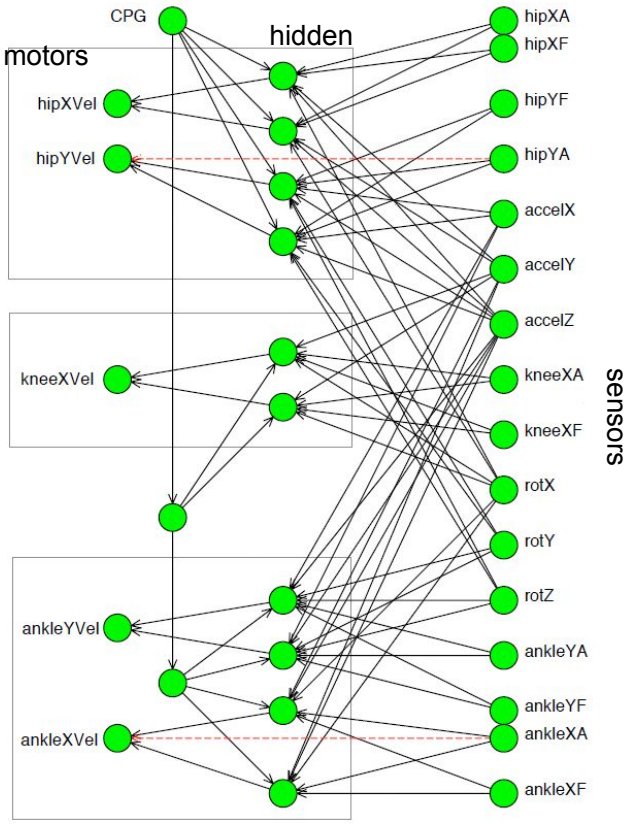


Ax = ankle spring/damper around x-axis
Kx = knee spring/damper around x-axis
Hx = hip spring-/damper around x-axis
Hy = hip spring/damper around y-axis
Ay = ankle spring/damper around y-axis

Each spring/damper has two parameters:
stiffness and damping coefficient

Q: How many body params?

Vaughan, E., Di Paolo, E., Harvey, I. (2004) The evolution of control and adaptation in a 3D powered passive dynamic walker. In *Artificial Life IX,* pp. 139-145.



Continuous Time Neural Network (CTNN)

Five **motors:**
one attached to each spring/damper.

CPG =

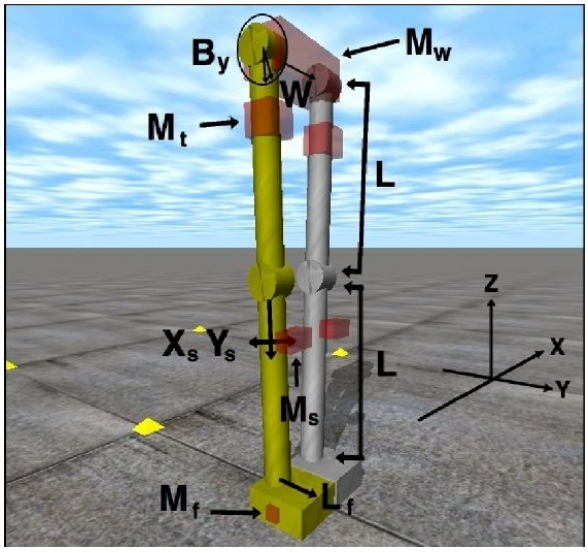
Central pattern generator:
emits a pulse at a regular time interval (introduces timing)

Two **hidden** nodes / per motor

Sensors:
*A = angle sensor
what is the angle of the joint?
*F = force sensor
how far is the spring from its resting length?
rot* = rotation about * axis
accel*=acceleration about * axis

The fitness function allows to control the gait used by the robot

$$f = d\left(\frac{1}{1+t}\right)\left(\frac{1}{1+x}\right)\left(\frac{1}{1+z}\right)\left(\frac{1}{1+r}\right)\left(\frac{1}{1+y}\right)$$



The Fitness Function:

Q: Does d, t, x, z, r, or y reward or punish?

f: fitness

d: distance travelled

t: torque used
(torque = rotational force)

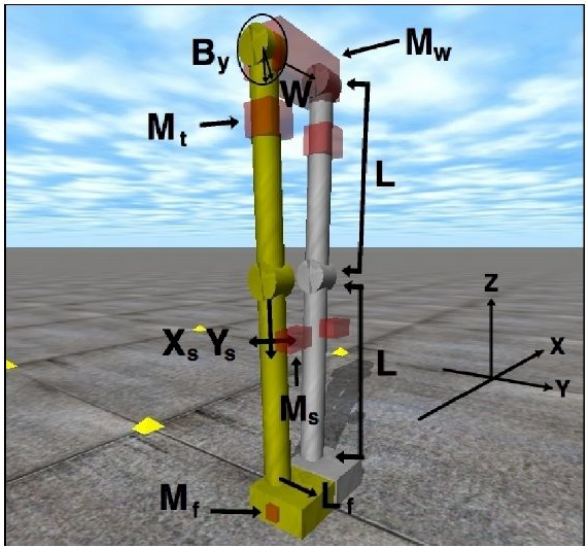
x: hip rotation about x-axis

z: hip rotation about z-axis

r: feet rotation about z-axis

y: hip rotation about y-axis

$$f = d\left(\frac{1}{1+t}\right)\left(\frac{1}{1+x}\right)\left(\frac{1}{1+z}\right)\left(\frac{1}{1+r}\right)\left(\frac{1}{1+y}\right)$$



Shaping:

1. **Evolve passive walking**

During evolution:
Evolve CPG -> motor params
Evolve body params

During evaluation:
Put robot on declined plane.
Supply one CPG pulse.
After pulse, turn off network

2. **Evolve hybrid walking**
3. **Evolve sensor-based walking**

$$f = d\left(\frac{1}{1+t}\right)\left(\frac{1}{1+x}\right)\left(\frac{1}{1+z}\right)\left(\frac{1}{1+r}\right)\left(\frac{1}{1+y}\right)$$

Shaping:

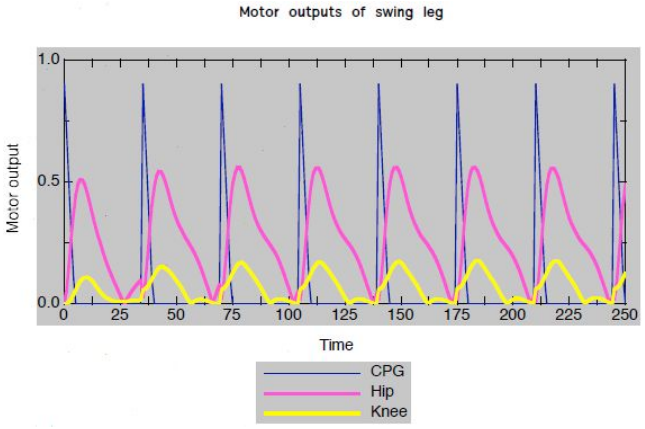
1. **Evolve passive walking**
2. **Evolve hybrid walking**

During evolution:
Evolve CPG→motor params
Evolve body params
Add (1/(1+v)) to fitness function
v = difference between passive and hybrid velocity
Gradually lower declined plane over generations

During evaluation:
Record time to first heel strike
Use this to set CPG frequency

3. **Evolve sensor-based walking**

$$f = d\left(\frac{1}{1+t}\right)\left(\frac{1}{1+x}\right)\left(\frac{1}{1+z}\right)\left(\frac{1}{1+r}\right)\left(\frac{1}{1+y}\right)$$

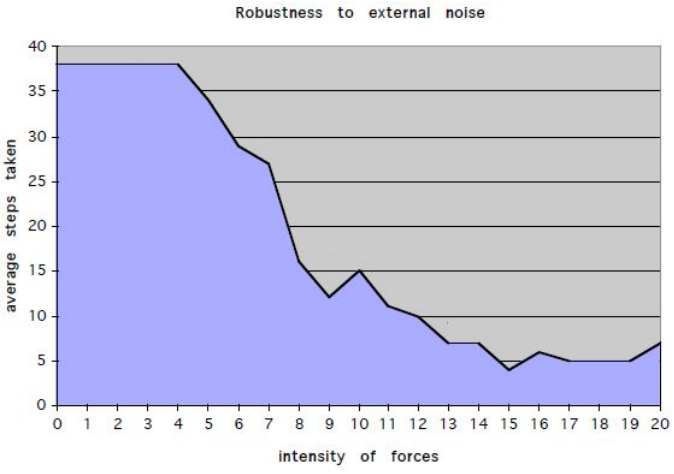


Shaping:

1. **Evolve passive walking**
2. **Evolve hybrid walking**
3. **Evolve sensor-based walking**

During evolution:
reconnect sensors to NN
with small connection weights
Evolve all NN param weights
Evolve body params
Use 1/(1+v)

During evaluation:
Same as before.



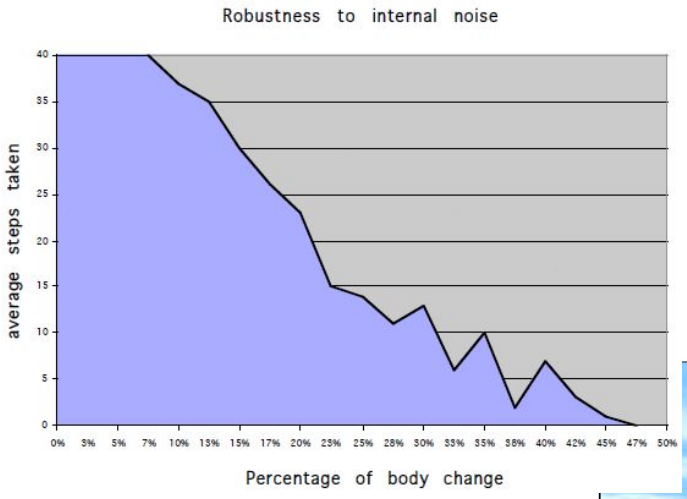
Robustness to **external** perturbations

Evolve for another 100 generations.

Small random force vectors applied to the robot while it moved (i.e. “wind”).

“Developed dynamic mechanisms to adapt to noise.”

“When pushed too far to one side, the machine was observed to adjust its foot placement by stepping inward to regain balance.”

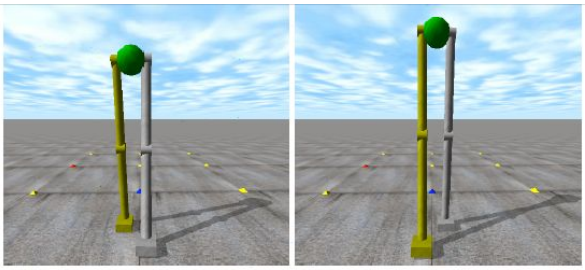


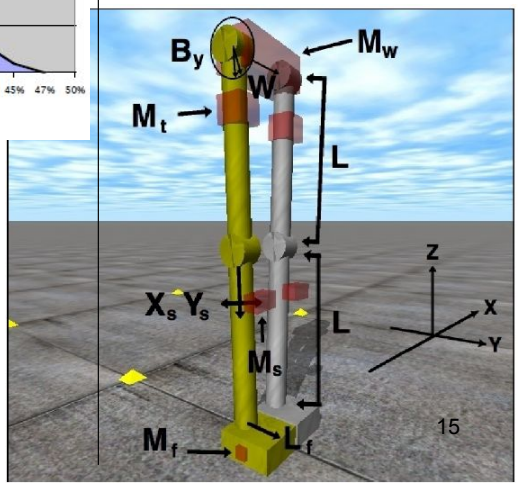
Robustness to **internal** perturbations

Evolve further.

Introduce “mistakes” when building the body.

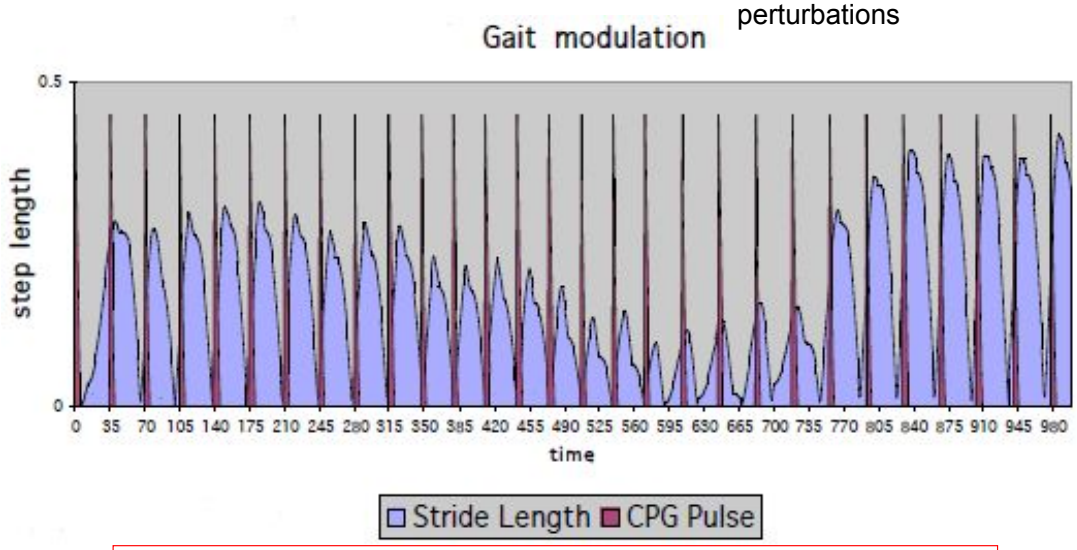
Random changes are made to the body params





15

Robustness to **CPG** perturbations



16

Modularity

Data/function hiding
in object-oriented programming:

The internal complexity of an object is hidden from the objects that call its public methods.

Class Pyrosim_Shape
public:
Create(…); Draw(…); Delete(…)

…this internal complexity will grow as the object becomes more complex, but interactions between objects should grow at a slower rate:

Class Pyrosim_Robot
public:
Create(…); Draw(…); Delete(…)
private:
CreateNN(…); DestroyNN(…);
CreateBody(…); DestroyBody(…);
CreateJoints(…); DestroyJoints(…);
ActuateJoints(…); CollectSensorReadings();

An important
concept in many domains.

Different definitions in different domains, but similar concept:

Tighter coupling within modules;
less coupling across modules.

A module in object-oriented programming: the object.

Modularity

Should be able to look at the NN and tell from the structure (the connectivity) if the network is modular

Why is
modularity important for evolutionary robotics?

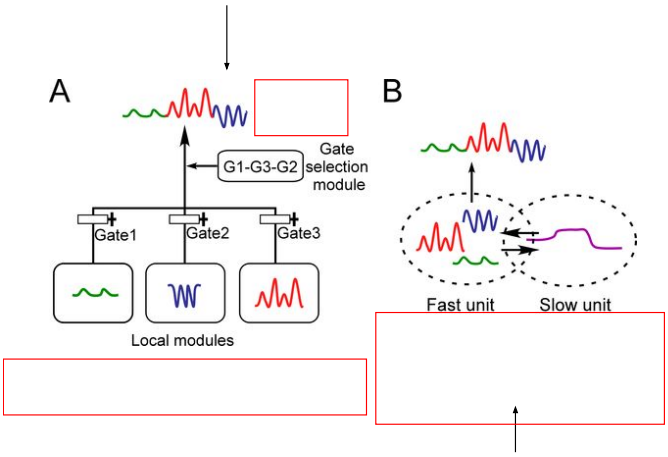
Structural modularity:

Reduces the quadratic increase in synapses as the number of neurons increases.

Functional modularity:

The size of the network does not need to increase with the number of motor primitives (i.e. simple actions the robot must perform)

Structural modularity in (neural) networks:
different parts of the network at different times.



Functional modularity in (neural) networks:
the network exhibits a fixed number of dynamic patterns.

Modularity

Why is
modularity important for
evolutionary robotics?

Structural modularity:

Reduces the quadratic
increase in synapses as the
number of neurons increases.

Functional modularity:

The size of the network
does not need to increase
with the number of
motor primitives (i.e. simple
actions the robot must perform)

Structural modularity

Given a task…

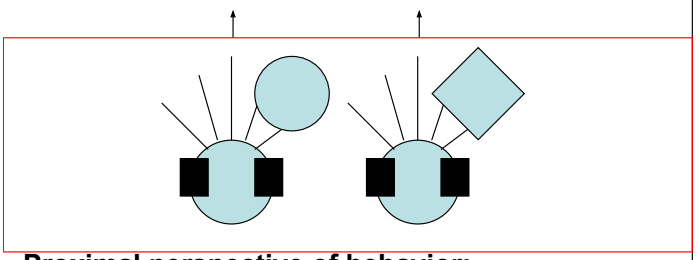
the roboticist
may break the task down into
several subtasks, and
assign a subnetwork to each
task.

the evolutionary algorithm
may create modules
appropriate for the
task.

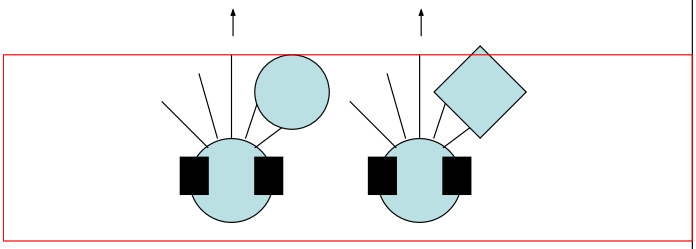
However, roboticist and
robot may have different
ideas about what counts
as a 'behavior'…

Structural modularity

Distal perspective of behavior:
how does behavior look from
the perspective of the distal observer.



Proximal perspective of behavior:
how does behavior look from
the perspective of the robot (i.e., from its sensors).



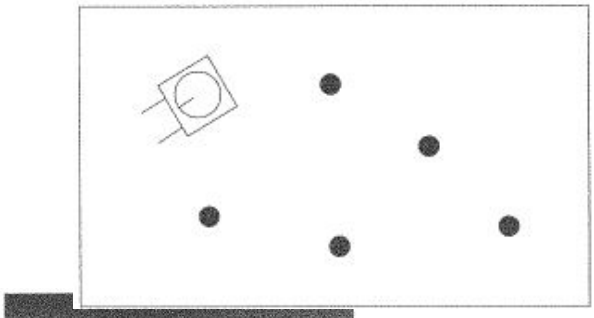
Given a task…

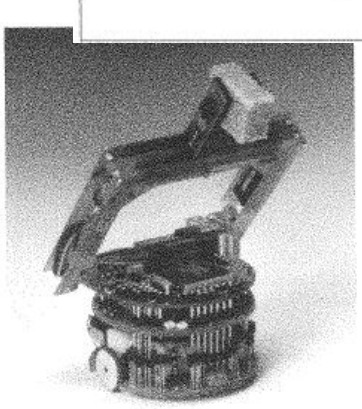
the roboticist
may break the task down into several subtasks, and assign a subnetwork to each task.

the evolutionary algorithm
may create modules appropriate for the task.

However, roboticist and robot may have different ideas about what counts as a 'behavior'…

Structural modularity





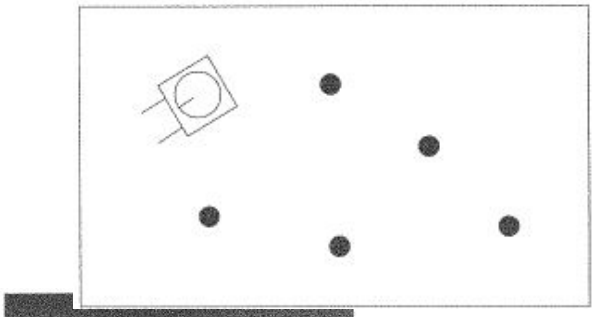
Research question:

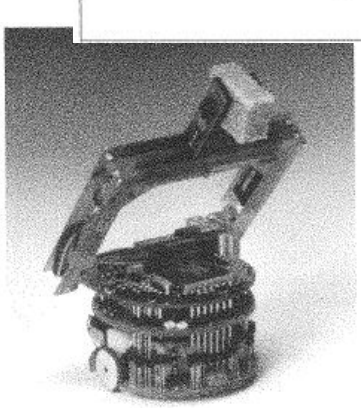
How to enable evolution to discover modules appropriate to the task?

[Today's assigned reading]

Task:
Pick up all objects and drop them over the wall and thus out of the arena.

Structural modularity

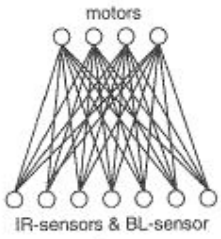




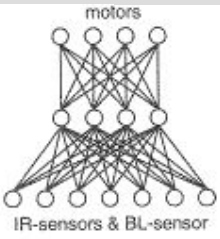
Task:
Pick up all objects and drop them over the wall and thus out of the arena.

Possible decomposition:
a. Explore environment,
avoid walls.
a1: if no sensors firing,
move forward.
a2: if left sensors firing,
turn right.
b. Recognize object, move
to position for pick-up.
c. Pick up object.
d. Move toward wall
(avoid other objects)
e. Recognize wall,
move close to it.
f. Release object.

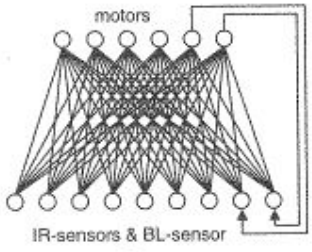
Structural modularity



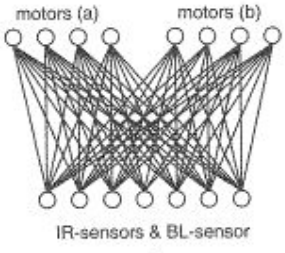
A



B



C



D

Five possible networks:

A: feedforward,
no hidden neurons

B: feedforward,
hidden neurons

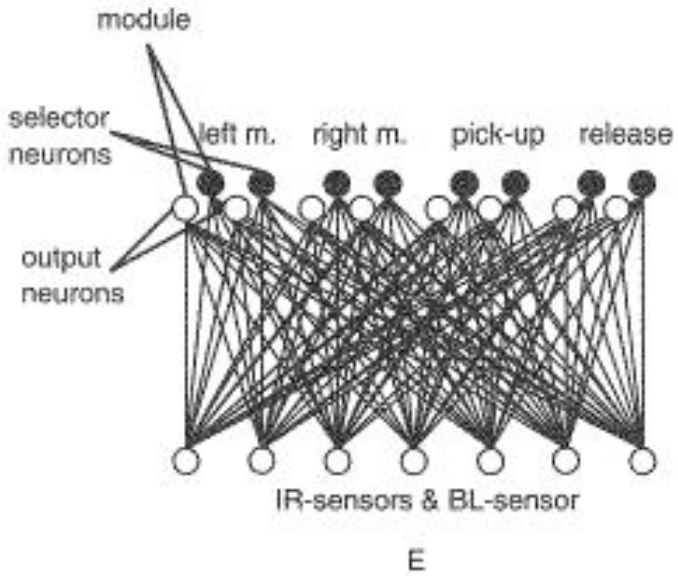
C: Recurrent,
hidden neurons

D: Two pre-designed
modules:

First module used
when gripper empty;

Second module used
when gripper full.

Structural modularity



Five possible networks:

E: Emergent modular network.

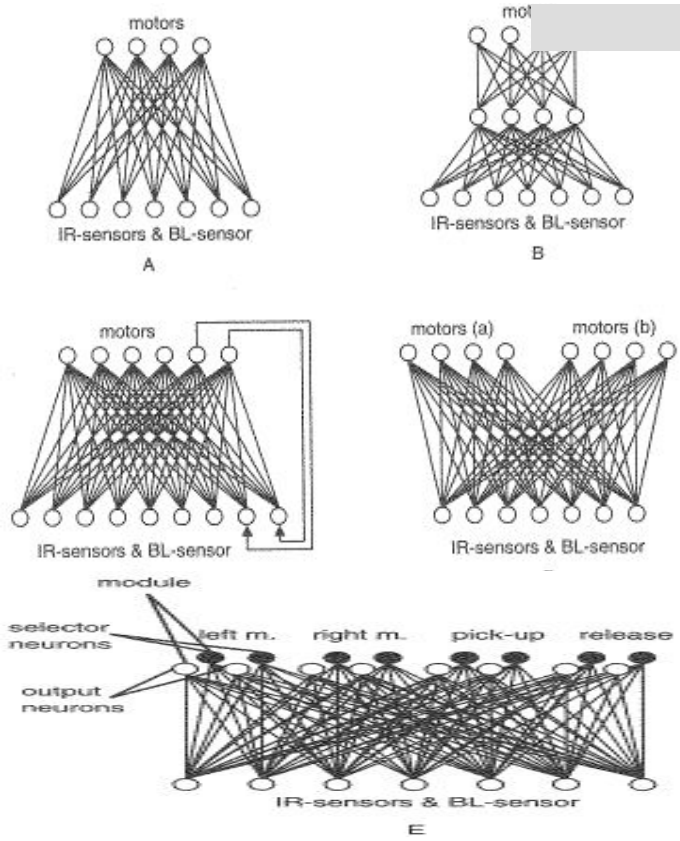
Four neurons per motor neuron

Two selector neurons,
two output neurons.

If first selector neuron >
second selector neuron:
first output neuron
controls motor neuron.
otherwise:
second output neurons
controls motor neuron.

How many ‘states’ are possible?
State = which selector neurons
control the motor neurons.

Structural modularity

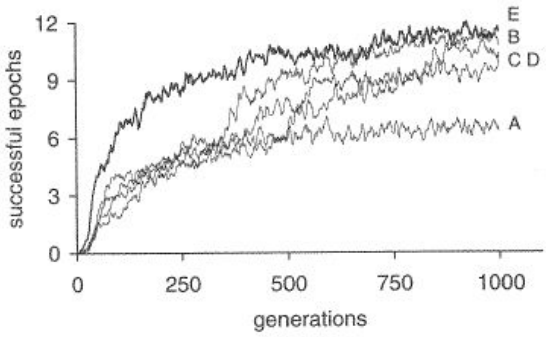


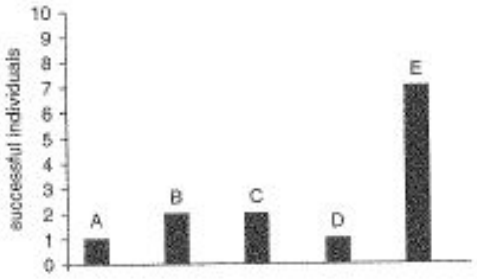
Can investigate the role of:

a) Internal units
(internal neurons can ‘recode’
sensor signals to cause
very different reactions to
very similar patterns.)

(b) Recurrent connections
(allows the robot to
react differently to the
same stimuli, if it experienced
different conditions just
previously.)

(c) Modularity
(divide continuously changing
sensor signals into groups,
to be processed by different
neural modules).





For the best controller from the 10 replications, how many successfully cleaned the arena?

Structural modularity

Results:

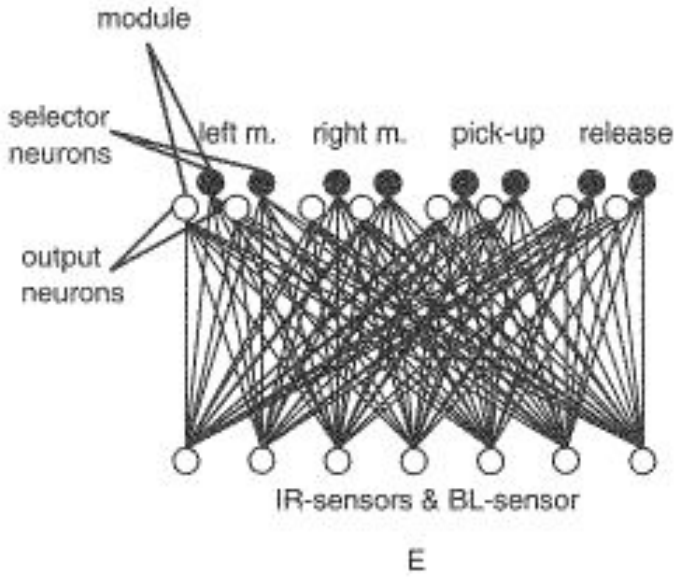
For each controller,

Robot put into arena 15 times, or epochs, at different positions and orientations.

1 epoch = 200 actions or object correctly released.

Controllers evolved for 1000 generations.

10 replications for each of the five controllers:
50 runs in total.



E

Structural modularity

Q: Is there a correspondence between evolved modules and distal behaviors (e.g. explore until object found?)

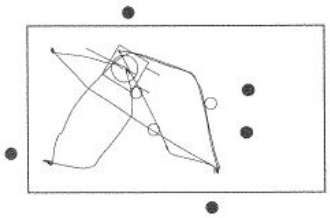
A: No.

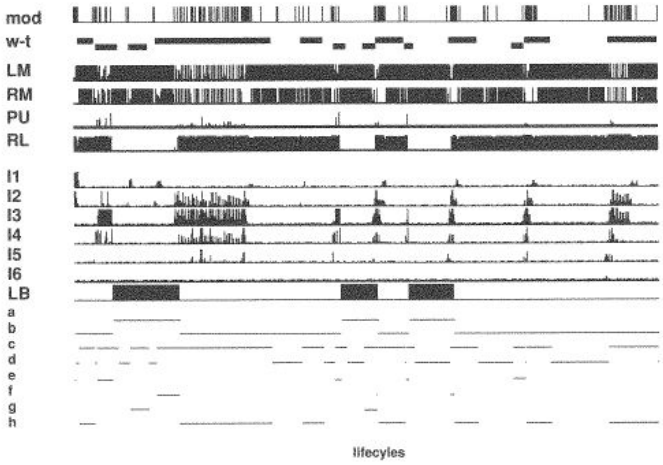
Results from one evolved controller E.

Controller has two states:

Only right motor is controlled by both output neurons;

for the other three motor neurons, one selector neuron always 'wins'.





Structural modularity

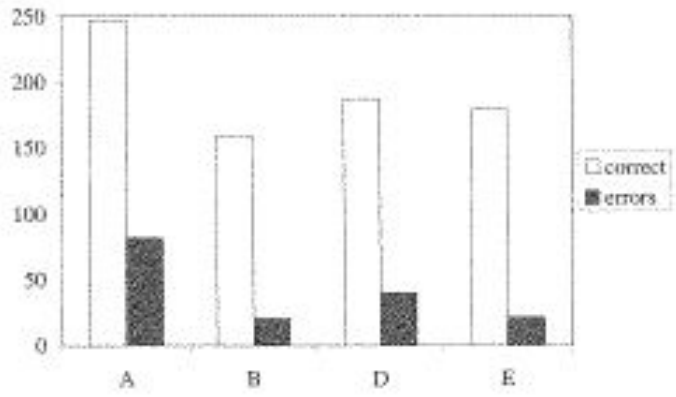
Mod: Which module controls the right motor?

w-t: what is perceived?
w=wall; t=target object

LM=left motor; RM=right motor
PU=pick up; RL=release

i1-i6: infrared sensors
LB = light barrier sensor

a: carrying object, look for wall
b: gripper empty, look for object
c: sees something: wall/object?
d: doesn't see anything
e: approaching object
f: approaching wall
g: avoiding an object
h: avoiding a wall.



Structural modularity

When evolved controllers are placed in robot at novel locations, how many errors are made?

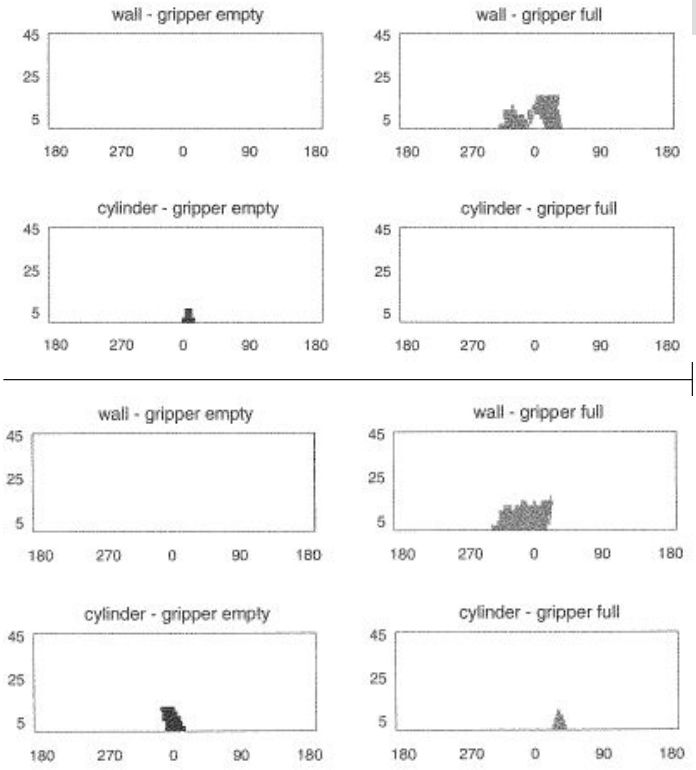
Novel stimuli:

Robot placed near object or wall at 180 different angles, 20 different distances, gripper full or empty. (180x20x2=7200 trials)

Robot with internal neurons (B) or emergent modules (E) makes fewer mistakes.

B, E: Two out of 10 networks never made mistakes.
A,D: All 10 made mistakes.
C: ?

Structural modularity



One evolved E controller that <u>never</u> makes mistakes:

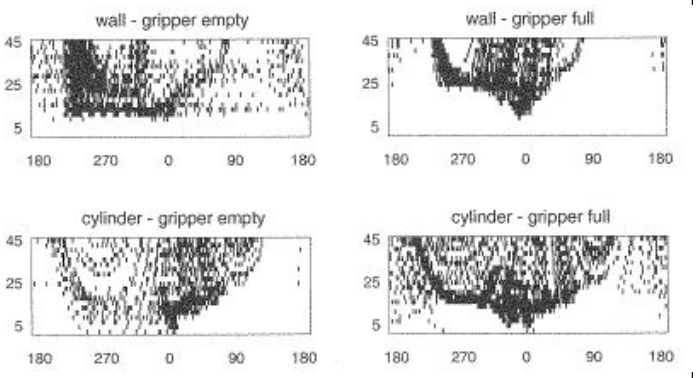
Black pixel: triggers pick-up behavior.
Gray pixel: triggers release behavior.
White pixel: neither is triggered.

One evolved E controller that makes mistakes:

Mistakenly releases object when it encounters an object close up on its right side.

However…

Structural modularity



…this controller never encountered this situation when behaving in the arena.

(black pixel: situation experienced when evaluated during evolution.)

(white pixel: situation <u>not</u> experienced when evaluated during evolution.)

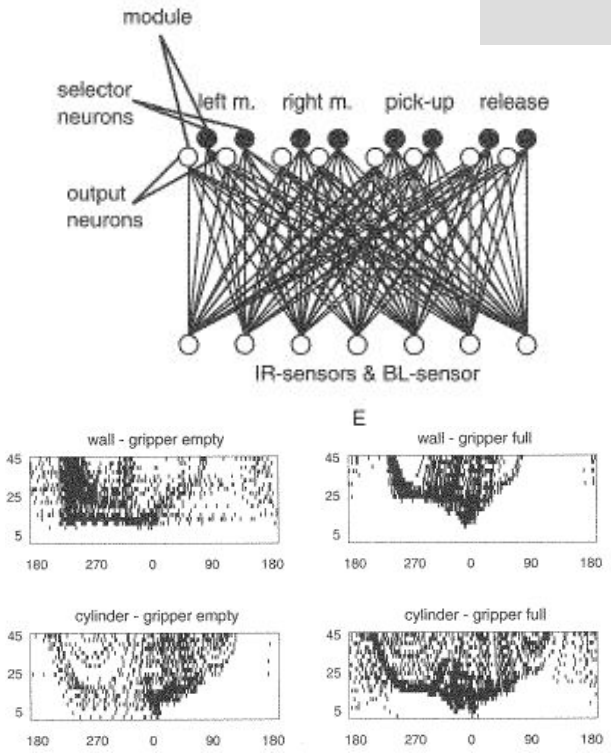


Fig. 6.10

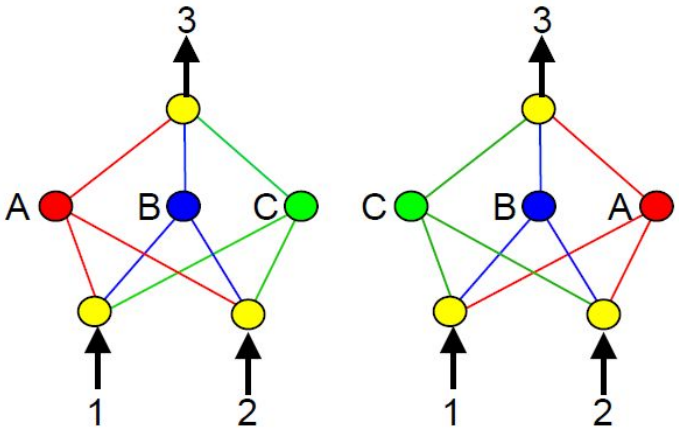
Structural modularity

Experiment was a success; evolution determined:

1. which sensors states would be experienced while moving
2. whether a behavior should be divided into modules (evolved usage of selector neurons)
3. What situations should trigger which modules (sensor -> selector neuron synaptic weights)
4. What each module should do (sensor -> output neuron synaptic weights)

17

NEAT / HyperNEAT



[A,B,C]
X[C,B,A]

Crossovers: [A,B,A] [C,B,C]
(both are missing information)

Invalid offsprings

Competing Conventions:

Two neural networks:

Both encode the same function;

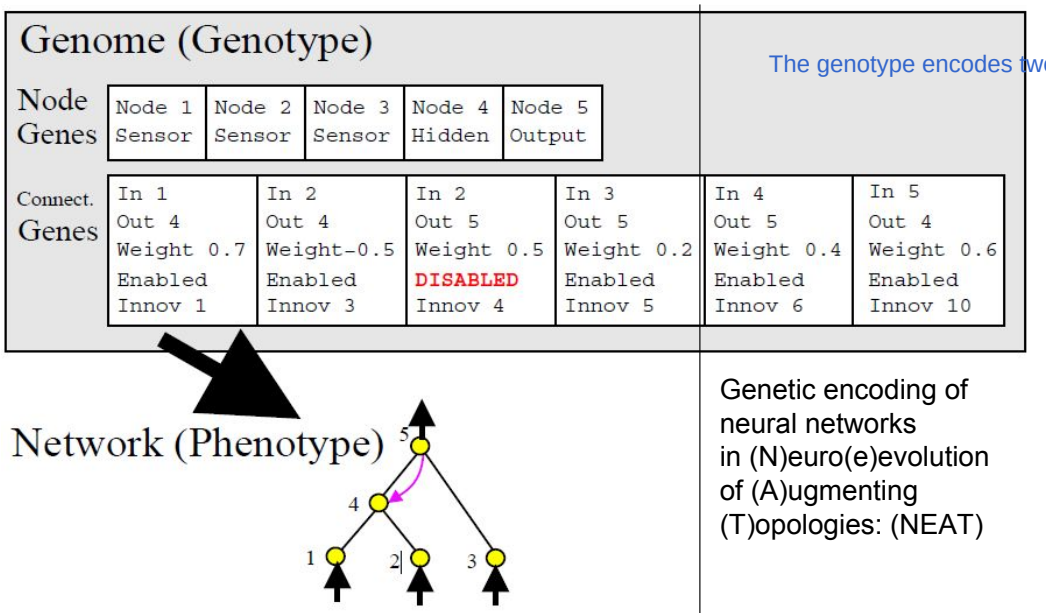
Have different conventions for doing so.

No matter how they're crossed, their children will lack information.

Stanley, K.O., Miikkulainen (2001) Evolving Neural Networks through Augmenting Topologies.

1

NEAT / HyperNEAT

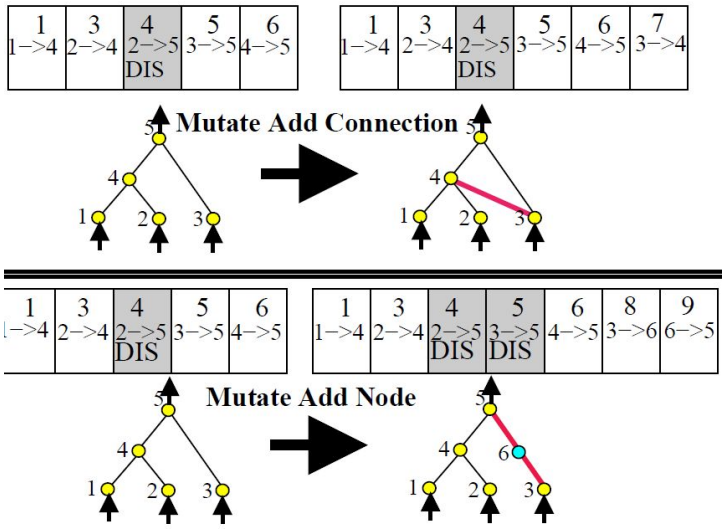


The genotype encodes two lists

Genetic encoding of neural networks in (N)euro(e)evolution of (A)ugmenting (T)opologies: (NEAT)

Stanley, K.O., Miikkulainen (2001) Evolving Neural Networks through Augmenting Topologies.

NEAT / HyperNEAT



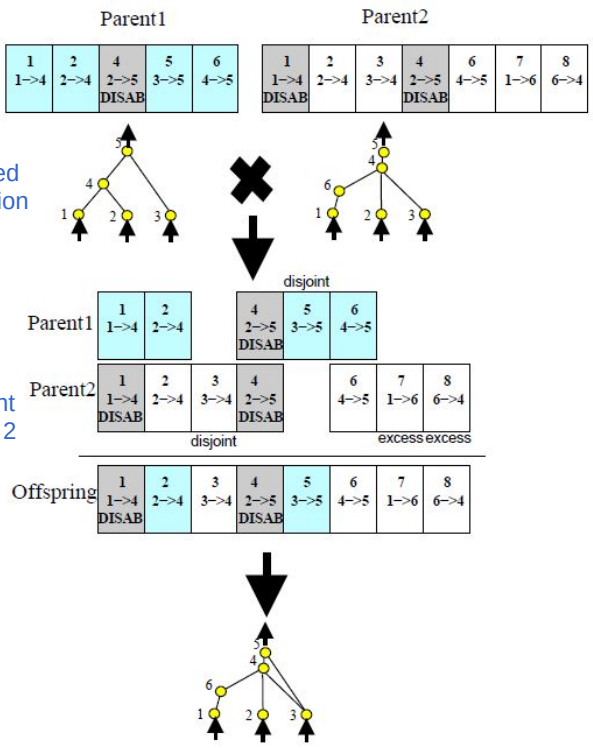
Historical Markings:

Keep a global counter; every time a neuron or synapse is added, assign the value of the counter, and increment it.

Genes/synapses can be disabled, but remain in the genome.

Stanley, K.O., Miikkulainen (2001) Evolving Neural Networks through Augmenting Topologies.

NEAT / HyperNEAT



NEAT designed so That crossover is

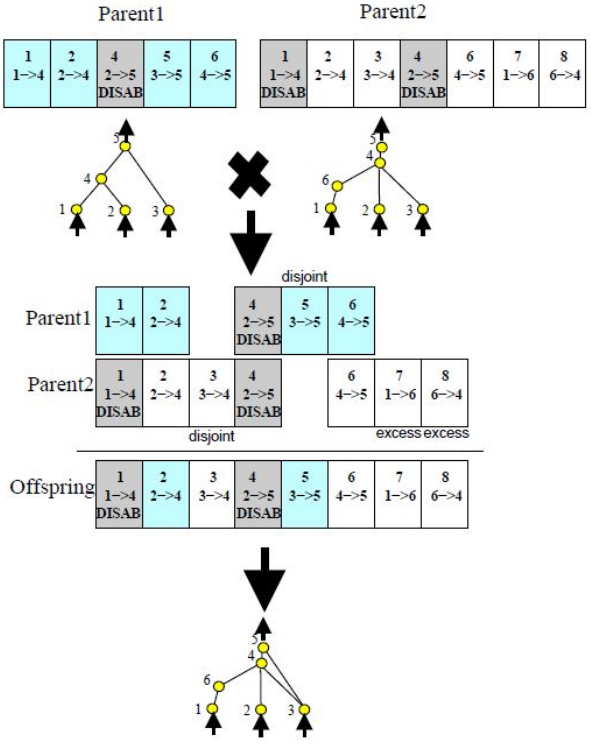
(1) Algorithmically simple

and

(2) Produces children that are similar to their parents.

