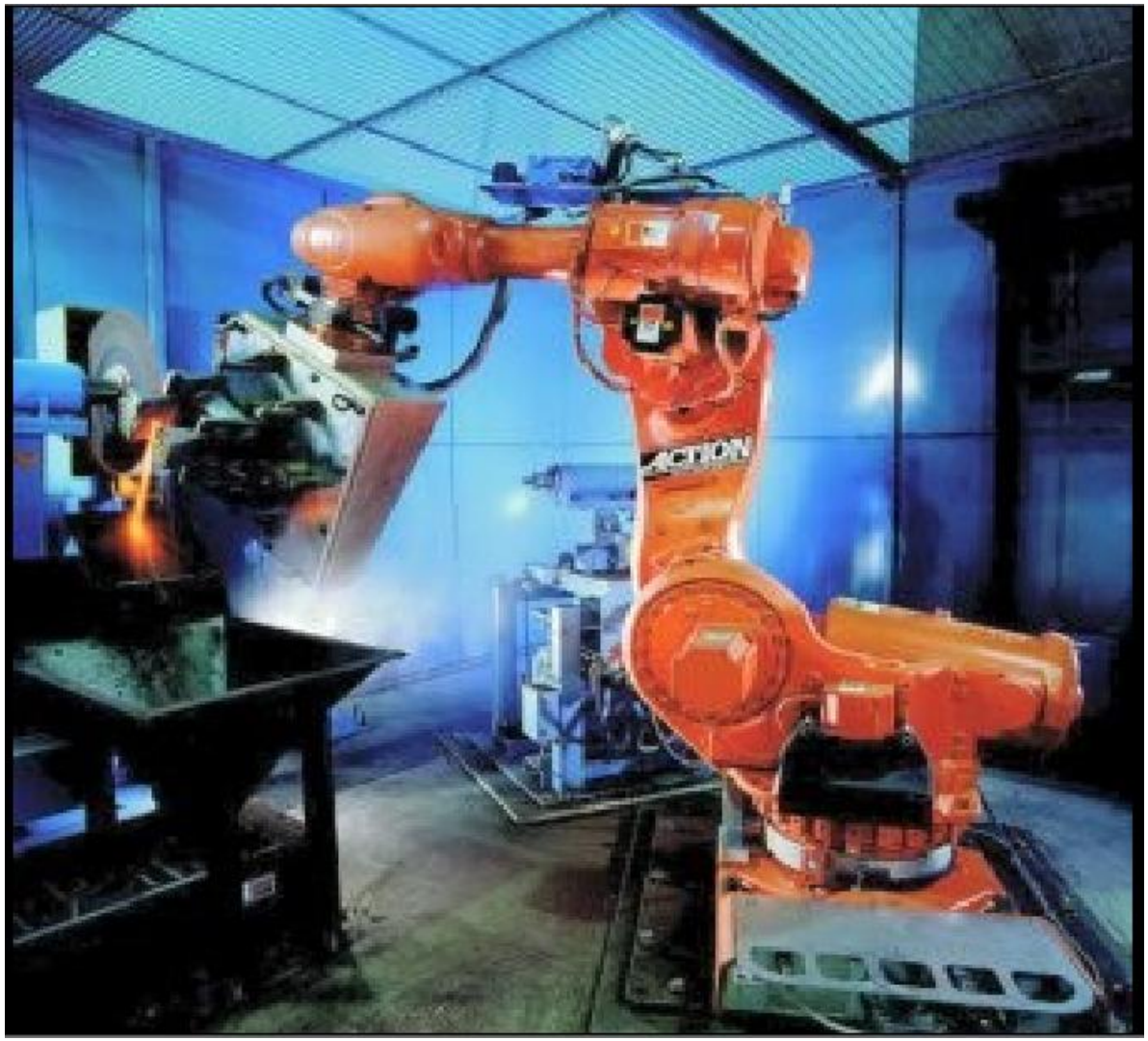
Master Recherche IAC
Robots et agents autonomes

Jamal Atif – Michèle Sebag
TAO
CNRS – INRIA – LRI, Université Paris-Sud

Jan. 11th, 2013

Case 1. Optimal control



Case 1. Optimal control, foll'd

Known dynamics and target behavior

1. state u, action $a \rightarrow$ new state u'
2. wanted: sequence of states

Approaches

- Inverse problem
- Optimal control

Challenges

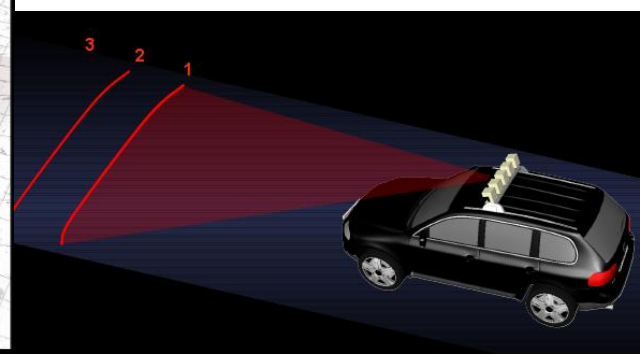
- Model errors, uncertainties
- Stability

Case 2. Reactive behaviors

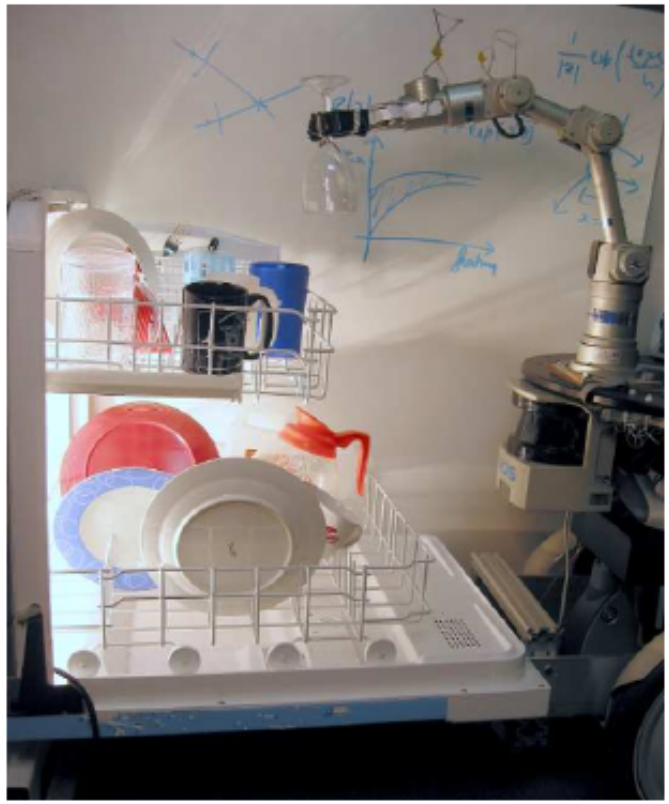
The 2005 Darpa Challenge

The terrain

The sensors



Case 3. Planning



An instance of reinforcement learning / planning problem

1. Solution = sequence of (state,action)
2. In each state, decide the appropriate action
3. ..such that in the end, you reach the goal

Case 3. Planning, foll'd

Approaches

- Reinforcement learning
- Inverse reinforcement learning
- **Direct policy search (= optimize the controller)**
  - **Evolutionary robotics**
  - **Preference-based RL**

Challenges

- Design the objective function (define the optimization problem)
- Solve the optimization problem
- Assess the validity of the solution

Overview

Policy search, formal background

Assumption

- We know the policy search space π: State $\mapsto$ Action
  For instance: Neural Nets, Decision list
- This search space Θ is parametric $\equiv \mathbb{R}^d$
- There exists a computable objective function to be optimized:

$$\theta \mapsto \pi_\theta \mapsto \text{ behavior } \mapsto \mathcal{F}(\theta)$$