Stanley, K.O., Miikkulainen (2001) Evolving Neural Networks through Augmenting Topologies.

NEAT / HyperNEAT



Take two parent NNs:

Line up connection genes according to their historical markings.

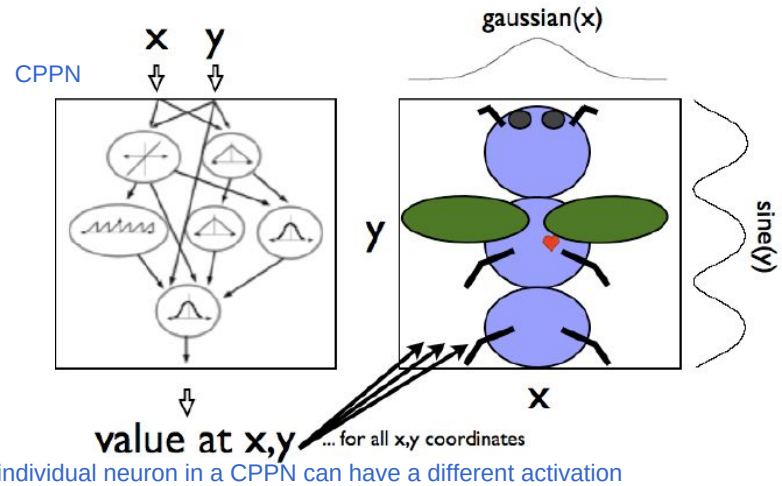
For matching genes, copy either gene into child at random.

Disjoint genes (those in the middle without a partner gene)

Excess genes (those at the end)

Stanley, K.O., Miikkulainen (2001) Evolving Neural Networks through Augmenting Topologies.

NEAT / HyperNEAT



Each individual neuron in a CPPN can have a different activation function.
Input layer receives inputs coordinates which are particular positions in the phenotype.
The same CPPN can be used at different resolutions.

CPPNs compose coordinates transformations.

HyperNEAT:

Evolves neural networks (compositional pattern-producing networks) that produce regular patterns spatially:

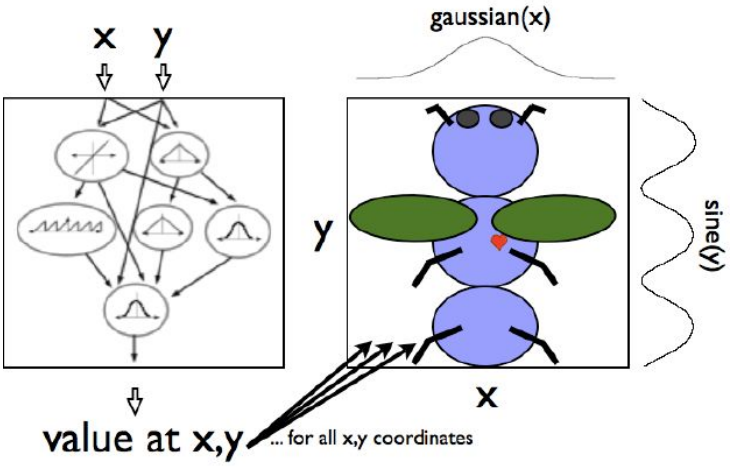
CPPNs take spatial coordinates as input

output determines the value at that position

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

L14

NEAT / HyperNEAT



HyperNEAT:

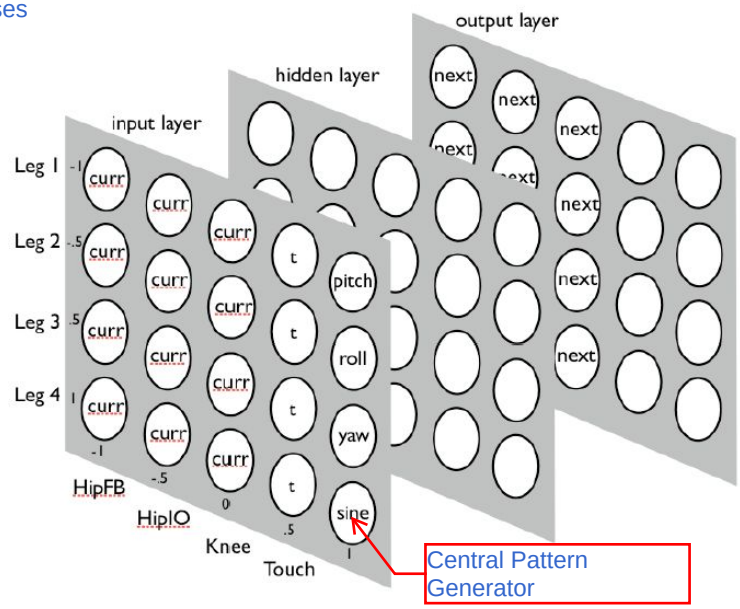
HyperNEAT "paints" regular patterns on to a hypercube.

The dimensionality of the hypercube is determined by the dimension of the input coordinates.

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

NEAT / HyperNEAT

HyperNEAT for robots is used to paint weights into synapses

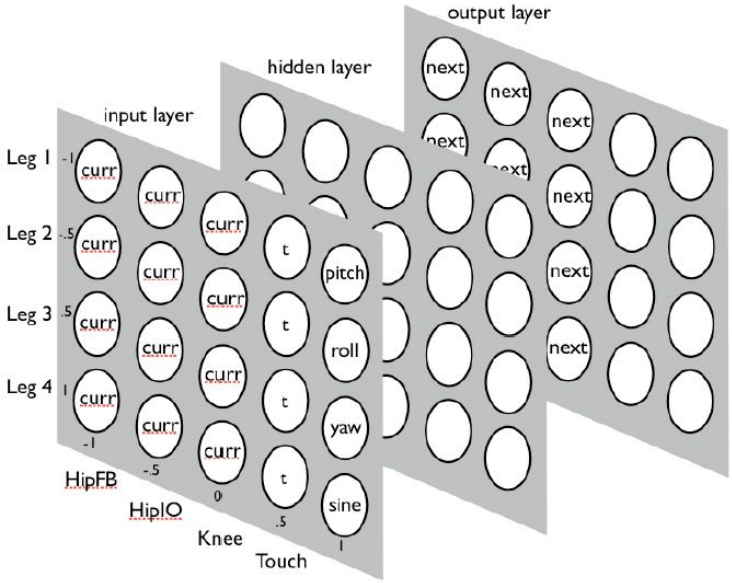


HyperNEAT can be used to "paint" weights on to synapses of a second neural network.

Requires that each neuron and synapse have a 3D location:

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

NEAT / HyperNEAT



HyperNEAT can be used to "paint" weights on to synapses of a second neural network.

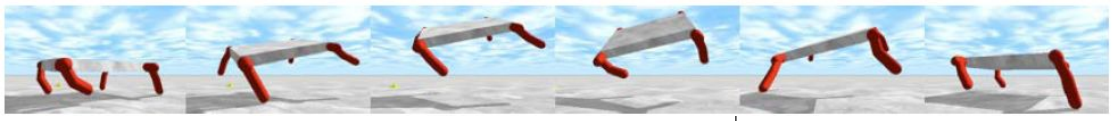
…why do this, if NEAT already evolves neural networks?

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

NEAT / HyperNEAT

HyperNEAT

FT-NEAT



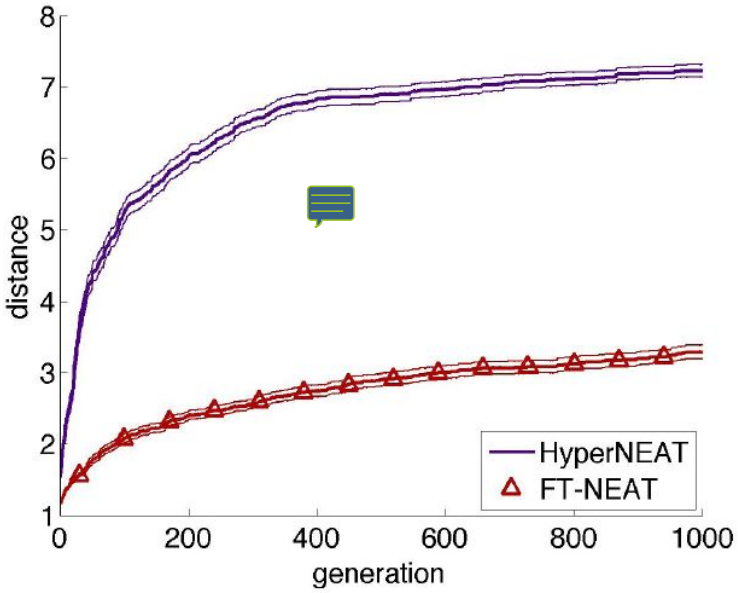
Compare HyperNEAT to NEAT:

FT-NEAT: Fixed Topology NEAT.

Use same NN as in HyperNEAT;
allow for mutation and crossover, but
not the addition/removal of neurons/synapses.

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

NEAT / HyperNEAT

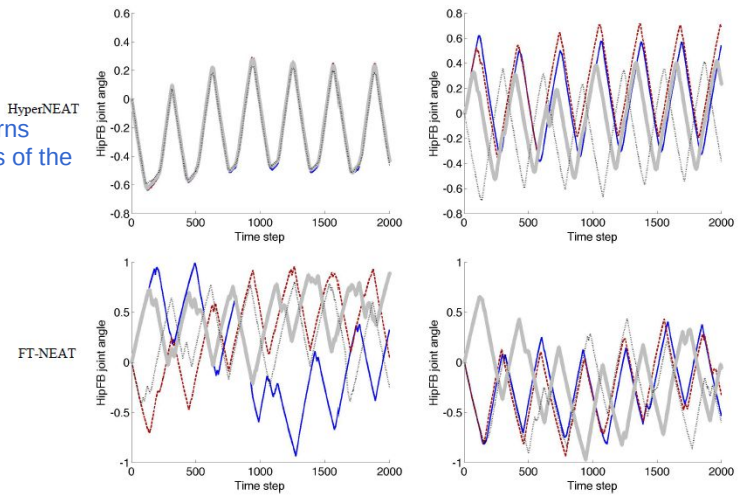


Results from evolving locomotion.

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

NEAT / HyperNEAT

There are regular patterns in the outputs of the legs



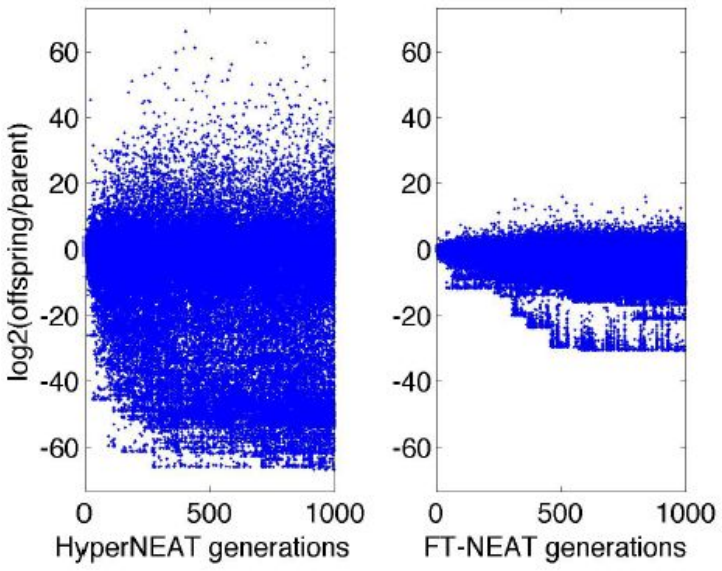
HyperNEAT repeatedly finds regular gaits;

FT-NEAT does not.

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

Mutation plot

NEAT / HyperNEAT



For mutations,

mean fitness of children compared to parents:

HyperNEAT:
FT-NEAT:

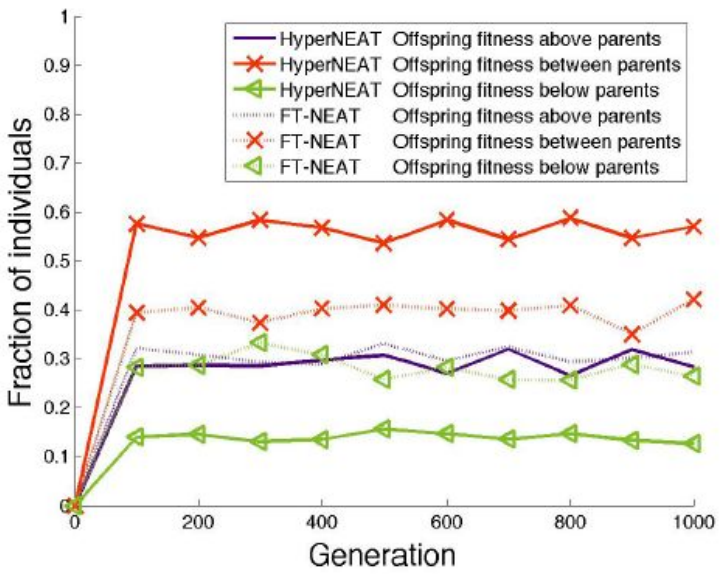
but HyperNEAT tends to create more fit children than FT-NEAT.

..why?

It also creates much worse children, but these are discarded.

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

NEAT / HyperNEAT



If children are produced by crossover…

In HyperNEAT, offspring tend to be more like their parents than in FT-NEAT.

…why?

Clune, J. *et al.* (2009) Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding

The radical envelope-of-noise hypothesis

Observation: Evolution will create controllers that exploit details of the simulation.
If those details do not exist in reality, the controller fails to cross the reality gap.

Hypothesis: Adding noise to a simulator in the right way forces evolution to create more robust controllers.

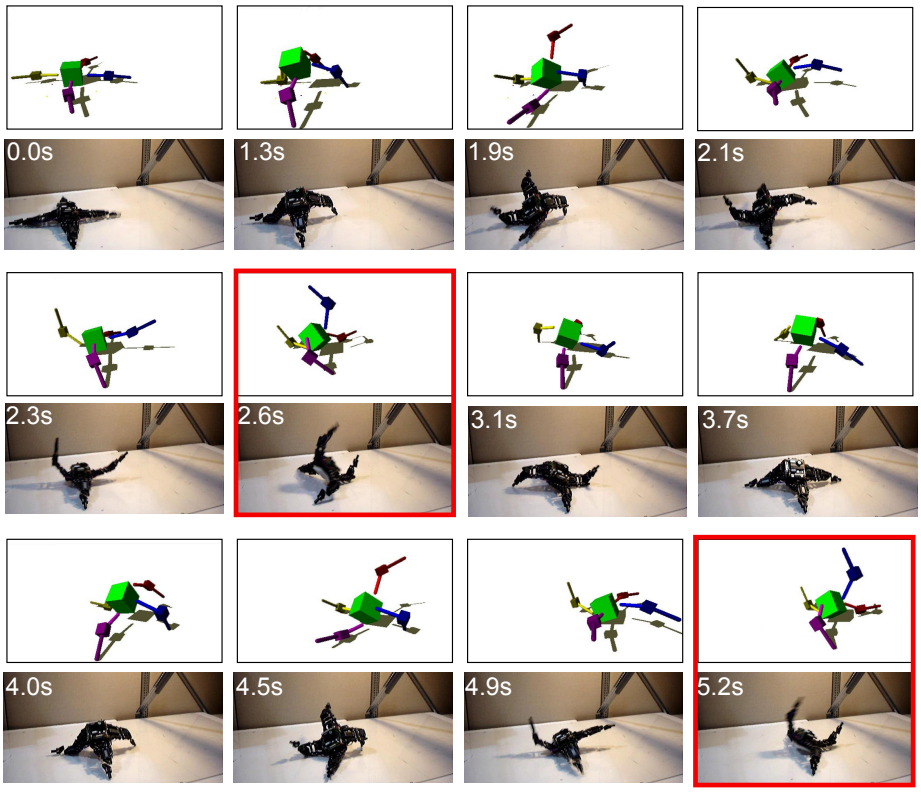
Question: What aspects of a simulation to noisify?
If the simulation is complex, many things must be noisified: friction, mass distribution, geometry, joint properties, sensor properties, motor properties.

Solution: Create a minimal simulation.

Minimal simulation: Simulation that contains as little details as possible.

Jakobi, N. (1997) Evolutionary Robotics and the Radical Envelope-of-Noise Hypothesis. *Adaptive Behavior*, 6(2): 325-368.

Generating behaviors using an optimized model



The radical envelope-of-noise hypothesis

Base set of robot-environment interactions:

those simulation aspects that have a basis in reality
(but will still be slightly different in reality)

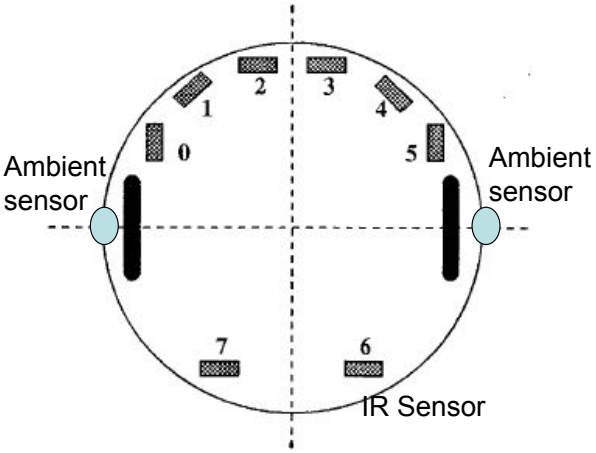
Implementation set of interactions:

those simulation aspects that *do not* have a basis in reality.
(might be completely different in reality)

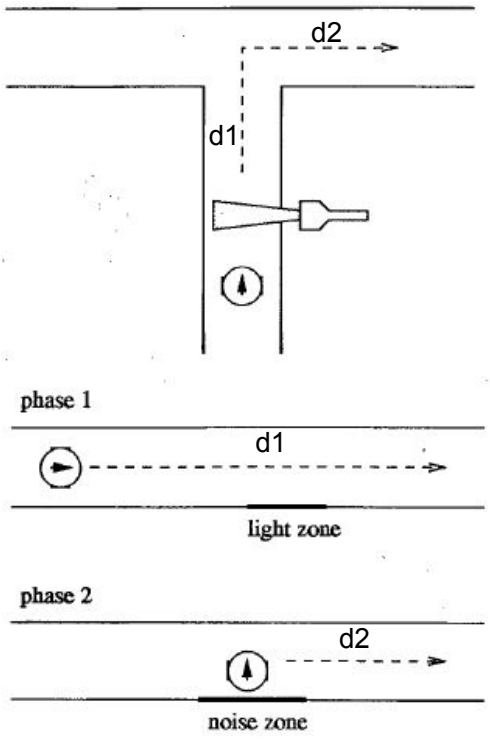
During evolution in simulation…

add a bit of noise to the base set
add 100% of noise to the implementation set
(so evolved controllers do not rely on them)

Radical Envelope-of-Noise Hypothesis



Radical Envelope-of-Noise Hypothesis



The task (requires memory)

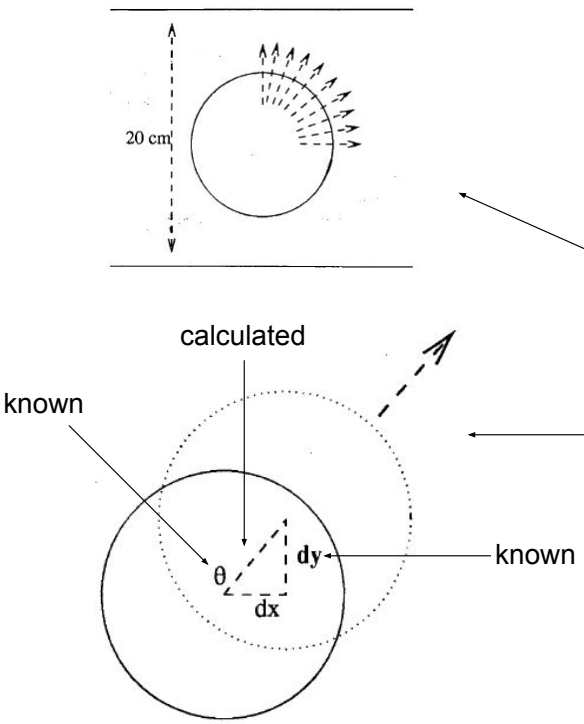
If light on right,
turn right at corner.

If light on left,
turn left at corner.

f = d1 + d2 + 100 correct way
at lights;
= d1 + d2 otherwise

Minimal simulation:

Robot can move through
two, infinitely long corridors.



Radical Envelope-of-Noise Hypothesis

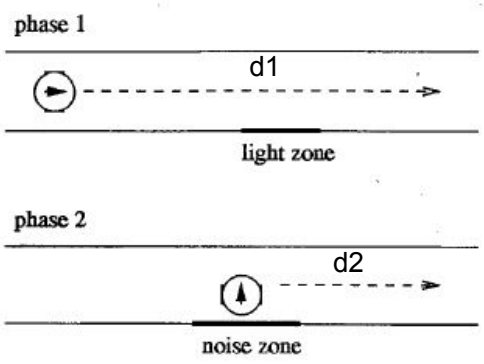
Minimal simulation (contd).

Simulation is made up of two look up tables:

Table I:
For robot's orientation, if move forward, update robot's x and y position.

Table II:
For robot's orientation θ and distance from the wall dy, how long is the line segment from IR sensor to wall?

Convert length to a sensor value.

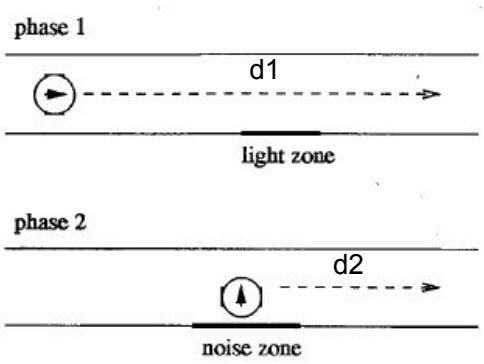


Radical Envelope-of-Noise Hypothesis

Base set (some basis in reality)

1. How robot moves in response to motor signals.
2. How IR sensors respond.
3. How the ambient sensors respond.

Add a bit of noise to these interactions.

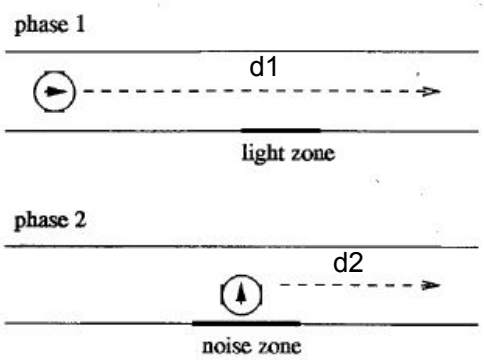


Radical Envelope-of-Noise Hypothesis

Implementation set (implementation details of simulation unlikely to match details in reality)

1. How IR sensors behave when the robot is at the T-junction.

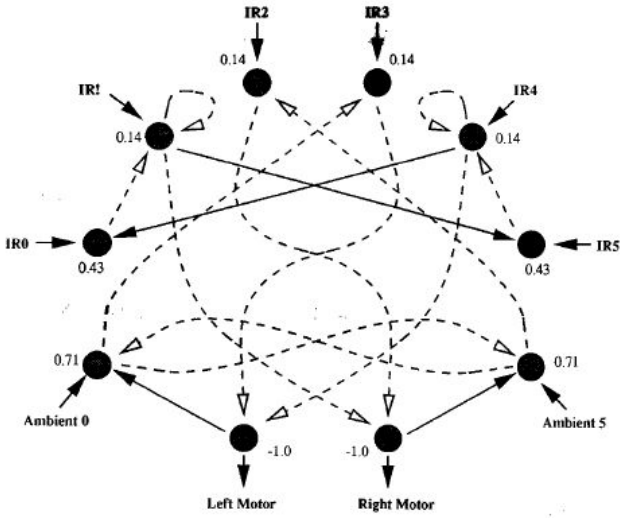
Add 100% noise:
If an IR sensor ray intersects the "noise zone", return a value randomly in sensor's [min,max] range.



Radical Envelope-of-Noise Hypothesis

Additional implementation details (some noise added)

2. Side from which light comes
3. Corridor width
4. Starting robot orientation
5. Length of light zone
6. Corridor length



Radical Envelope-of-Noise Hypothesis

Evolved controllers:

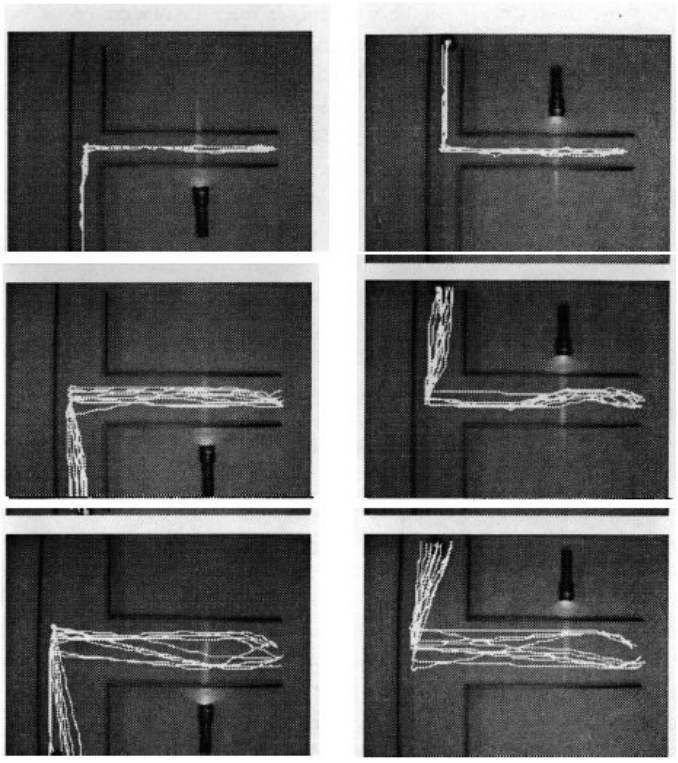
Bilateral symmetry imposed.

Solid arrows: Excitatory links

Dashed arrows: Inhibitory links

Numbers: Activation thresholds

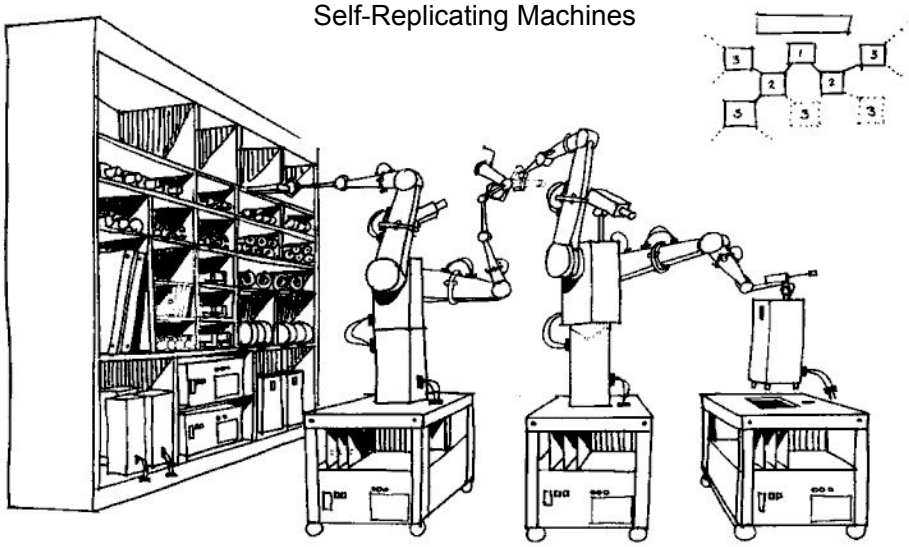
Q: Where is the memory of the light?



Radical Envelope-of-Noise Hypothesis

Results

Self-Replicating Machines



Source: NASA Conference Publication 2255 (1982), based on the Advanced Automation for Space Missions NASA/ASEE summer study Held at the University of Santa Clara in Santa Clara, California, from June 23-August 29, 1980.

Original idea: John von Neumann, 1948;
often referred to as “von Neumann machines”
(different from von Neumann architecture)

The Golem Project

The New York Times

THURSDAY, AUGUST 31, 2000

Scientists Report They Have Made Robot That Makes Its Own Robots

By KENNETH CHANG

For the first time, computer scientists have created a robot that designs and builds other robots, almost entirely without human help.

In the short run, this advance could lead to a new industry of inexpensive robots customized for specific tasks. In the long run — decades at least — robots may one day be truly regarded as “artificial life,” able to reproduce and evolve, building improved versions of themselves.

Such durable, adaptive robots, astronomers have suggested, could someday be sent into space to explore the galaxy or search for other life.

But the quest to create artificial life also revives concerns that computer scientists could eventually create a robotic species that would supplant biological life, including humans.

“Some things we probably can do we shouldn’t do,” said Bill Joy, chief scientist at Sun Microsystems, who wrote a recent article warning of the power of emerging technologies.

Brandeis University

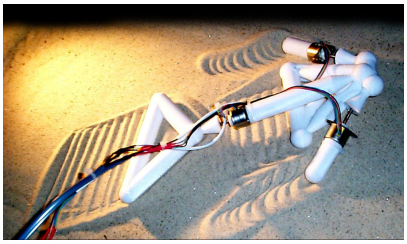
The “Arrow” left a trail as it crawled across a bed of sand.

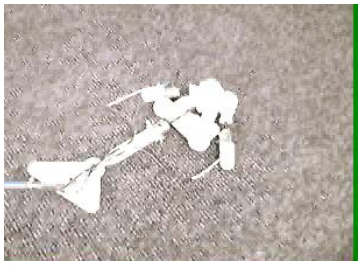
were not manufactured by humans.”

Dr. Pollack and Dr. Lipson, a research scientist, report their results in today’s issue of the journal Nature.

“This is the first example of pretty much 100 percent automated evolution of a machine,” said Dr. Philip Husbands, a professor of artificial intelligence at the University of Sussex in England. “It’s a rather primitive example, but it’s the first step to something that could be quite significant.”

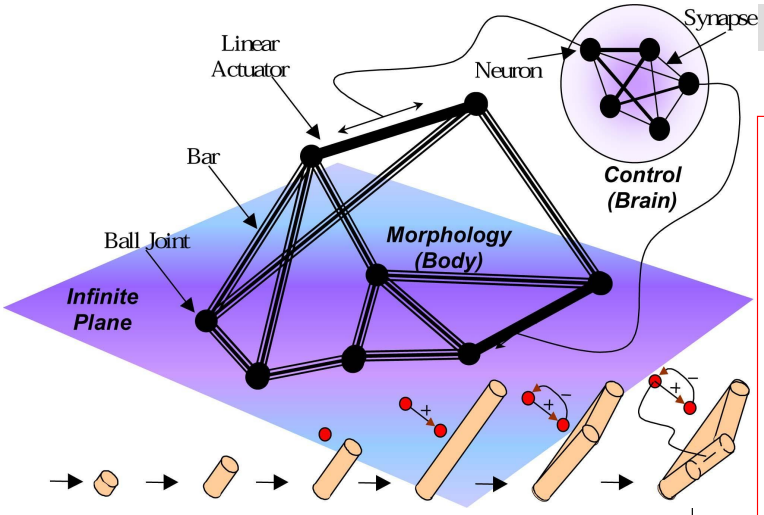
In the future, the technique could





Lipson, H., Pollack J. B., (2000), "Automatic Design and Manufacture of Artificial Lifeforms", Nature 406, pp. 974-978.

The Golem Project



Step 1:

Evolve a robot to locomote in simulation (evolve body and brain)

Step 2:

Manufacture robot using 3D printer

Step 3:

Snap in motors and electronics

Genotype encoding:

The Golem Project

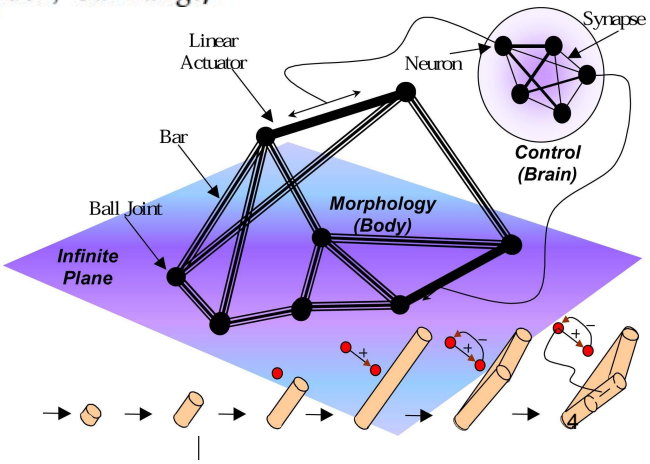
robot := ⟨vertices⟩⟨bars⟩⟨neurons⟩⟨actuators⟩

vertex := ⟨x, y, z⟩

bar := ⟨vertex 1 index, vertex 2 index, relaxed length, stiffness⟩

neuron := ⟨threshold, synapse coefficients of connections to all neurons⟩

actuator := ⟨bar index, neuron index, bar range⟩



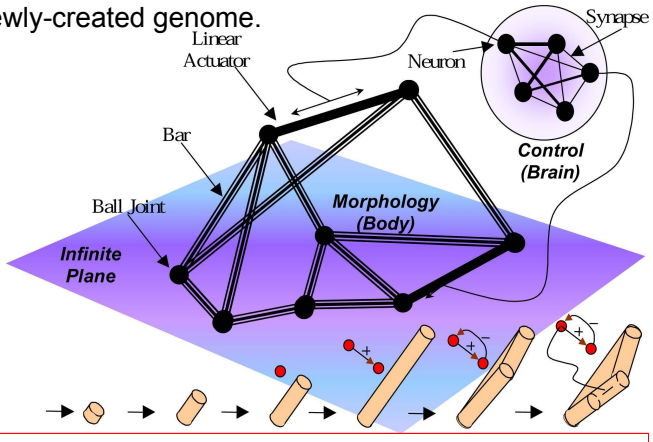
Genotype encoding: **The Golem Project**

Initial population: 200 null genotypes.
Fitness: Displacement of the center of mass
Evaluation period: “Fixed number (12) of cycles of its neural control”?

Genotype encoding: **The Golem Project**

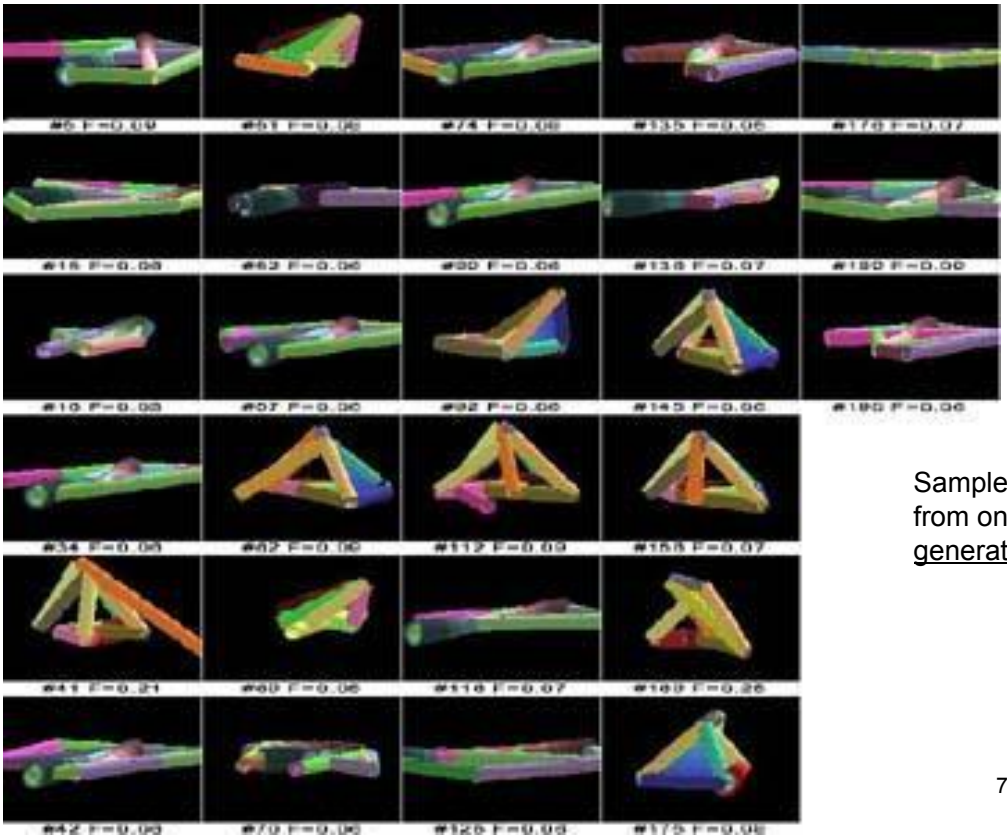
Mutations: 1. Change the length of a bar (10%)
2. Change a synaptic weight (10%)
3. add or 4. remove a dangling bar (1%)
5. add or 6. remove unconnected neuron (1%)
7. split vertex into two and add a small bar (3%)
8. split bar into two and add vertex (3%)
9. attach or 10. detach neuron to bar (3%)

One mutation was applied to each newly-created genome.



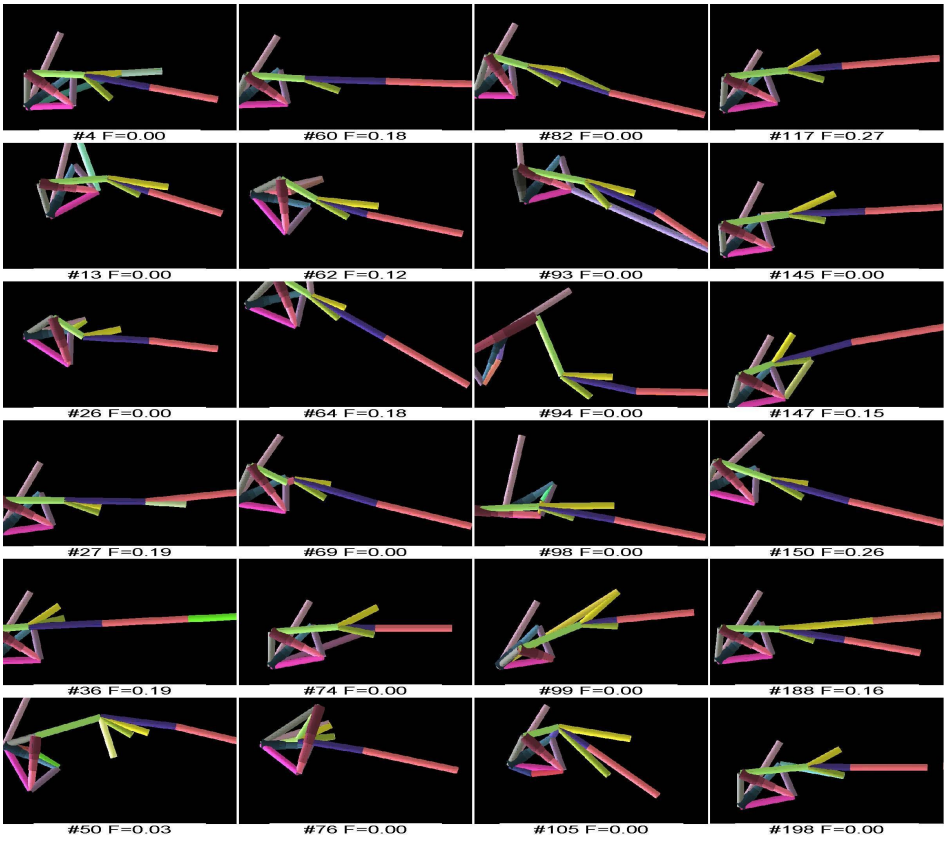
Q: Mutation sequence:

3 -> 1 -> 1,5 -> 1,5 -> 7, 2 -> 9



Samples from one generation.

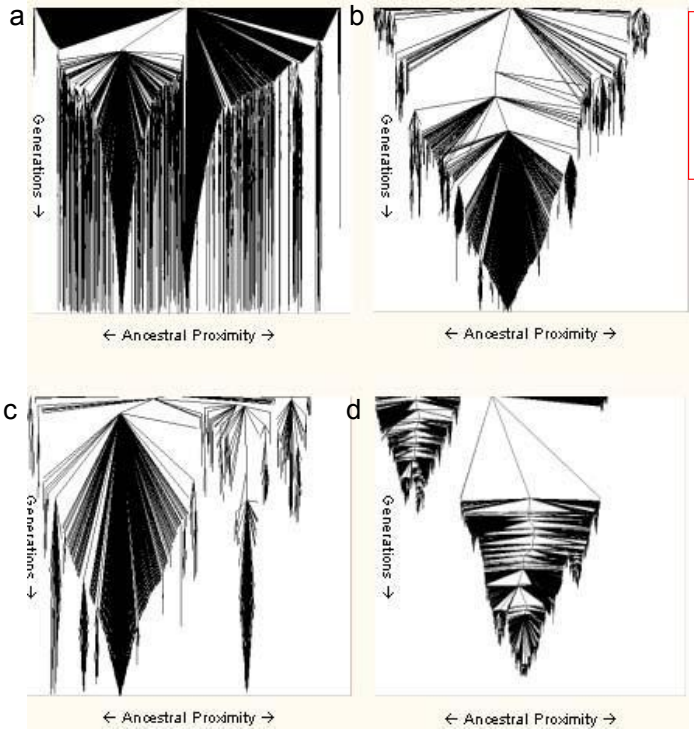
7



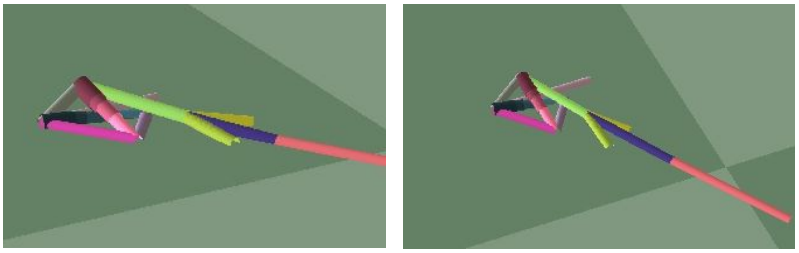
Samples from one evolutionary run.

8

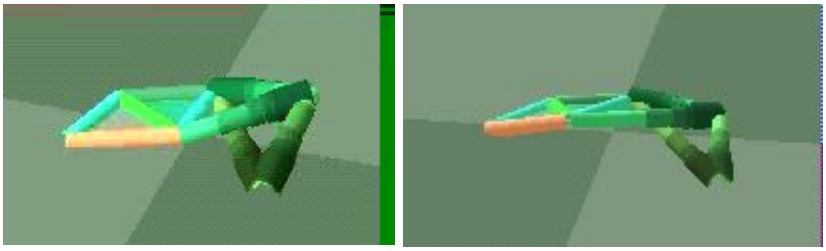
Another way of visualizing artificial evolution: phylogenetic trees
Node = individual; link = parent-child relationship
Horizontal distance between two nodes = similarity of ancestry



a: divergence
b: convergence
c: emergence of species
d: extinction



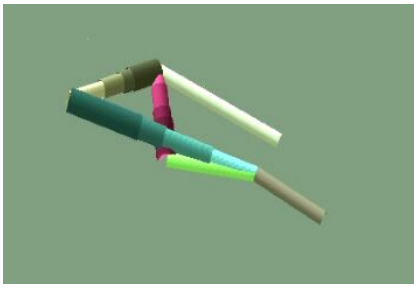
"Snakes"

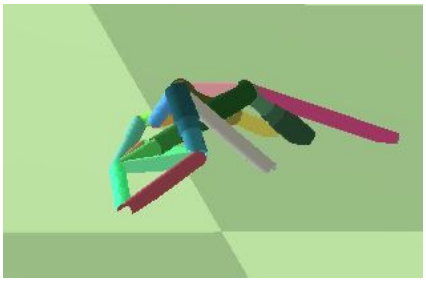


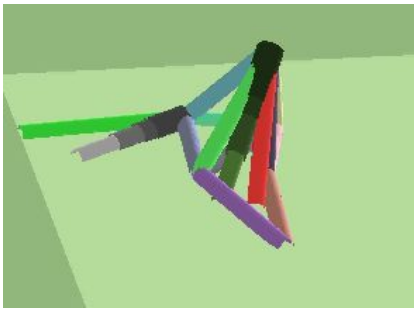
"Crabs"

The Golem Project

Samples from two evolutionary runs.





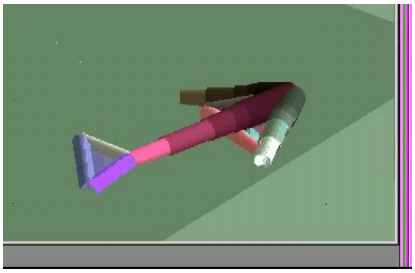


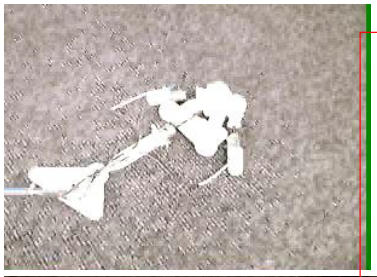
The Golem Project

Samples from several evolutionary runs.

Crossing the reality gap:

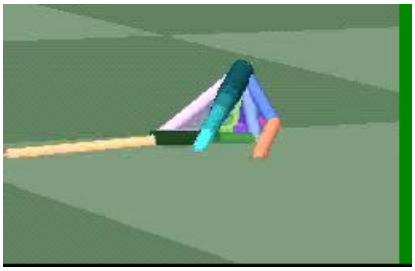
The Golem Project





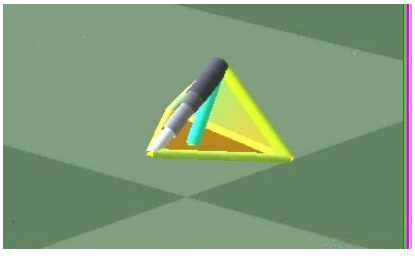
Arrow:
Virtual distance

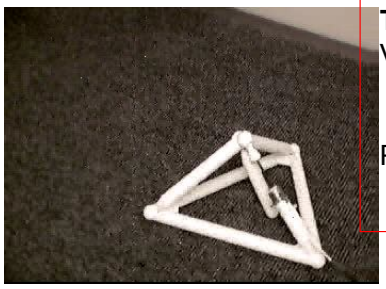
Physical distance



Pusher:
Virtual distance

Physical distance



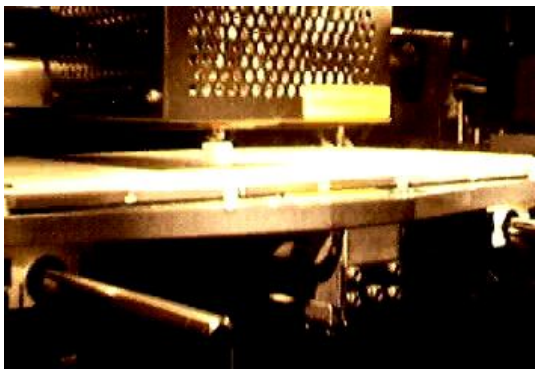


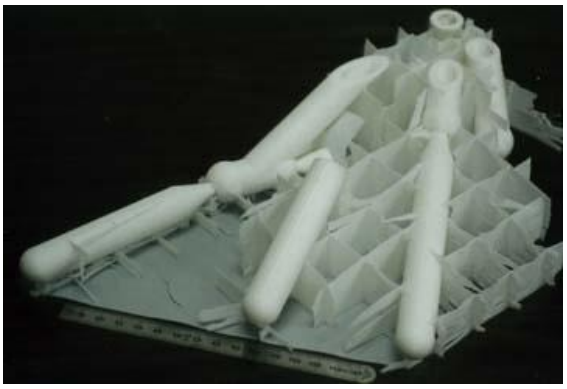
Tetrahedron:
Virtual distance

Physical distance

Crossing the reality gap:

The Golem Project



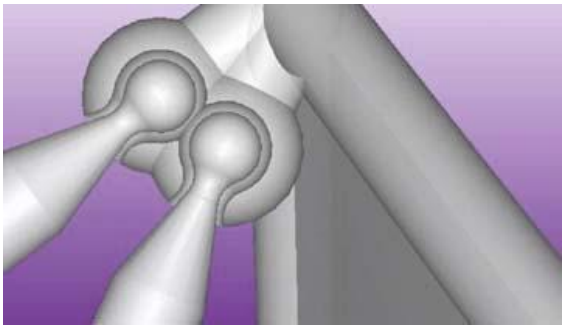


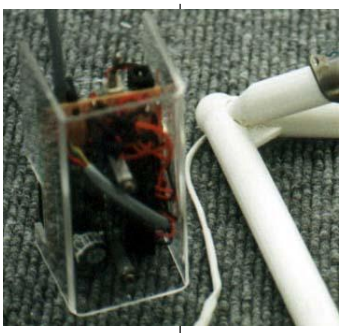
Step 1:
Evolve a robot to locomote
in simulation (evolve body and brain).

Step 2:
Manufacture robot using 3D printer

Step 3:
Snap in motors and electronics

Crossing the reality gap:



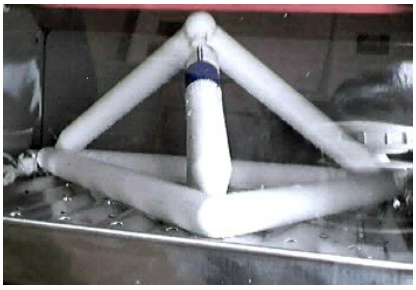


Step 1:
Evolve a robot to locomote
in simulation (evolve body and brain).

Step 2:
Manufacture robot using 3D printer

Step 3:
Snap in motors and electronics

Crossing the reality gap:



Reduce,
Reuse,
Recycle…

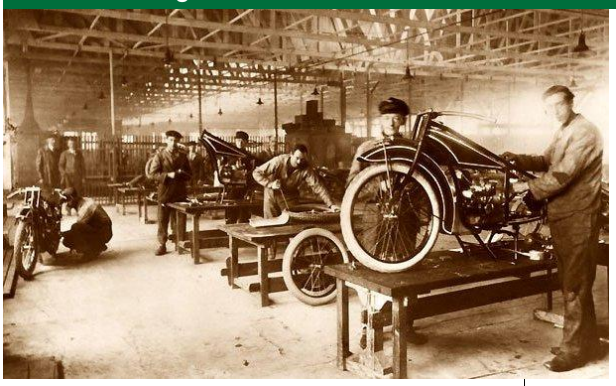
Reduce,
Reuse,
Recycle…

Fab@Home: Open source 3D printing technology, revolutionizing manufacturing

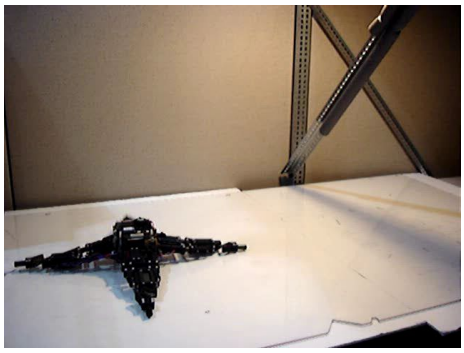
Rep Rap: A machine that makes a machine that makes a machine that makes…

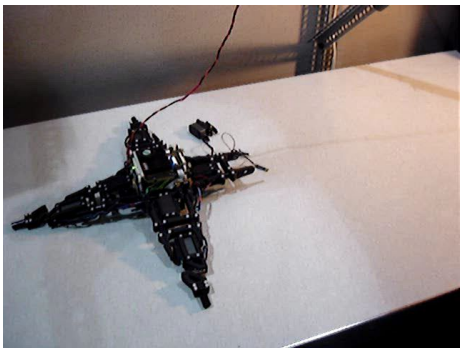
Q: How could evolutionary algorithms be used to expand the capabilities of this technology?

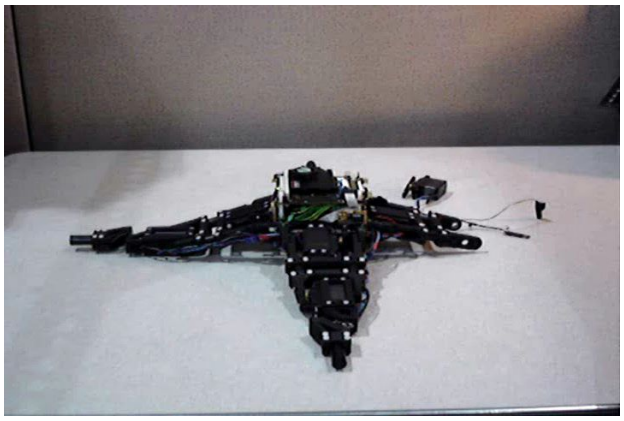
Manufacturing versus Construction



Resiliency: adaptation to constant change L17







Evolutionary robotics approaches L17

Evolve the controller of a robot to automatically discover (near-)optimal behavior

Three existing approaches to evolutionary robotics:

Evolve controllers directly on a physical robot.
Requires 100s or 1000s of physical evaluations.

Create a simulation of the robot, and perform some or all of controller evolution in simulation before transferal to the physical device.
Requires a human to hand craft the simulator;
"Reality gap" problem.

Adapt controllers on the physical robot from an original, hand-created controller
Requires a human to hand craft the original controller.

Evolutionary robotics approaches L17

Evolve the controller of a robot to automatically discover (near-)optimal behavior

Three existing approaches to evolutionary robotics:

Evolve controllers directly on a physical robot.
Requires 100s or 1000s of physical evaluations.

Create a simulation of the robot, and perform some or all of controller evolution in simulation before transferal to the physical device.
Requires a human to hand craft the simulator;
"Reality gap" problem.

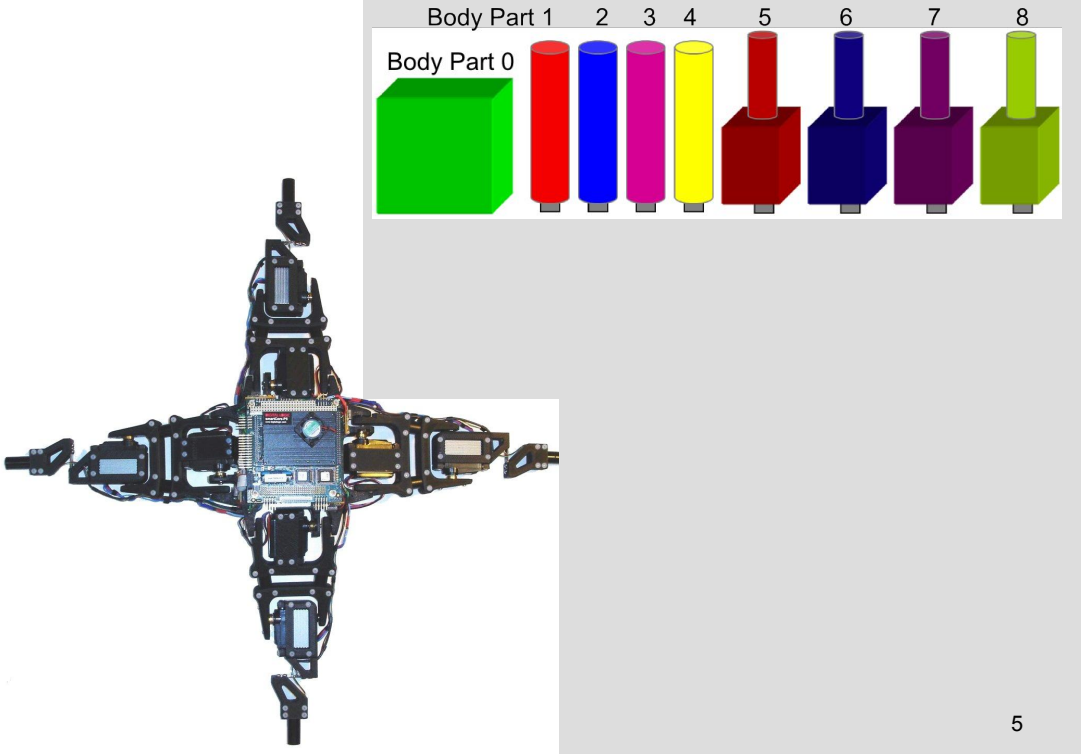
Adapt controllers on the physical robot from an original, hand-created controller
Requires a human to hand craft the original controller.

Alternative approach—The Estimation-Exploration Algorithm (EEA):

1. Use the experiences of the robot to evolve an accurate simulator (estimation)
2. Use the evolved simulator to evolve a useful controller (exploration)

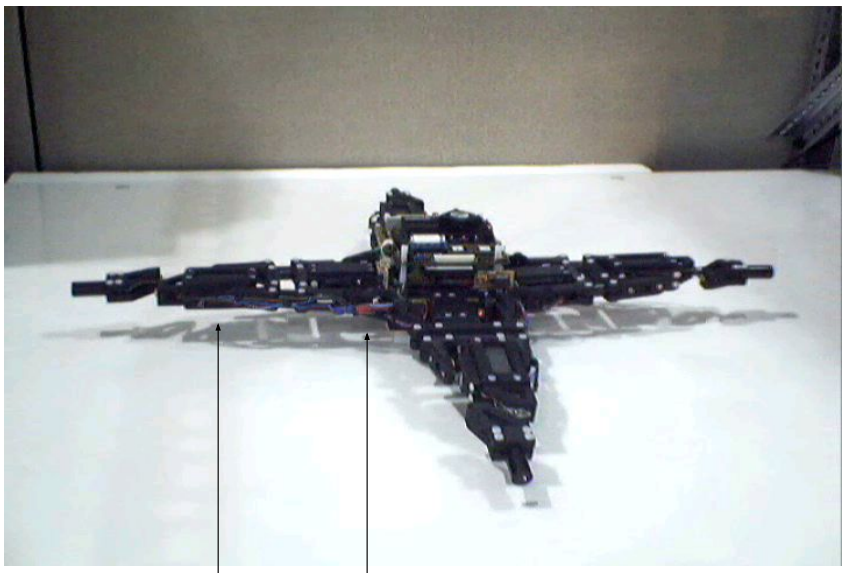
4

Typical experiment L17

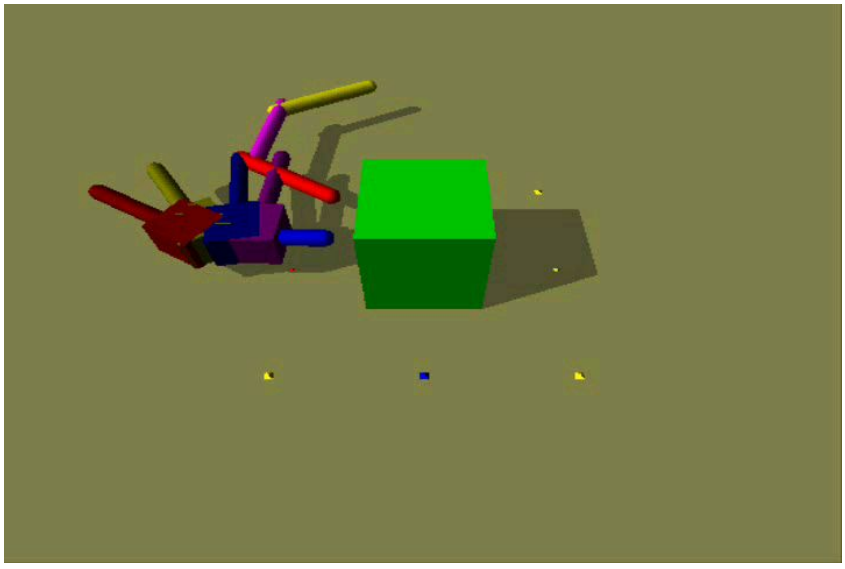


5

Typical experiment L17

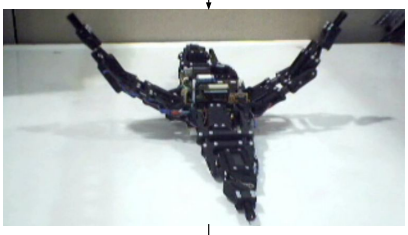


Typical experiment L17



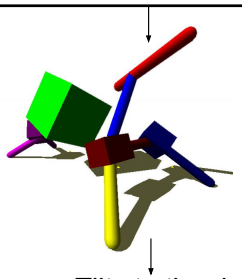
Evaluating candidate self-models

+30 −30 −30 +30 +30 +30 +30 +30



Does not tilt

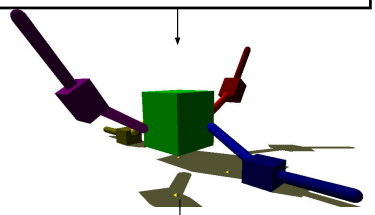
+30 −30 −30 +30 +30 +30 +30 +30



Tilts to the right

Higher error

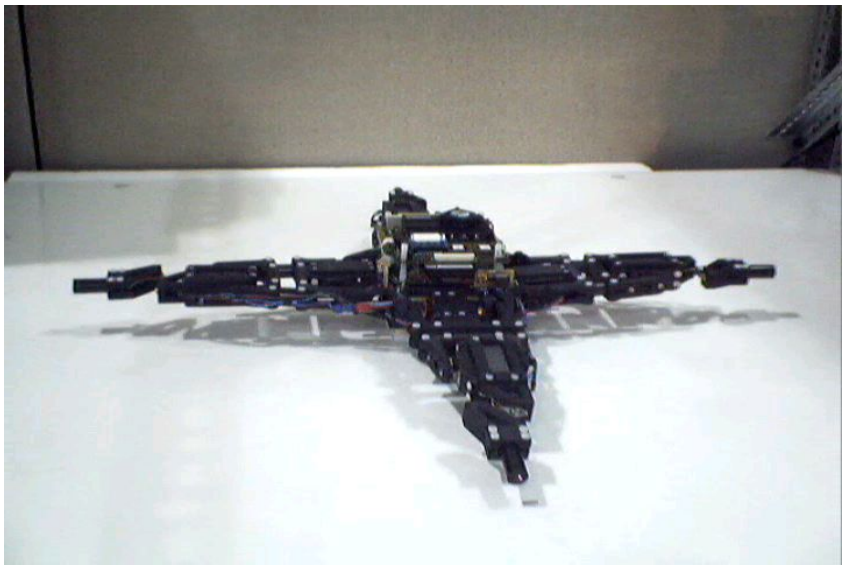
+30 −30 −30 +30 +30 +30 +30 +30



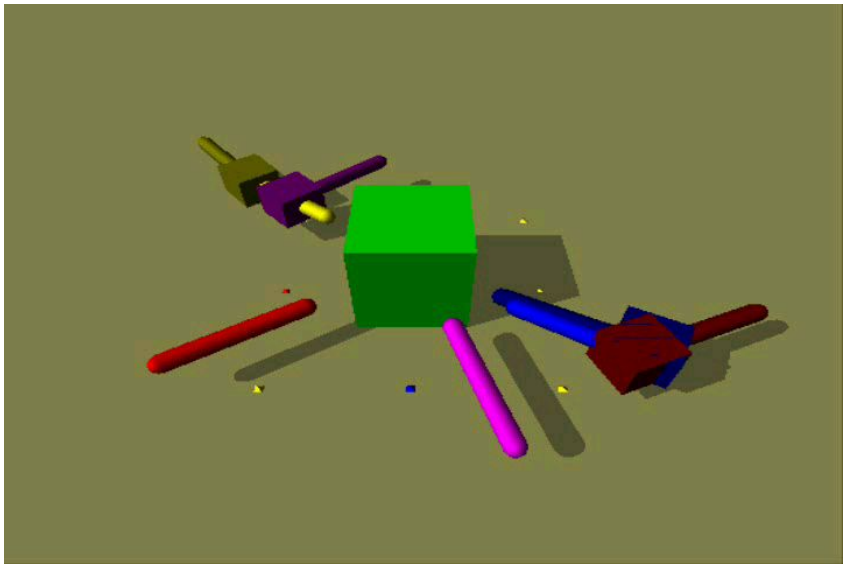
Does not tilt

Lower error

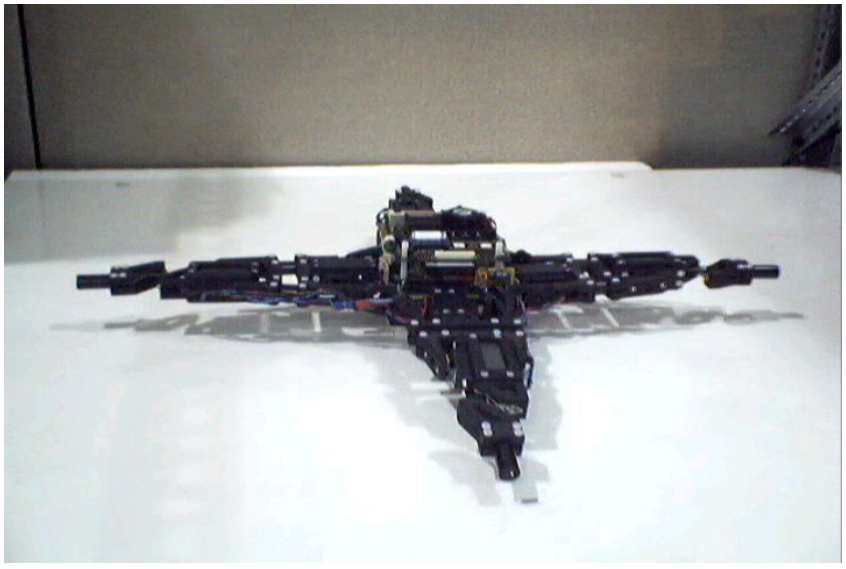
Typical experiment



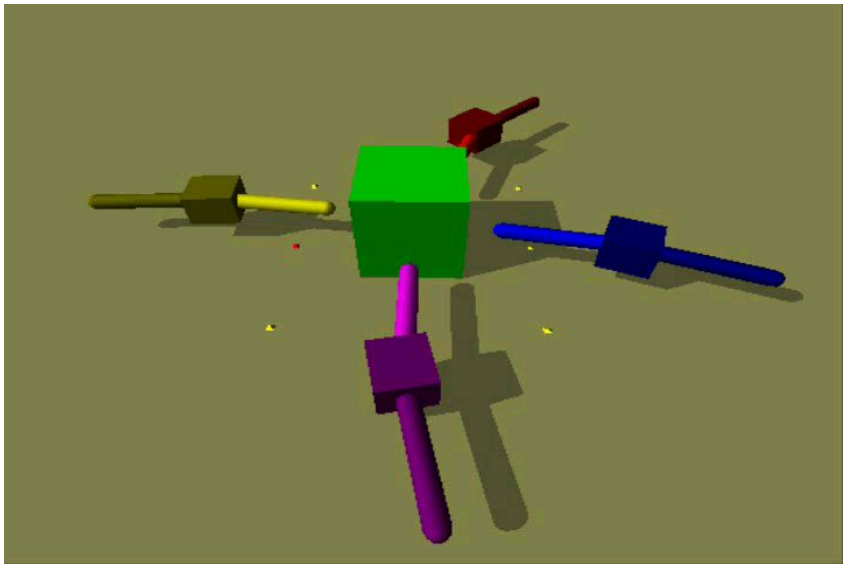
Typical experiment L17



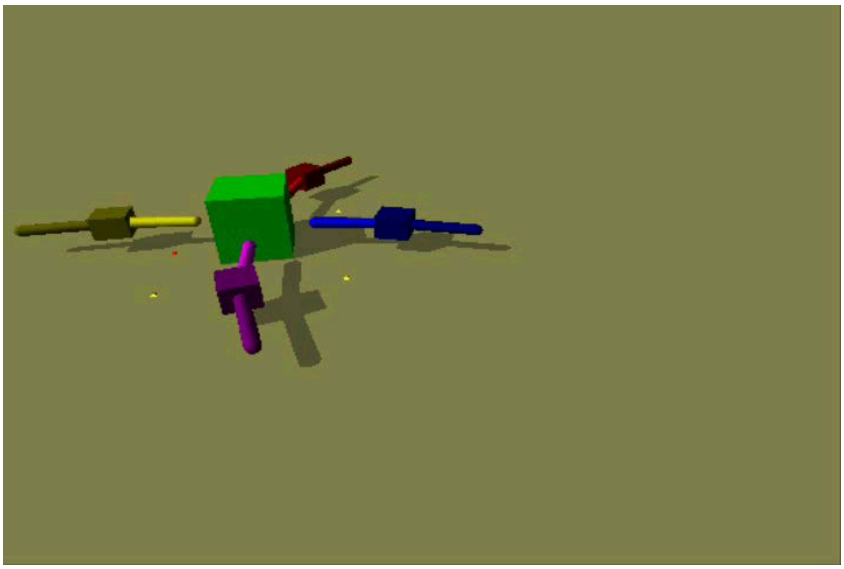
Typical experiment L17



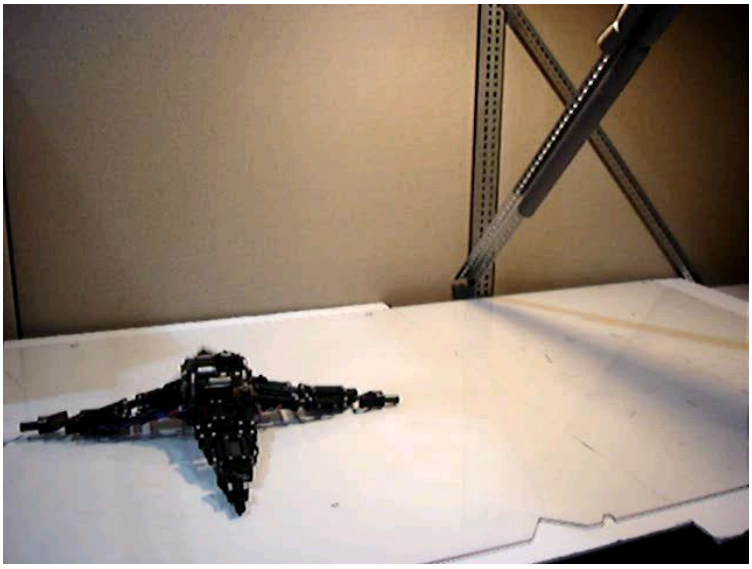
Typical experiment L17



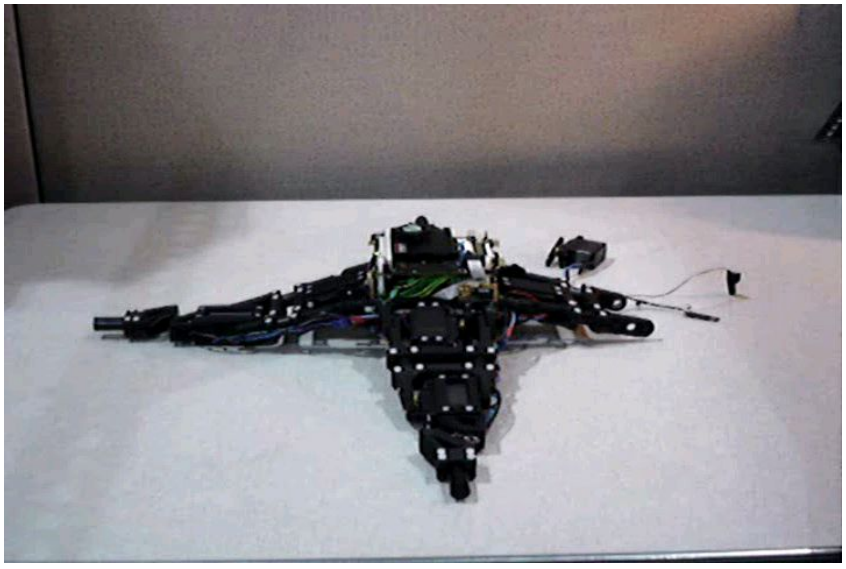
Typical experiment L17



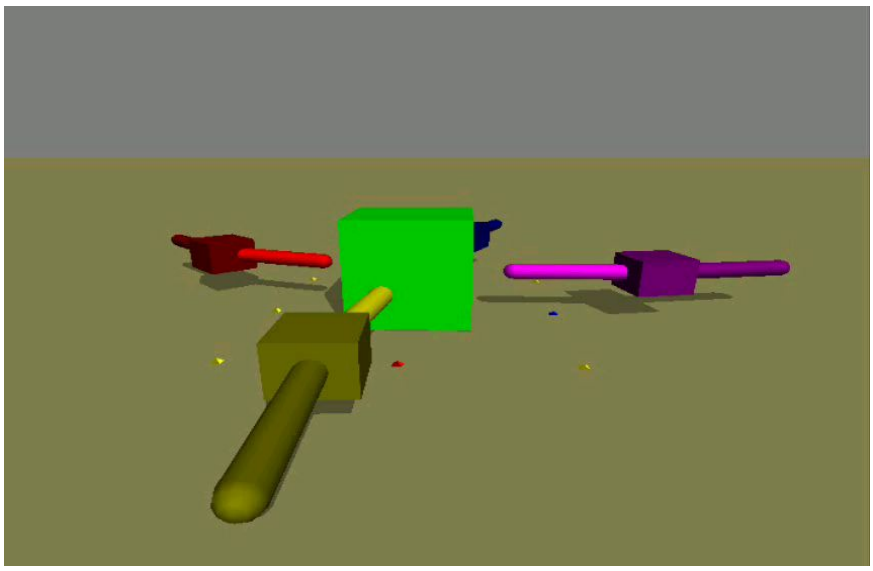
Typical experiment L17

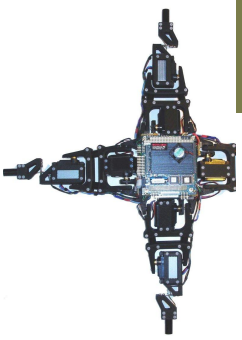


Typical experiment L17

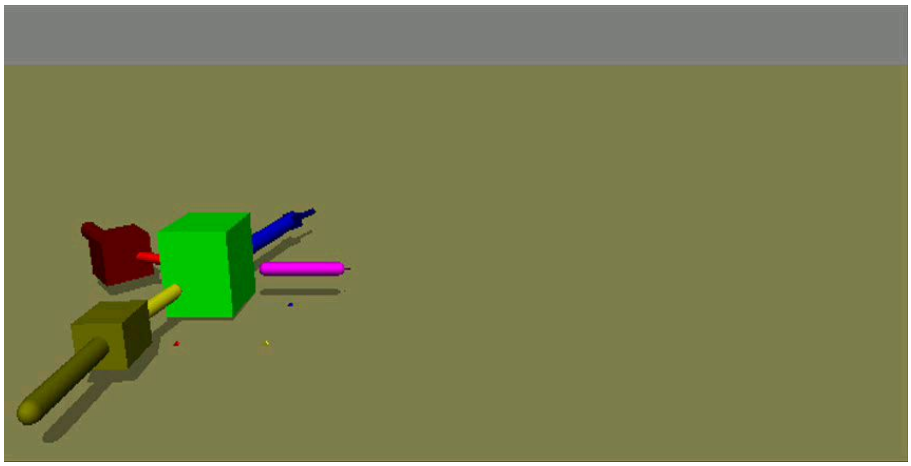


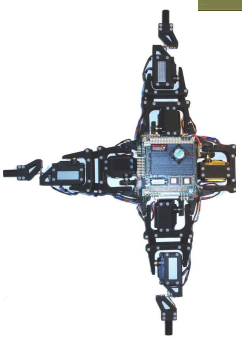
Typical experiment L17



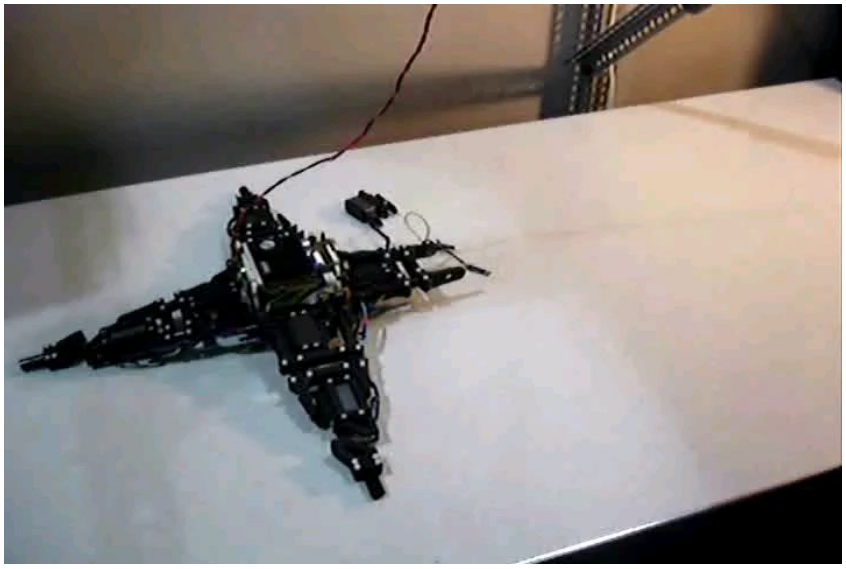


Typical experiment L17





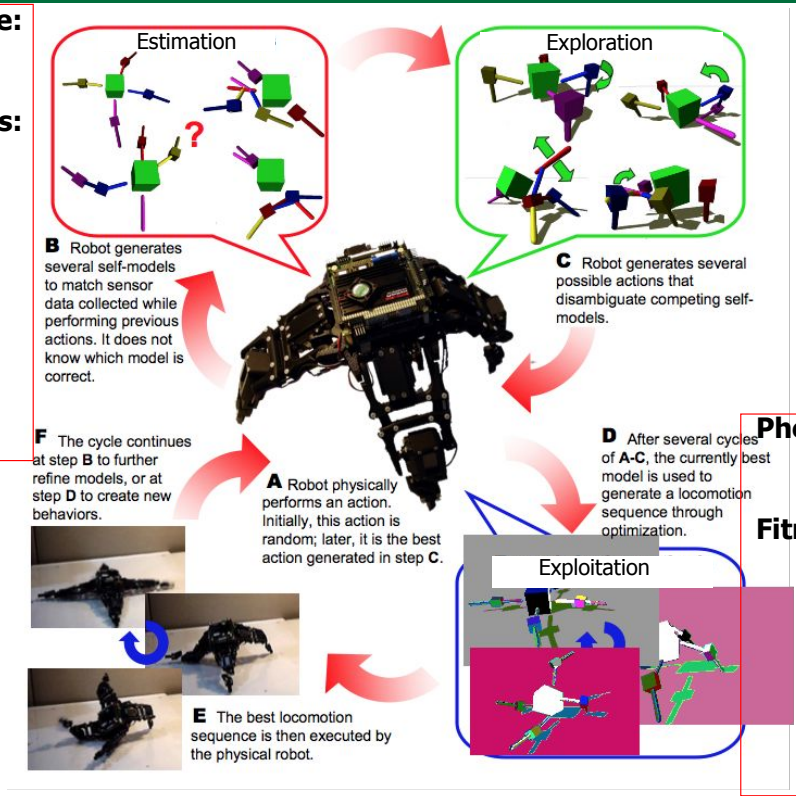
Typical experiment L17



The Estimation-Exploration Algorithm (EEA) applied to a single robot L17

Phenotype:

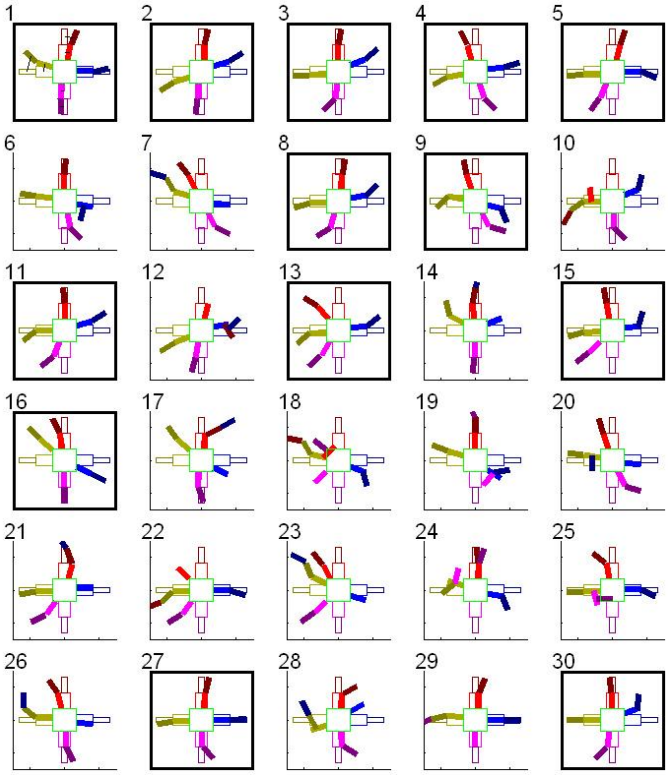
Fitness:



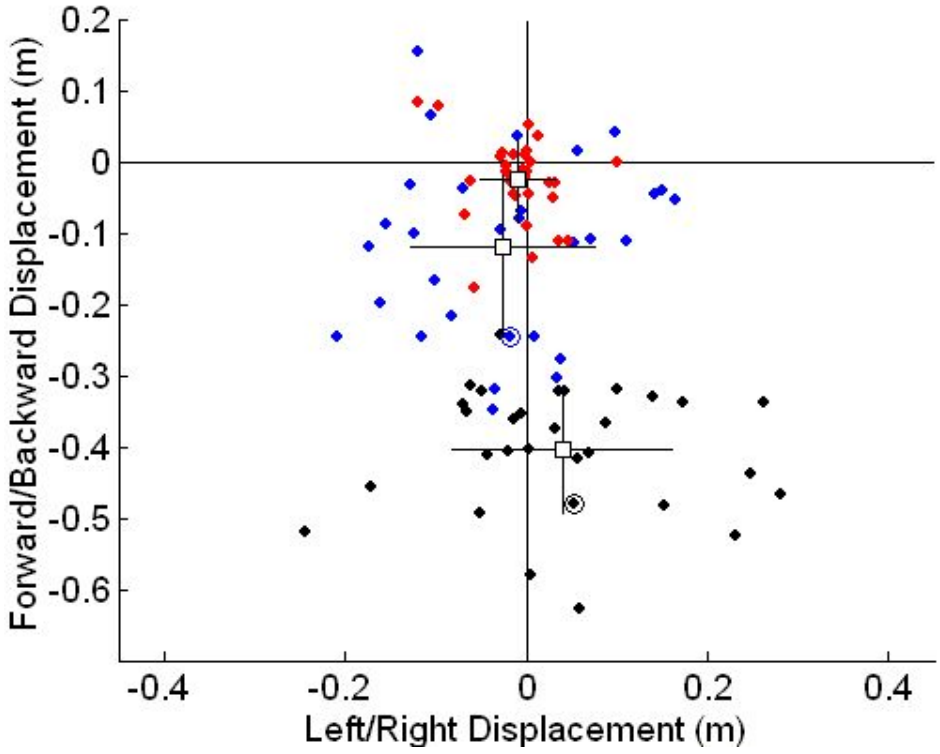
Phenotype:

Fitness:

Intelligent testing: 13 out of 30 runs prodeuce successful models



Generating behaviors using an optimized model



22

Transferability

Question: Could the transferability of a controller be added to the fitness function?

How to define transferability?

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Transferability

Question: Could the transferability of a controller be added to the fitness function?

How to define transferability?

Solution:

Use of two fitness objectives:

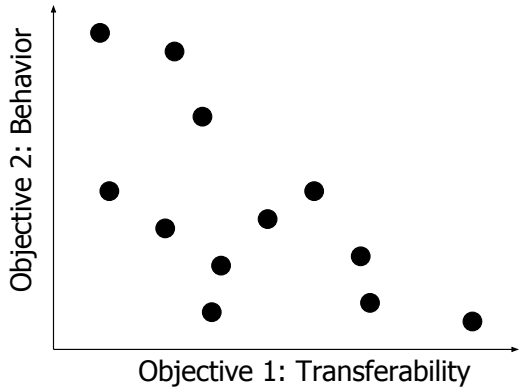
- rewards for the desired behavior

- rewards for similarities between simulated and real behavior

** There is not best way to combine two different fitness to obtain a good behavior.

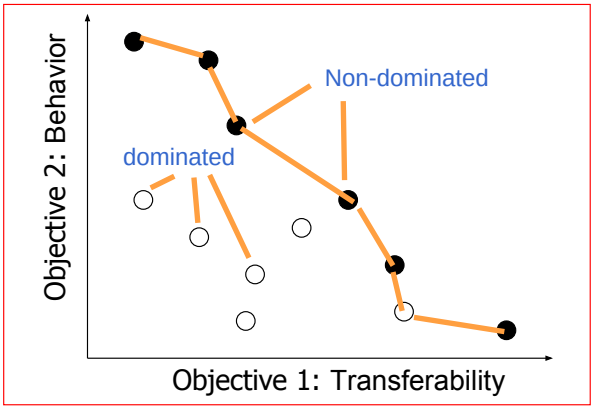
Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Multi-objective optimization (MOO)



Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

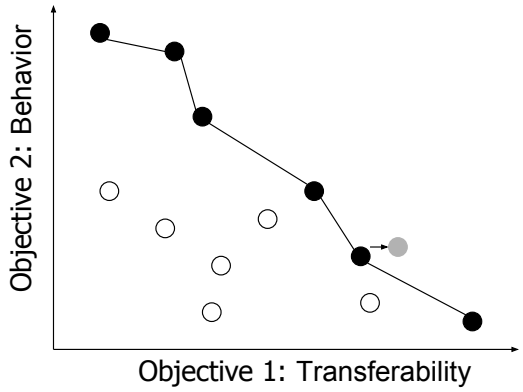
Multi-objective optimization (MOO)



solution s = { dominated : there exists another solution, t, for which all objectives o_t >= all objectives o_s
non-dominated : otherwise

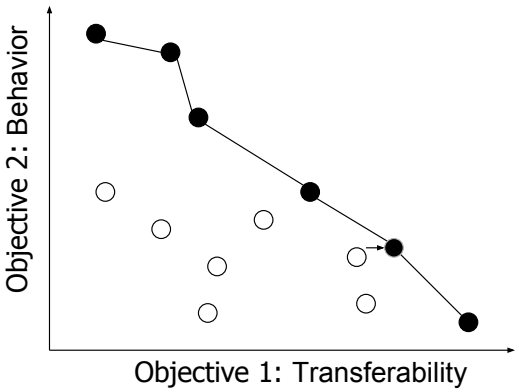
Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Multi-objective optimization (MOO)



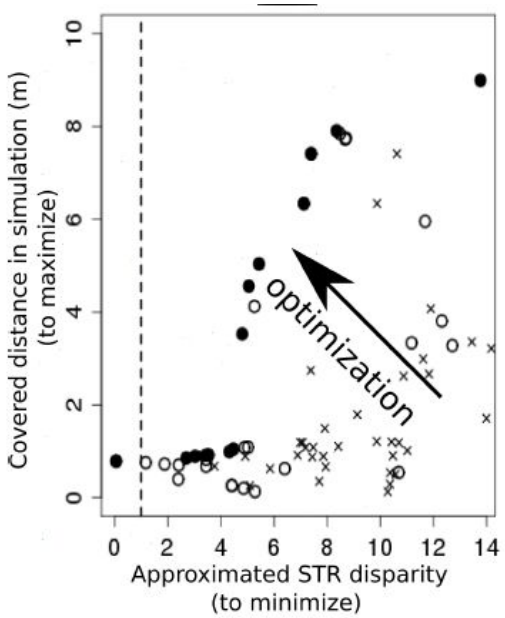
Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Multi-objective optimization (MOO)



Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Multi-objective optimization (MOO)



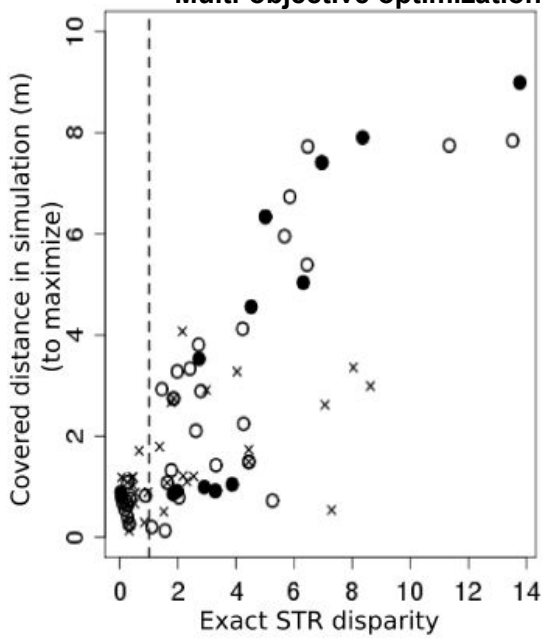
x = controller at generation i

● = non-dominated controller at generation i+1

○ = dominated controller at generation i+1

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Multi-objective optimization (MOO)



Q: Why low distance → low disparity?

x = controller at generation i

● = non-dominated controller at generation i+1

○ = dominated controller at generation i+1

Transferability

Question: Could the transferability of a controller be added to the fitness function?