An optimization problem

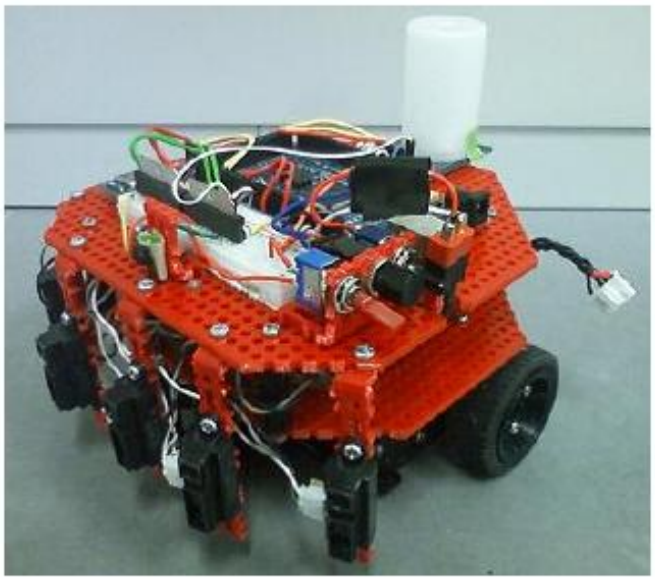
$$\text{Find } \theta^* = \mathit{argmax}\{\mathcal{F}(\theta\}$$

Specificities

- Noisy optimization (actuators, motors) and partially observable setting
- Can (must) incorporate prior knowledge
  search space structure; initialization; objective function

Example:
swarm robots moving in column formation

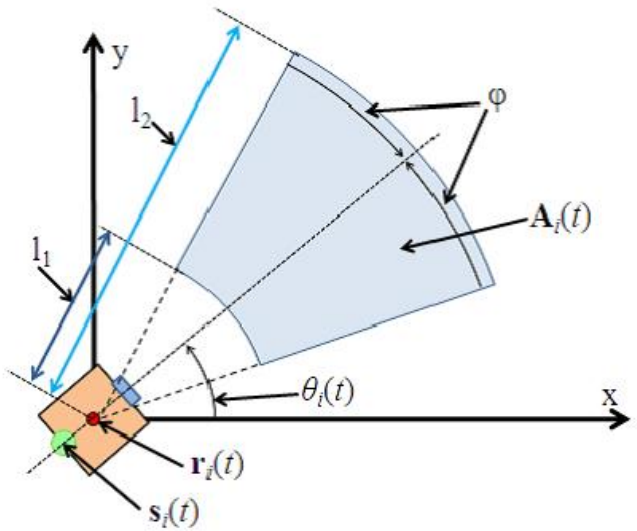
Robot



Example, foll'd

Representation

Constants		
	l1	blind zone
	l2	sensor range
	ϕ	Vision angular range
Variables(t)		
	$r(t), s(t)$	positions
	$\theta(t)$	angular direction



Example of a (almost manual) controller

CONTROLLER OF A ROBOT

Info. from the image sensors	Info. from the IR sensors		
	$0 \leqq x_{\text{IR}} < \beta_0$	$\beta_0 \leqq x_{\text{IR}} < \beta$	$\beta \leqq x_{\text{IR}}$
$0 \leqq x_{\text{image}} \leqq \alpha$	move backward or turn right	turn left	
$\alpha < x_{\text{image}} < (19 - \alpha)$	move backward or turn right	stop	move forward
$\alpha \leqq x_{\text{image}} \leqq 19$	move backward or turn right	turn right	
preceding robot NOT FOUND	move backward or turn right	move forward	

Toward defining $\mathcal{F}$

- The i-th robot follows the k-th robot at time t iff the center of gravity of k belongs to the perception range of i ($\mathbf{s}_k(t) \in \mathbf{A}_i(t)$).
- The i-th robot is a leader if i) it does not follow any other robot; ii) there exists at least one robot following it.
- A column is a subset $\{i_1, \ldots i_K\}$ such that robot i_{k+1} follows robot i_k and robot i_1 is a leader.
- A deadlock is a subset $\{i_1, \ldots i_K\}$ such that robot i_{k+1} follows robot i_k and robot i_1 follows robot i_K.

Milestones

1. From θ to π_θ trivial
2. From π_θ to the robot behavior
3. From the robot behavior to evaluating $\mathcal{F}(\theta)$
4. From trials $(\{(\theta_t, \mathcal{F}(\theta_t))\}$ to θ^*

Milestone 1
From the controller π_θ to the robot behavior

How

- In silico = in simulation
  - Main approach for evolutionary robotics
  - No way, says the roboticist **reality gap**
- In situ: embeds the policy on the robot, and sees.
  - The robot breaks before long
  - Makes it difficult to compute $\mathcal{F}(\theta)$.
- Both Hod Lipson & Bongard 2006

Milestone 1

Bottleneck: Accurate predictions

- World model: what is out there.
  SLAM, Simultaneous Localization and Mapping
  Long term planning

- Forward model: what will happen if robot selects action a in state s
  Local model of itself **Short term planning**

- Uncertainties about e.g. sensors or actuators models, initial localization.

Milestone 1

Bottleneck: Accurate predictions, follow'd,

- Partially observable effects
  ex., in the case of swarms: there are many robots
  does robot *Bob* know robot *Alice*'s plans ? If yes, centralized resolution
  Else, *Alice*'s behavior is impredictible (and *Bob* can't predict with certainty what will be in his vision cone).
- $\rightarrow$ non deterministic model.
  thus, the behavior is a random variable; $\mathcal{F}(\theta)$ becomes an expectation,

$$\mathbb{E}_{\sim\pi_\theta}[\mathcal{F}(\text{behavior})]$$

Milestone 2
From the robot behavior to $\mathcal{F}(\theta)$

How

- In simulation: define computable $\mathcal{F}$
  by trials and errors (fitness shaping)
  manual (see section evolutionary robotics)
- In situ:
  - Interactive
  - Manual
  - Measurements (e.g. data mining on the videos).

Milestone 3
Optimisation

How

- Gradient-based approaches — Direct Policy Search
- Black-box optimization — Evolutionary Robotics
- Surrogate optimization — Preference reinforcement learning

What is optimized

- policy $\equiv \theta$
- Value function. (satisfies Bellman equation)
- Energy function $H(s, a)$ (same use, but no Bellman)

$$\pi(s) = argmax_a\{H(s.a)\}$$

Overview

Direct policy search, formal background

Assumption

- Function $\mathcal{F}(\theta)$, to be optimized

Pros

- No divergence, even under function approximation
- Policies are much simpler to represent
  (a neural net)
- Partial observability does not hurt convergence
  increases computational cost and harms long-term value

Cons

- Lost convergence to the globally optimal policy
- Lost the Bellman constraint $\rightarrow$ larger variance

Direct policy search, principles

Recall: Policy return estimate

$$V(s) = \mathbb{E}[\sum_t \gamma^t r(s_t)|s_0 = s]$$

or long term average reward

$$V(s) = lim_{T\to\infty} \frac{1}{T} \mathbb{E}[\sum_t r(s_t)|s_0 = s]$$

Assumption: ergodic Markov chain

(After a while, the initial state does not matter).