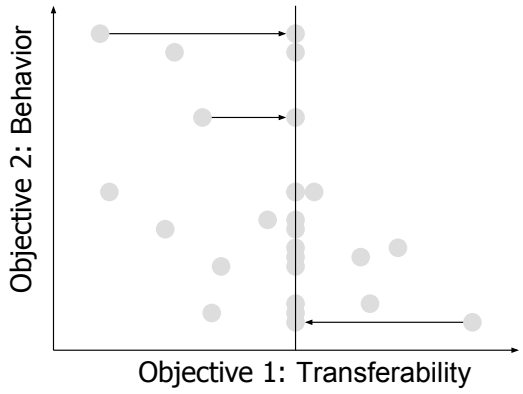
How to define transferability?

Solution: Use two fitness objectives:
One that rewards for the desired behavior,
another that punishes for differences between simulated and real behavior (transferability)

Problem 1: …but to compute transferability for each evolved controller, have to evaluate each on the real robot!

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

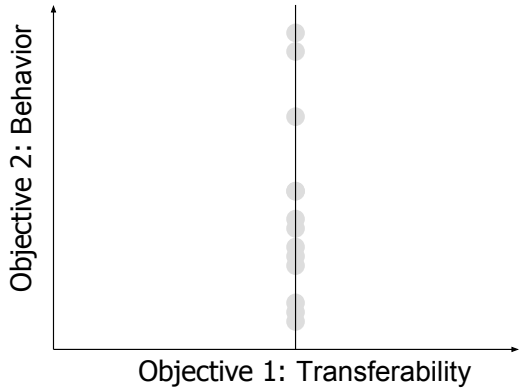
Multi-objective optimization (MOO)



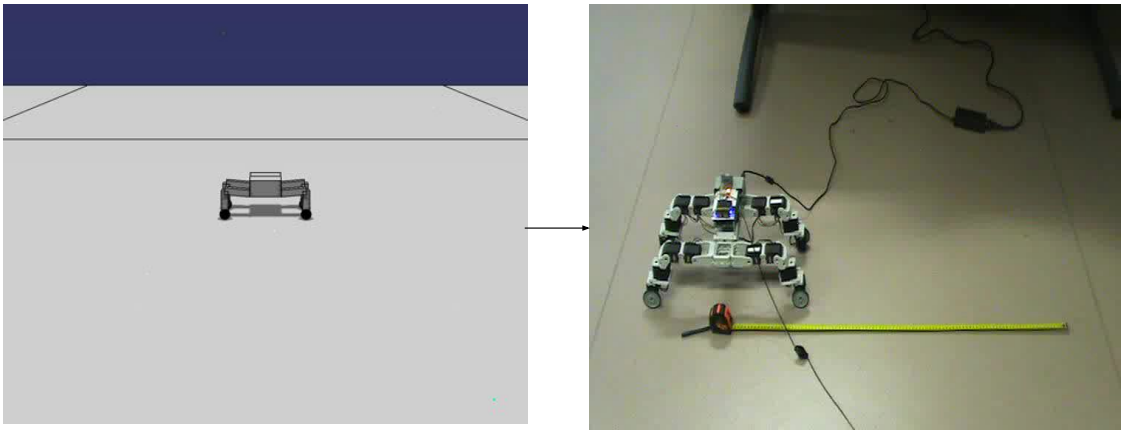
Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

L18

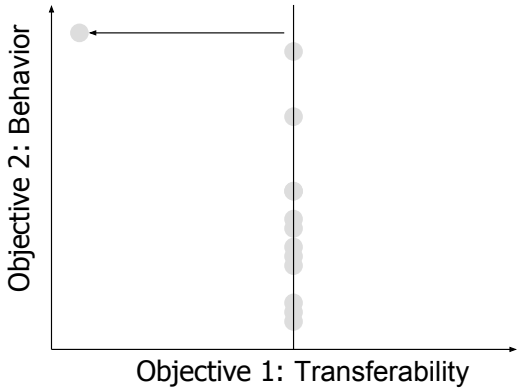
Multi-objective optimization (MOO)



Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

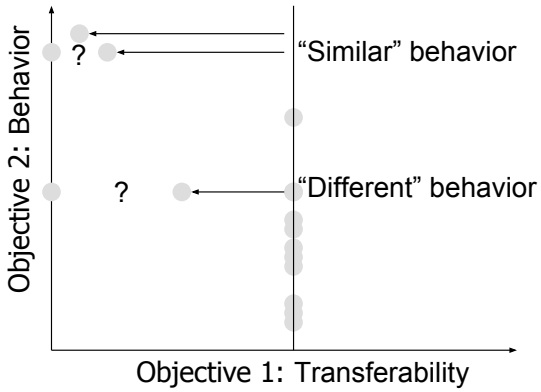


Multi-objective optimization (MOO)



Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Multi-objective optimization (MOO)



What you want to do is sent the controller that is more different

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Transferability

Question:	Could the transferability of a controller be added to the fitness function?
	How to define transferability?
Solution:	Use two fitness objectives: One that rewards for the desired behavior, another that punishes for differences between simulated and real behavior (transferability)
Problem 1:	…but to compute transferability for each evolved controller, have to evaluate each on the real robot!
Solution 1:	Estimate transferability of a controller based on behavioral similarity to a tested controller.

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Transferability

Question: Could the transferability of a controller be added to the fitness function?

How to define transferability?

Solution: Use two fitness objectives:
One that rewards for the desired behavior,
another that punishes for differences between simulated and real behavior (transferability)

Problem 1: …but to compute transferability for each evolved controller, have to evaluate each on the real robot!

Problem 2: …how to measure the similarity between behaviors?

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Transferability

Problem 2: …how to measure the similarity between behaviors?

Solution 2: Define a distance measure between any two behaviors.

Define behavior b as a vector: b = (f1, f2, f3)
f1 = distance traveled by robot
f2 = mean height of the robot during travel
f3 = final orientation of the robot

You can compute the behaviorar distance between two controllers by taking the euclidean distance between the two vectors.

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Transferability

Problem 2: …how to measure the similarity between behaviors?

Solution 2: Define a distance measure between any two behaviors.

Define behavior b as a vector: b = (f1, f2, f3)
f1 = distance traveled by robot
f2 = mean height of the robot during travel
f3 = final orientation of the robot

Question: How to define the transferability of controller i?

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

L18

Transferability

Problem 2: …how to measure the similarity between behaviors?

Solution 2: Define a distance measure between any two behaviors.

Define behavior b as a vector: b = (f1, f2, f3)
f1 = distance traveled by robot
f2 = mean height of the robot during travel
f3 = final orientation of the robot

Question: How to define the transferability of controller i?
Solution: The smaller the distance between the two behavioral vectors for a controller the more transferable it is

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Transferability

Problem 1: …but to compute transferability for each evolved controller, have to evaluate each on the real robot!
Solution 1: Estimate transferability of a controller based on behavioral similarity to a tested controller.

Estimated transferability of controller c =

Use the behaviorar vectors for controllers.
Compute the weighted average by taking the transferability of a controller and divided it by the behavioral distance between the controller in simulation and a controller previously sent.
Controller that are closer will have a larger influne in the estimation of the transferability

CT = set of already-transferred controllers

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Transferability

Problem 1: …but to compute transferability for each evolved controller, have to evaluate each on the real robot!
Solution 1: Estimate transferability of a controller based on behavioral similarity to a tested controller.
Problem 2: …how to measure the similarity between behaviors?
Solution 2: Define a distance measure between any two behaviors.

Problem 3: Which controller to try in reality next?

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

Transferability

Problem 1: …but to compute transferability for each evolved controller, have to evaluate each on the real robot!
Solution 1: Estimate transferability of a controller based on behavioral similarity to a tested controller.
Problem 2: …how to measure the similarity between behaviors?
Solution 2: Define a distance measure between any two behaviors.

Problem 3: Which controller to try in reality next?
Solution 3: The one most "different" from those in the population.

Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

L18

Transferability

Problem 1: …but to compute transferability for each evolved controller, have to evaluate each on the real robot!
Solution 1: Estimate transferability of a controller based on behavioral similarity to a tested controller.
Problem 2: …how to measure the similarity between behaviors?
Solution 2: Define a distance measure between any two behaviors.

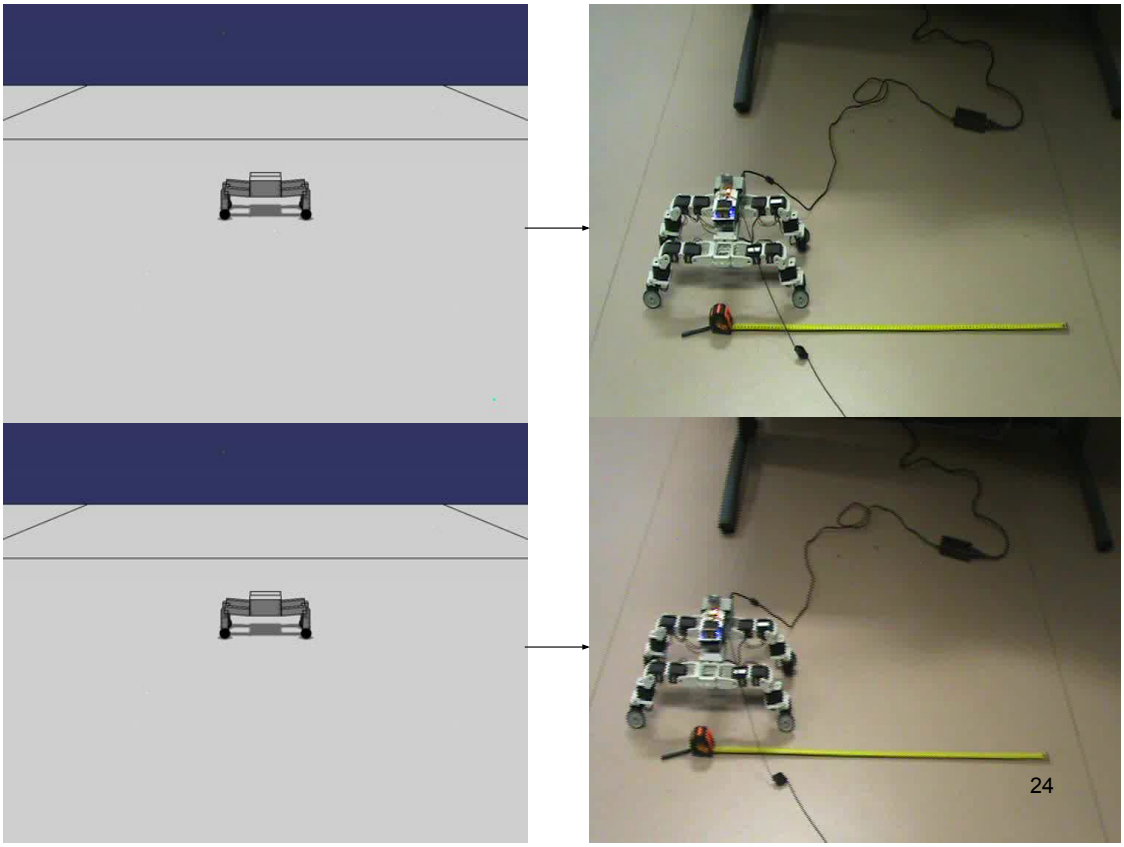
Problem 3: Which controller to try in reality next?
Solution 3: The one most "different" from those in the population.

diversity(c) = min b_dist(c,c_i)

Look for the minimun of the current controller from all the controller that have already been sent across the gap.

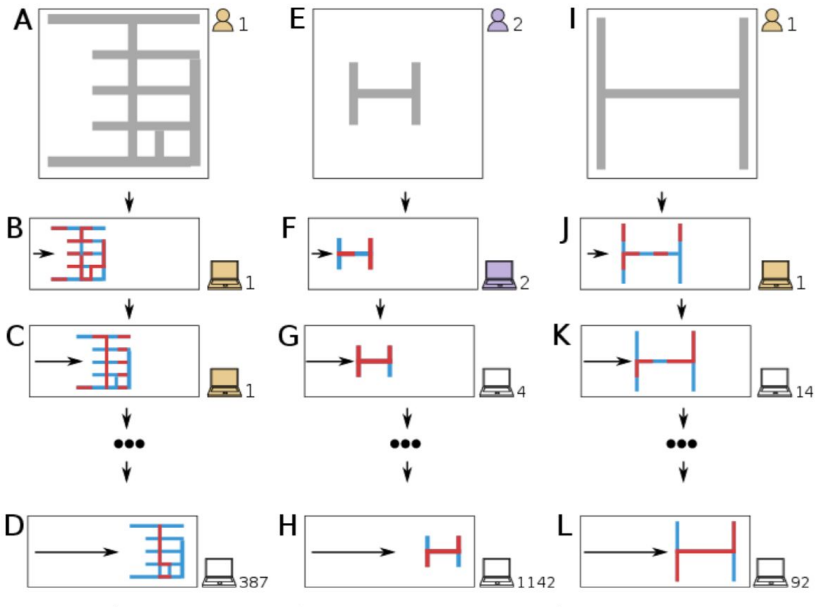
Zero means that the behavior of the controller is similar

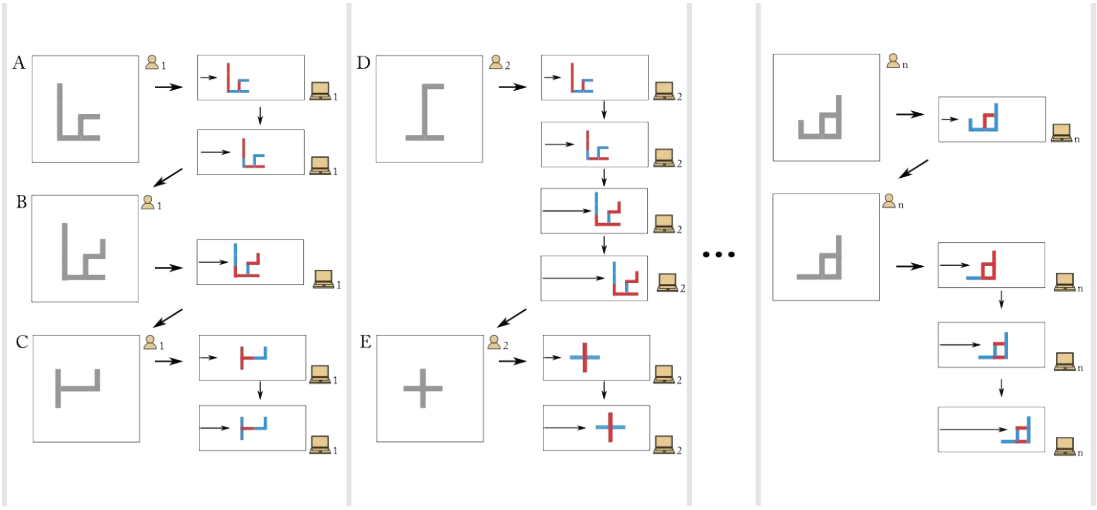
Koos, S. and Mouret, J.-B. and Doncieux, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.
GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.

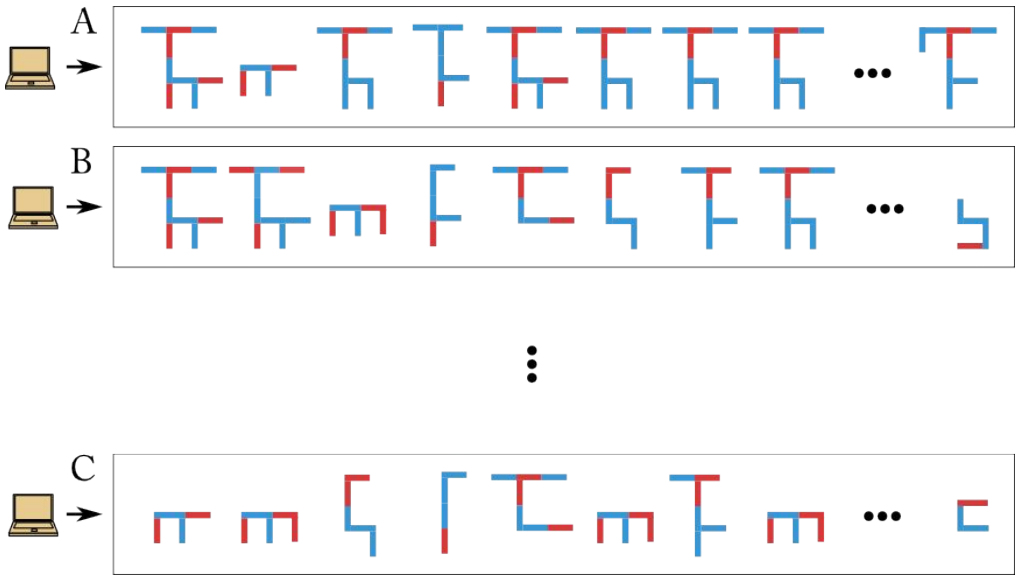


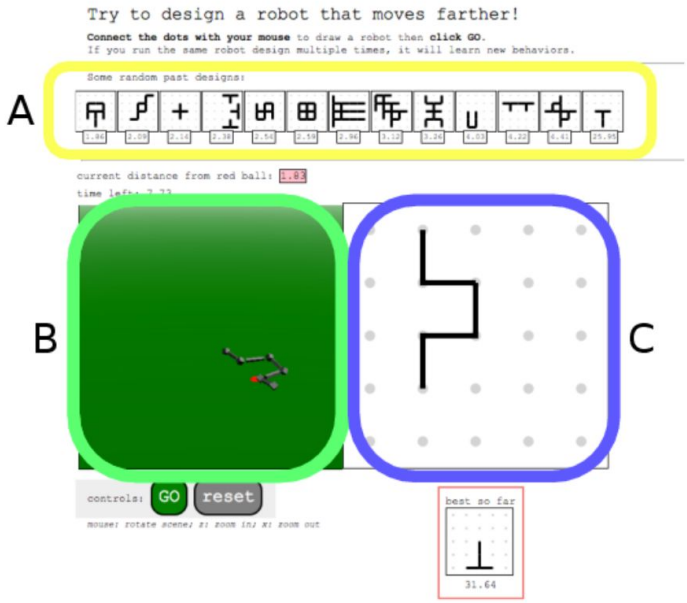
Crowdsourcing Robotics:

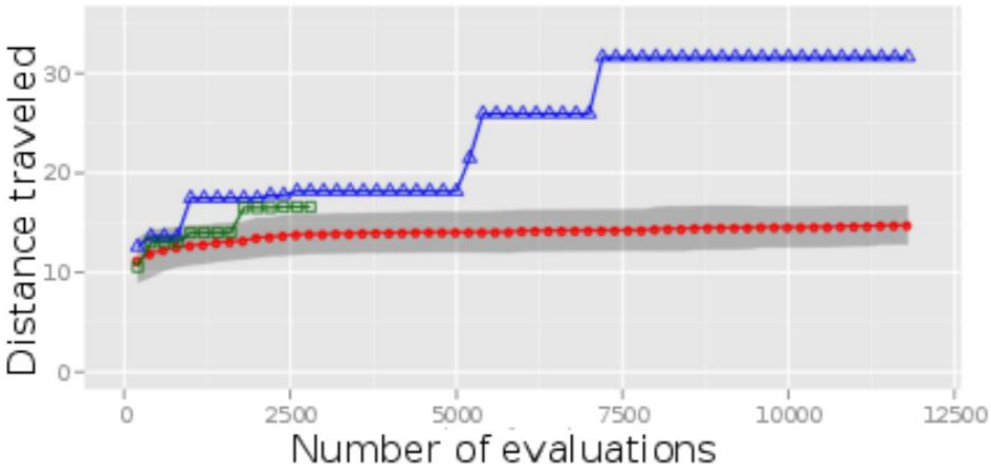
The “DotBot” Project

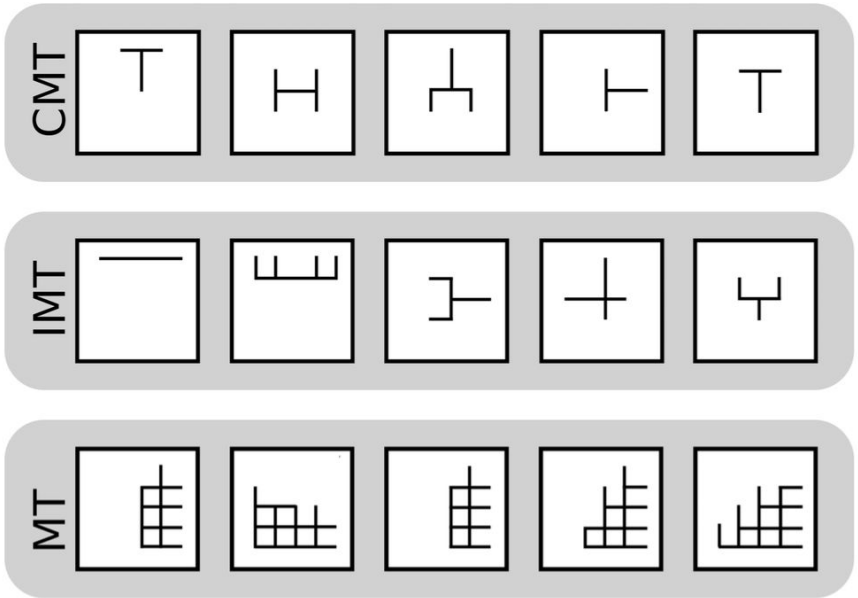

A
1
E
2
I
1
B
1
F
2
J
1
C
1
G
4
K
14
D
387
H
1142
L
92

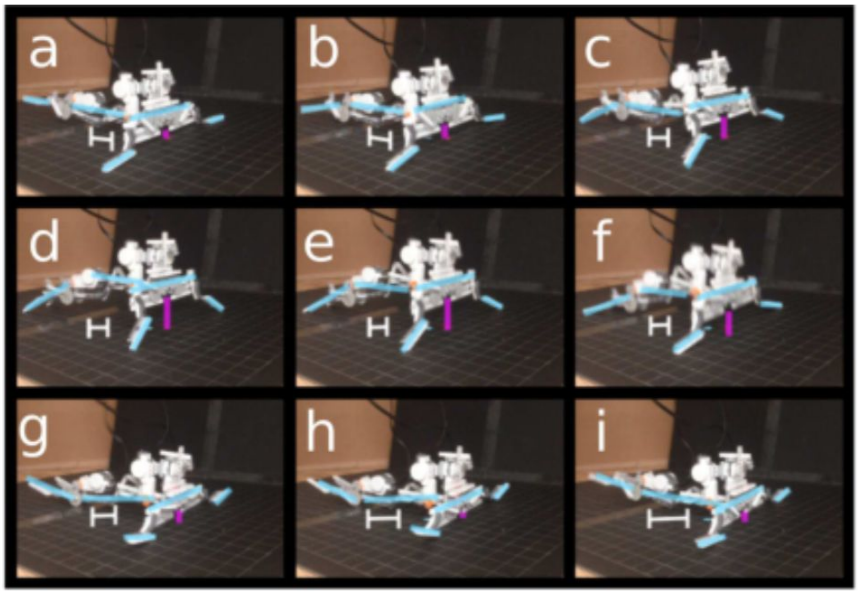

A
B
C
D
E

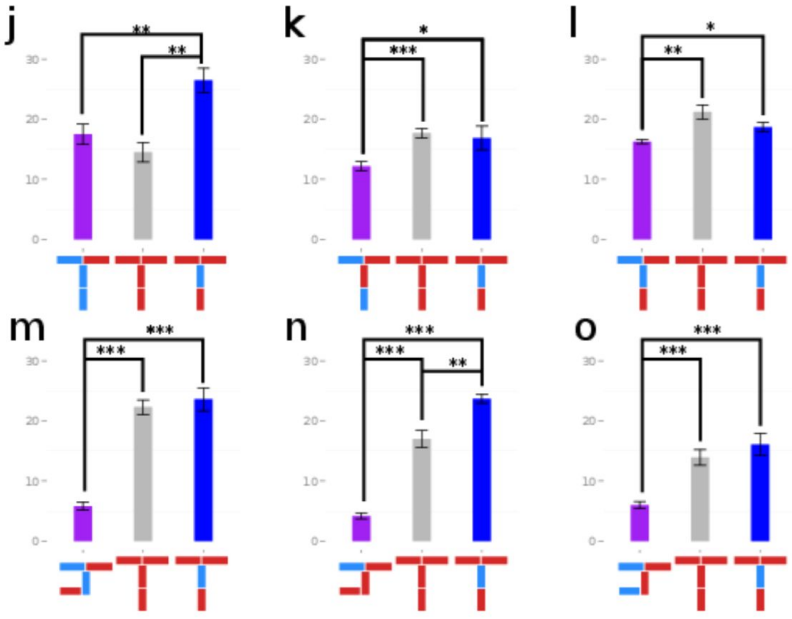

A
B
C


Try to design a robot that moves farther!
Connect the dots with your mouse to draw a robot then click GO.
If you run the same robot design multiple times, it will learn new behaviors.
Some random past designs:
A
current distance from red ball:
B
C
controls:
GO
reset
best so far
31.64


Distance traveled
30
20
10
0
0
2500
5000
7500
10000
12500
Number of evaluations


CMT
IMT
MT


a
b
c
d
e
f
g
h
i


j
k
l
m
n
o
**
**
*
***
*
**
***
***
***
***
**
***
***
***
30
20
10
0

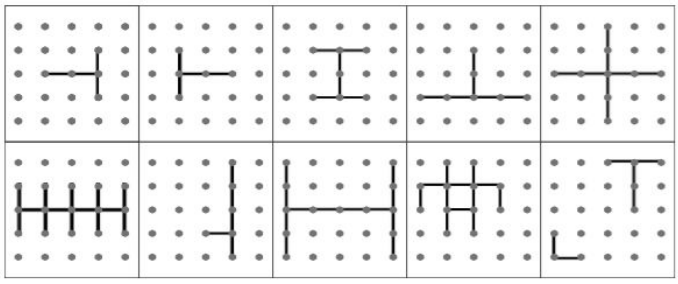
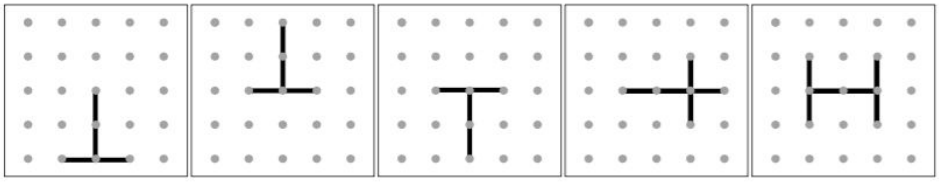
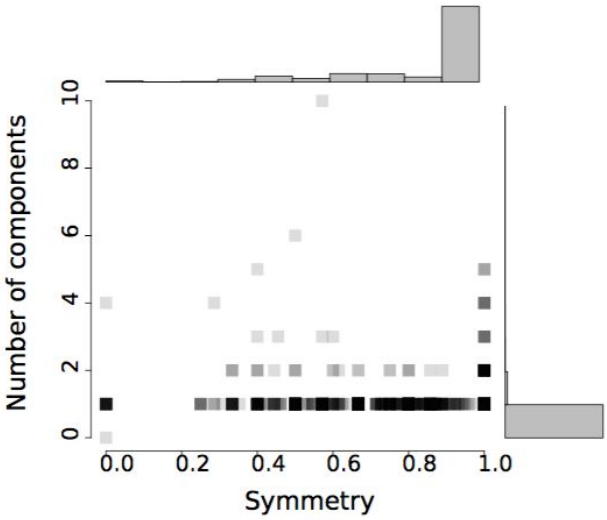
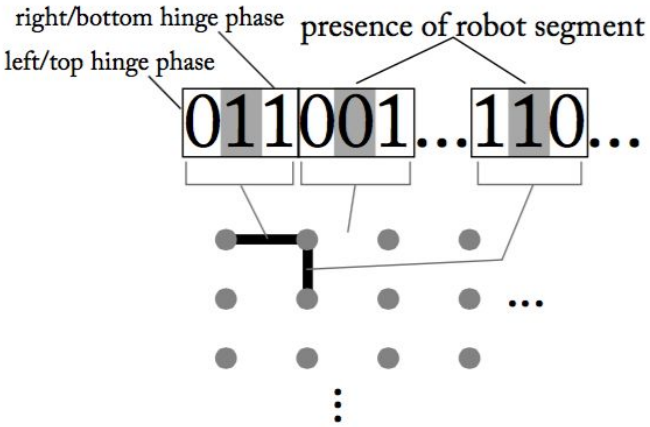
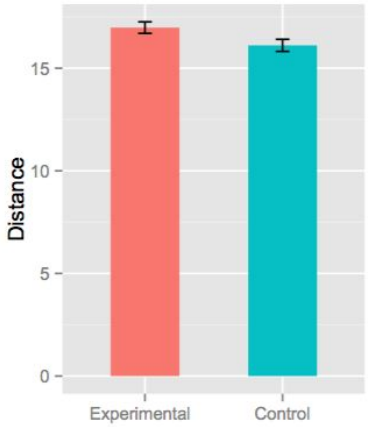


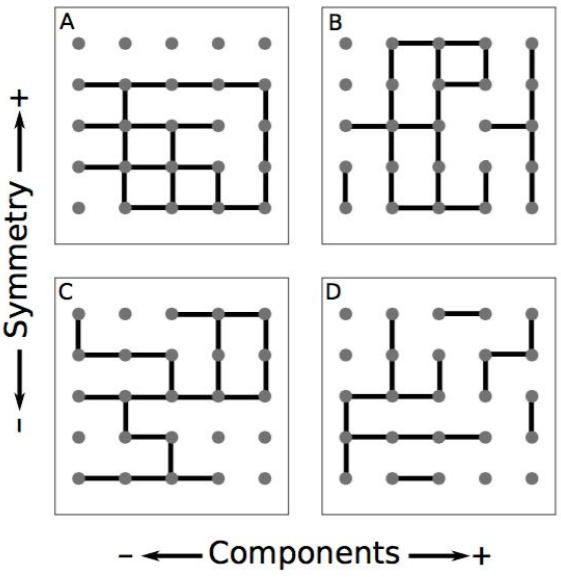
Figure 4.3: Ten examples of user designs of robot bodies.




Number of components
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0
Symmetry


right/bottom hinge phase
left/top hinge phase
presence of robot segment
011001...110...


Distance
15
10
5
0
Experimental
Control


A
B
C
D
Symmetry
+
-
Components
-
+

It uses genetic programming

Variable	Symbol
Maximal matching	M_{max}
Number of connected components	c
Maximum degree	D_{max}
Minimum degree	D_{min}
Number of limbs	L
Not a chordal graph	C
Symmetry	S
Number of segments	G
Average Degree	D_{ave}
Average degree connectivity	D_{con}
Average clustering	T_{ave}

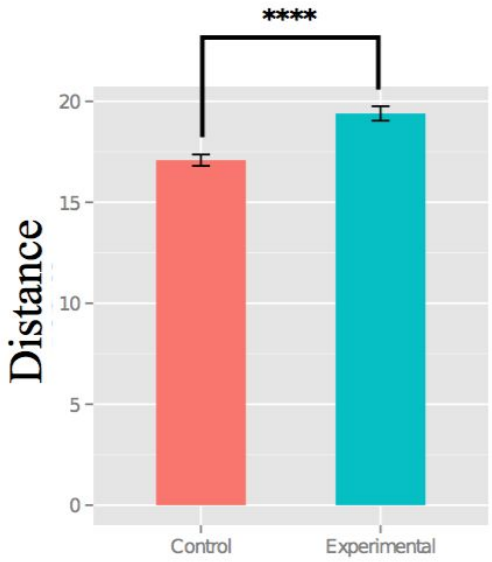
(a) Variables

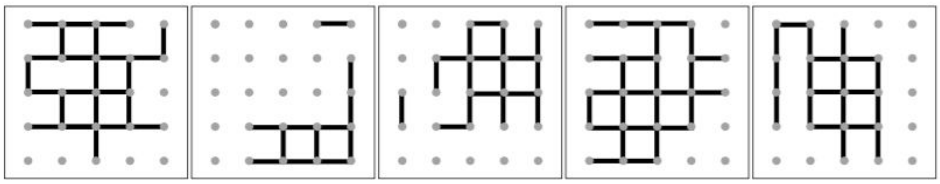
Function Name	Symbol
Addition	$+$
Subtraction	$-$
Multiplication	$\cdot$
Division	$/$
Logistic function	$\sigma()$
Indicator function	$I()$
Cosine	$cos()$
Sine	$sin()$
Tangent	$tan()$
Exponential	$exp()$
Natural Logarithm	$log()$
Power	x^y
Square root	$\sqrt{}$
Gaussian	$G()$
Less than or equal	$\leq$
Greater than or equal	$\geq$

(b) Functions

Equations produced by genetic programming that replicated the human design and optimiza them

	Solution	Error
1*	$I(D_{max} \leq L) \cdot min(G(D_{conn}), S) + 0.177min(c + 6.800D_{conn}, D_{max}^{4.510-max(S,log(L))})$	0.771
2	$0.0.041 \cdot c \cdot S \cdot I(L \geq 0.044 \cdot c \cdot D_{max}) + min(c^{0.350}, D_{max})$	0.773
3	$min(D_{max} \cdot \sigma(M_{max}), 2.187) - 0.085 \cdot M_{max}$	0.804
4	$min(3.150, D_{max} + 0.170 \cdot M_{max}) - 0.144 \cdot M_{max}$	0.806
5	$\sqrt{D_{max}} - (0.003 \cdot N \cdot c^2)^S \cdot cos(D_{max})$	0.786
6	$min(3.054, D_{max} \cdot min(M_{max}, 1.325)) - 0.125 \cdot M_{max}$	0.807
7	$4.374 \cdot c \cdot min(0.033, D_{con}) + I(D_{max} \geq 3) + min(S, I(D_{max} \leq L))$	0.781
8	$0.053 \cdot c \cdot S \cdot I(L \geq 0.044 \cdot c \cdot D_{max}) + min(\log c + S, D_{max})$	0.773
9	$D_{con} + S \cdot I(L \geq D_{max}) + 0.196 \cdot min(3.471^{D_{max}}, c) - I(L \geq G)$	0.771
10	$2.819 \cdot \sigma(S) \cdot min(3.180, D_{max}) - 0.141 \cdot M_{max} \cdot min(M_{max}, D_{max}) - 2.960 \cdot I(3.887 - L \geq M_{max})$	0.779


****
Distance
20
15
10
5
0
Control
Experimental



A potentially scalable solution to the symbol grounding problem.

Josh Bongard

Morphology, Evolution and Cognition Laboratory
University of Vermont
www.cs.uvm.edu/∼jbongard

April 5, 2016

The Chinese Room Problem.

Searle, John (1980). "Minds, Brains and Programs", *Behavioral and Brain Sciences* 3(3): 417–457

The Cyc Project.

Douglas Lenat (1984—present)

The concept names in Cyc are known as *constants*. Constants start with an optional "#$" and are case-sensitive. There are constants for:

- Individual items known as *individuals*, such as #$BillClinton or #$France.
- *Collections*, such as #$Tree-ThePlant (containing all trees) or #$EquivalenceRelation (containing all equivalence relations). A member of a collection is called an *instance* of that collection.
- *Functions*, which produce new terms from given ones. For example, #$FruitFn, when provided with an argument describing a type (or collection) of plants, will return the collection of its fruits. By convention, function constants start with an upper-case letter and end with the string "Fn".
- *Truth Functions* which can be applied to one or more other concepts and return either true or false. For example, #$siblings is the sibling relationship, true if the two arguments are siblings. By convention, truth function constants start with a lower-case letter. Truth functions may be broken down into logical connectives (such as #$and, #$or, #$not, #$implies), quantifiers (#$forAll, #$thereExists, etc.) and predicates.

The most important predicates are #$isa and #$genls. The first one describes that one item is an instance of some collection, the second one that one collection is a subcollection of another one. Facts about concepts are asserted using certain CycL *sentences*. Predicates are written before their arguments, in parentheses:

```
(#$isa #$BillClinton #$UnitedStatesPresident)
```

"Bill Clinton belongs to the collection of U.S. presidents" and

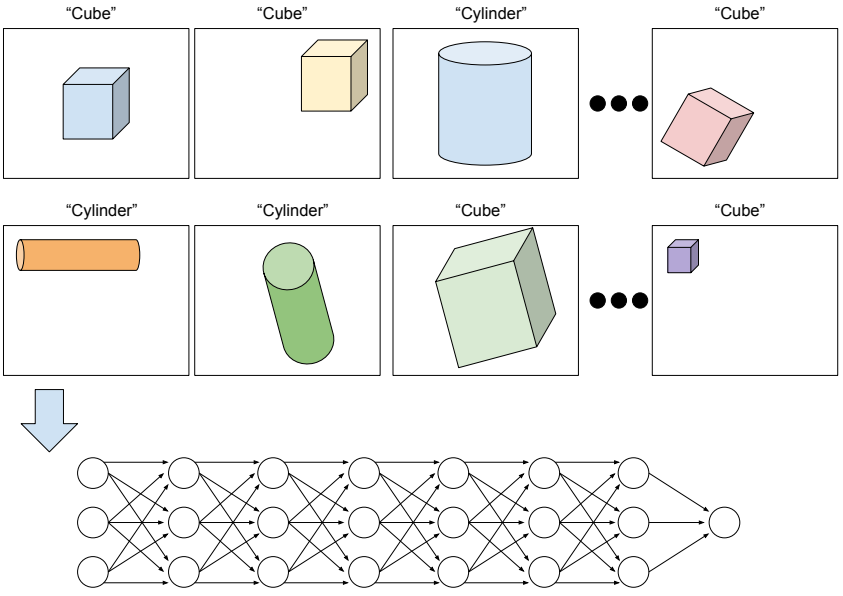
```
(#$genls #$Tree-ThePlant #$Plant)
```

"All trees are plants".

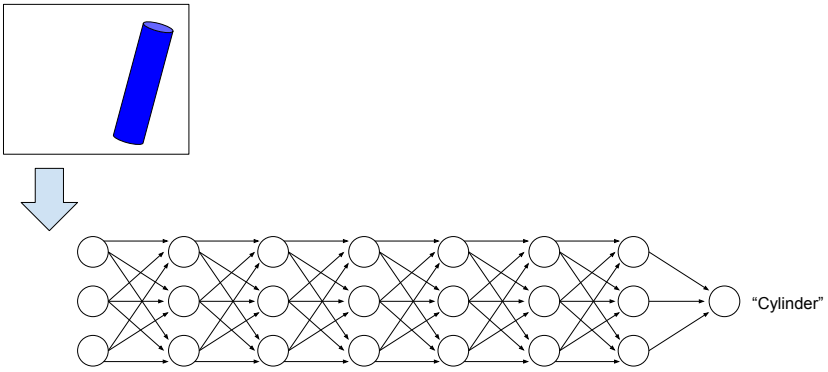
```
(#$capitalCity #$France #$Paris)
```

"Paris is the capital of France."

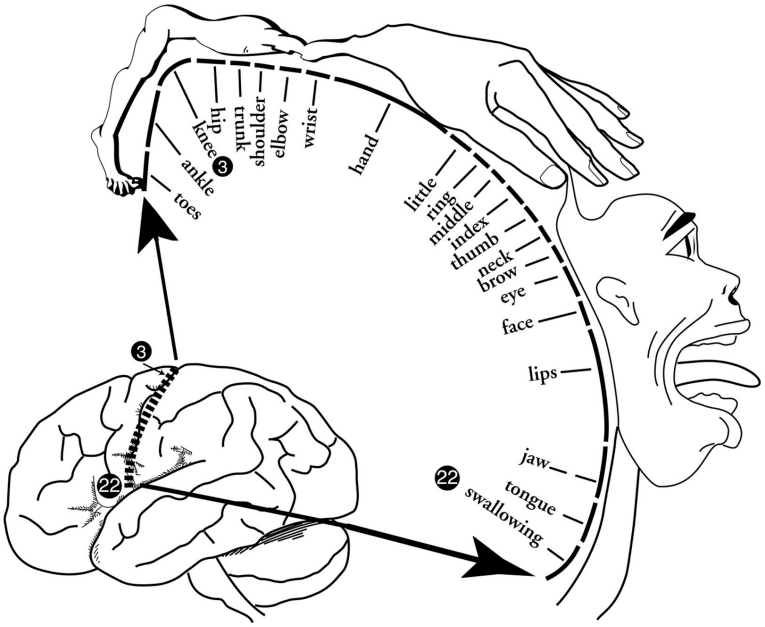
Non-embodied categorization.



Non-embodied categorization.

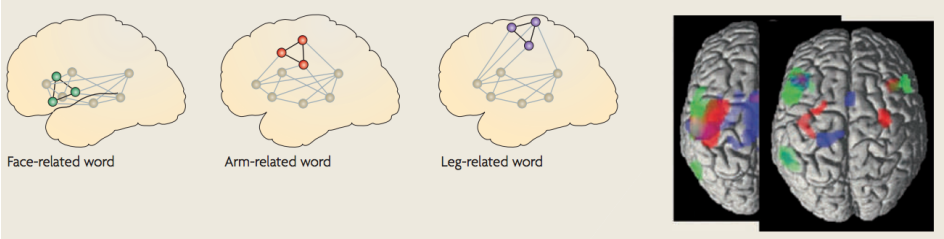


The brain's motor strip.



Motoric words activate the brain's motor strip.

Pulvermüller & Fadiga, (2010). Active perception: sensorimotor circuits as a cortical basis for language. *Nature Reviews Neuroscience*, 11(5), 351—360.