- $V(s)$ does not depend on s
- One can estimate the percentage of time spent in state s

$$q(\theta, s) = Pr_\theta(S = s)$$

Another policy return estimate

expected average reward

$$V = \mathbb{E}_\theta[r(S)] = \sum_s r(s) q(\theta, s)$$

Direct policy search, Algorithm

1. $\mathcal{F}(\theta) = \mathbb{E}_\theta[r(S)] = \sum_s r(s)q(\theta, s)$
2. Compute or estimate the gradient, $\nabla\mathcal{F}(\theta)$
3. Use it: (can do better)

$$\theta_{t+1} = \theta_t + \alpha\nabla\mathcal{F}(\theta)$$

Computing the derivative

$$\nabla V = \nabla(\sum_s r(s)q(\theta, s)) = \sum_s r(s)\nabla q(\theta, s)$$

Then:

$$\nabla V = \mathbb{E}_\theta[r(S)\frac{\nabla q(\theta, S)}{q(\theta, S)}]$$

Unbiased estimate of the gradient ($\widehat{\text{integral}}$ = empirical sum)

$$\hat{\nabla} V = \frac{1}{N}\sum_i r(s_i)\frac{\nabla q(\theta, s_i)}{q(\theta, s_i)}$$

Computing the derivative, foll'd

Using trajectories $((s_t, r(s_t)))$:
Given observations et rewards,

$$\frac{\nabla q(\theta, s_t)}{q(\theta, s_t)} = \sum_{i=0}^{t-1} \frac{\nabla p_\theta(s_i, s_{i+1})}{p_\theta(s_i, s_{i+1})}$$

where $p_\theta(s_i, s_j)$ is the probability of going from s_i to s_j with π_θ.

Eligibility trace

$$z_0 = 0; \qquad z_t = z_{t-1} + \frac{\nabla p_\theta(s_{t-1}, s_t)}{p_\theta(s_{t-1}, s_t)}$$

Computing the derivative, foll'd

Approximations truncated: biased

$$z_t = \sum_{k=t-n}^{t-1} \frac{\nabla p_\theta(s_k, s_{k+1})}{p_\theta(s_k, s_{k+1})}$$

or

$$z_t = \beta z_{t-1} + \frac{\nabla p_\theta(s_{t-1}, s_t)}{p_\theta(s_{t-1}, s_t)}$$

Quality

$$\hat{\nabla}_\beta V = \frac{1}{T} \sum_t r(s_t) z_t$$

Baxter Bartlett 2001

$$lim_{\beta \to 1} \hat{\nabla}_\beta V = \nabla V$$

Role of β: tradedoff bias/variance.

Discussion

Pros

- Many achievements: fine manipulation (peg-in-hole), learning biped walking with integrated trajec- tory generation and execution, first results using a real humanoid robot.

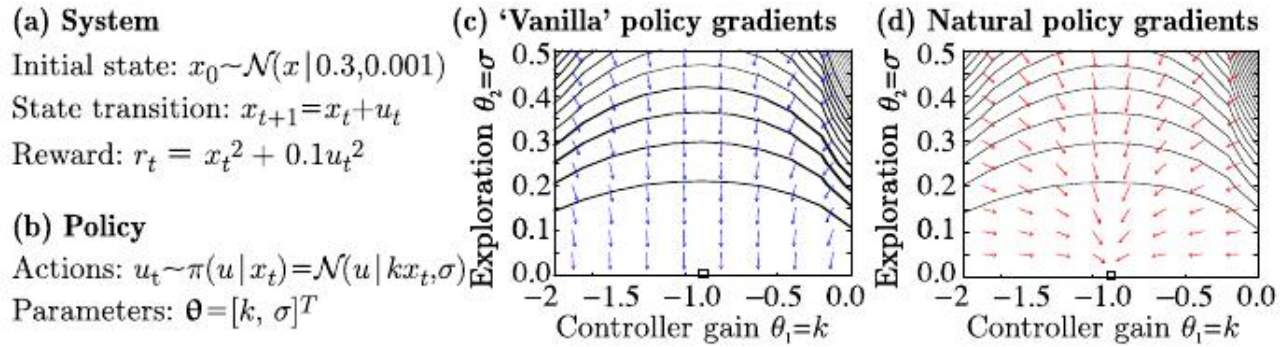
Cons

- Finite state space
- Adversely affected by reward variance

More

Natural Actor Critic

Peters Schaal 2003, 2008



Importance Sampling

Peshkin Shelton 2002; Tang Abbeel 2010

$$V(\theta) \propto \sum_i \frac{\pi_\theta(\text{trajectory } i)}{q_{reference}(\text{trajectory } i)} r(\text{trajectory } i)$$

Overview

Evolutionary Robotics, Milestones

1. Select the search space Θ
2. Define the objective function $\mathcal{F}(\theta)$ Sky is the limit: controller; morphology of the robot; co-operation of several robots...
3. Define a computable objective function
   in simulation, in-situ, reality gap
4. Optimize: Evolutionary Computation (EC); variants thereof
5. Test the found solution

1. Search Space

Neural Nets

- Universal approximators; continuity; generalization hoped for.
- Fast computation
- Can include priors in the structure
- Feedforward: reactive; Recurrent, with internal state

Critical issues

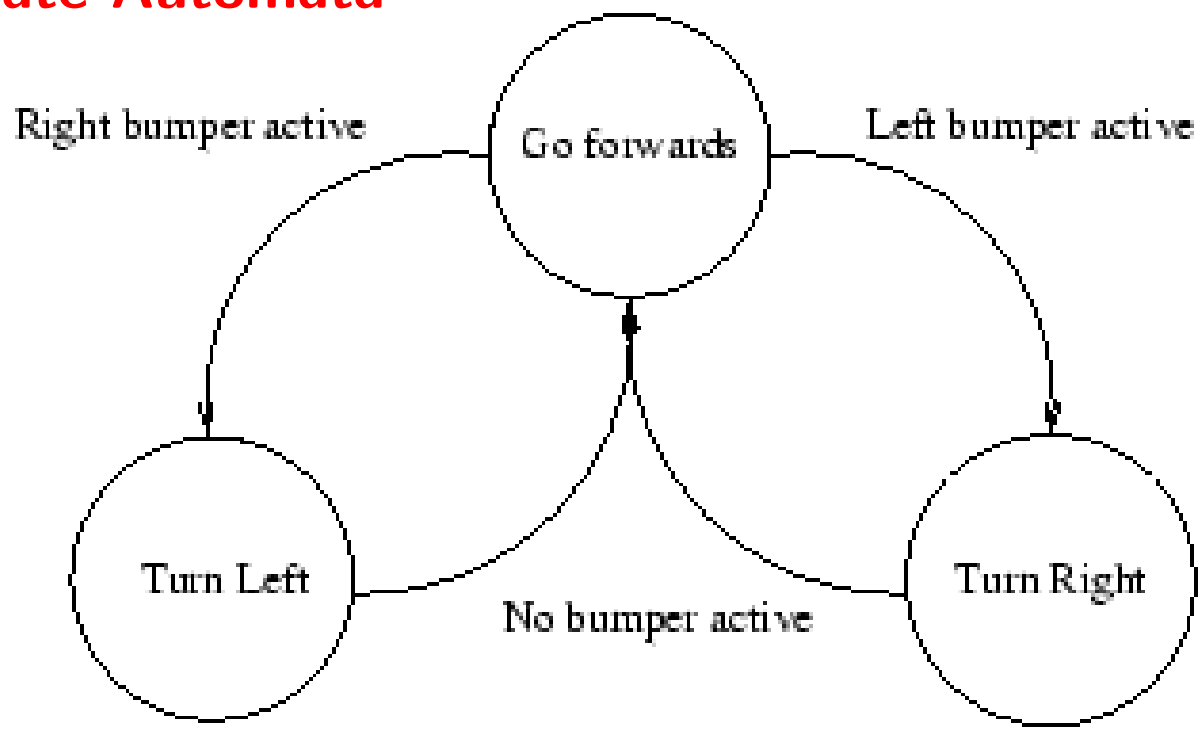
- Find the structure;
  (structured EC much more difficult)
  See NEAT and HyperNEAT Stanley Miikkulainen, 2002
  NeuroEvolution of Augmented Topology