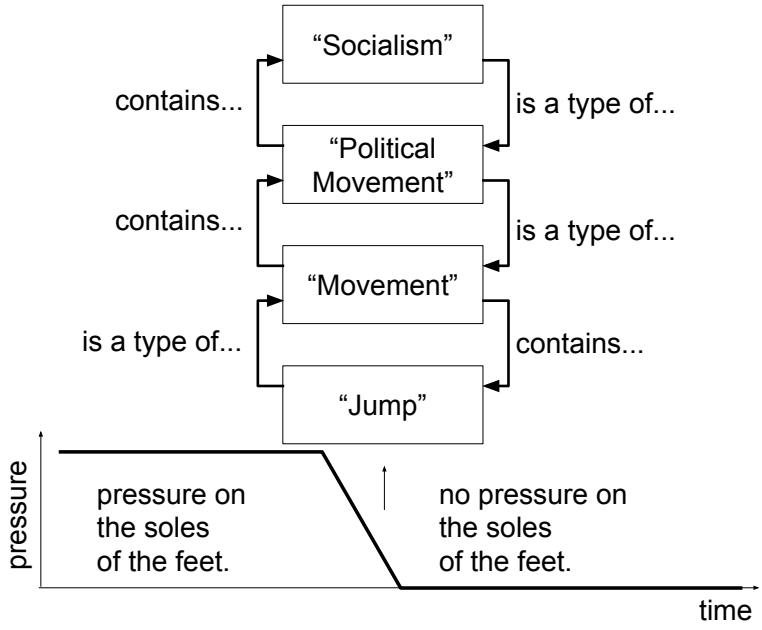


- Face-related words (green): "talk", "lick".
- Arm-related words (red): "grasp", "pick".
- Leg-related words (blue): "walk", "kick".

George Lakoff: embodied metaphors.

- Don't **jump** to conclusions.
- Don't **look back** in anger.
- ...

Symbols grounded...?

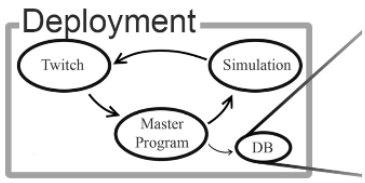


Grounding symbols using *Twitch Plays Robotics.*

Act. Observe. Learn. Predict.

Grounding symbols using *Twitch Plays Robotics*.

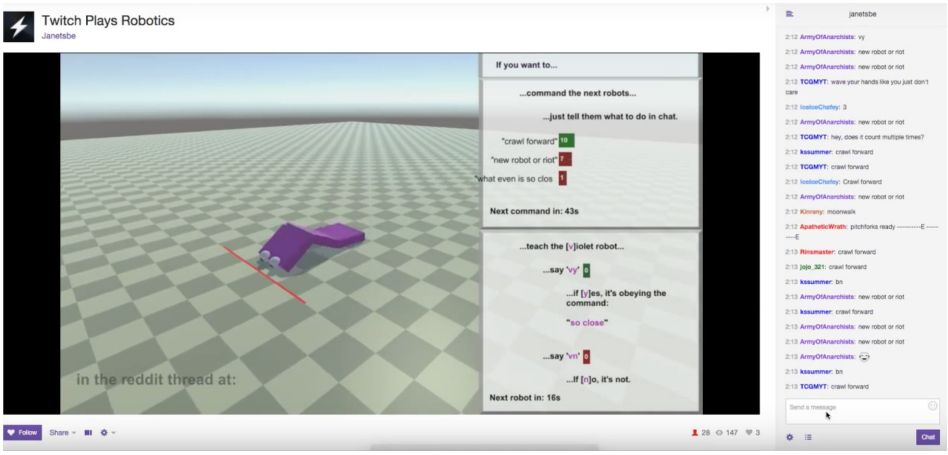
Act. Observe. Learn. Predict.



Simulation: Robots simulated in Unity Physics Engine.
Twitch: Video captured from simulation and fed to Twitch.
Master Program:controllers-->simulation; crowd response -->DB

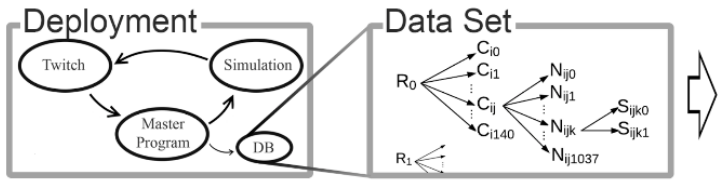
Grounding symbols using *Twitch Plays Robotics*.

Act. Observe. Learn. Predict.



Grounding symbols using *Twitch Plays Robotics*.

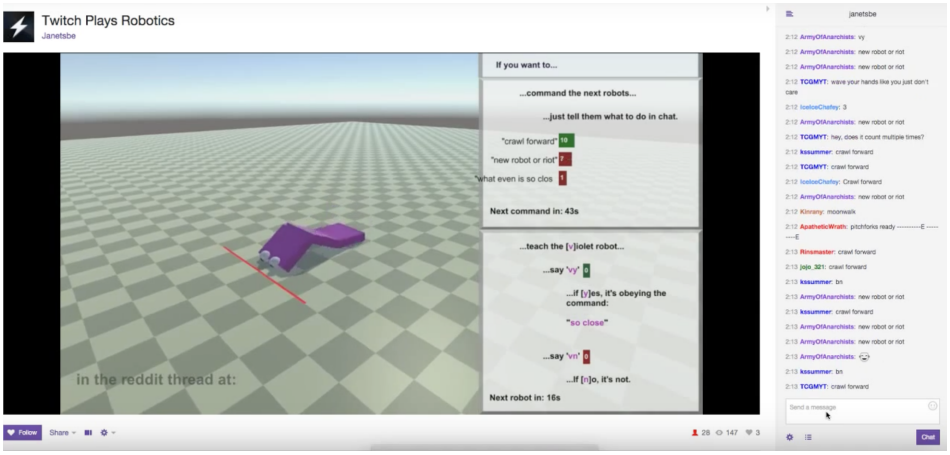
Act. **Observe.** Learn. Predict.



Simulation: Robots simulated in Unity Physics Engine.
Twitch: Video captured from simulation and fed to Twitch.
Master Program: controllers-->simulation; crowd response -->DB
$\mathbf{R_i}$: Robot i.
$\mathbf{C_{ij}}$: Command j issued to robot i.
$\mathbf{N_{ijk}}$: Controller k evaluated under command j.
$\mathbf{S_{ijk0}}$: Negative reinforcement signals issued under controller k.
$\mathbf{S_{ijk1}}$: Positive reinforcement signals issued under controller k.

Grounding symbols using *Twitch Plays Robotics*.

Act. **Observe.** Learn. Predict.



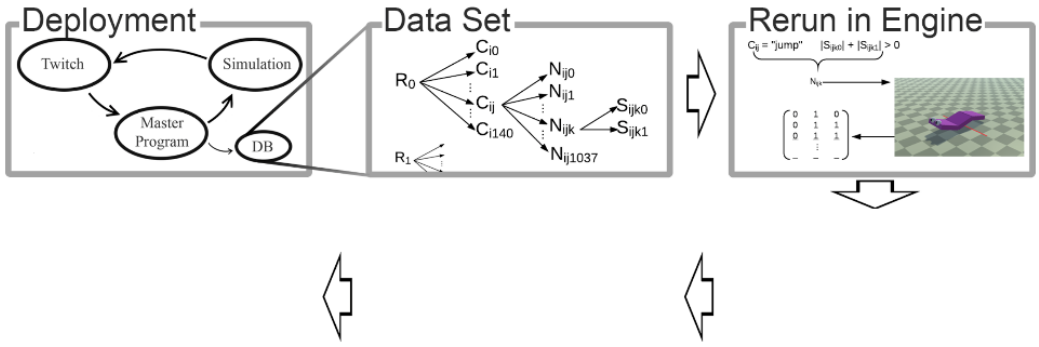
Grounding symbols using *Twitch Plays Robotics*.

Act. **Observe.** Learn. Predict.

General Figures	**Values**
Subjects	424
Robot evaluations sent	57,108
Robot evaluations observed	$\geq$ 6,388
Subject inputs	16,449
Commands	**Values**
Candidate commands entered	8943
Candidate commands/subject	$\approx$ 21.1
Distinct commands issued	266
Most frequently issued commands	jump (385) walk forward (58) move forward (41) run (26) crawl forward (20).
Reinforcement signals	**Values**
Reinforcement signals entered	7503
Mean reinforcement signals/eval.	$\approx$ 1.18
Mean reinf. signals/subject	$\approx$ 17.7
Proportion of positive reinforcement	$o = 0.28$

Grounding symbols using *Twitch Plays Robotics*.

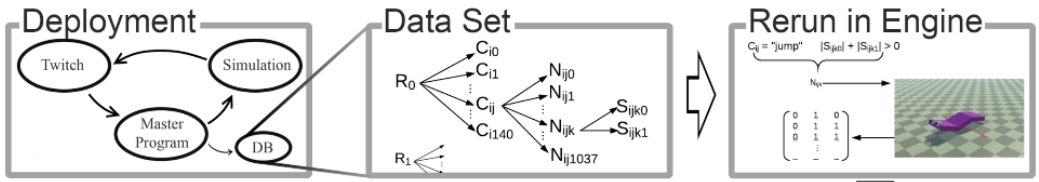
Act. Observe. **Learn.** Predict.



Simulation: Robots simulated in Unity Physics Engine.
Twitch: Video captured from simulation and fed to Twitch.
Master Program: controllers-->simulation; crowd response -->DB
R_i: Robot i.
C_{ij}: Command j issued to robot i.
N_{ijk}: Controller k evaluated under command j.
S_{ijk0}: Negative reinforcement signals issued under controller k.
S_{ijk1}: Positive reinforcement signals issued under controller k.

Grounding symbols using *Twitch Plays Robotics.*

Act. Observe. **Learn.** Predict.

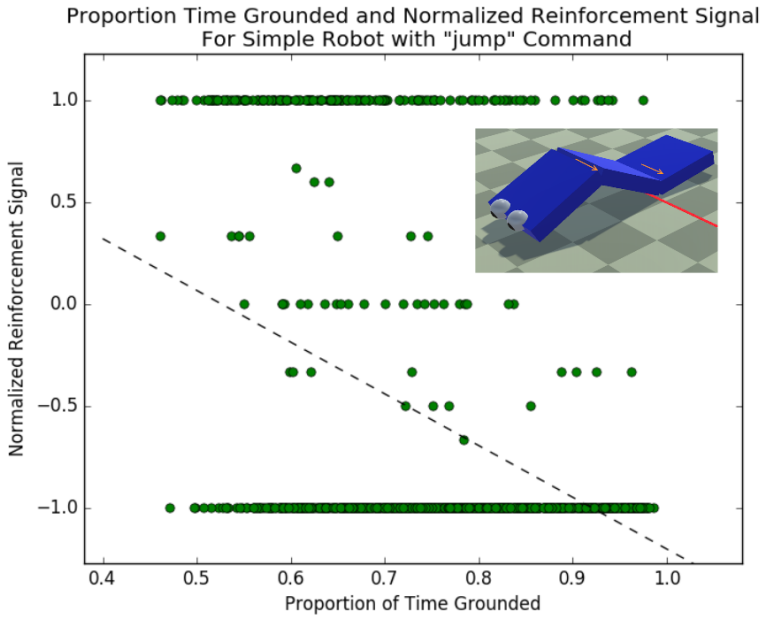


Simulation: Robots simulated in Unity Physics Engine.
Twitch: Video captured from simulation and fed to Twitch.
Master Program: controllers-->simulation; crowd response -->DB
$\mathbf{R_i}$: Robot i.
$\mathbf{C_{ij}}$: Command j issued to robot i.
$\mathbf{N_{ijk}}$: Controller k evaluated under command j.
$\mathbf{S_{ijk0}}$: Negative reinforcement signals issued under controller k.
$\mathbf{S_{ijk1}}$: Positive reinforcement signals issued under controller k.

$\mathbf{T^n_{ijkt}}$: Value of nth touch sensor at time t.
$\mathbf{o_{ijk}}$: Crowd response to controller k.

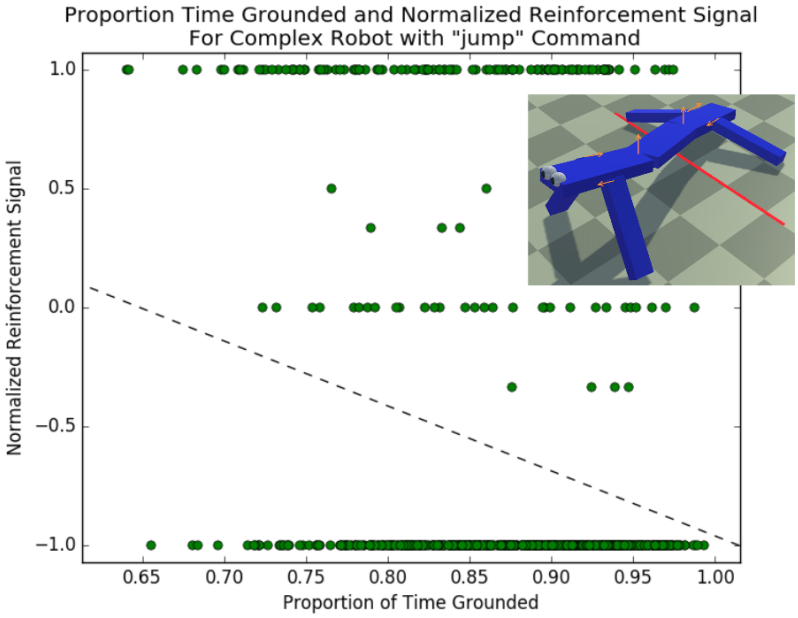
Grounding symbols using *Twitch Plays Robotics.*

Act. Observe. **Learn**. Predict.



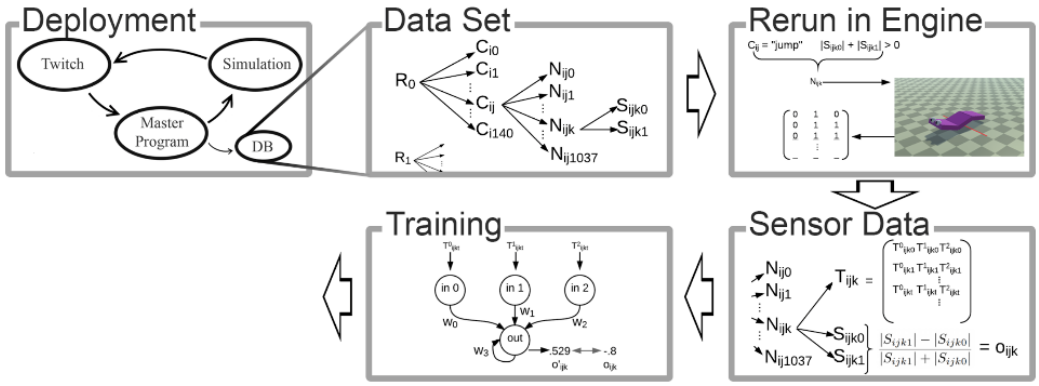
Grounding symbols using *Twitch Plays Robotics.*

Act. Observe. **Learn**. Predict.



Grounding symbols using *Twitch Plays Robotics.*

Act. Observe. **Learn.** Predict.

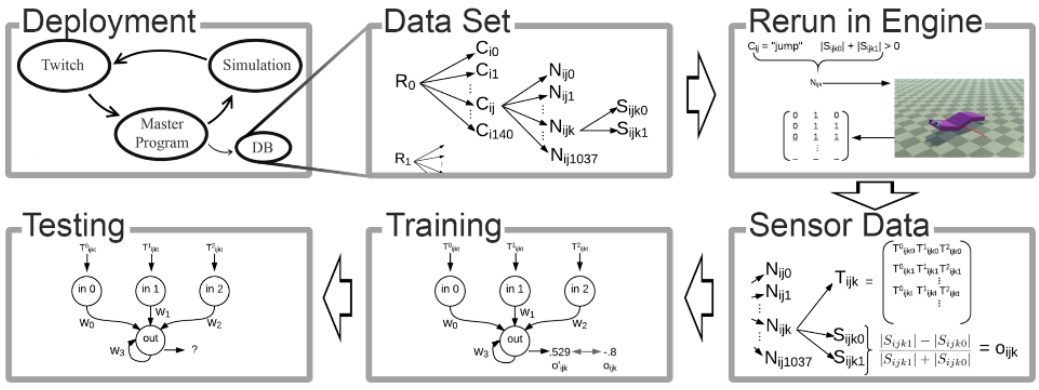


Simulation: Robots simulated in Unity Physics Engine.
Twitch: Video captured from simulation and fed to Twitch.
Master Program:controllers-->simulation; crowd response -->DB
$\mathbf{R_i}$:Robot i.
$\mathbf{C_{ij}}$:Command j issued to robot i.
$\mathbf{N_{ijk}}$:Controller k evaluated under command j.
$\mathbf{S_{ijk0}}$:Negative reinforcement signals issued under controller k.
$\mathbf{S_{ijk1}}$:Positive reinforcement signals issued under controller k.

$\mathbf{T^n_{ijkt}}$:Value of nth touch sensor at time t.
$\mathbf{o_{ijk}}$:Crowd response to controller k.
in p:pth input neuron
out:output neuron
$\mathbf{w_q}$:qth synaptic weight
$\mathbf{o'_{ijk}}$:Predicted crowd response.

Grounding symbols using *Twitch Plays Robotics*.

Act. Observe. Learn. **Predict**.



Simulation: Robots simulated in Unity Physics Engine.
Twitch: Video captured from simulation and fed to Twitch.
Master Program:controllers-->simulation; crowd response -->DB
$\mathbf{R_i}$:Robot i.
$\mathbf{C_{ij}}$:Command j issued to robot i.
$\mathbf{N_{ijk}}$:Controller k evaluated under command j.
$\mathbf{S_{ijk0}}$:Negative reinforcement signals issued under controller k.
$\mathbf{S_{ijk1}}$:Positive reinforcement signals issued under controller k.

$\mathbf{T^n_{ijkt}}$:Value of nth touch sensor at time t.
$\mathbf{o_{ijk}}$:Crowd response to controller k.
in p:pth input neuron
out:output neuron
$\mathbf{w_q}$:qth synaptic weight
$\mathbf{o'_{ijk}}$:Predicted crowd response.
?:How good was the prediction?

Grounding symbols using *Twitch Plays Robotics*.

Act. Observe. Learn. **Predict**.

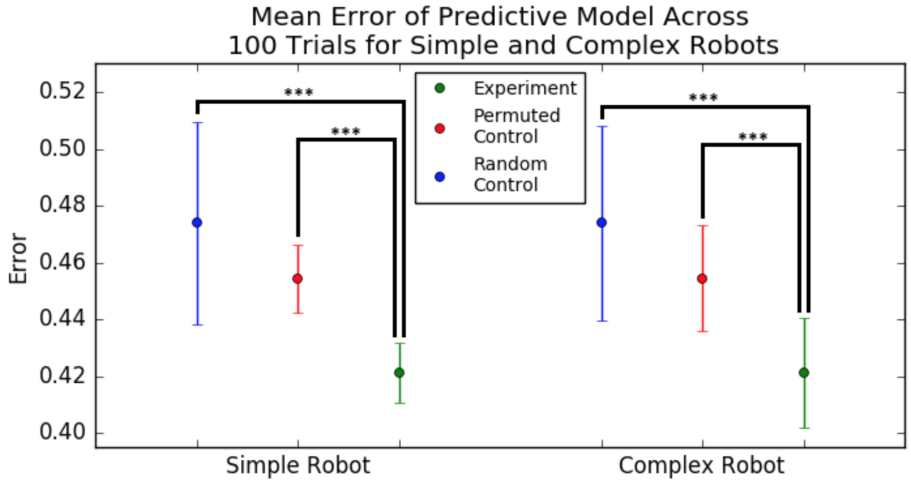
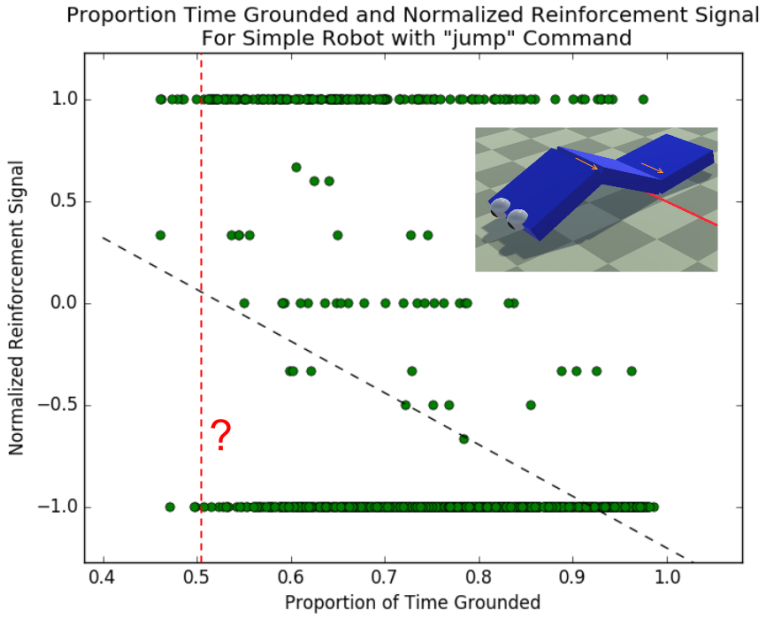


Figure 4: Results from the predictive model on unseen test data, the permuted test data set, and random RNNs on test data. *** denotes $p < .001$ as reported by a Student's T-test.

Grounding symbols using *Twitch Plays Robotics*.

Act. Observe. Learn. **Predict**.

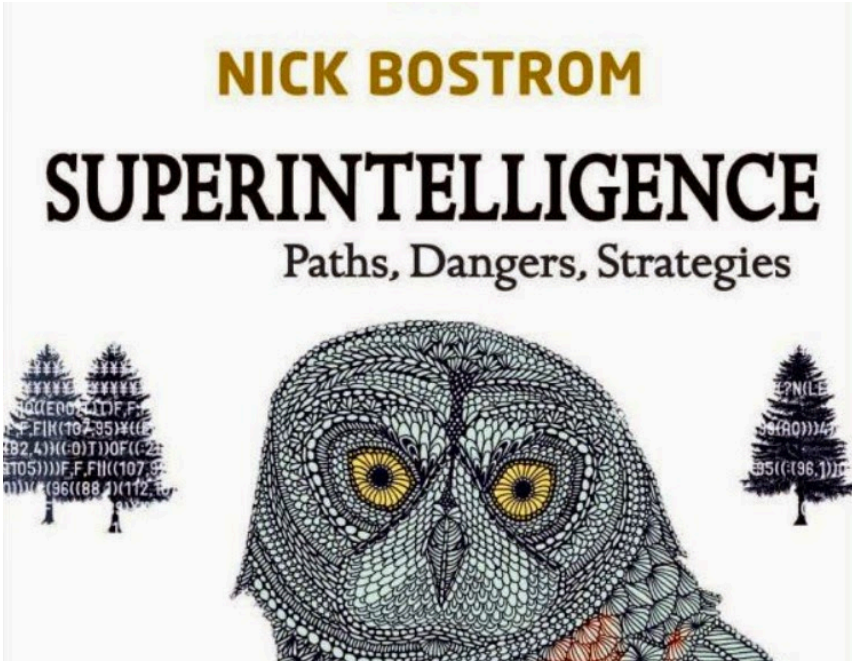


Scaling up the grounding of symbols.

- Jump
- ...
- Don't **jump** to conclusions.

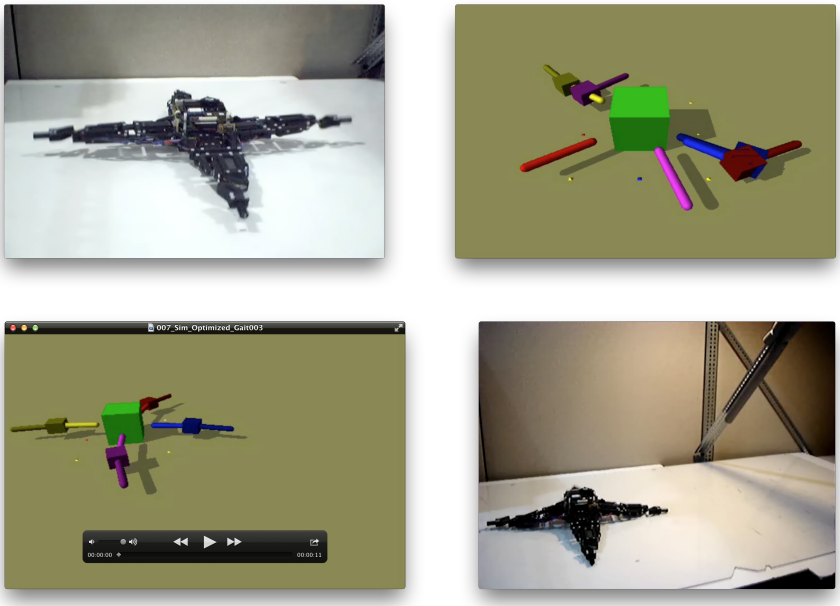
N. Bostrom. *Superintelligence*, (2014).

Superintelligent AI's may be unsafe because of **pathological instantiation**.



Resilient Machines (The Intact Robot)

Bongard, Zykov & Lipson, 2006, *Science*



Yudkowsky: CEV could combat perverse instantiation.

E. Yudkowsky. 2004. Coherent extrapolated volition. *The Singularity Institute*, San Francisco.

- A friendly AI should implement policies in a way that...
- ...despite our differences... **[coherent]**
- ...our future selves... **[extrapolated]**
- ...would be likely to approve of those policies. **[volition]**

Yudkowsky: CEV could combat perverse instantiation.

E. Yudkowsky. 2004. Coherent extrapolated volition. *The Singularity Institute*, San Francisco.

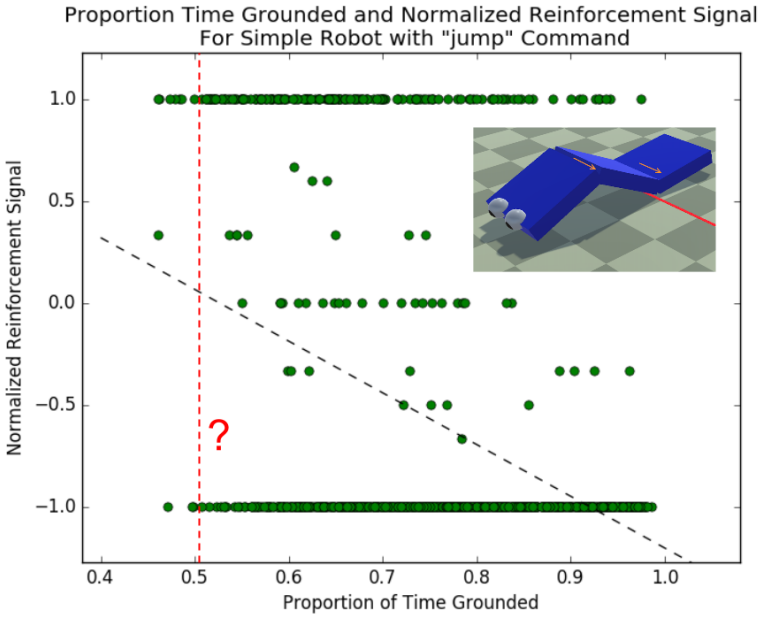
- A friendly AI should implement policies in a way that...
- ...despite our differences... **[coherent]**
- ...our future selves... **[extrapolated]**
- ...would be likely to approve of those policies. **[volition]**

Challenges:

- Requires an accurate model of future human behavior.
- Policy implemented now cannot cause future humans to vote as the model predicted.
- Thus, model must also predict that future humans would signal remorse that a given policy was not implemented now.

Grounding symbols using *Twitch Plays Robotics*.

Act. Observe. Learn. **Predict**.



Anetsberger & Bongard (2016). *Proceedings of ALife*. In press.

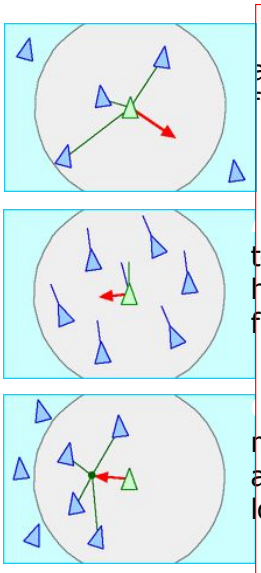
L21

Evolutionary Robotics for Swarms

Evolutionary Robotics for Swarms

Example: Controllers for a Robot Swarm

Design a controller to be placed into a swarm of Unmanned Aerial Vehicles, such that they swarm:



steer to avoid crowding local flockmates

steer towards the average heading of local flockmates

steer to move toward the average position of local flockmates

Evolutionary Robotics for Swarms

Example: Controllers for a Robot Swarm

Design a controller to be placed into a swarm of Unmanned Aerial Vehicles, such that they swarm:

Evolutionary Robotics for Swarms

Question: What tasks cannot be solved by a single robot?
Answer: Group hunting

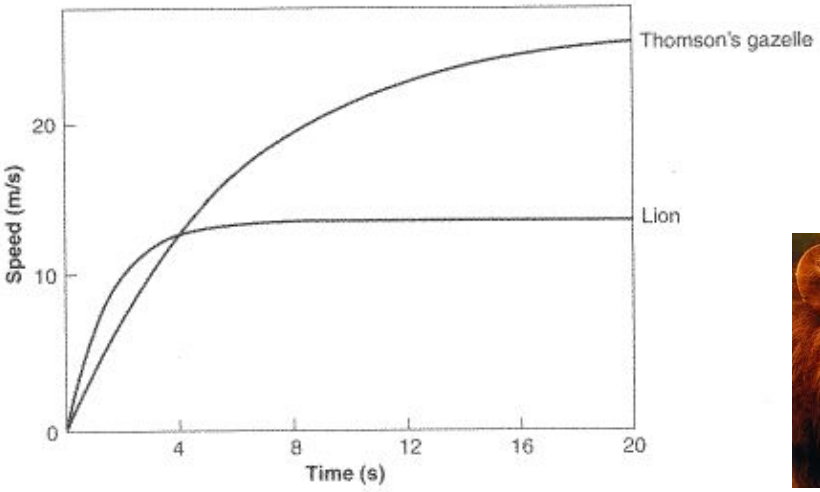


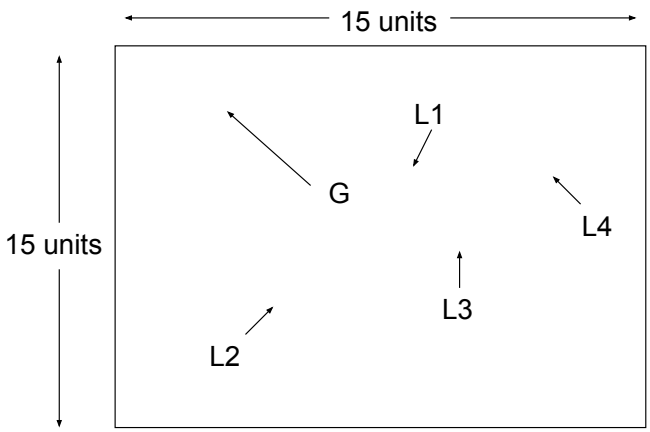
Fig. 1.1. Graphs of speed against time for lions and Thomson's gazelle, calculated from films of lions attacking prey. The curves were obtained by fitting to the data equations of the form $v = v_{max}[1 - \exp(-kt)]$, where v is the speed at time t, v_{max} is the speed that is approached asymptotically, and k is a constant. Redrawn from Elliott et al. (1977).

Evolutionary Robotics for Swarms

Question: Can we evolve behaviors for a team of predators?
How?
Will they evolve to cooperate?

Luke, S., Spector, L. (1996) Evolving teamwork and coordination with genetic programming. In Procs. of the First Annual Conference on Genetic Programming, pp. 150-156.

Evolutionary Robotics for Swarms

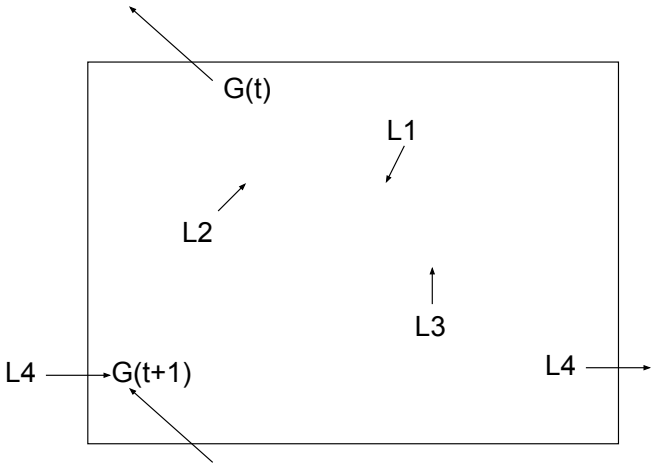


G = gazelle

L1 = lion 1
L2 = lion 2
L3 = lion 3
L4 = lion 4

Luke, S., Spector, L. (1996) Evolving teamwork and coordination with genetic programming. In Procs. of the First Annual Conference on Genetic Programming, pp. 150-156.

Evolutionary Robotics for Swarms



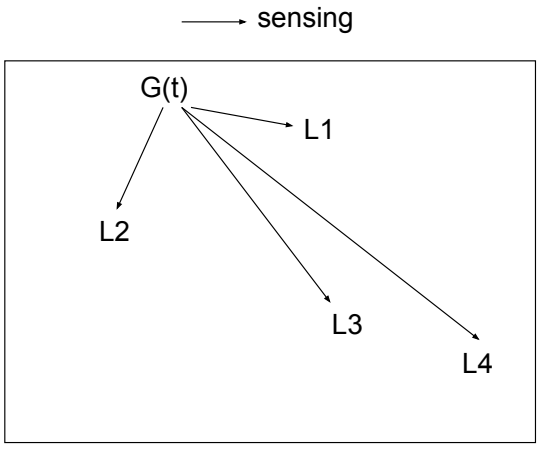
Savannah is toroidal:

Sensing and moving beyond the edge “wraps around” to the opposing side.

sensing
moving

Luke, S., Spector, L. (1996) Evolving teamwork and coordination with genetic programming. In Procs. of the First Annual Conference on Genetic Programming, pp. 150-156.

Evolutionary Robotics for Swarms



Gazelle's behavior *b*:

$||max|| = \sqrt{(w/2)^2 + (h/2)^2}$

$b = -\sum_{v \in V} (v/||v||)\,(||max|| - ||v||)$

Luke, S., Spector, L. (1996) Evolving teamwork and coordination with genetic programming. In Procs. of the First Annual Conference on Genetic Programming, pp. 150-156.

Lion's behavior L_i: Encoded as a tree that operates on 2D vectors:

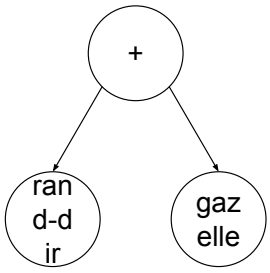
Name	# Args	Description
`last`	0	One unit in the direction the lion went last. On the first move, return a random normal vector.
`rand-dir`	0	A random normal vector.
`gazelle`	0	The smallest vector from the lion to the gazelle.
`+`	2	Add two vectors.
`-`	1	Negate a vector.
`*2`	1	Multiply the magnitude of a vector by 2.
`/2`	1	Divide the magnitude of a vector by 2.
`->90`	1	Rotate a vector clockwise 90 degrees.
`rand`	1	Given a vector, return a vector in the same direction, whose magnitude varies randomly between 0 and the magnitude of the original vector.
`inv`	1	"Invert" a vector's magnitude, so that small vectors are "more significant" than large vectors. This is done similarly to the method used in the gazelle algorithm: Let v be the vector, and $\|max\| = \sqrt{(w/2)^2 + (h/2)^2}$, where w is the width of the world (15 units), and h is the height of the world (also 15 units). Then inv returns $\frac{v}{\|v\|}(\|max\| - \|v\|)$.
`ifdot`	4	Evaluate the first and second arguments. If their dot product is greater than or equal to 0, then evaluate and return the third argument, else evaluate and return the fourth argument.
`if>=`	4	Evaluate the first and second arguments. If the magnitude of the first argument is greater than or equal to the magnitude of the second argument, then evaluate and return the third argument, else evaluate and return the fourth argument.

Q: If you were a single lion chasing the gazelle, what is the best strategy?

Luke, S., Spector, L. (1996) Evolving teamwork and coordination with genetic programming. In Procs. of the First Annual Conference on Genetic Programming, pp. 150-156.

Lion's behavior L_i: Encoded as a tree that operates on 2D vectors:

Name	# Args	Description
nearest	0	The vector from the lion nearest the gazelle, to the gazelle.
lion	0	A vector from the lion to its nearest neighbor lion.
rlion	0	A vector from the lion to the first neighbor lion encountered in a clockwise sweep. The sweep begins in the direction the lion moved last. All the minimal vectors to each lion are gathered, and the first one found in the sweep is returned.
llion	0	A vector from the lion to the first neighbor lion encountered in a counterclockwise sweep. The sweep begins in the direction the lion moved last. All the minimal vectors to each lion are gathered, and the first one found in the sweep is returned.



Q: If you had these additional four sensors, what is the best strategy?

Luke, S., Spector, L. (1996) Evolving teamwork and coordination with genetic programming. In Procs. of the First Annual Conference on Genetic Programming, pp. 150-156.

Lion's behavior L_i: Encoded as a tree that operates on 2D vectors:

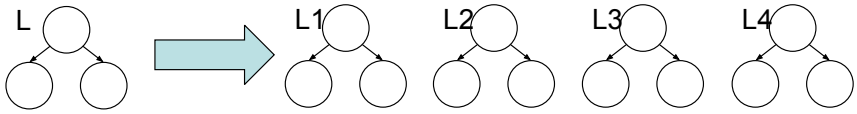
Name	# Args	Description
lion-1	0	A vector from the lion to lion #1.
lion-2	0	A vector from the lion to lion #2.
lion-3	0	A vector from the lion to lion #3.
lion-4	0	A vector from the lion to lion #4.

Q: If you had these additional four sensors, what is the best strategy?

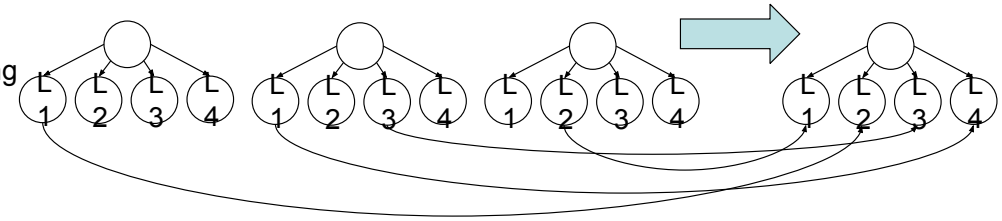
Luke, S., Spector, L. (1996) Evolving teamwork and coordination with genetic programming. In Procs. of the First Annual Conference on Genetic Programming, pp. 150-156.

Q: How to evolve teams? Three possible ways:

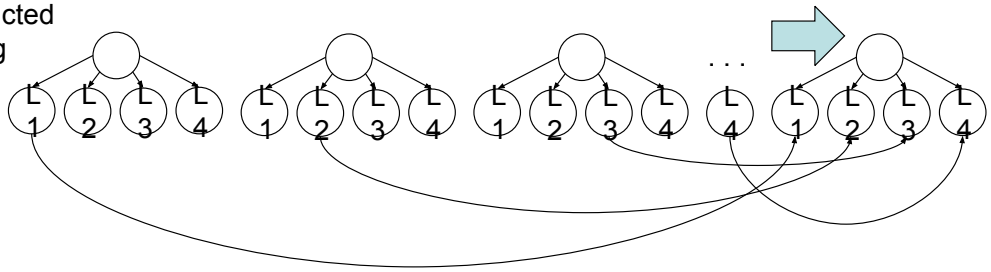
1. Cloning



2. Free breeding



3. Restricted breeding



Results:

1200 Evolutionary runs

100 runs for each of three sensing capabilities and three team-construction methods

100 runs: One lion with evolved behavior
100 runs: One randomly-moving lion
100 runs: Four randomly-moving lions

51 generations, population size = 500, max tree size = 70, max tree depth = 17,

For each new tree: created by crossover = 90% probability
created by mutation = 10% probability

For each team evaluation: Place gazelle, lions, randomly
Each moves 15 times.

Fitness function: Fitness = 0 : lion <=1 unit from gazelle
Fitness =||nearest lion – gazelle|| - 1 : otherwise

Trial Results

Sensing	Restricted Breeding Average	Restricted Breeding Best	Free Breeding Average	Free Breeding Best	Clones Average	Clones Best
Deictic	1.65	0.13	2.03	0.23	1.52	0.20
Name–Based	1.33	0.03	1.79	0.07	1.93	0.22
None	2.20	0.49	2.23	0.50	2.18	0.45

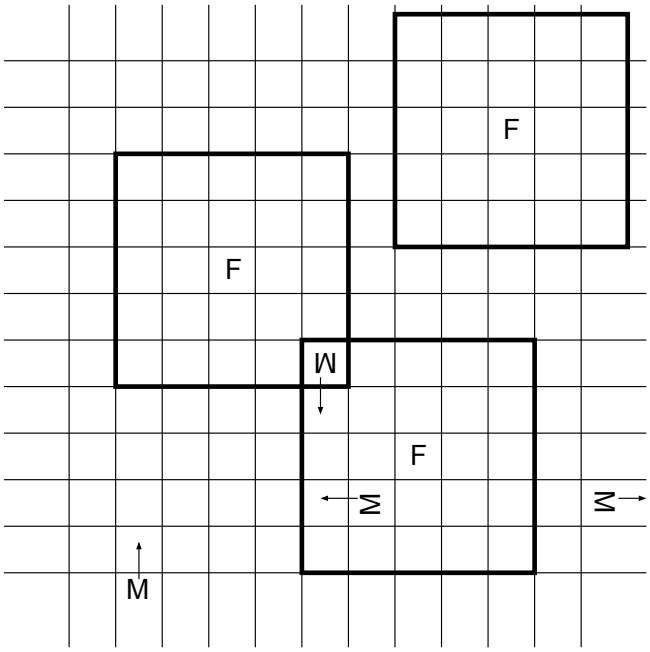
Control Results

	One Lion Average	One Lion Best	One Random Average	One Random Best	Four Random Average	Four Random Best
Controls	7.39	5.43	7.87	6.74	4.41	2.57

Observations:
1. One evolved lion doesn't do much better than a random lion
2. Four lions do better than one lion (random or evolved)
3. For clones, name-based sensing was worse than deictic sensing; why?
4. For restricted breed, name-based sensing was best; why?

Q: What else besides distance to Lion i might a lion want to know?

Evolution of Communication



Grid world:

200x200 squares (toroidal)

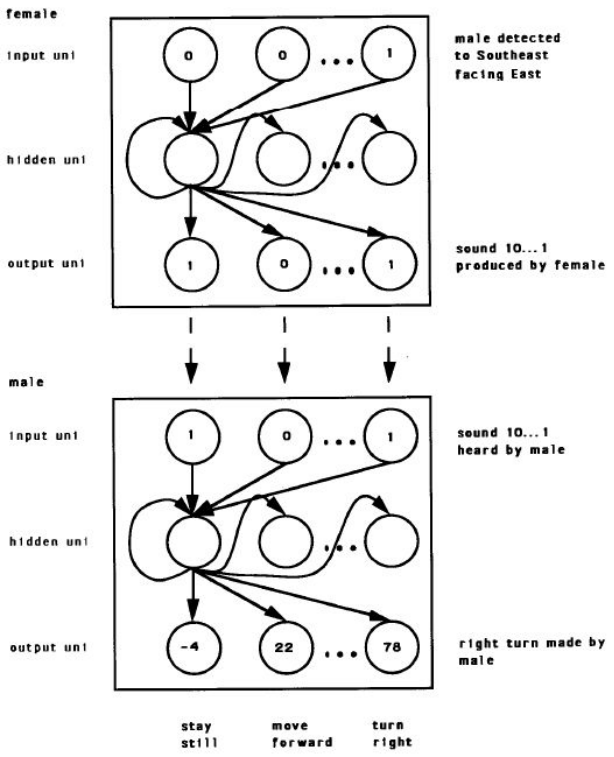
800 females, 800 males

Females are deaf and immobile, but can signal

Males are blind but can hear, cannot signal, but are mobile.

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

Evolution of Communication



Each male and female starts with its own random neural network

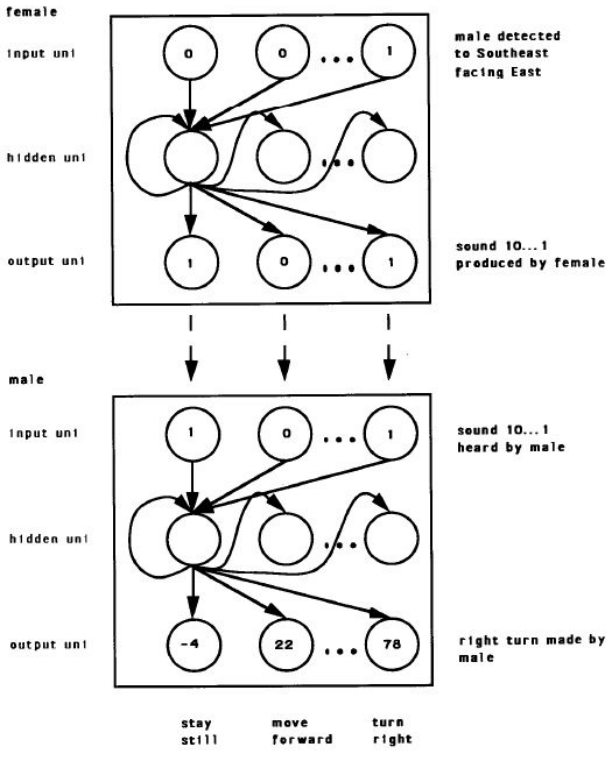
Female network:
input:

output:

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

L22

Evolution of Communication



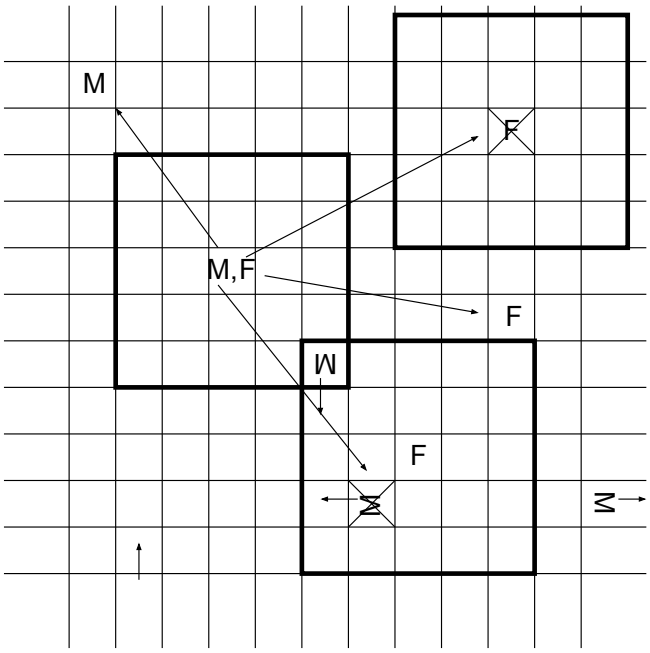
Each male and female starts with its own random neural network

Male network:
input:

output:

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

Evolution of Communication



If male finds female (occupy same cell)…

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

Observed evolutionary changes:

1. Males wandered randomly; females signaled randomly.

TABLE 1 Responses of randomly generated male neural nets to female signals are random (percentage of males making each response at Time = 100).

Signal	Move Forward	Turn Right	Turn Left	Stand Still
000	25	38	9	28
001	19	25	31	25
010	28	22	26	24
011	29	25	29	17
100	25	27	26	22
101	25	25	26	24
110	19	20	31	30
111	22	27	21	30

Observed evolutionary changes:

1. Males wandered randomly; females signaled randomly.

2. Males that stood still went extinct.

TABLE 2 Few males interpret a female signal to mean "stand still" (percentage of males making each response at Time = 5000).

Signal	Move Forward	Turn Right	Turn Left	Stand Still
000	74	20	5	1
001	81	12	7	0
010	67	18	14	1
011	79	11	19	1
100	80	7	12	0
101	75	11	14	0
110	56	21	23	0
111	70	14	16	0

L22

Observed evolutionary changes:

1. Males wandered randomly; females signaled randomly.

2. Males that stood still went extinct.

3. Males evolved to move in straight lines; ignore signals.

TABLE 3 Males ignore their inputs and simply move forward (percentage of males making each response at Time = 7500).

Signal	Move Forward	Turn Right	Turn Left	Stand Still
001	98	2	0	0
001	99	0	0	1
010	98	1	1	0
011	100	0	0	0
100	100	0	0	0
101	99	1	0	0
110	98	1	1	0
111	99	0	1	0

Observed evolutionary changes:

1. Males wandered randomly; females signaled randomly.

2. Males that stood still went extinct.

3. Males evolved to move in straight lines; ignore signals.

4. Males evolved to turn when on same row or column as female;
females have evolved to signal this information;
females that do not use this signal go extinct.

TABLE 4 Males evolve that interpret 101 as "turn left," 110 as "turn right," and the remaining patterns as "move forward" (percent of males making each response at Time = 15,000).

Signal	Move Forward	Turn Right	Turn Left	Stand Still
000	97	2	0	1
001	100	0	0	0
010	98	1	1	0
011	98	0	1	1
100	100	0	0	0
101	22	77	1	0
110	5	2	93	0
111	97	0	3	0

Observed evolutionary changes:

1. Males wandered randomly; females signaled randomly.

2. Males that stood still went extinct.

3. Males evolved to move in straight lines; ignore signals.

4. Males evolved to turn when on same row or column as female;
females have evolved to signal this information;
females that do not use this signal go extinct.

5. Females evolve to use this signal in more situations. →

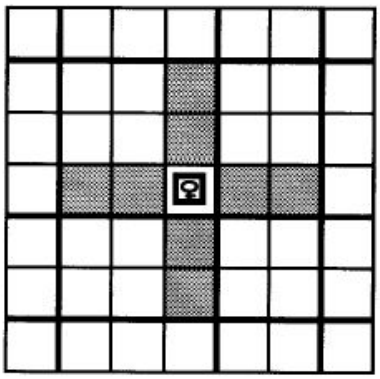
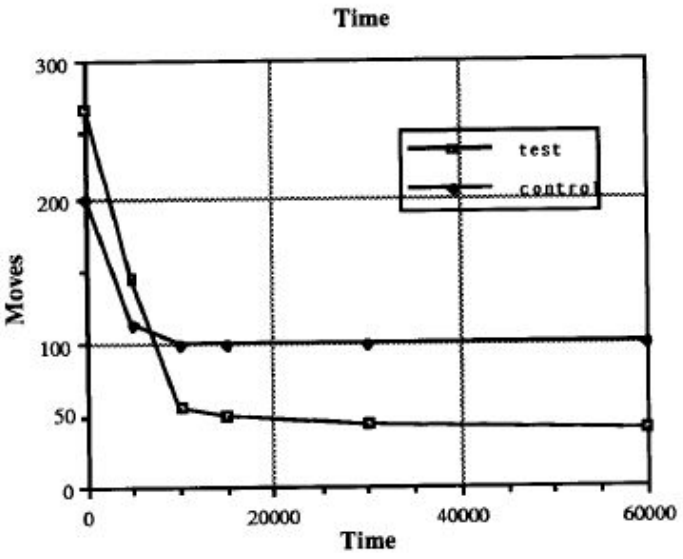


FIGURE 3 Females evolve that tell males in cross-shaped (shaded) area to continue turning until the male is facing the female. Males entering the "receptive field" but outside of the cross-shaped area are told to go straight. Such males will eventually enter the shaded area and from there be guided in to the female.

Evolution of Communication



Control set of runs:

Males were deafened; could not hear female's signals.

Q: Why the initial faster rate of improvement in the control case?

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

L22

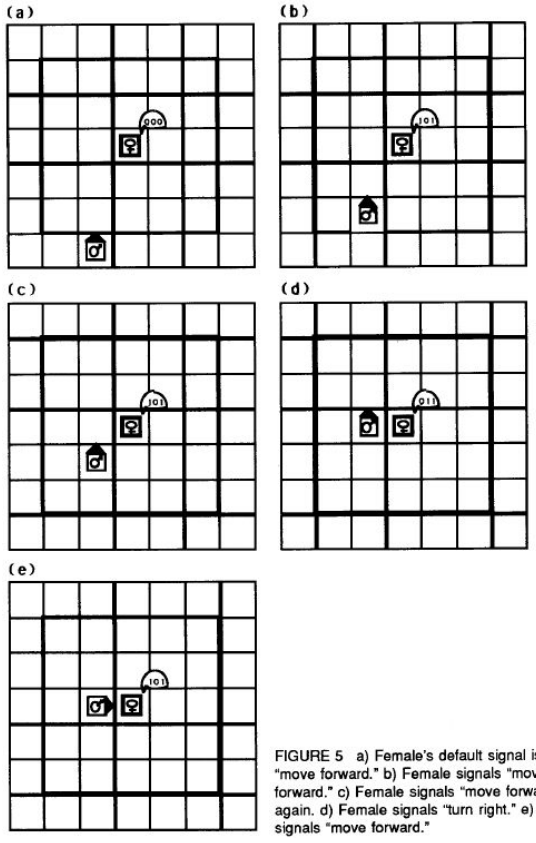
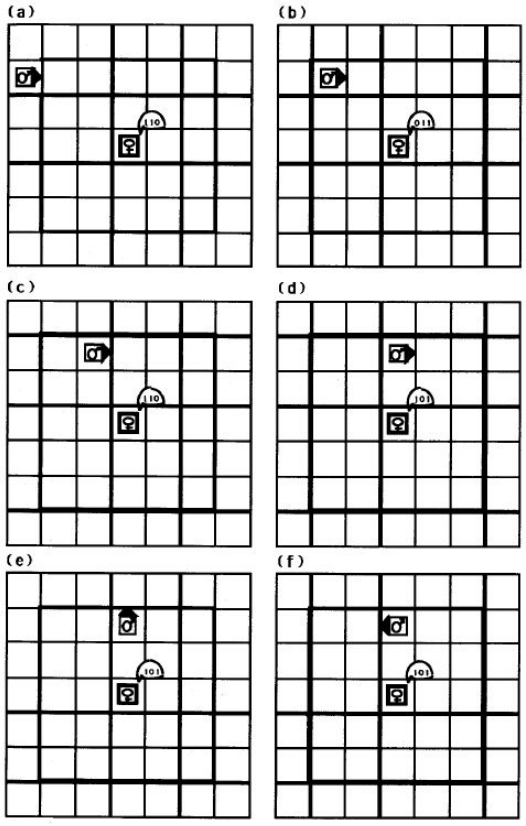


FIGURE 5 a) Female's default signal is "move forward." b) Female signals "move forward." c) Female signals "move forward" again. d) Female signals "turn right." e) Female signals "move forward."

Example of an evolved female signal / male response pairing.

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.



Example of a better evolved female signal / male response pairing.

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

Evolution of Communication

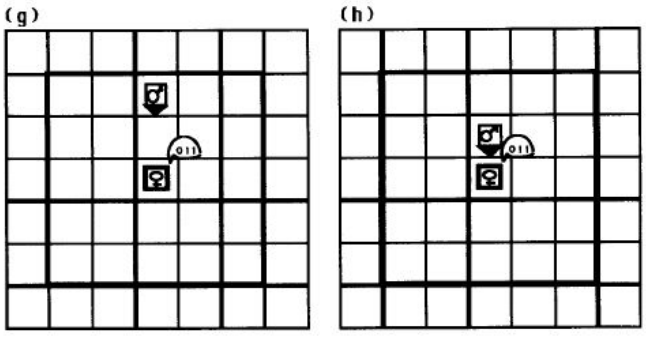


FIGURE 6 (continued) a) Male approaches "receptive field." b) Female signals "move forward." c) Female indicates "move forward" with another signal. d) Female signals "turn left" since she lacks a signal for "turn right." e) Female signals "turn left" again. f) Female signals "turn left" again. g) Female signals "move forward." h) Female signals "move forward" again.

Example of a better evolved female signal / male response pairing.

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

TABLE 5 Frequency of Male Signal Responses (time = 0)

4:000	6:0001	5:0002	7:0003	4:0010	6:0011	4:0012	4:0013
12:0020	1:0021	3:0022	4:0023	8:0030	8:0031	2:0032	7:0033
5:0100	7:0101	7:0102	5:0103	7:0110	5:0111	3:0112	8:0113
2:0120	7:0121	6:0122	7:0123	3:0130	9:0131	6:0132	7:0133
8:0200	6:0201	4:0202	3:0203	11:0210	3:0211	4:0212	8:0231
1:0220	7:0221	6:0222	5:0223	8:0230	5:0231	8:0232	2:0233
8:0300	7:0301	2:0302	9:0303	4:0310	9:0311	9:0312	4:0313
10:0320	7:0321	3:0322	5:0323	12:0330	7:0331	6:0332	4:0333
8:1000	6:1001	6:1002	7:1003	7:1010	4:1011	7:1012	2:1013
7:1020	7:1021	0	7:1023	4:1030	8:1031	7:1032	8:1033
11:1100	4:1101	4:1102	6:1103	10:1110	3:1111	3:1112	8:1113
3:1120	5:1121	6:1122	4:1123	3:1130	8:1131	2:1132	7:1133
11:1200	4:1201	6:1202	5:1203	5:1210	6:1211	8:1212	5:1213
7:1220	5:1221	8:1222	8:1223	5:1230	4:1231	7:1232	5:1233
8:1300	4:1301	9:1302	7:1303	8:1310	11:1311	8:1312	6:1313
4:1320	3:1321	10:1322	6:1323	5:1330	2:1331	6:1332	7:1333
7:2000	9:2001	7:2002	11:2003	3:2010	9:2011	5:2012	6:2013
12:2020	8:2021	5:2022	7:2023	7:2030	6:2031	7:2032	10:2033
7:2100	5:2101	8:2102	3:2103	4:2110	7:2111	4:2112	5:2113
11:2120	11:2121	6:2122	6:2123	5:2130	8:2131	4:2132	9:2133
7:2200	4:2201	3:2202	10:2203	7:2210	8:2211	4:2212	7:2213
8:2220	7:2221	9:2222	11:2223	4:2230	5:2231	3:2232	5:2233
6:2300	8:2301	6:2302	7:2303	7:2310	7:2311	5:2312	6:2313
6:2320	4:2321	7:2322	4:2323	3:2330	3:2331	8:2332	3:2333
8:3000	6:3001	8:3002	4:3003	8:3010	4:3011	9:3012	9:3013
4:3020	8:3021	4:3022	9:3023	6:303	10:3031	6:3032	5:3033
9:3100	5:3101	7:3102	3:3103	14:3110	8:3111	5:3112	8:3113
7:3120	5:3121	5:3122	12:3123	3:3130	6:3131	6:3132	6:3133
4:3200	10:3201	6:3202	4:3203	12:3210	6:3211	10:3212	0
14:3220	9:3221	3:3222	12:3223	4:3230	6:3231	7:3232	7:3233
8:3300	4:3301	4:3302	9:3303	8:3310	8:3311	4:3312	3:3313
4:3320	6:3321	11:3322	6:3323	4:3330	5:3331	6:3332	6:3333

Female can emit one of four signals:

00, 01, 10, 11

For a given female signal, males can do 1 of 4 actions:

0: stay still
1: go forward
2: turn left
3: turn right

←Initial random responses of males

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

TABLE 6 Frequency of Male Signal Responses (time = 8000)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1:0101	0	0	0	0	0	1:0113
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1:0222	0	0	0	0	0
0	0	0	0	0	0	1:0312	1:0313
0	0	0	0	0	0	0	1:0333
2:1000	0	0	0	0	73:1011	0	1:1013
1:1020	0	0	0	0	0	2:1032	6:1033
0	1:1101	0	1:1103	1:1110	70:1111	92:1112	64:1113
0	131:1121	14:1122	2:1123	0	3:1131	0	24:1133
1:1200	0	0	0	0	323:1211	5:1212	2:1213
1:1220	4:1221	2:1222	12:1223	0	0	0	2:1233
4:1300	2:1301	1:1302	0	5:1310	547:1311	74:1312	1:1313
0	89:1321	0	1:1323	0	13:1331	0	1:1333
0	0	0	0	0	0	0	0
0	1:2021	0	1:2023	0	0	0	0
0	0	0	0	0	0	0	1:2113
0	1:2121	0	1:2123	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1:2332	0
0	0	1:3002	0	0	0	0	0
0	0	0	1:3023	0	0	0	0
1:3100	0	0	0	0	0	0	0
1:3120	1:3121	0	0	1:3130	0	0	0
0	0	0	0	0	0	0	0
1:3220	0	0	0	0	0	0	0
0	0	1:3302	0	0	0	0	0
0	0	1:3322	0	0	1:3331	0	0

Males can do 1 of 4 actions:

0: stay still
1: go forward
2: turn left
3: turn right

The 547 males with response ‘1311’ responded to females that emitted only signals #1 and #2.

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

TABLE 7 Frequency of Male Signal Responses (time = 10,000)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1:0101	0	0	0	0	0	1:0113
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1:0313
0	0	0	0	0	0	0	0
1:1000	0	0	0	0	45:1011	0	22:1013
1:1020	0	0	0	0	0	1:1032	0
0	0	0	1:1103	0	55:1111	90:1112	127:1113
0	78:1121	5:1122	1:1123	0	0	0	53:1133
0	0	0	0	0	321:1211	2:1212	3:1213
1:1220	2:1221	4:1222	4:1223	0	0	0	1:1233
2:1300	0	1:1302	0	1:1310	613:1311	42:1312	20:1313
0	85:1321	0	0	0	7:1331	0	1:1333
0	0	0	0	0	0	0	0
0	1:2021	0	0	0	0	0	0
0	0	0	0	0	0	0	1:2113
0	0	0	1:2123	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1:3023	0	0	0	0
1:3100	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1:3322	0	0	1:3331	0	0

Males can do 1 of 4 actions:

0: stay still
1: go forward
2: turn left
3: turn right

The 127 males with response '1113' responded to females that emitted only signals #3 and #4.

Two groups of females; Males 1311 and 1113 only responded to one of these groups.

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

TABLE 8 Frequency of Male Signal Responses (time = 12,000)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1:0313
0	0	0	0	0	0	0	0
1:1000	0	0	0	0	33:1011	0	3:1013
1:1020	0	0	0	0	0	0	1:1033
0	0	0	0	0	40:1111	35:1112	169:1113
0	25:1121	1:1122	0	0	1:1311	0	23:1133
0	0	0	0	0	406:1211	17:1212	20:1213
0	3:1221	0	0	0	0	0	0
0	0	1:1302	0	0	627:1311	9:1312	128:1313
0	53:1321	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1:2021	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1:3322	0	0	0	0	0

Males can do 1 of 4 actions:

0: stay still
1: go forward
2: turn left
3: turn right

Q: How can we tell that from this figure?

Q: What do you think will happen in subsequent generations?

TABLE 9 Frequency of Male Signal Responses (time = 14,000)

0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0
1:1000	0	0	0	0	11:1011	0	1:1013
1:1020	0	0	0	0	0	0	0
0	0	0	0	0	27:1111	88:1112	286:1113
0	16:1121	1:1122	0	0	2:1131	0	26:1133
0	0	0	0	1:1210	310:1211	78:1212	20:1213
0	3:1221	0	0	0	0	0	0
0	1:1301	1:1302	0	0	470:1311	20:1312	217:1313
0	18:1321	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1:2021	0	0	0	0	0	0
0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0

Female can emit
one of four signals:

00, 01, 10, 11

For a given female signal,
Males can do 1 of 4 actions:

0: stay still
1: go forward
2: turn left
3: turn right

TABLE 10 Frequency of Male Signal Responses (time = 16,000)

0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0
1:1000	0	0	0	0	7:1011	0	1:1013
1:1020	0	0	0	0	0	0	0
0	0	0	0	0	27:1111	125:1112	192:1113
0	10:1121	1:1122	0	0	1:1131	0	3:1133
0	0	0	0	1:1210	397:1211	102:1212	218:1213
0	0	0	0	0	0	0	0
0	0	1:1302	0	0	326:1311	28:1312	155:1313
0	2:1321	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1:2021	0	0	0	0	0	0
0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

TABLE 11 Frequency of Male Signal Responses (time = 20,000)

0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	8:1111	72:1112	87:1113
0	5:1121	0	0	0	0	0	0
0	0	0	0	0	222:1211	171:1212	262:1213
0	0	0	0	0	0	0	0
0	0	0	0	0	211:1311	45:1312	517:1313
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0

Female can emit
one of four signals:

00, 01, 10, 11

For a given female signal,
Males can do 1 of 4 actions:

0: stay still
1: go forward
2: turn left
3: turn right

TABLE 12 Frequency of Male Signal Responses (time = 30,000)

0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	5:1211	231:1212	183:1213
0	0	0	0	0	0	0	0
0	0	0	0	0	1:1311	40:1312	1140:1313
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

TABLE 13 Frequency of Male Signal Responses (time = 40,000)

0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1600:1313
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	0	0	0

Female can emit
one of four signals:

00, 01, 10, 11

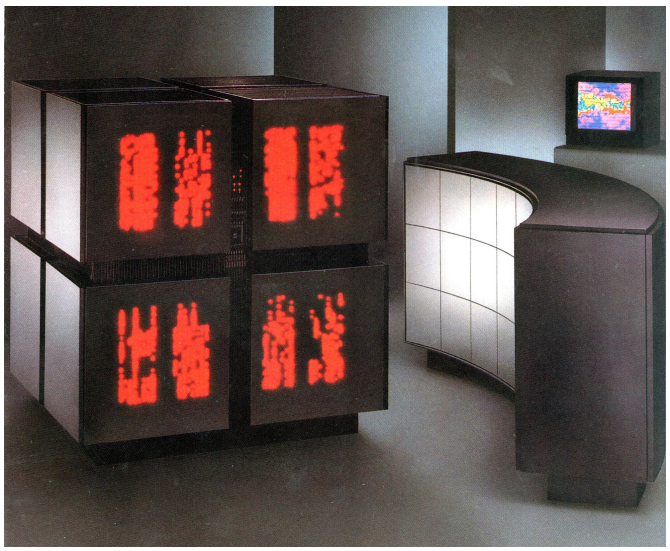
For a given female signal,
Males can do 1 of 4 actions:

0: stay still
1: go forward
2: turn left
3: turn right

Werner, G.M., Dyer, M.G. (1991)
Evolution of Communication in Artificial Organisms
Artificial Life II, Studies in the Sciences of Complexity.

First attempt to evolve robot morphology and control

Sims, K. (1994)
Evolving 3D Morphology
And Behavior by Competition.
Artificial Life IV, 28-39.



First attempt to evolve robot morphology and control

Evolution performed on Connection Machine-5:

1024 cores

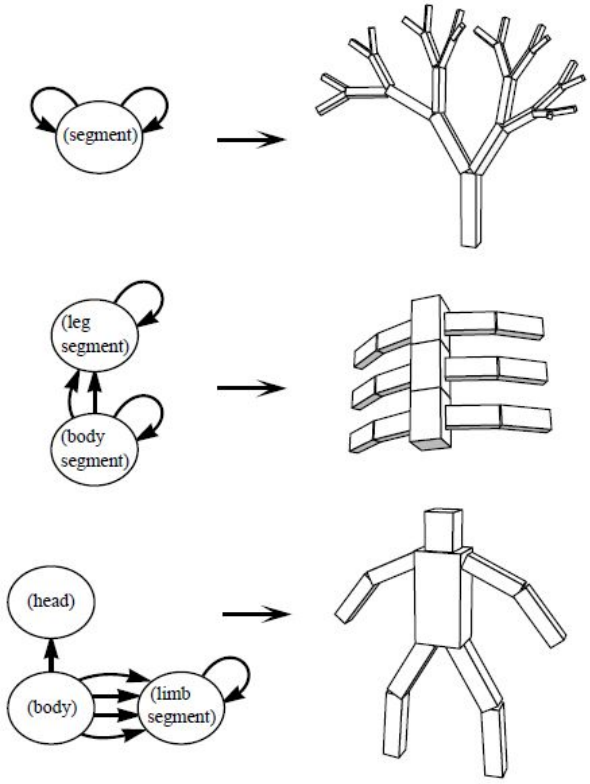
Rpeak of 131.0 Gflops/sec

Mac Pro: ~91 Gflops/sec

Population size = 300
# of generations = 100

~ 3 hours for a typical run

Sims, K. (1994)
Evolving 3D Morphology
And Behavior by Competition.
Artificial Life IV, 28-39.



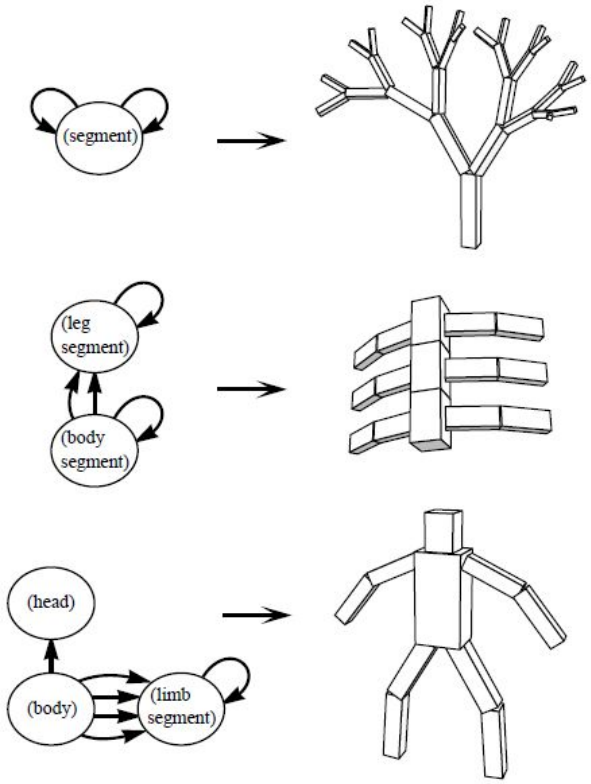
First attempt to evolve robot morphology and control

Genotypes:

Phenotypes:

Genotype-to-phenotype mapping:

Sims, K. (1994)
Evolving 3D Morphology
And Behavior by Competition.
Artificial Life IV, 28-39.



First attempt to evolve robot morphology and control

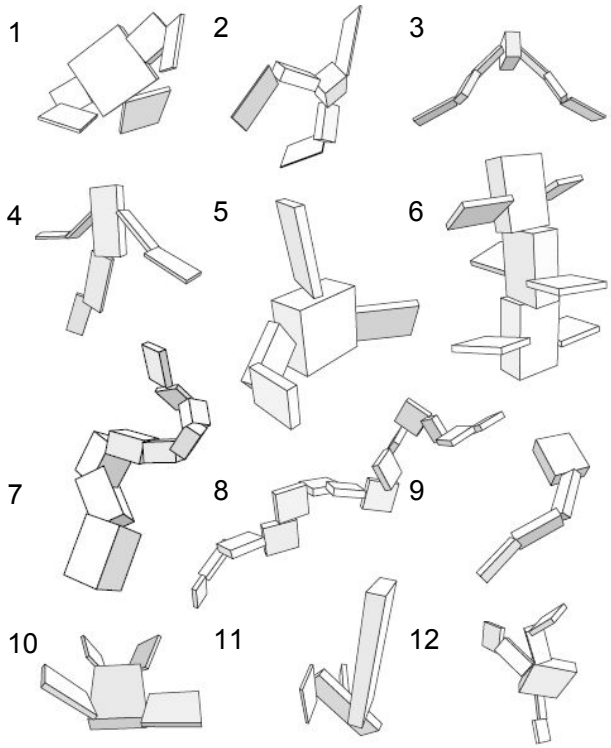
Genotype:

Node labels:
- body part dimensions
- joint-type: hinge, ball/socket…
- joint-limits
- recursive-limit (RL):

```
if ( RL>0 ):
    RL = RL-1
    follow all outgoing edges.
```

Edge labels:
- change in position
- change in orientation
- change in scale
- reflection
- terminal-only flag (TO):

```
if ( TO==true &&
     parent->RL==0 ):
    follow edge;
```

First attempt to evolve robot morphology and control

Genotype:

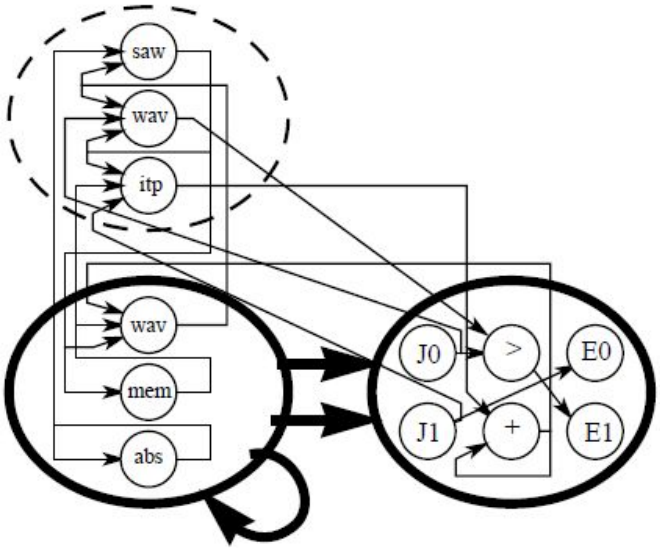
Node labels:
- body part dimensions
- joint-type: hinge, ball/socket…
- joint-limits
- **recursive-limit (RL):**

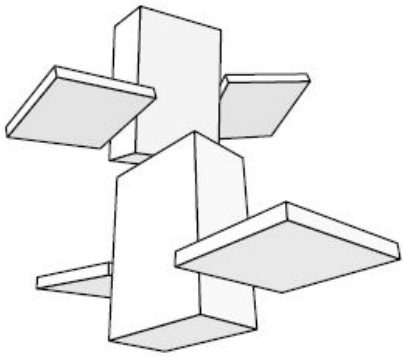
```
if ( RL>0 ):
    RL = RL-1
    follow all outgoing edges.
```

Edge labels:
- position
- orientation
- scale
- reflection
- **terminal-only flag (TO):**

```
if ( TO==true &&
     parent->RL==0 ):
    follow edge;
```

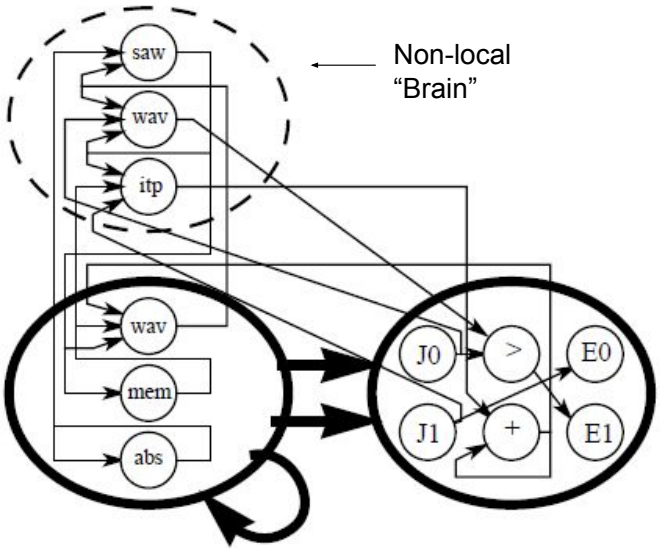
First attempt to evolve robot morphology and control





Sims, K. (1994)
Evolving 3D Morphology
And Behavior by Competition.
Artificial Life IV, 28-39.

First attempt to evolve robot morphology and control



Non-local
"Brain"

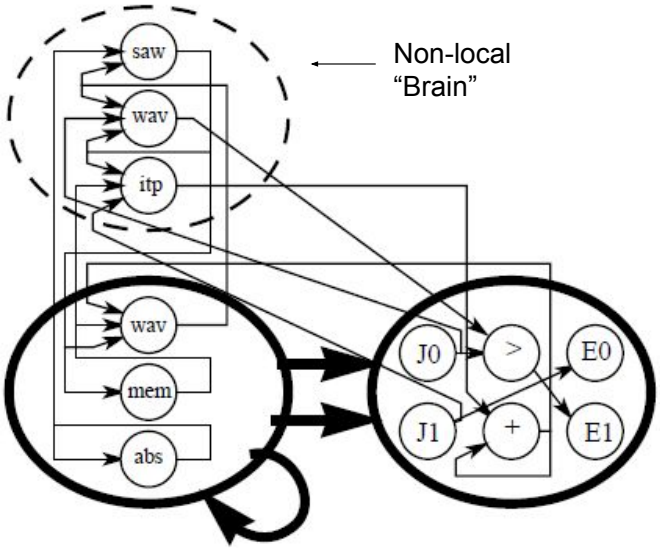
Genotypes:

Nested, directed, multigraphs:

Each graph node contains another graph.

Embedded graph describes the local neural circuitry for that body part.

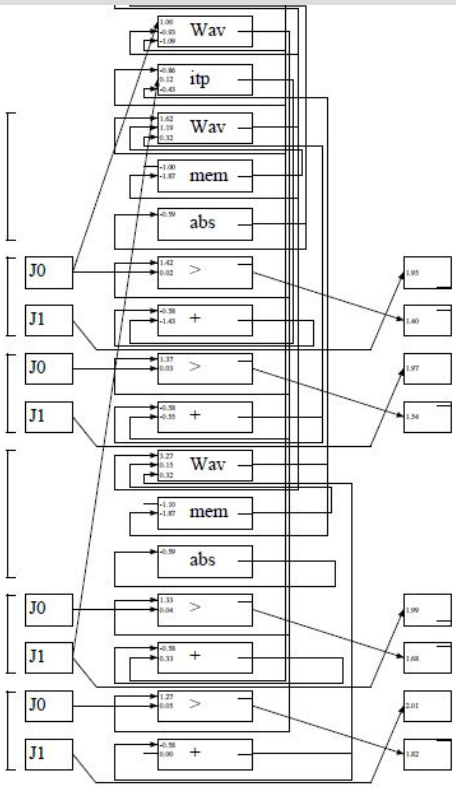
JX = joint sensor
CX = contact sensor
PX = photosensor
EX = effector
(X=degree-of-freedom)



First attempt to evolve robot morphology and control

Neurons:

Sum
Product
Divide
Sum-threshold
Greater-than
Sign-of, Min, Max, Abs,
If,
Interpolate (itp)
Sin, cos, atan,
Log,
Expt,
Sigmoid,
Integrate,
Differentiate,
Smooth,
Memory
Oscillate-wave (wav)
Oscillate-saw (saw).



First attempt to evolve robot morphology and control

Genotypes:

Nested, directed, multigraphs:

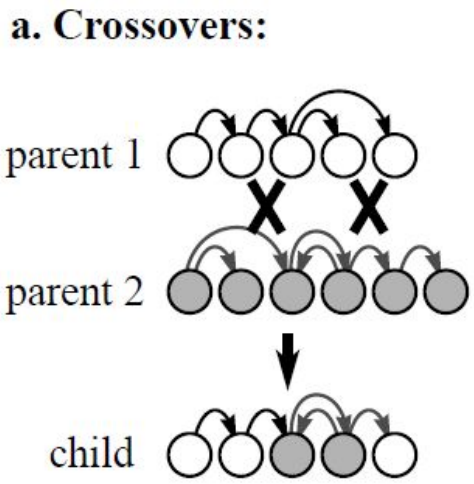
Each graph node contains another graph.

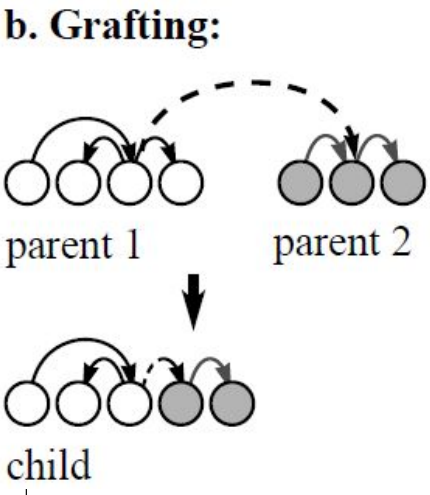
Embedded graph describes the local neural circuitry for that body part.

JX = joint sensor
CX = contact sensor
PX = photosensor
EX = effector
(X=degree-of-freedom)

Sims, K. (1994)
Evolving 3D Morphology
And Behavior by Competition.
Artificial Life IV, 28-39.

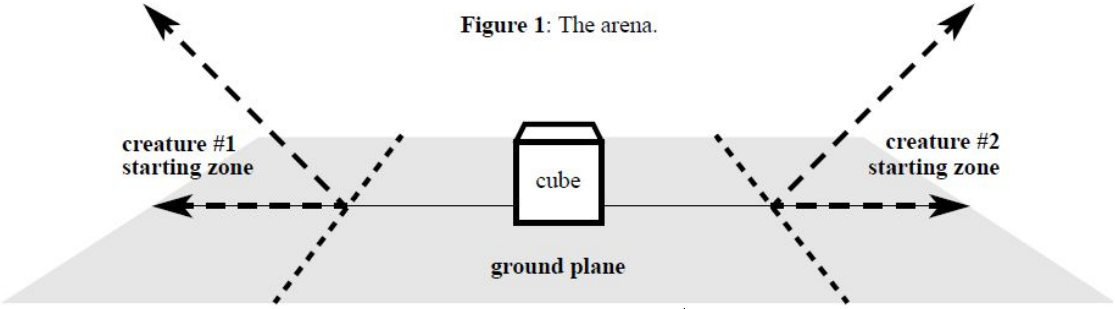
First attempt to evolve robot morphology and control





Sims, K. (1994)
Evolving 3D Morphology
And Behavior by Competition.
Artificial Life IV, 28-39.

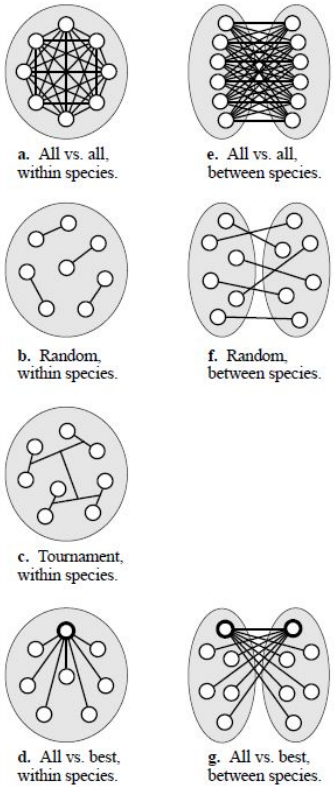
First attempt to evolve robot morphology and control



Two robots compete:

Sims, K. (1994)
Evolving 3D Morphology
And Behavior by Competition.
Artificial Life IV, 28-39.

First attempt to evolve robot morphology and control



Robots can compete within a single species (a-d),

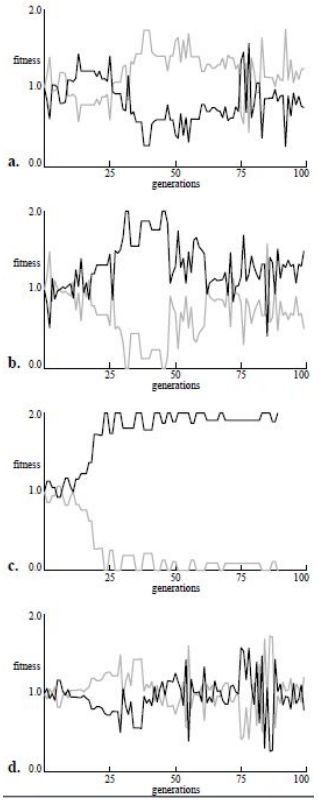
or

robots can compete between species (e-g)

What is the disadvantage of 'all-vs-all', 'random', and 'tournament'?

Sims, K. (1994) Evolving 3D Morphology And Behavior by Competition. *Artificial Life IV*, 28-39.

First attempt to evolve robot morphology and control



Results from four evolutionary runs.

Grey line:

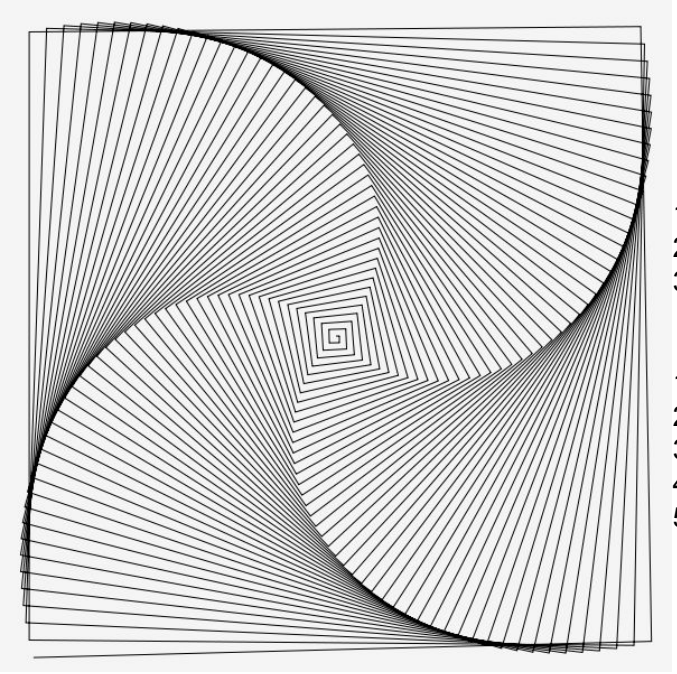
Average fitness of species 1;

Black line:

Average fitness of species 2.

Sims, K. (1994) Evolving 3D Morphology And Behavior by Competition. *Artificial Life IV*, 28-39.

Turtle Graphics (from computer graphics)



Using A Formal Grammar To Evolve Robot Bodies and Brains

Turtle moves over a 2D surface.

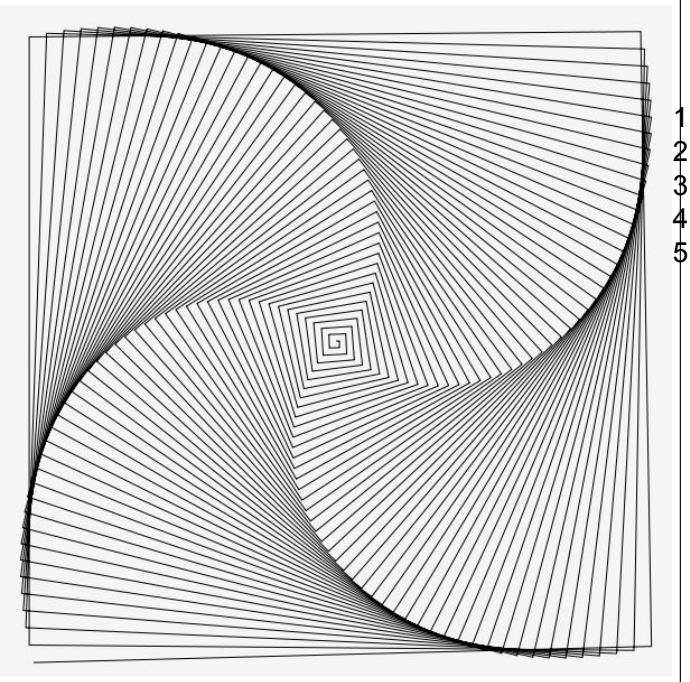
Possesses:

1. Position
2. Orientation
3. Pen, which can be up or down.

"Code" for drawing:

1. Move forward 10 paces
2. Turn left 95 degrees
3. Drop pen
4. Move forward
5. Lift pen…

Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.



Using A Formal Grammar To Evolve Robot Bodies and Brains

1. Drop pen
2. For i = 1 to 1000
3. forward 1 + i/1000
4. turn 85 degrees left
5. Lift pen

Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.