1. Search Space, foll'd

Classifier Systems

if	(true)	then leftSpeed=2; rightSpeed=2;
if	(leftSensor>threshold)	then rightSpeed=0;
if	(rightSensor>threshold)	then leftSpeed=0;
if	(leftSensor>threshold) and (rightSensor>threshold)	then leftSpeed=-2; rightSpeed=-2;

Finite State Automata



1. Search Space, foll'd

Genetic Programming: trees made of

- Nodes (operators) $\mathcal{N}$
- Leaves (operands) $\mathcal{T}$

$$\text{Search space } \Omega = \text{Trees}(\mathcal{N}, \mathcal{T})$$

Examples :

- $\begin{cases} \mathcal{N} = \{+, \times\} \\ \mathcal{T} = \{X, \mathcal{R}\} \\ \Omega = \text{ Polynoms of } X. \end{cases}$
- $\begin{cases} \mathcal{N} = \{ \text{ if-then-else, while-do, repeat-until,..}\} \\ \mathcal{T} = \{\text{expressions, instructions}\} \\ \Omega = \text{ Programs} \end{cases}$

Key issues:

- Variable length genoms
- MORE $\neq$ BETTER

Genetic Programming and Embryogenesis

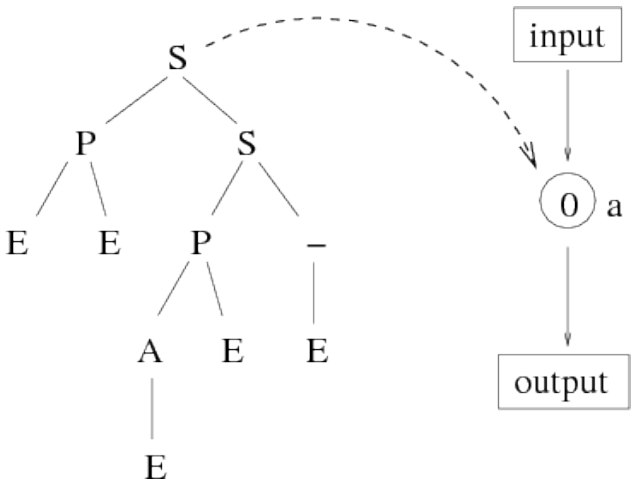
Principle

- An embryo
- .. develops along a program (= tree)
- Optimize the embryogenesis program

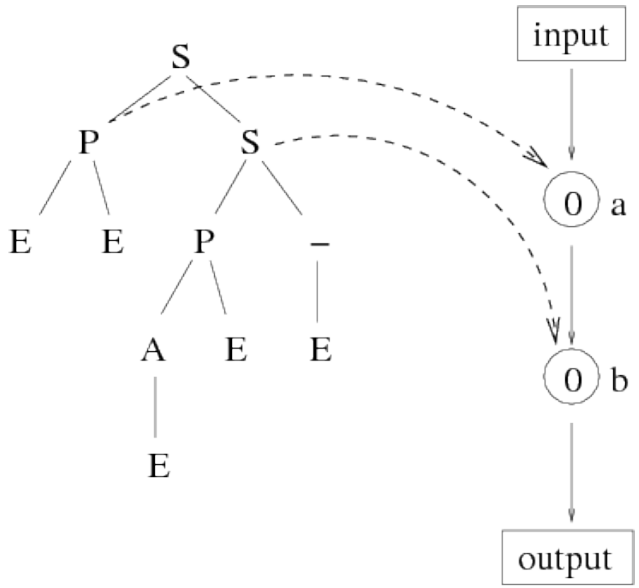
Operators

$\mathcal{T}$	E	stops
$\mathcal{N}$	S	Sequential Division
	P	Parallel Division
	A	Increase neuron threshold
	O	Decrease neuron threshold
	$+$	Increase weight on an edge
	$-$	Decrease weight
	C	Cut edge
	I	**Increase edge index**
	D	**Decrease edge index**
	R	**Back to the top of the tree (recursion)**
	W	Wait