Using A Formal Grammar To Evolve Robot Bodies and Brains

Formal grammar:

1. Define an alphabet V that contains elements that can be replaced (variables)

2. Define a starting 'sentence' composed of elements from V
   (eg. 'AB')

3. Define a set of production rules:
   rule is composed of two strings:
   the predecessor and
   the successor
   (eg. A → AB
   B → A)

4. If a symbol C in V does not match any predecessor, then C is a constant.

5. Apply the productions rules to the start sentence.

6. Continue to apply the rules until the sentence only contains constants.

Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.

Using A Formal Grammar To Evolve Robot Bodies and Brains

Example 1: Algae

Lindenmayer's original L-system for modelling the growth of algae.

variables : A B
constants : none
start : A
rules : (A → AB), (B → A)

which produces:

n = 0 : A
n = 1 : AB
n = 2 : ABA
n = 3 : ABAAB
n = 4 : ABAABABA
n = 5 : ABAABABAABAAB
n = 6 : ABAABABAABAABABAABABA
n = 7 : ABAABABAABAABABAABABAABAABABAABAAB

Example 1: Algae, explained

```
n=0:          A           start (axiom/initiato
             / \
n=1:        A   B         the initial single A
           /|    \
n=2:      A B     A        former string AB with
         /| |     |\
n=3:    A B A     A B      note all A's producin
       /| | |\    |\ \
n=4: A B A A B   A B A     ... into an A one gen
```

A = algae
B = bud (cannot reproduce)

A → AB is a model of growth: algae 'spawns' a bud.

B → A models maturation: bud becomes adult algae capable of reproduction.

Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.

Using A Formal Grammar To Evolve Robot Bodies and Brains

Lindenmayer system =

A system in which ‘turtle graphics’ commands serve as the elements in a formal grammar.

Example: “Drawing” Cantor dust.

variables : A B

constants : none

start : A {starting character string}

rules : (A → ABA), (B → BBB)

Let *A* mean "draw forward" and *B* mean "move forward".

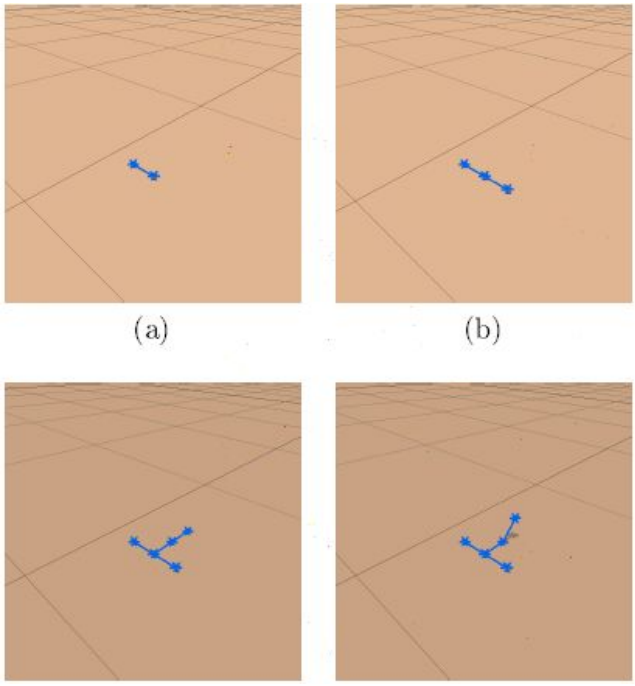
Using A Formal Grammar To Evolve Robot Bodies and Brains

Given the right L-system, one can produce structures that look like biological plants:

Example: Fractal plant

variables : X F
constants : + −
start : X
rules : (X → F-[[X]+X]+F[+FX]-X),
(F → FF)

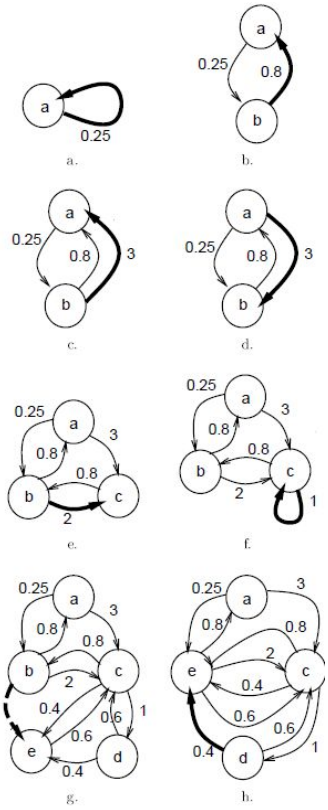
F means "draw forward",
- means "turn left 25°", and
+ means "turn right 25°".
X does not correspond to any drawing action.
[= ‘push’ position and angle
] = ‘pop’ position and angle



Building bodies with L-systems

[= push state
] = pop state
{block}(n) = repeat enclosed block *n* times.
forward: create bar forward
backward: move backward
Revolute-1(n): move forward, add joint around Z
Revolute-2(n): move forward, add joint around Y
up(n): rotate heading +n*90° about turtle's X axis
down(n): rotate –n*90° around X
left/right(n): … around Y
counter/clockwise(n): … around Z
(n = rate of oscillation of motor)

Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.



Building brains with L-systems

Turtle 'lives' on synapses.

If turtle's 'home' link connects neuron A to neuron B…

[= push 'home' link
] = pop new 'home' link

decrease-weight(n): subtract n from weight of current link.

duplicate(n): Create new link from A to B with weight n.

loop(n): Create new link from B to itself with weight n.

merge(n): Merge neurons A and B into single neuron, C. new 'home' link is nth input link to C.

parent(n): move to nth input link to A.

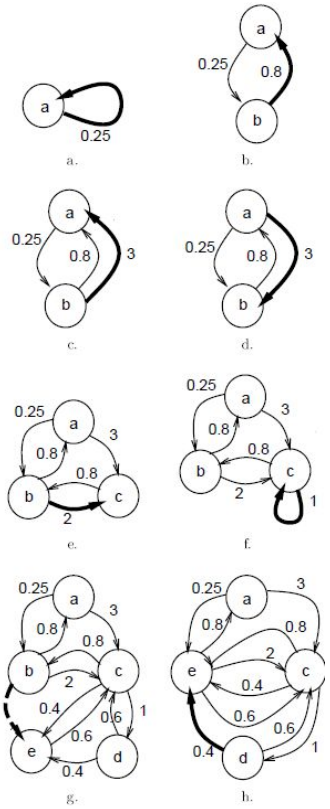
next(n): move to nth output link from B.

reverse: link changed to point from B to A.

output(n): neuron A sends commands to the closest joint.

split(n): create new neuron C; attach edge from A to C; and create a new edge from C to B with weight *n*.

Connecting Body and Brain



In the formal grammar, throw 'body' and 'brain' elements together.

Turtle 'points' to both a body part and a synapse.

If a 'joint' element is called, output(1) is also called:

connects part of the neural network to the joint.

L24

Parametric Lindenmayer systems

For example, the L-system,

$$\begin{aligned} a: &\ \rightarrow a\ b \\ b: &\ \rightarrow b\ a \end{aligned}$$

if started with the symbol a, produces the following strings,

$$a$$
$$ab$$
$$abba$$
$$abbabaab$$

A parametric L-system [Lindenmayer, 1974] is a class of L-systems in which production rules have parameters and algebraic expressions can be applied when parameter values to successors. Parameter values can also be used in determining which production rule to apply. For example, the P0L-system,

$$\begin{aligned} a(n): &\ (n > 1) &&\rightarrow a(n-1)\ b(n) \\ a(n): &\ (n \leq 1) &&\rightarrow a(0) \\ b(n): &\ (n > 2) &&\rightarrow b(n/2)\ a(n-1) \\ b(n): &\ (n \leq 2) &&\rightarrow b(0) \end{aligned}$$

When started with $a(4)$, the P0L-system produces the following sequence of strings,

$$a(4)$$
$$a(3)b(4)$$
$$a(2)b(3)b(2)a(3)$$
$$a(1)b(2)b(1.5)a(2)b(0)a(2)b(3)$$
$$a(0)b(0)b(0)a(1)b(2)b(0)a(1)b(2)b(1.5)a(2)$$
$$a(0)b(0)b(0)a(0)b(0)b(0)a(0)b(0)b(0)a(1)b(2)$$
$$a(0)b(0)b(0)a(0)b(0)b(0)a(0)b(0)b(0)a(0)b(0)$$

Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.

Parametric Lindenmayer systems

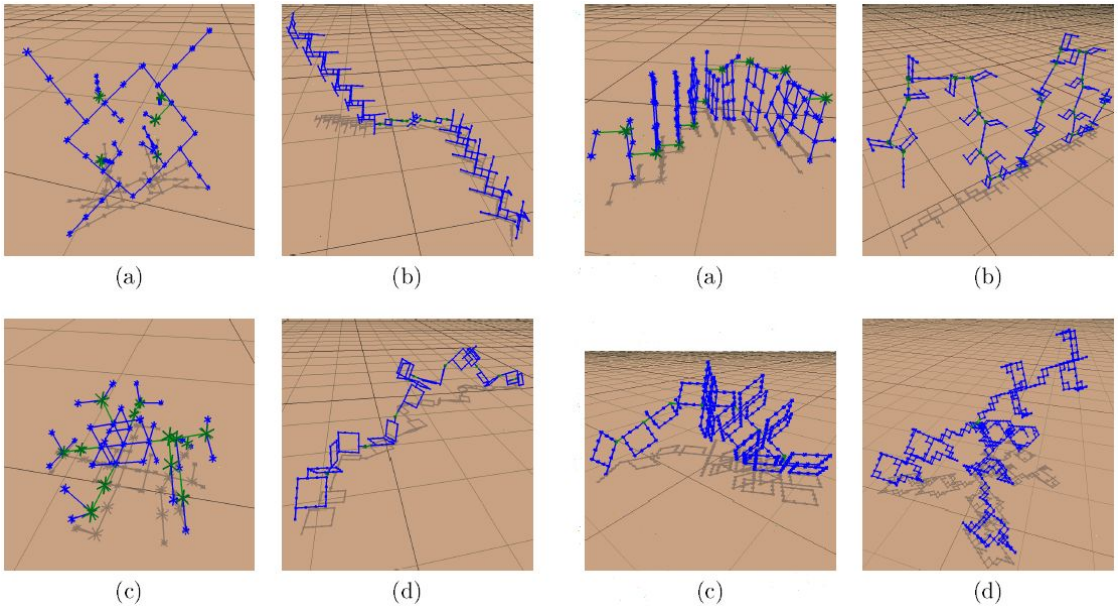
Advantages of a parametric L-system are a P0L-system can produce a family of structures, with the specific structure created being determined by the starting parameters. Similarly, parameters can be used so that repeating patterns of connections will have different weights. An example of a P0L-system for a network is,

$P0(n0)$:
$n0 > 1.0 \rightarrow \quad P1(n0)\ P0(n0-1)$
$n0 > 0.0 \rightarrow \quad loop(1)\ P1(1)\ parent(1)\ merge(1)$

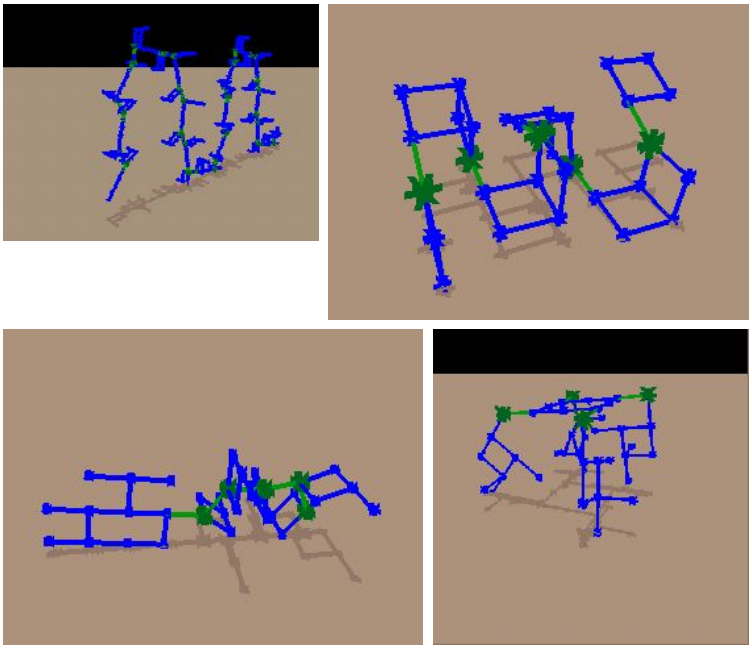
$P1(n0)$:
$n0 > 1.0 \rightarrow \quad split(0.8)\ duplicate(n0)\ reverse$
$n0 > 0.0 \rightarrow \quad \{split(0.6)\ duplicate(0.4)\}(2)\ reverse$

This L-system consists of two productions, each containing two condition-successor pairs and when started with P0(3) produces the sequence of four strings: *a*, *P1(3) P0(2)*; *b*, *split(0.8) duplicate(3) reverse P1(2) P0(1)*; *c*, *split(0.8) duplicate(3) reverse split(0.8) duplicate(2) reverse loop(1) P1(1) parent(1) merge(1)*; and *d*, *split(0.8) duplicate(3) reverse split(0.8) duplicate(2) reverse loop(1) {split(0.6) duplicate(0.4)} reverse parent(1) merge(1)*. This last is interpreted as: *split(0.8) duplicate(3) reverse split(0.8) duplicate(2) reverse loop(1) split(0.6) duplicate(0.4) split(0.6) duplicate(0.4) reverse parent(1) merge(1)*, and the network that it constructs is shown in figure 2.h.

Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.



Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.



Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.

Generative vs. Non-generative Genotype to phenotype Mappings

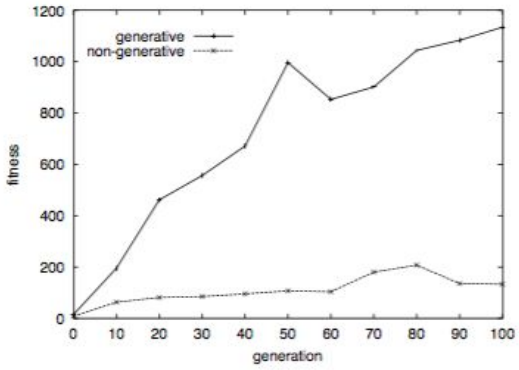


Figure 3: Generative vs. Non-generative Encodings

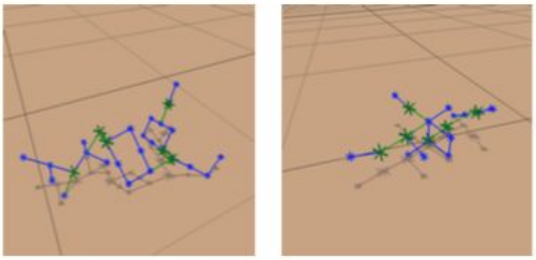
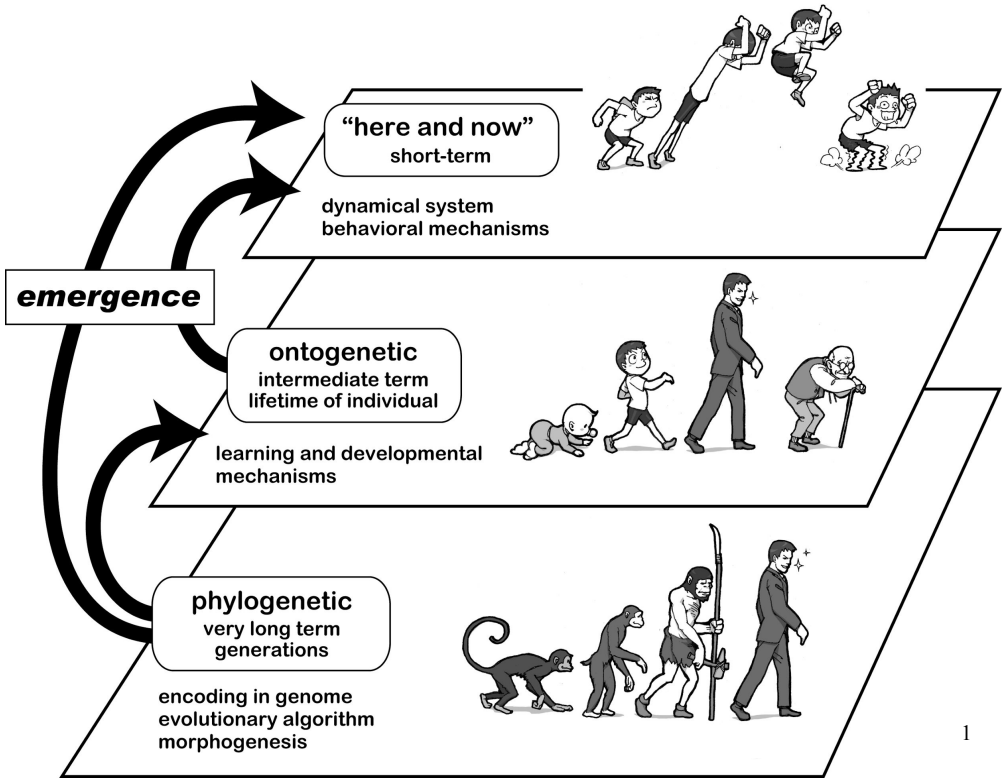
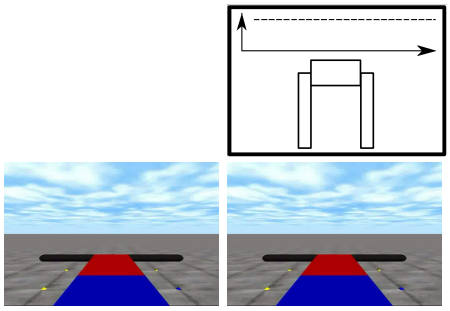


Figure 4: Results with Non-generative Encoding

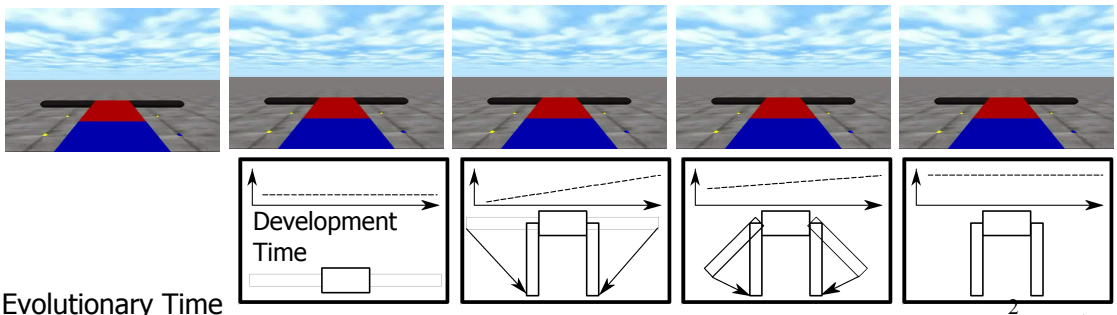
Hornby, G.S., Pollack, J. (2001) Creating high-level components with a generative representation for body-brain evolution. *GECCO 2001*.





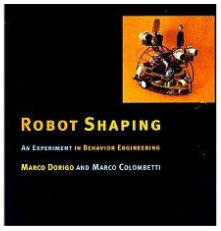
No Morphological Scaffolding

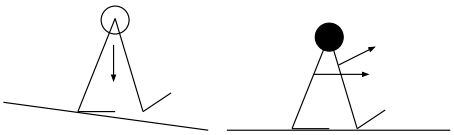
Morphological Scaffolding

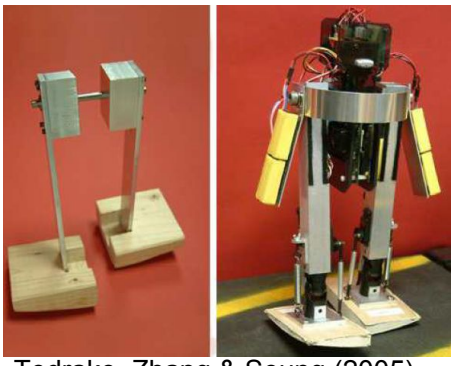


Evolutionary Time

Robot Shaping
(Dorigo & Colombetti, 1997)



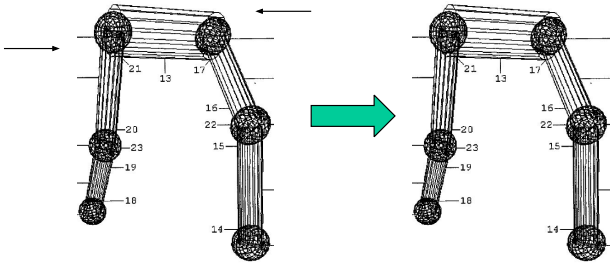


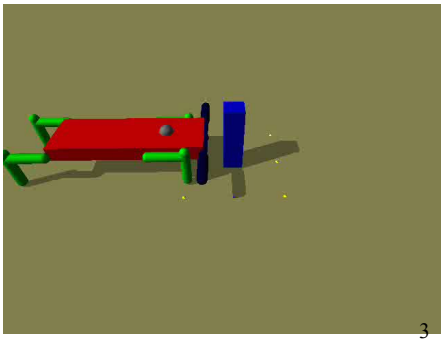


Tedrake, Zhang & Seung (2005)

Environmental Shaping

Reil & Husbands (2002)



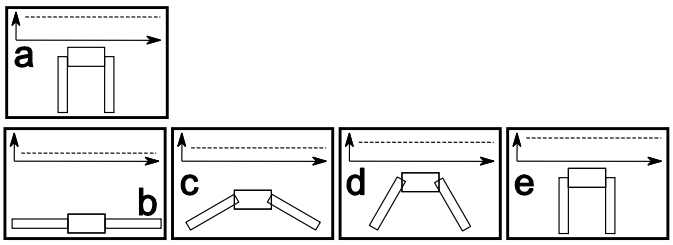


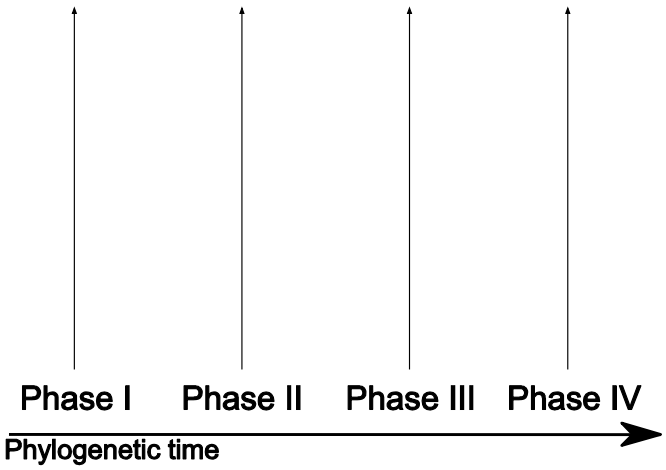
Bongard (2008)

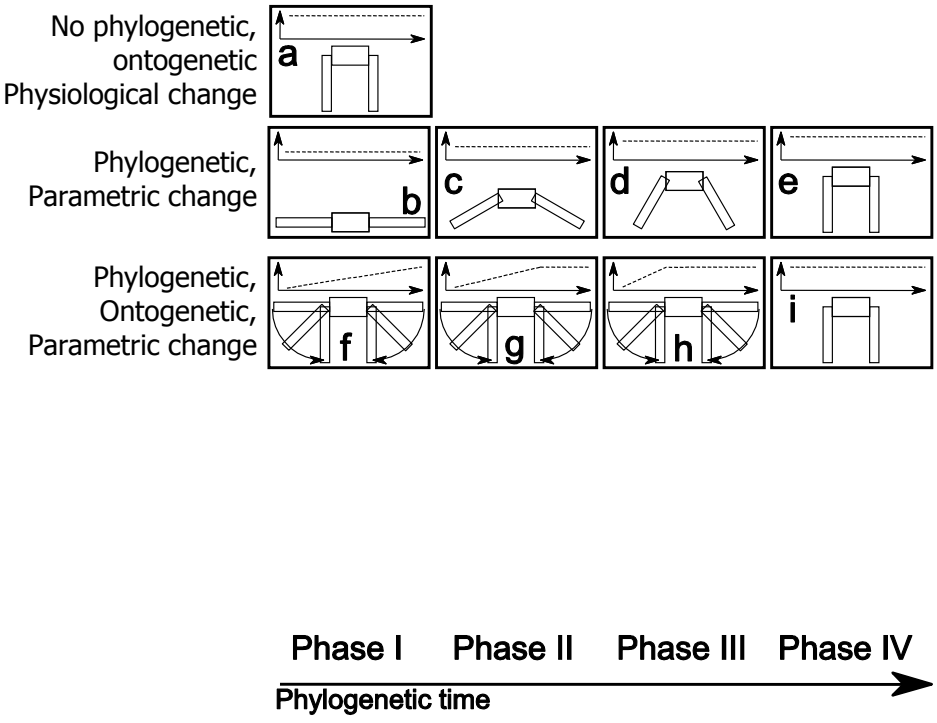
Why Evolve Morphology? L25

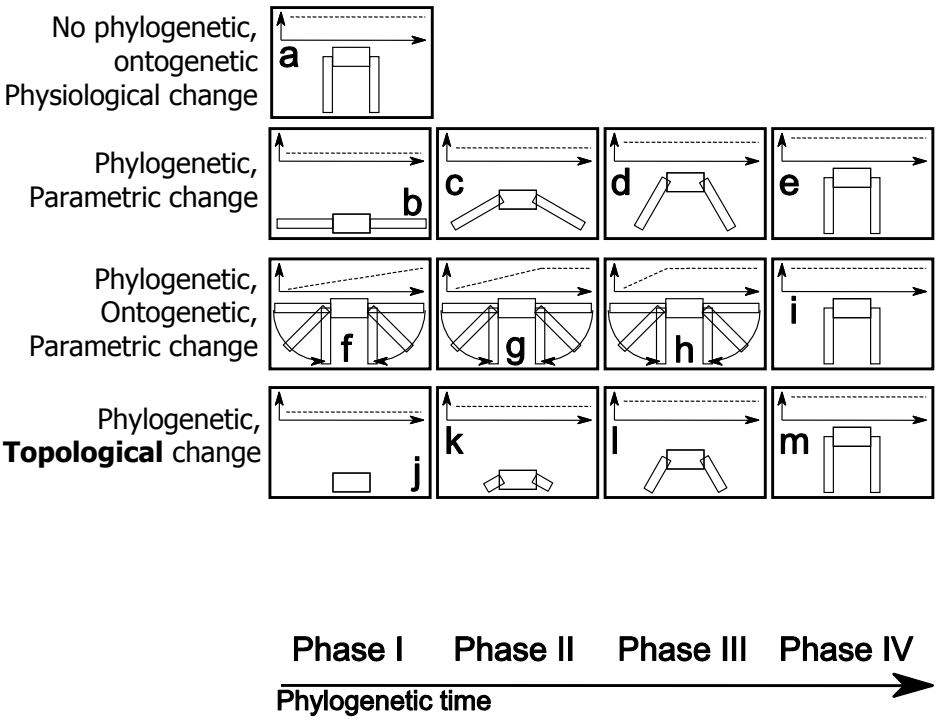
No phylogenetic, ontogenetic Physiological change

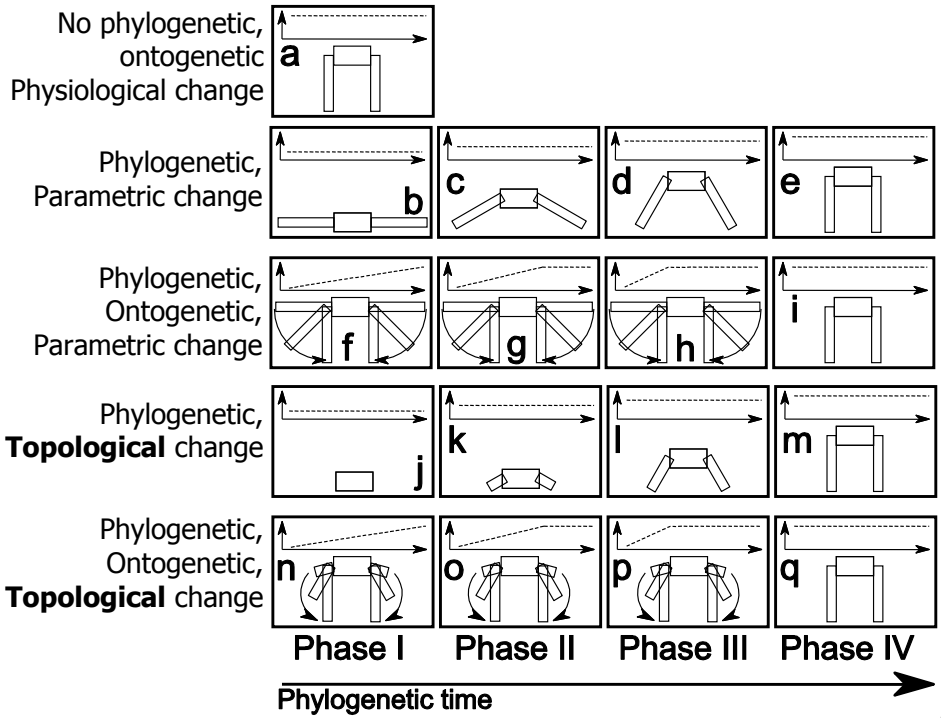
Phylogenetic, Parametric change

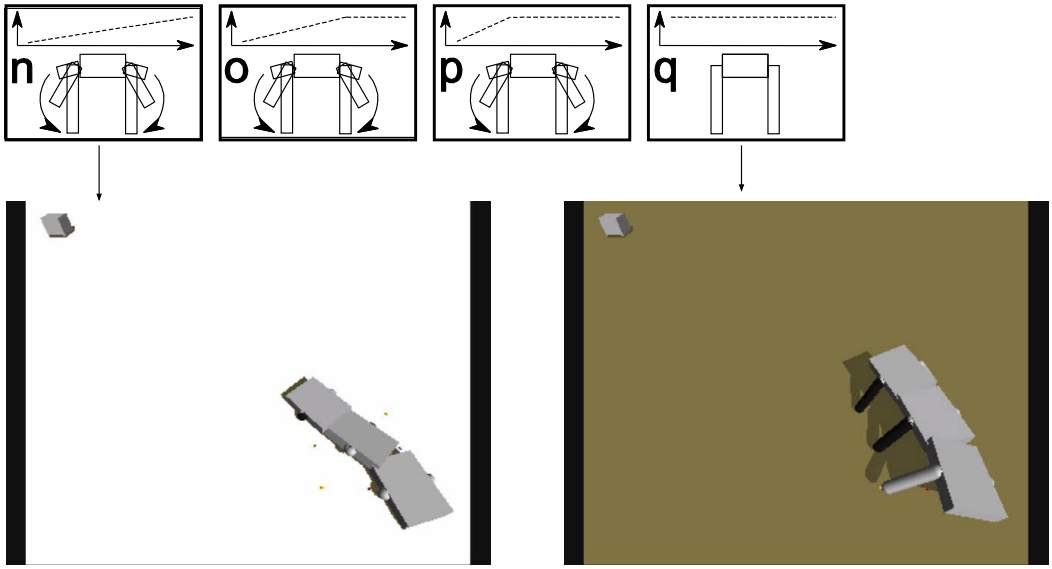




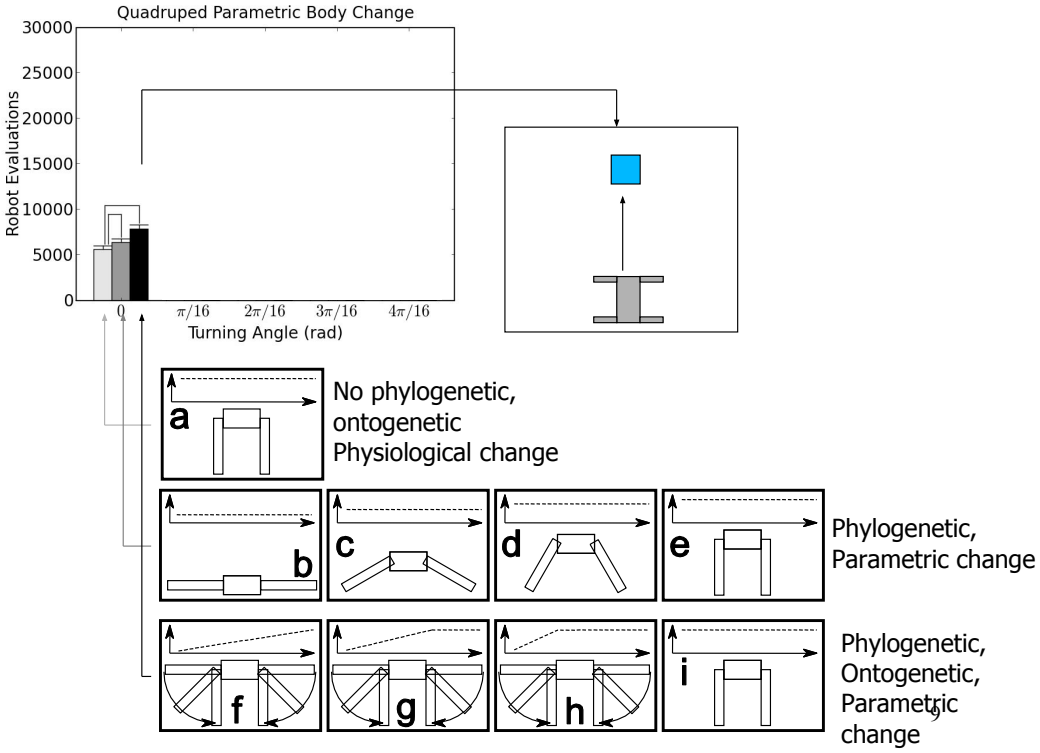

No phylogenetic, ontogenetic Physiological change
Phylogenetic, Parametric change
Phylogenetic, Ontogenetic, Parametric change
a
b
c
d
e
f
g
h
i
Phase I
Phase II
Phase III
Phase IV
Phylogenetic time


No phylogenetic, ontogenetic Physiological change
Phylogenetic, Parametric change
Phylogenetic, Ontogenetic, Parametric change
Phylogenetic, **Topological** change
a
b
c
d
e
f
g
h
i
j
k
l
m
Phase I
Phase II
Phase III
Phase IV
Phylogenetic time

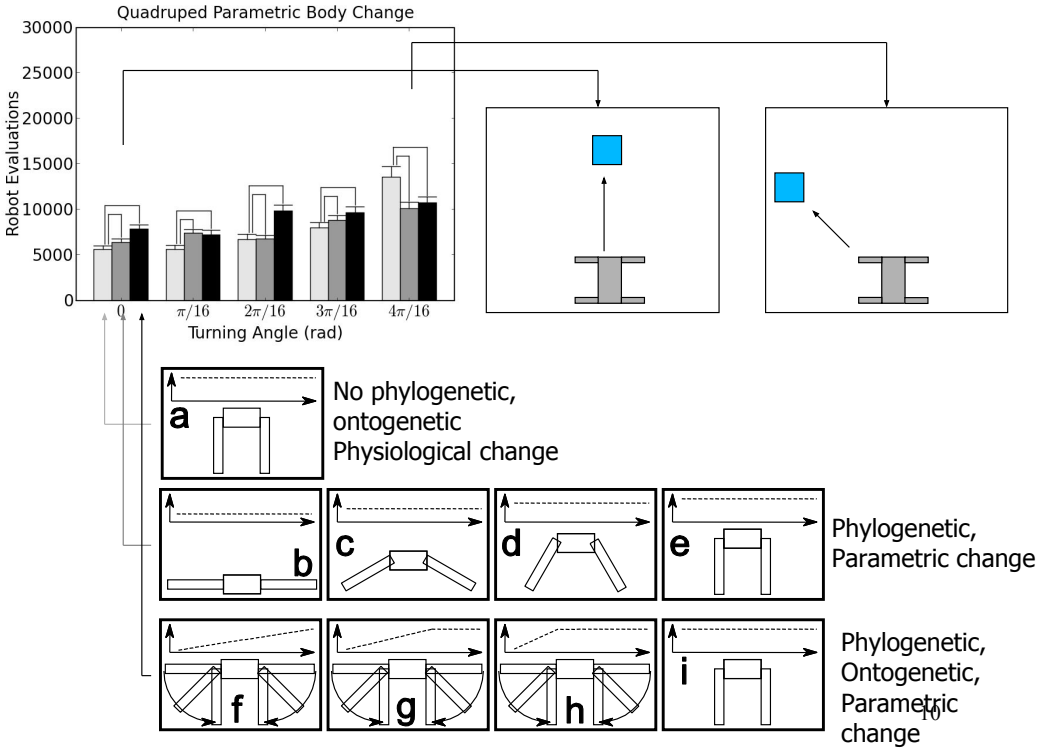

No phylogenetic,
ontogenetic
Physiological change
a
Phylogenetic,
Parametric change
b
c
d
e
Phylogenetic,
Ontogenetic,
Parametric change
f
g
h
i
Phylogenetic,
Topological change
j
k
l
m
Phylogenetic,
Ontogenetic,
Topological change
n
o
p
q
Phase I
Phase II
Phase III
Phase IV
Phylogenetic time

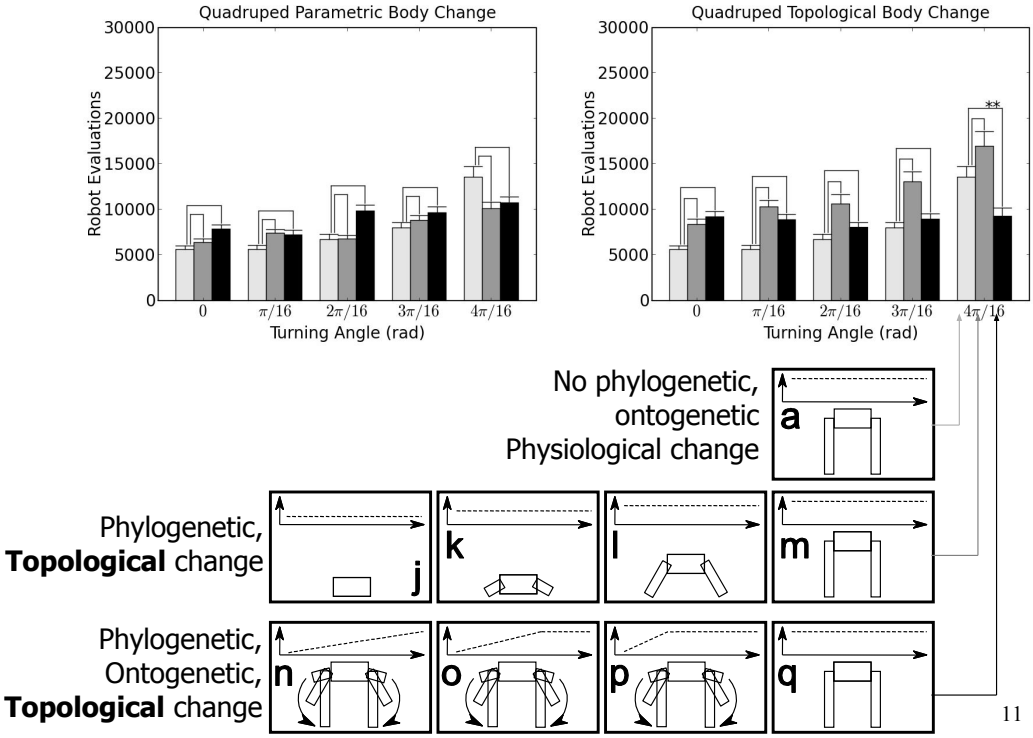

n
o
p
q

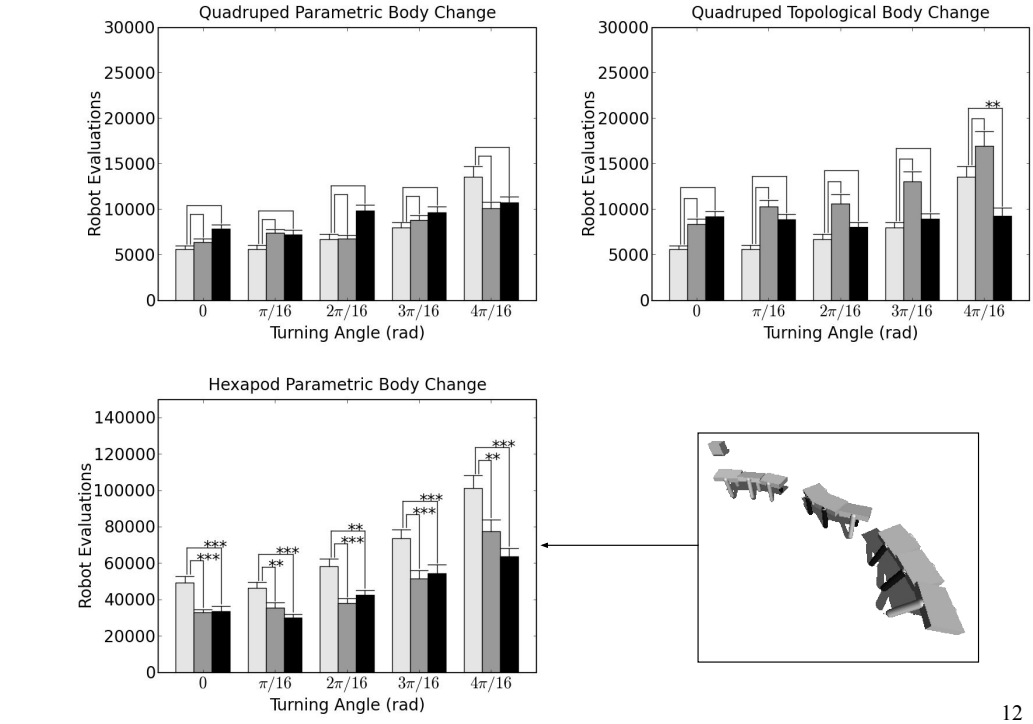
Why Evolve Morphology? L25



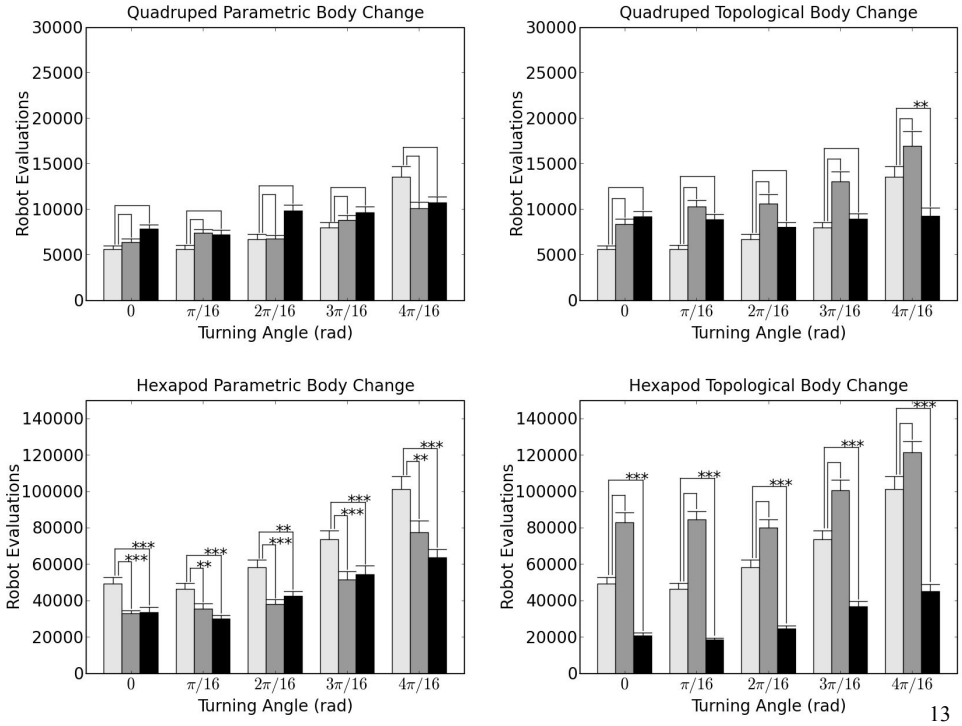
Why Evolve Morphology? L25




Quadruped Parametric Body Change
Quadruped Topological Body Change
Robot Evaluations
Turning Angle (rad)
0
π/16
2π/16
3π/16
4π/16
No phylogenetic,
ontogenetic
Physiological change
a
Phylogenetic,
Topological change
j
k
l
m
Phylogenetic,
Ontogenetic,
Topological change
n
o
p
q