Embryogenesis, 1

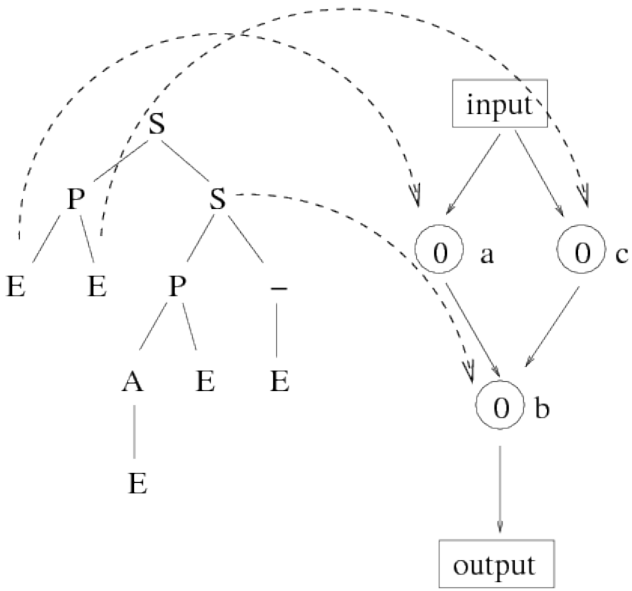


Tree and embryo,

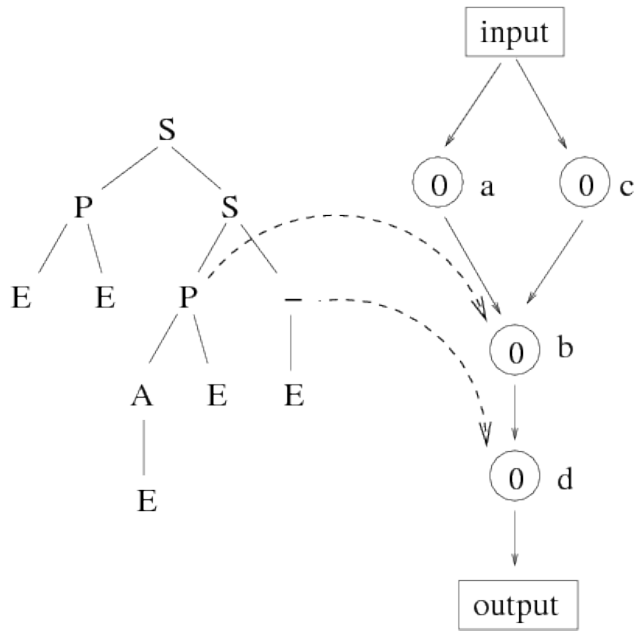


step 1

Embryogenesis, 2

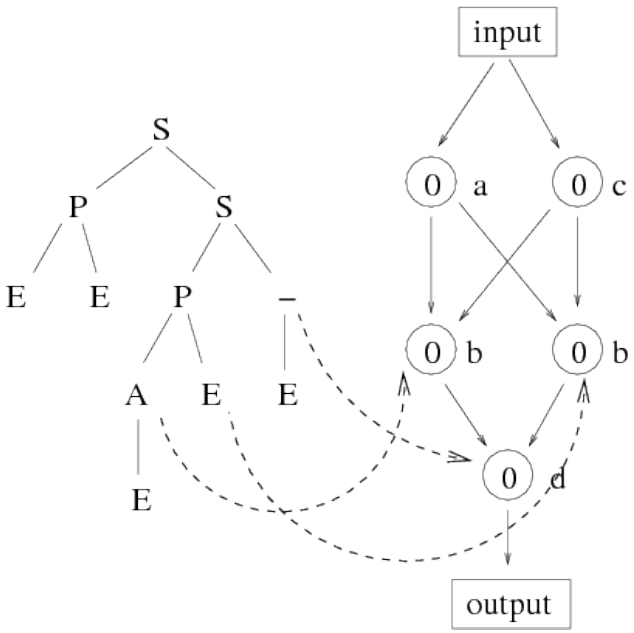


step 2

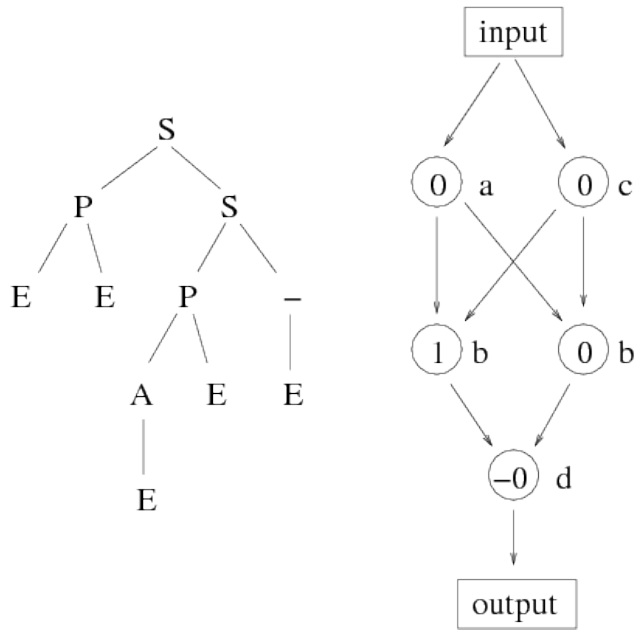


step 3, 4, 5

Embryogenesis, 3



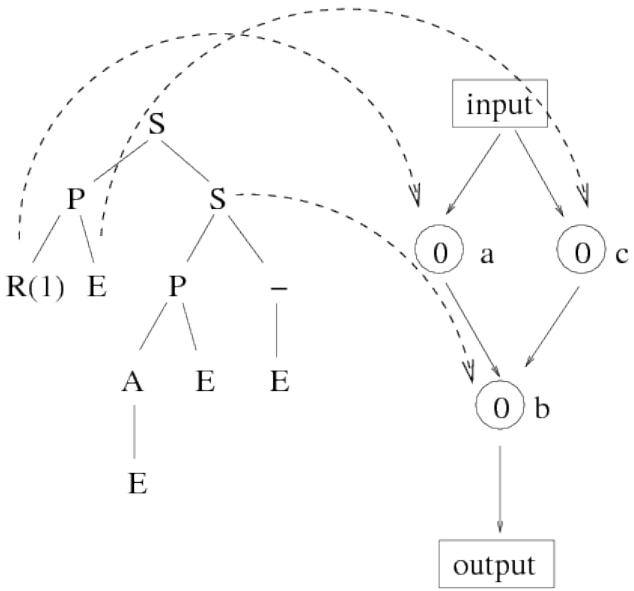
step 6



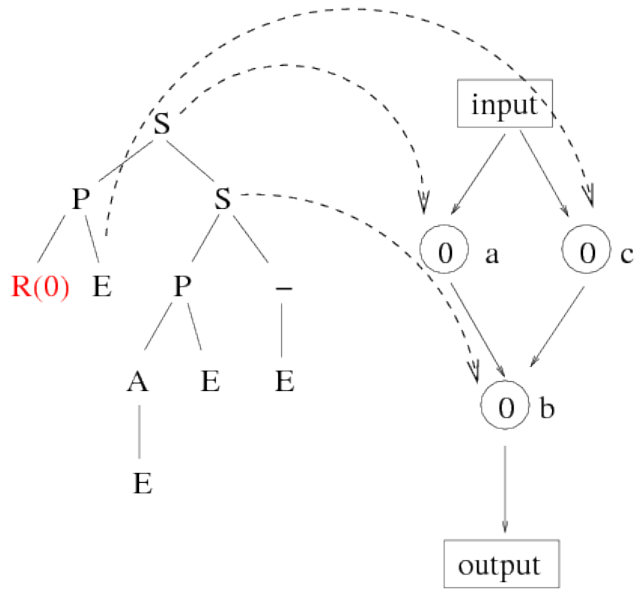
final

Embryogenesis, 4

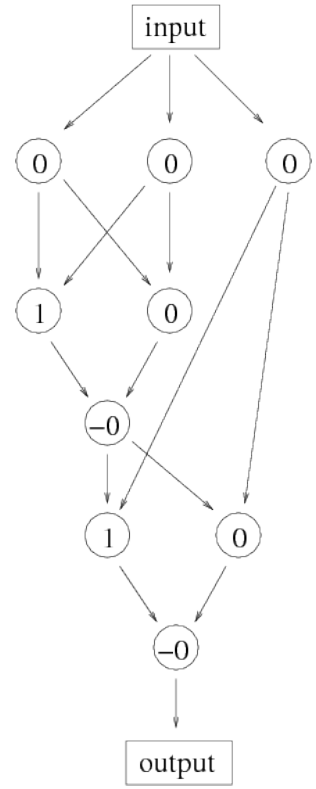
With recursion



step 2



step 3



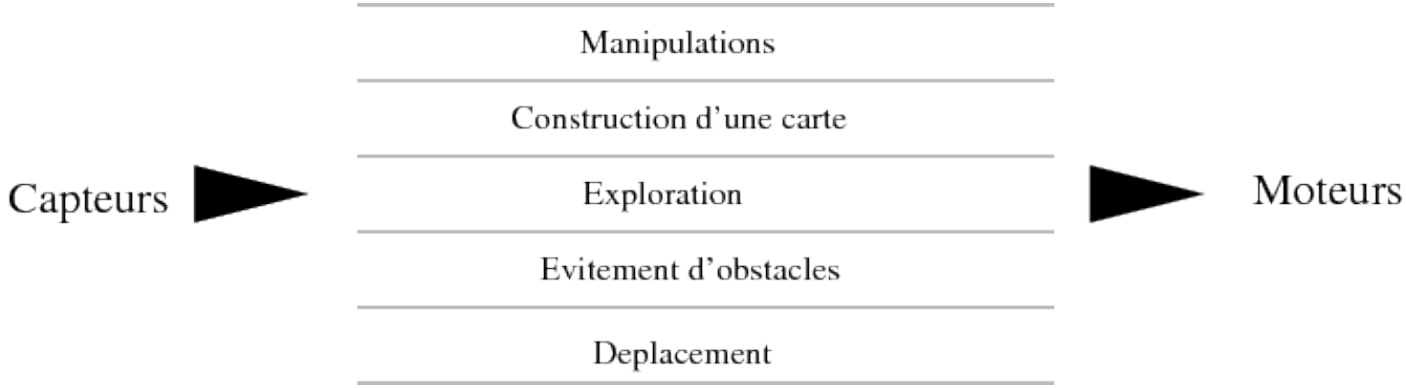
final

Overview

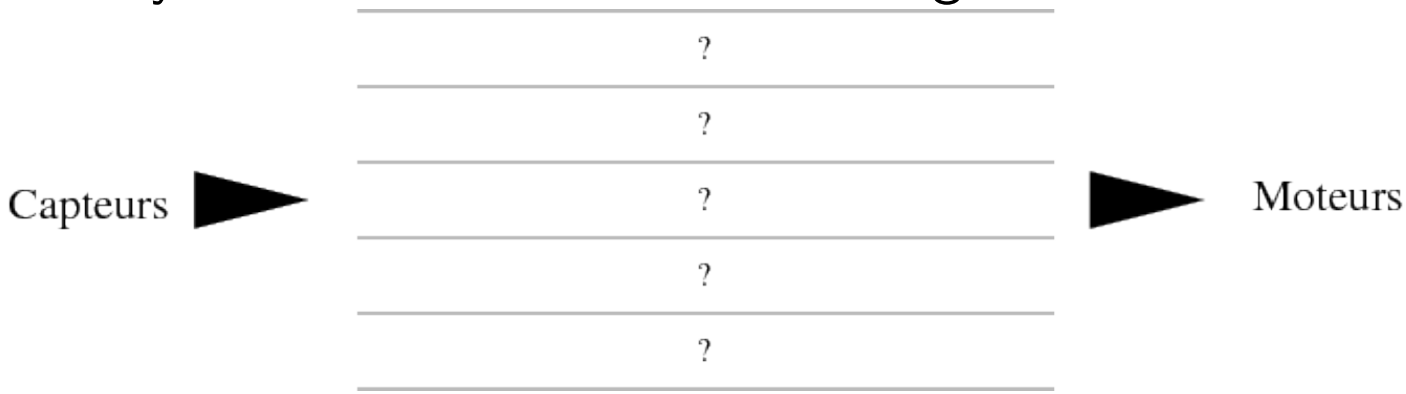
2. Objective

The promise: no need to decompose the goal

- Behavioral robotics — hand crafted decomposition

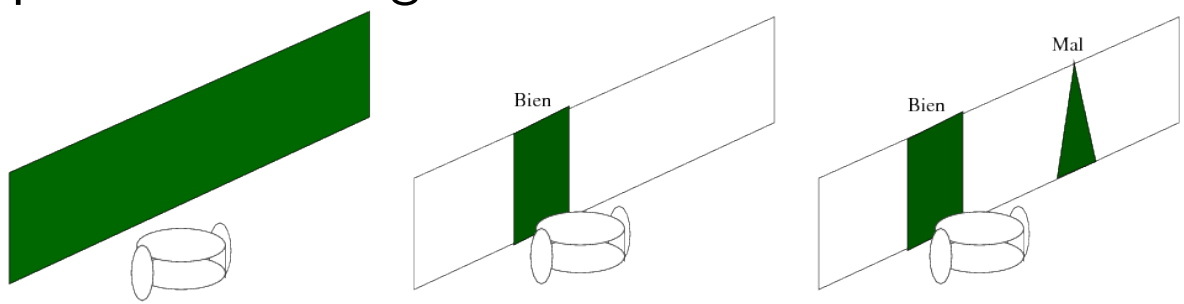


- Evolutionary robotics — emergence of a structure



In practice: bootstrap

- All initial (random) individuals are just incompetent
- Fitness landscape: Needle in the Haystack ? (doesn't work)
- Start with something simple
- Switch to more complex *during evolution*
- Example: visual recognition



2. Objective, foll'd

- Fonctional vs behavioral

  state of controller vs distance walked

- Implicit vs explicit

  Survival vs Distance to socket

- Internal vs external information

  Sensors, ground truth

- Co-evolution: e.g. predator/prey

  performance depends on the other robots

State of art

- Standard: function, explicit, external variables
- In-situ: behavioral, implicit, internal variables
- Interactive: behavioral, explicit, external variables

2. Objective, foll'd

Fitness shaping

- Obstacle avoidance
- Obstacle avoidance, and move !
- Obstacle avoidance, and (non circular) move !!

Finally

Floreano Nolfi 2000

$$\mathcal{F} = \int_{T_{exp.}} V(1 - \sqrt{\Delta v})(1 - i)$$

- V sum of wheel speed $r_i \in [-0.5, 0.5]$

$\rightarrow$ move

- $\Delta v = |r_1 + r_2|$

$\rightarrow$ ahead

- i maximum (normalised) of sensor values

$\rightarrow$ obstacle avoidance

Behavioral, internal variables, explicit

Result analysis

- First generations
  - Most rotate
  - Best ones slowly go forward
  - No obstacle avoidance
  - Perf. depends on starting point
- After $\approx$ 20 gen.
  - Obstacle avoidance
  - No rotation
- Thereafter, gradually speed up

Result analysis

- Max. speed 48mm/s (true max = 80)

  Inertia, bad sensors

- Never stuck in a corner

  contrary to Braitenberg

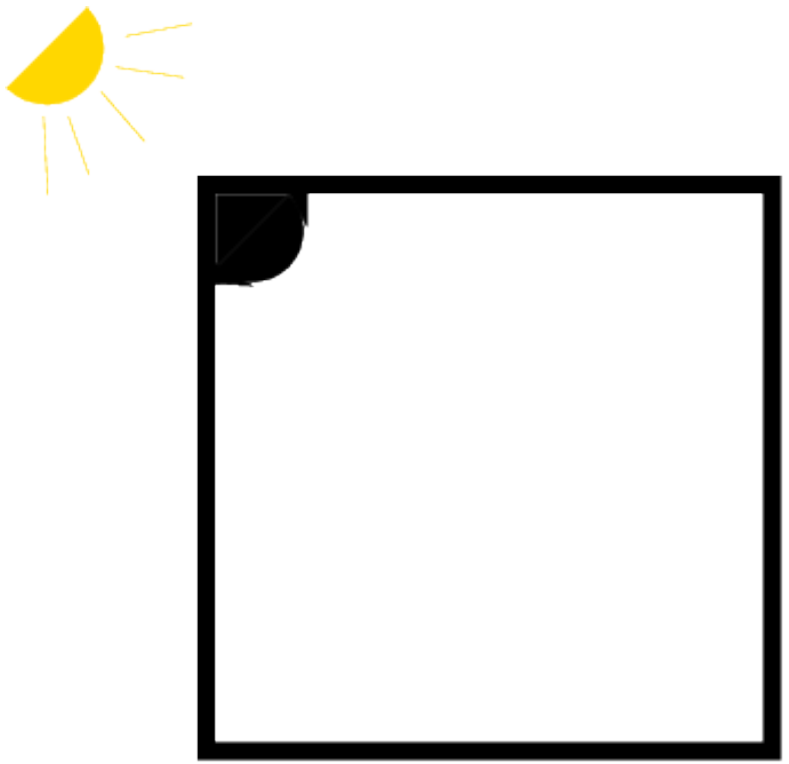
Going further

- Changing environment
- Changing robotic platform
- From simulation to real-world

Fast adaptation

Explore and recharge

Not a reactive behavior



- Battery gets empty in 20s in white zone
- recharges in black zone
- But no reward in black zone

Explore and recharge, 2

- A ground sensor

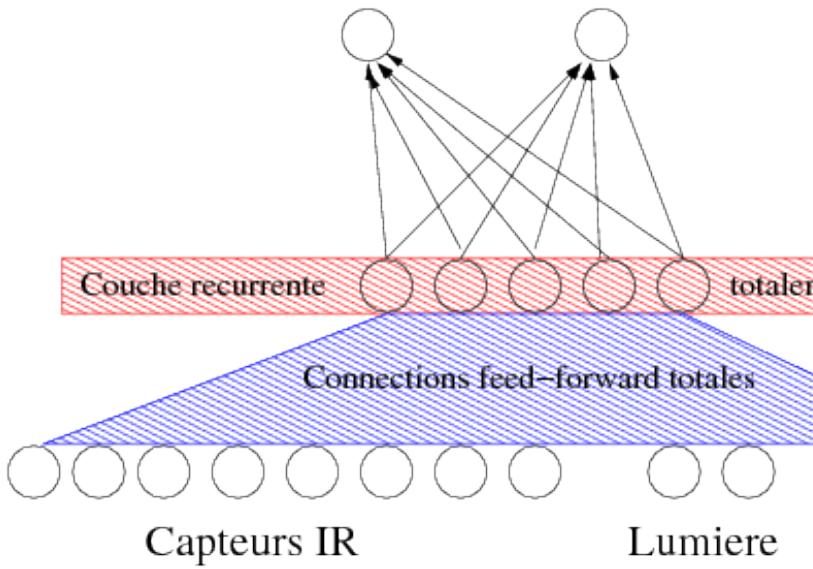
  $\rightarrow$ sees whether the ground is white or black