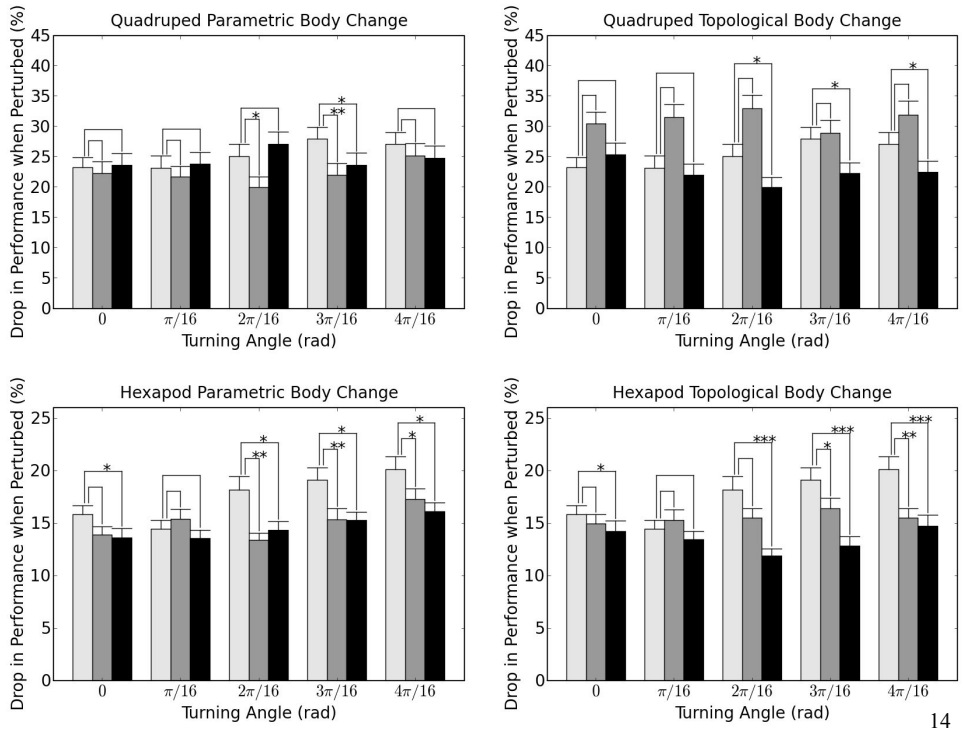

Quadruped Parametric Body Change
Quadruped Topological Body Change
Robot Evaluations
Turning Angle (rad)
Hexapod Parametric Body Change

Why Evolve Morphology? L25

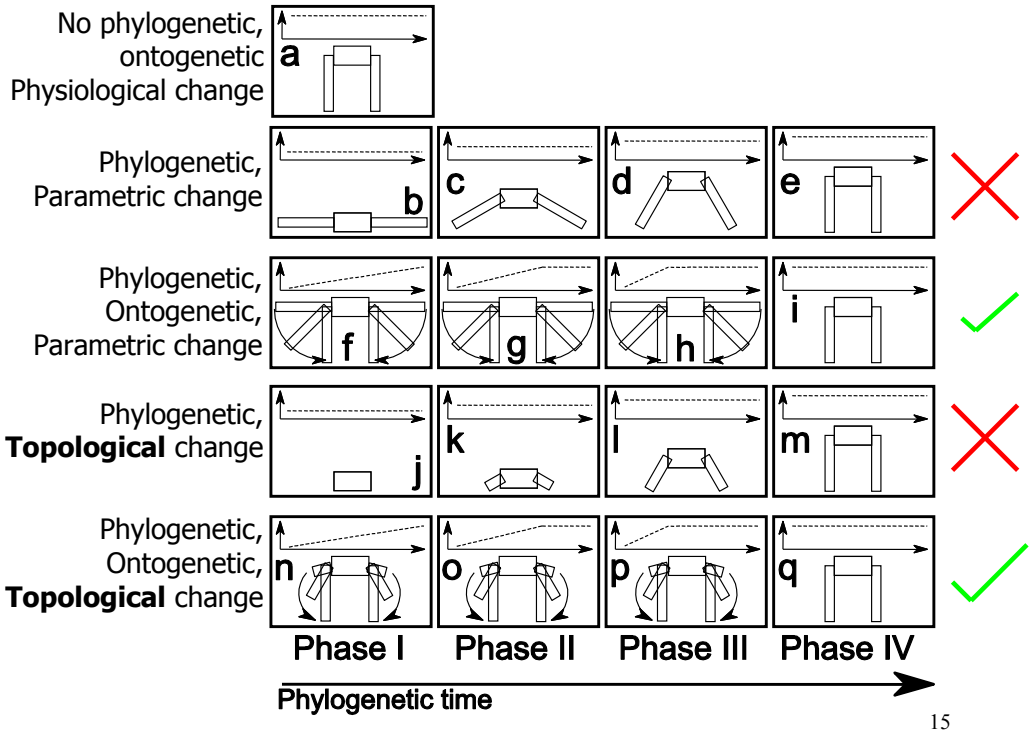


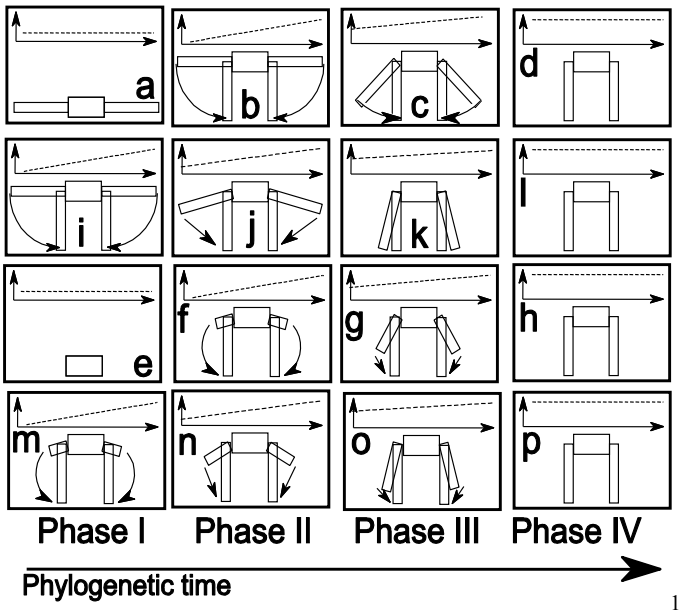
13

Why Evolve Morphology? L25

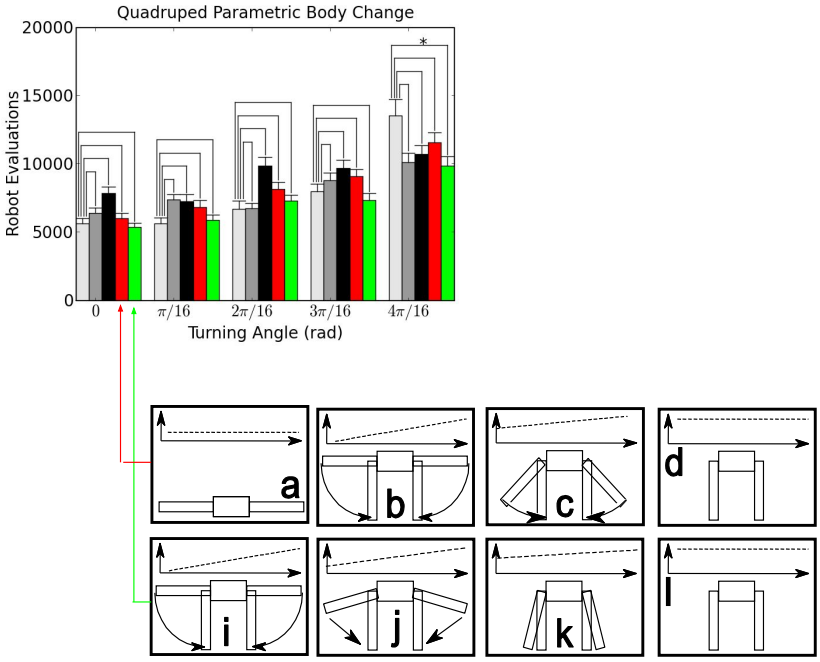


14

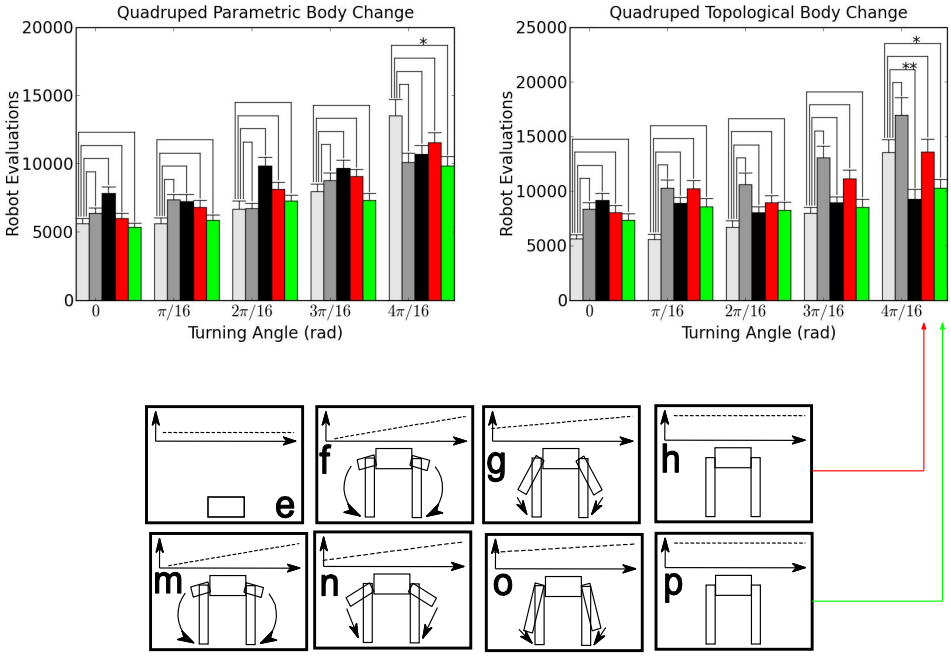

No phylogenetic, ontogenetic Physiological change
a
Phylogenetic, Parametric change
b
c
d
e
Phylogenetic, Ontogenetic, Parametric change
f
g
h
i
Phylogenetic, **Topological** change
j
k
l
m
Phylogenetic, Ontogenetic, **Topological** change
n
o
p
q
Phase I
Phase II
Phase III
Phase IV
Phylogenetic time


a
b
c
d
i
j
k
l
e
f
g
h
m
n
o
p
Phase I
Phase II
Phase III
Phase IV
Phylogenetic time

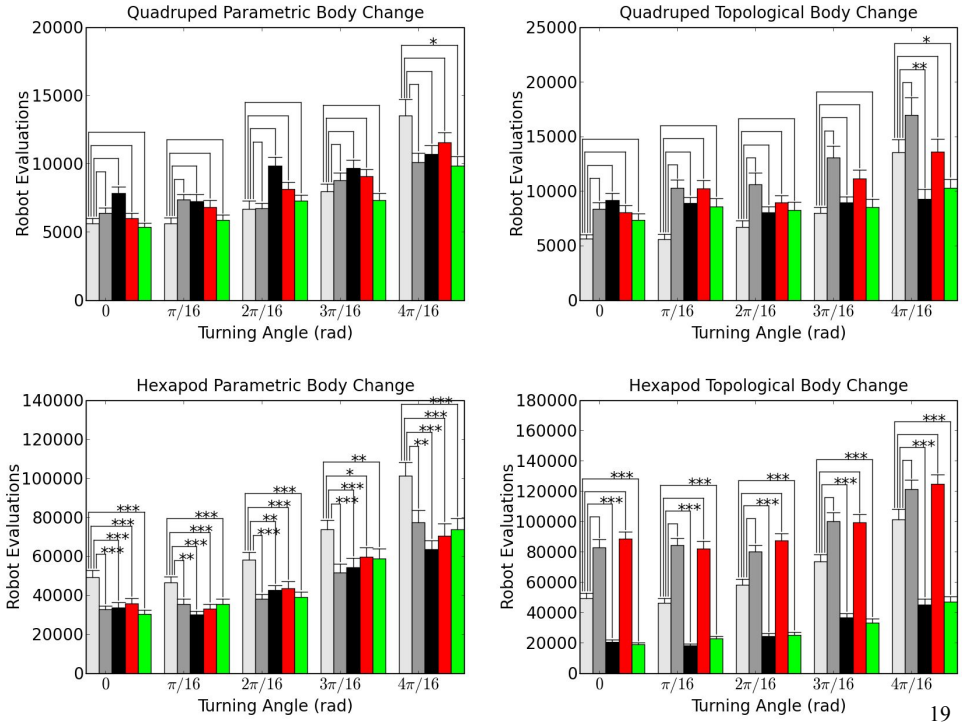
Why Evolve Morphology? L25



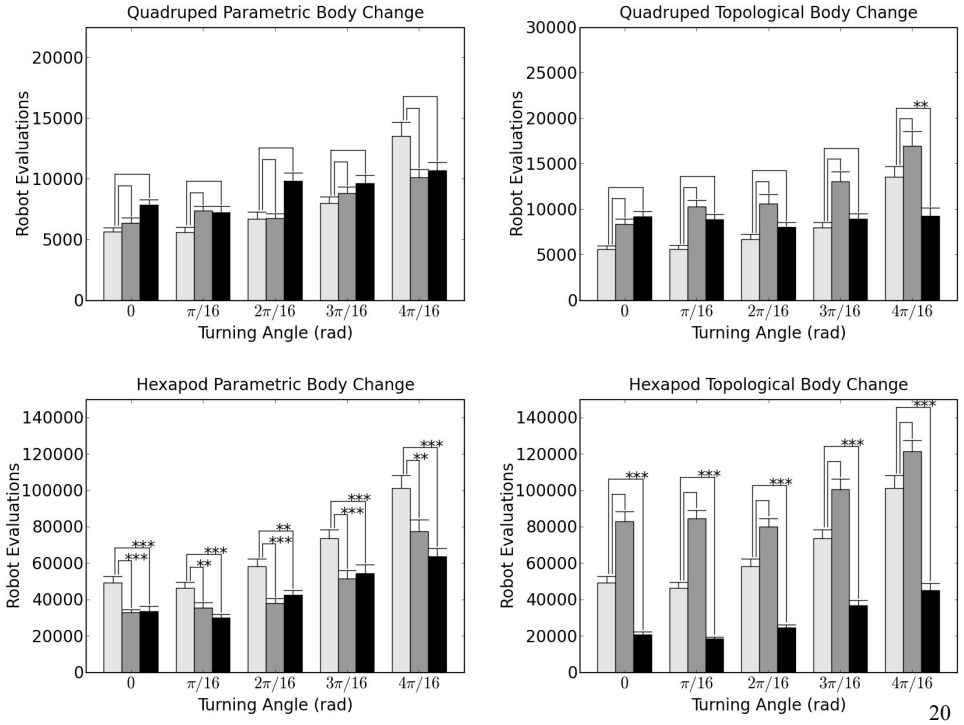
Why Evolve Morphology? L25



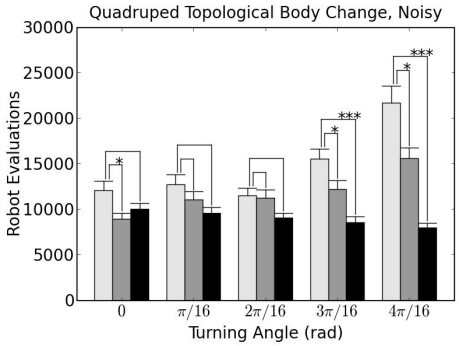
Why Evolve Morphology? L25



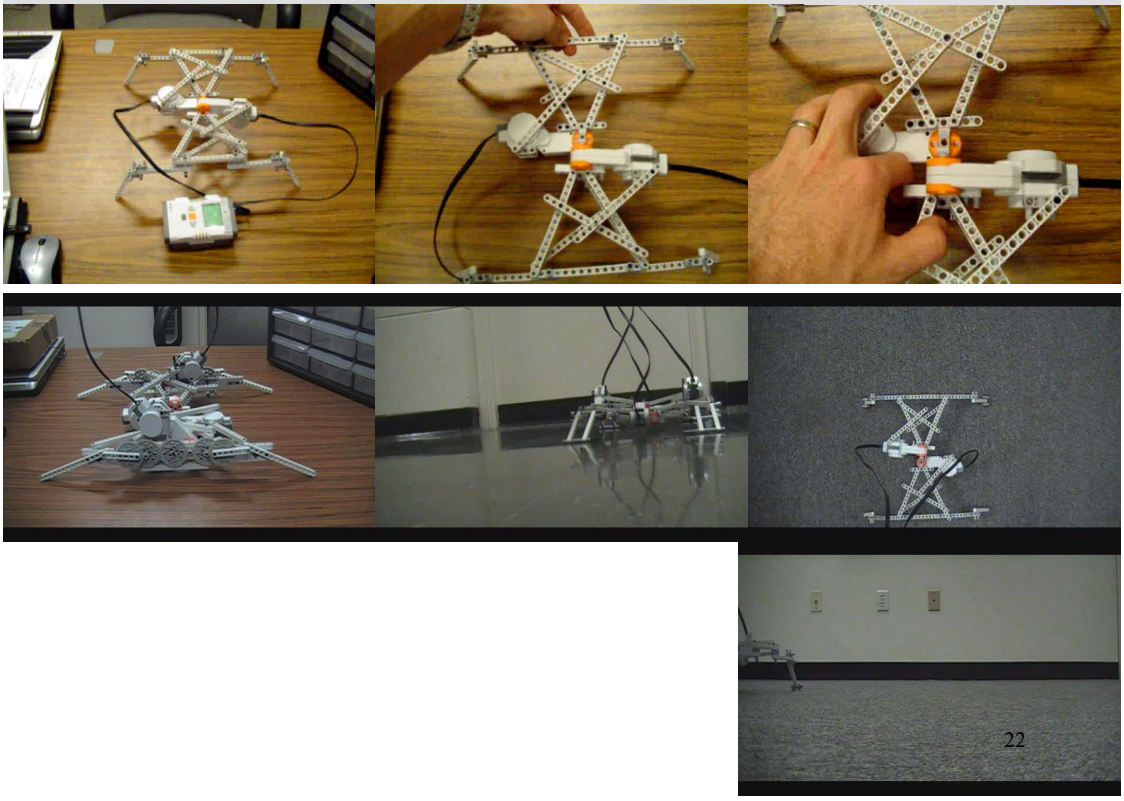
Why Evolve Morphology? L25



Why Evolve Morphology? L25



Why Evolve Morphology? L25



Robots that can adapt like animals.

Video Summary:

http://youtu.be/T-c17RKh3uE

Example gaits:

http://youtu.be/IHQgnpSphEI

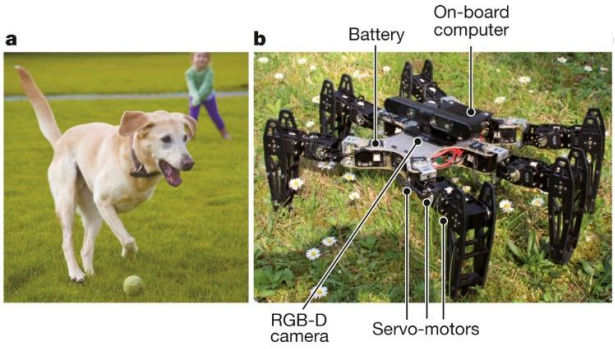
A Cully *et al*. *Nature* **521**, 503-507 (2015) doi:10.1038/nature14422

Using the Intelligent Trial and Error (IT&E) algorithm, robots, like animals, can quickly adapt to recover from damage.

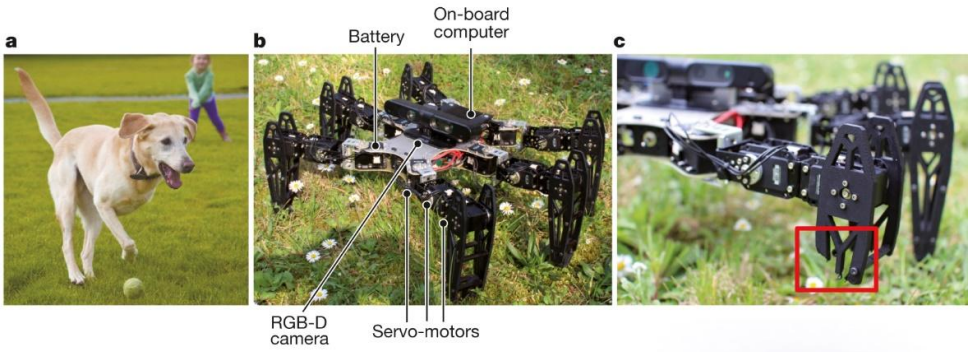
a

A Cully *et al*. *Nature* **521**, 503-507 (2015) doi:10.1038/nature14422

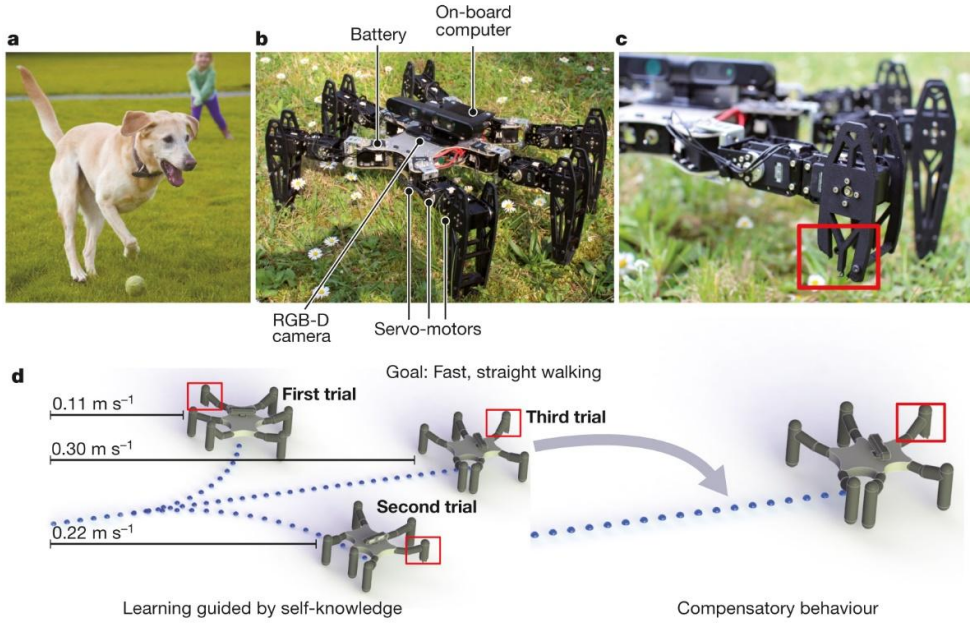
Using the Intelligent Trial and Error (IT&E) algorithm, robots, like animals, can quickly adapt to recover from damage.



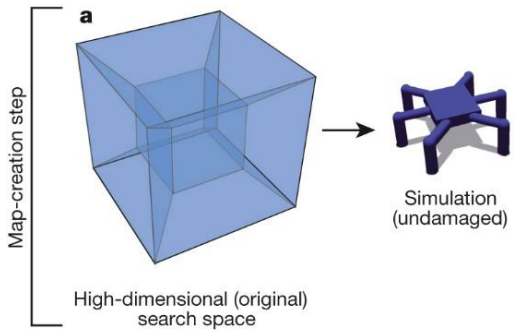
Using the Intelligent Trial and Error (IT&E) algorithm, robots, like animals, can quickly adapt to recover from damage.



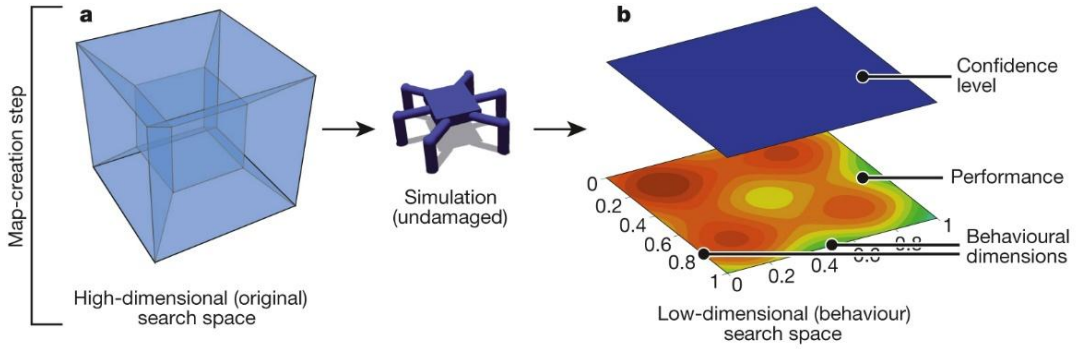
Using the Intelligent Trial and Error (IT&E) algorithm, robots, like animals, can quickly adapt to recover from damage.



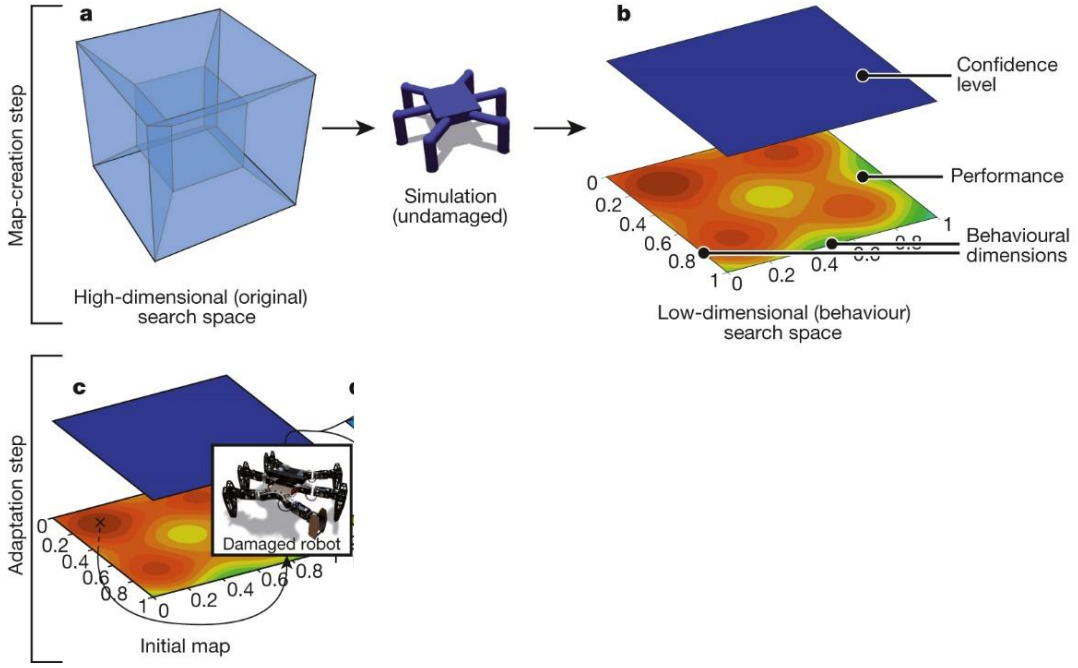
The two steps of IT&E. L26



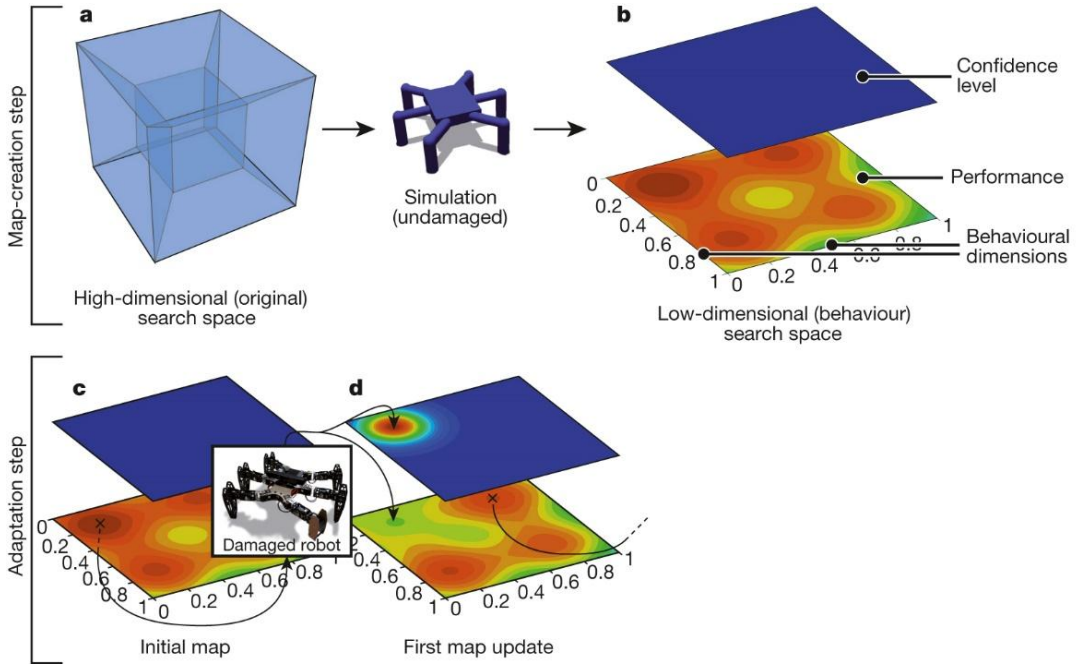
The two steps of IT&E. L26



The two steps of IT&E. L26



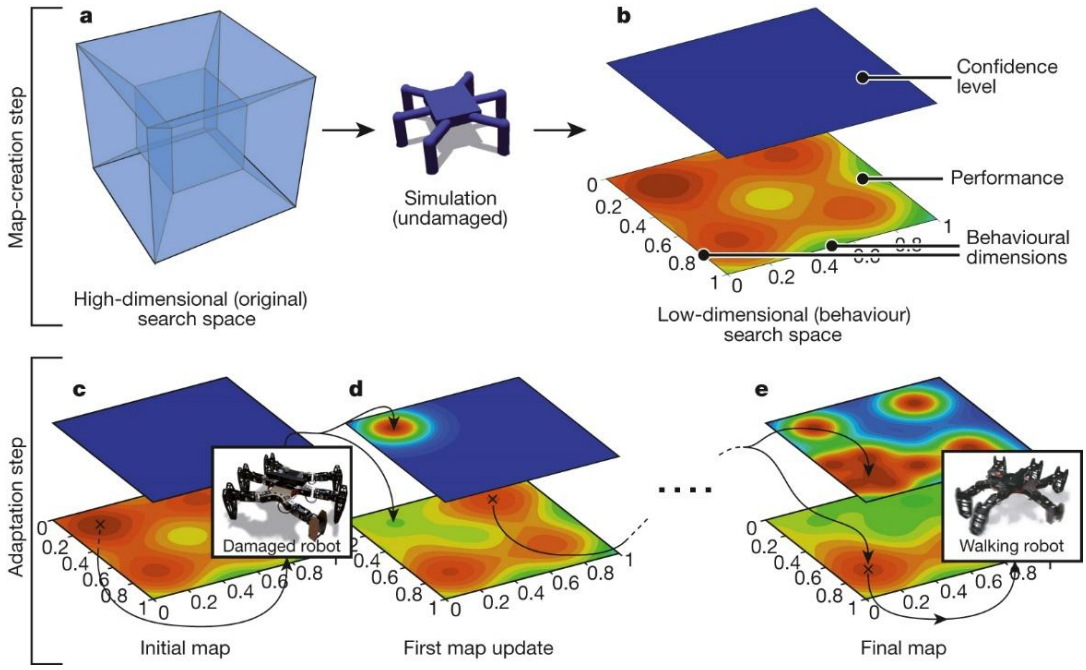
The two steps of IT&E. L26



A Cully *et al*. *Nature* **521**, 503-507 (2015) doi:10.1038/nature14422

nature

The two steps of IT&E. L26

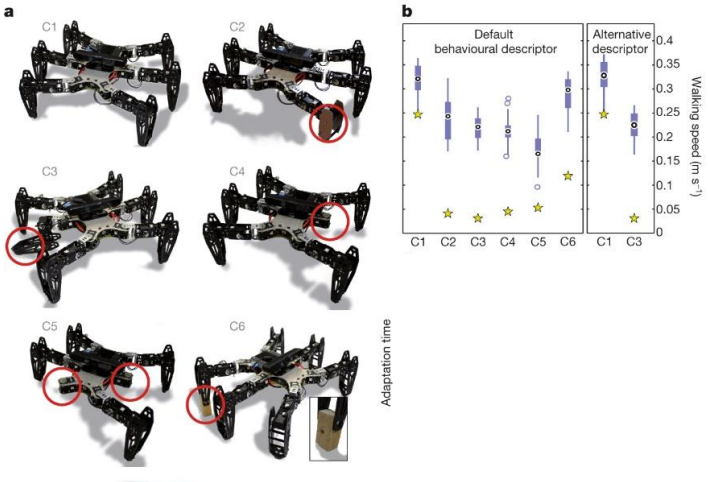


A Cully *et al*. *Nature* **521**, 503-507 (2015) doi:10.1038/nature14422

nature

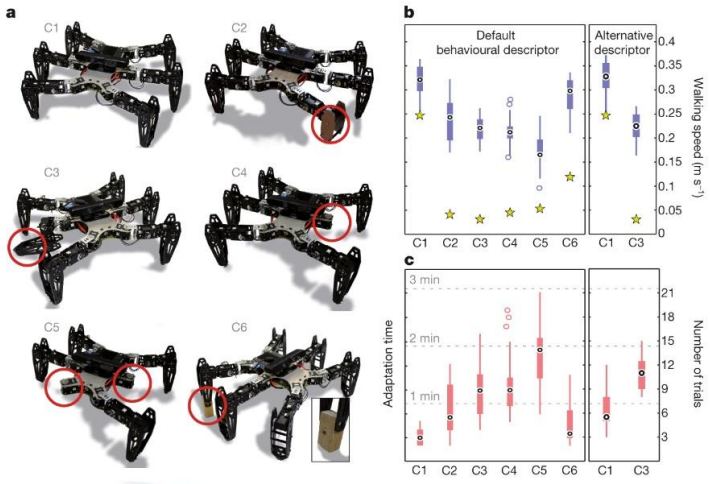
Main experiments and results. L26

Main experiments and results. L26



Main experiments and results.

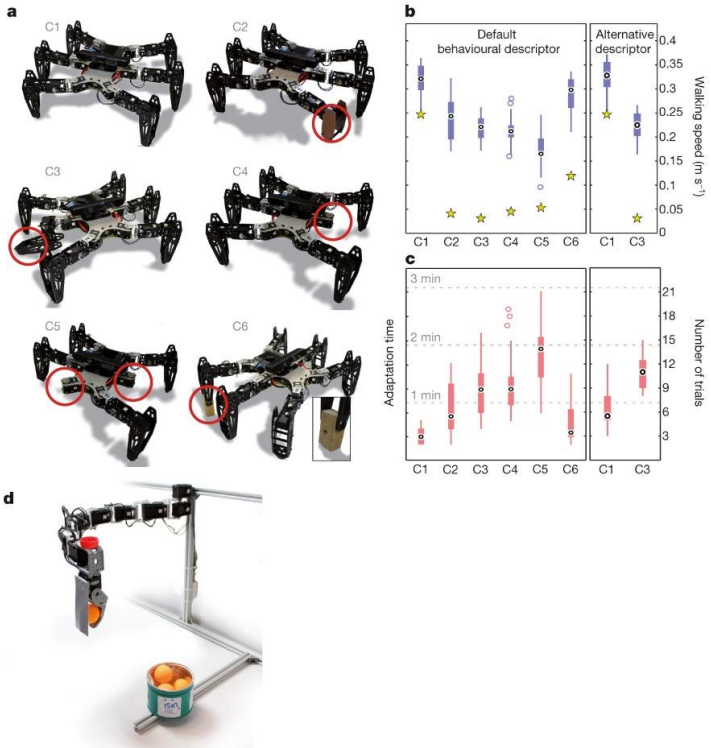
L26



A Cully *et al*. *Nature* **521**, 503-507 (2015) doi:10.1038/nature14422

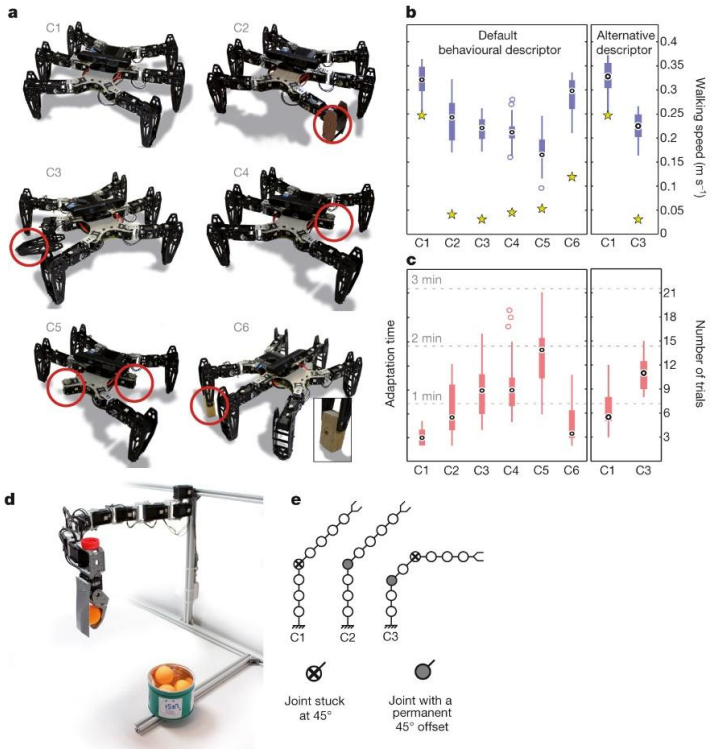
Main experiments and results.

L26



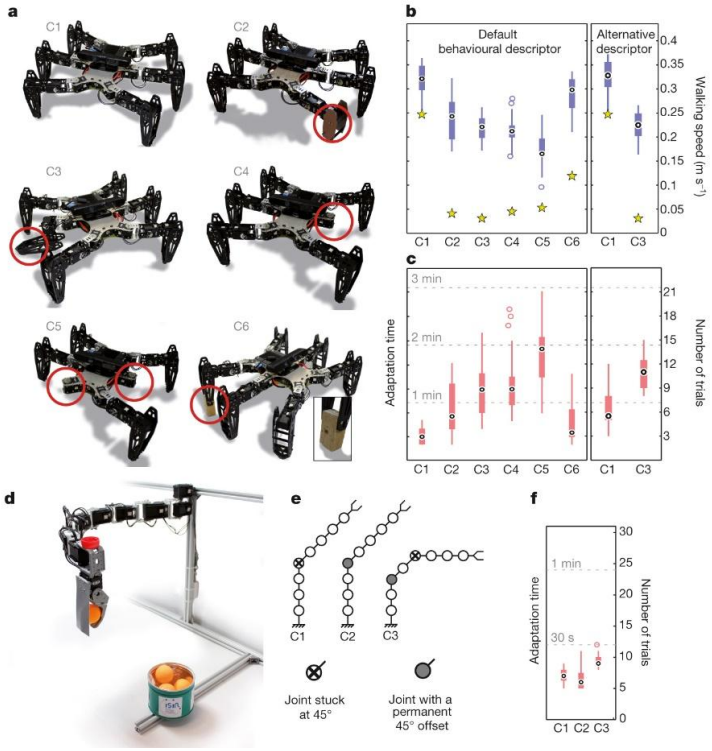
A Cully *et al*. *Nature* **521**, 503-507 (2015) doi:10.1038/nature14422

nature



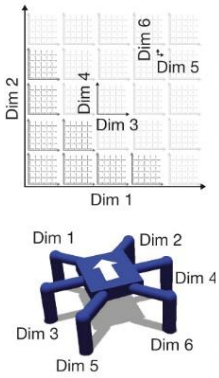
A Cully *et al*. *Nature* **521**, 503-507 (2015) doi:10.1038/nature14422

Main experiments and results. L26

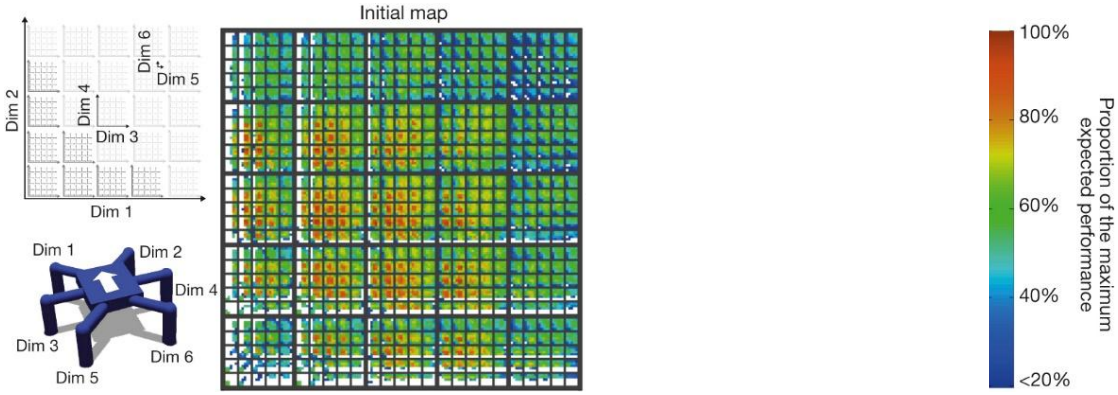


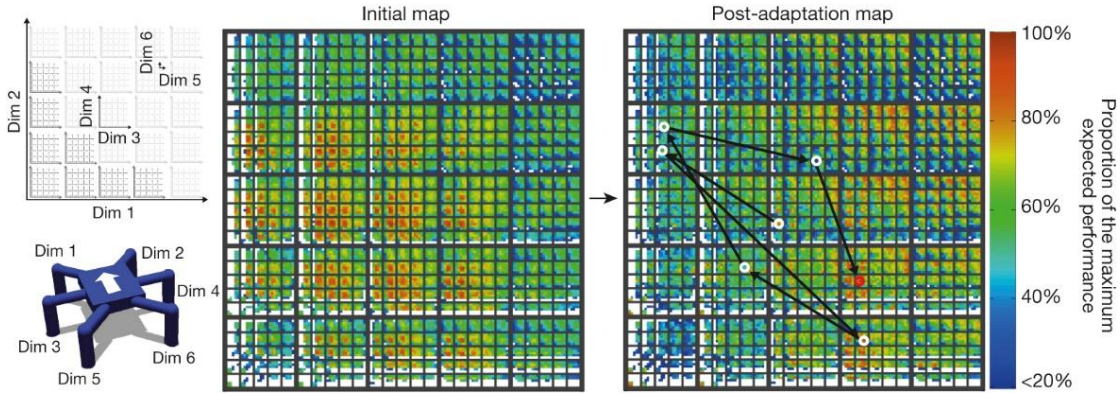
A Cully *et al*. *Nature* **521**, 503-507 (2015) doi:10.1038/nature14422

An example behaviour–performance map L26



An example behaviour–performance map. L26




Initial map
Post-adaptation map
Dim 1
Dim 2
Dim 3
Dim 4
Dim 5
Dim 6
100%
80%
60%
40%
<20%
Proportion of the maximum expected performance