- 2 sensors passive mode

  $\rightarrow$ ambiant light

Search space: Elman network

- Optimize weights
- Recurrent NN, thus with internal state
- Optimize in situ



Explore and recharge, 2

Performance

$$\mathcal{F} = \int_{\text{White zone}} V(1-i)$$

- Lifetime requires a good recharge strategy
- V cumulative wheel speed $r_i \in [-0.5, 0.5]$

$\rightarrow$ move

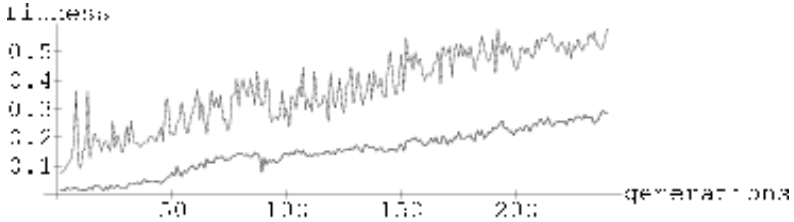
- i maximum (normalised) of sensor values

$\rightarrow$ obstacle avoidance

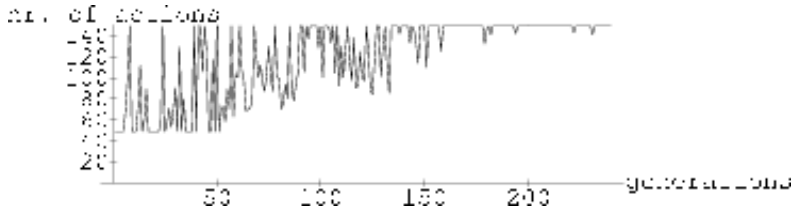
Behavioral, internal, explicit + implicit

Result analysis

During evolution



Fitness (best and average)

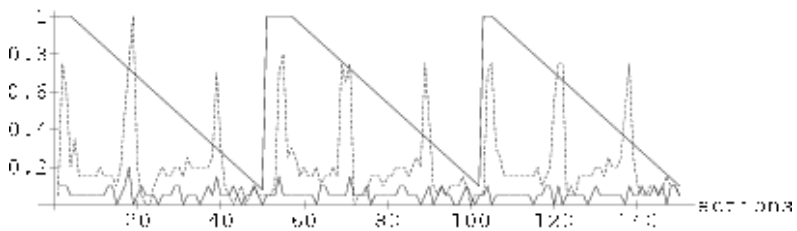


Lifetime

Inspecting best behavior

methods inspired from neurophysiology/ethology

Instrumenting the robot

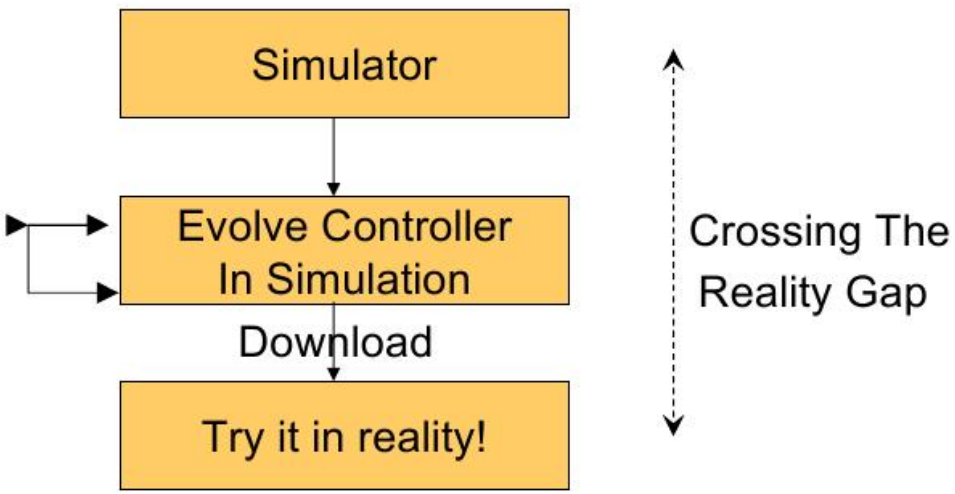


Battery and motor state along lifetime

Overview

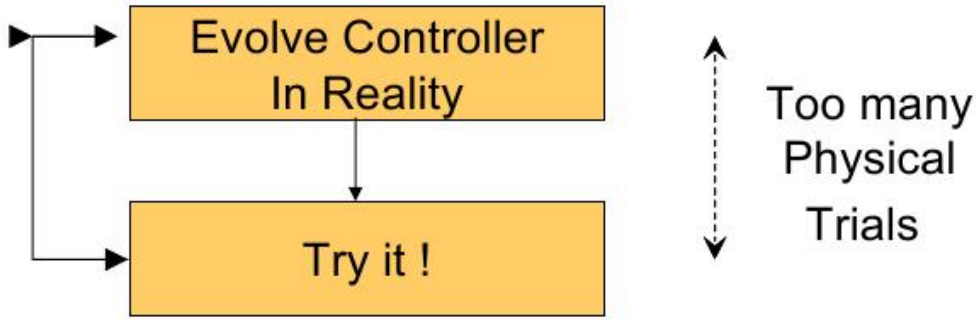
Reality gap

- What if simulator does not reflect the robot or the environment ?
- Optimizes the wrong function

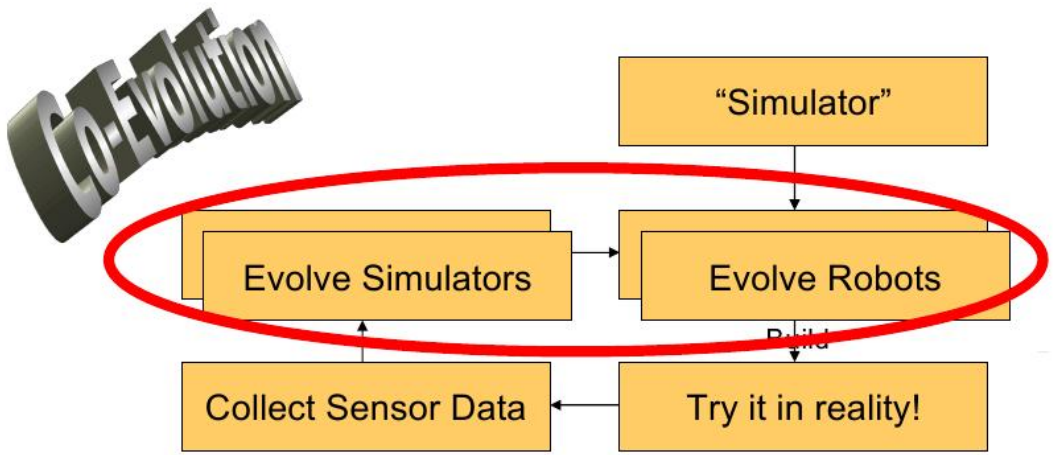


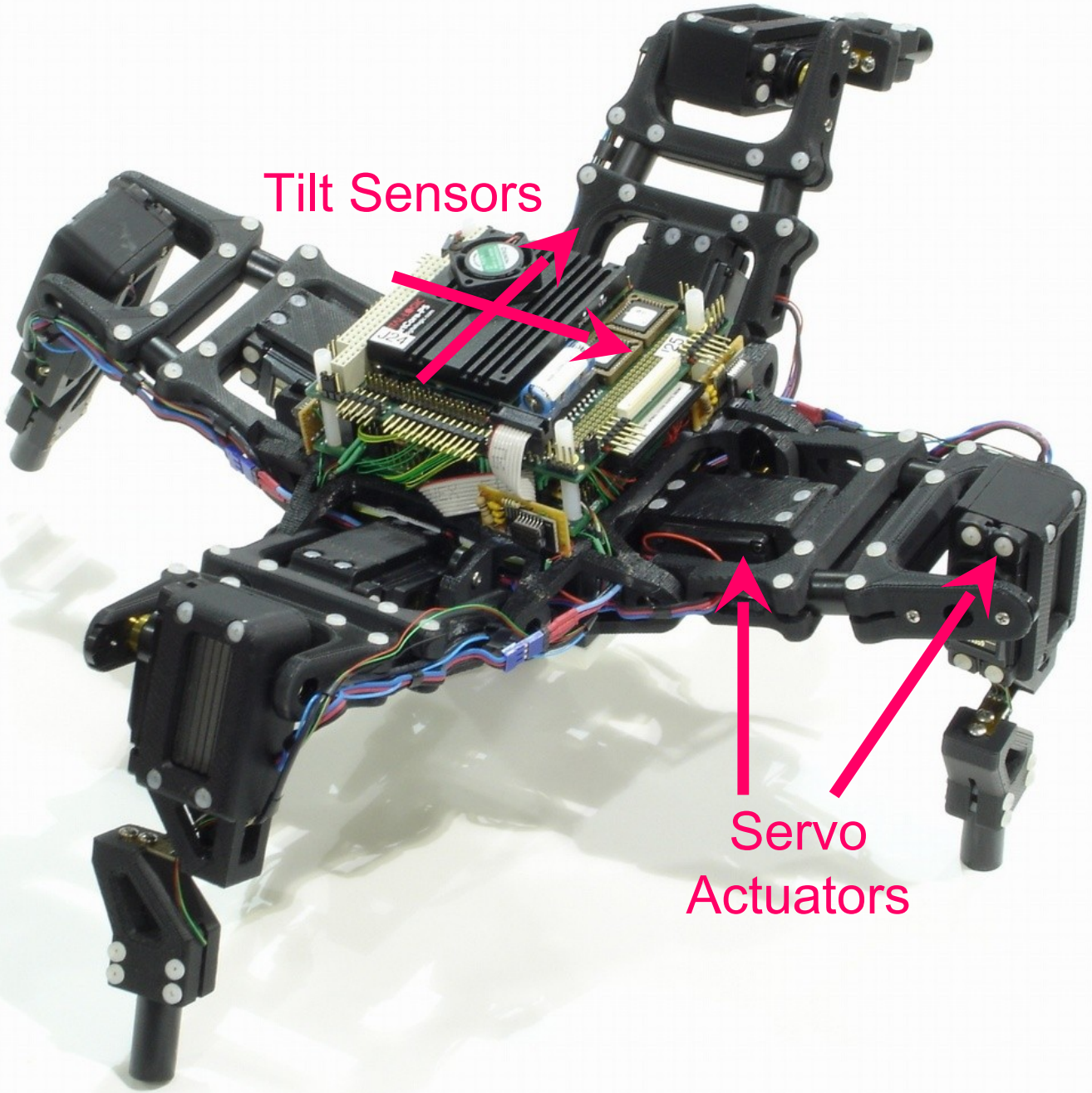
Reality gap, 2

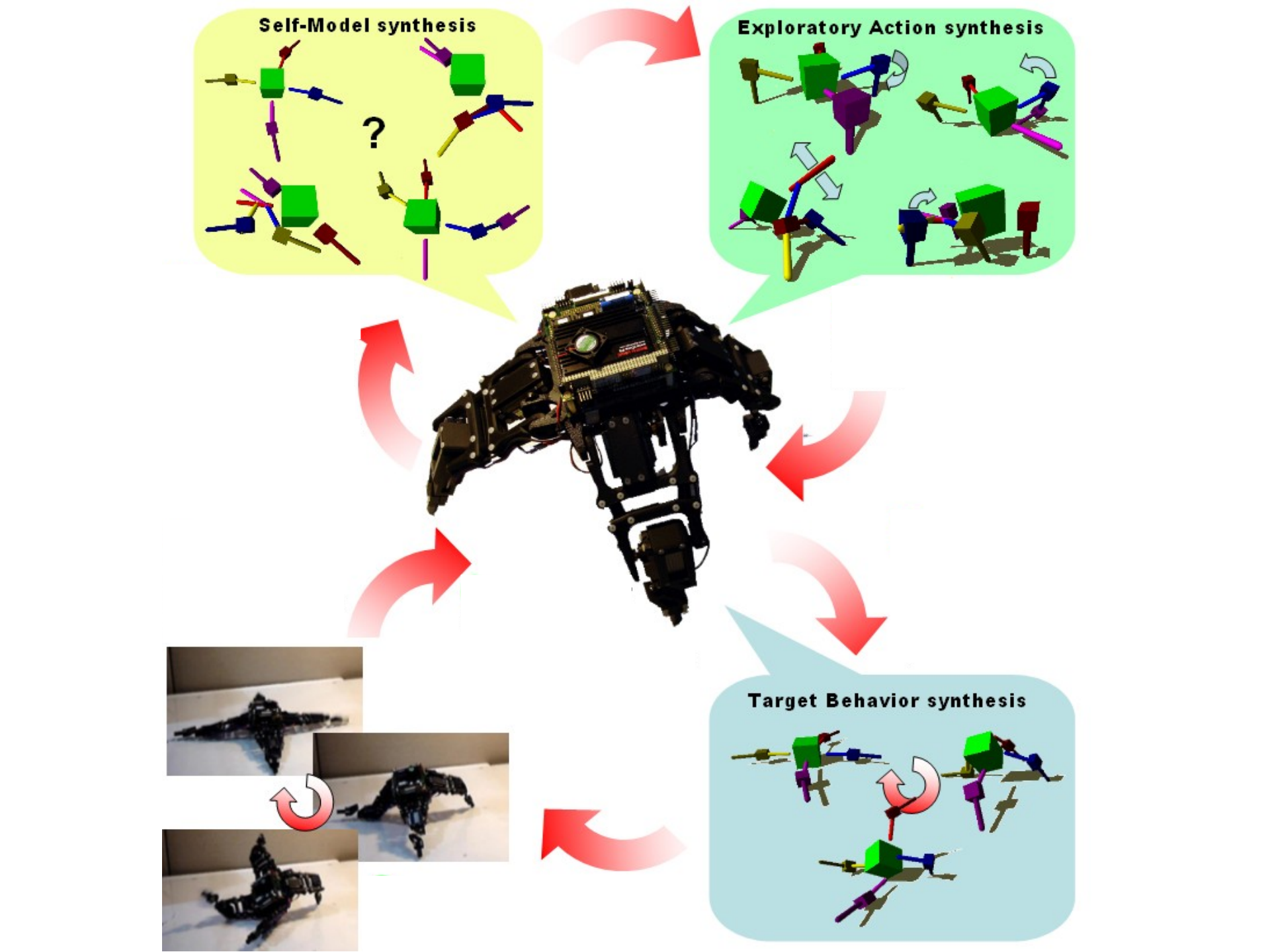
Against in-situ



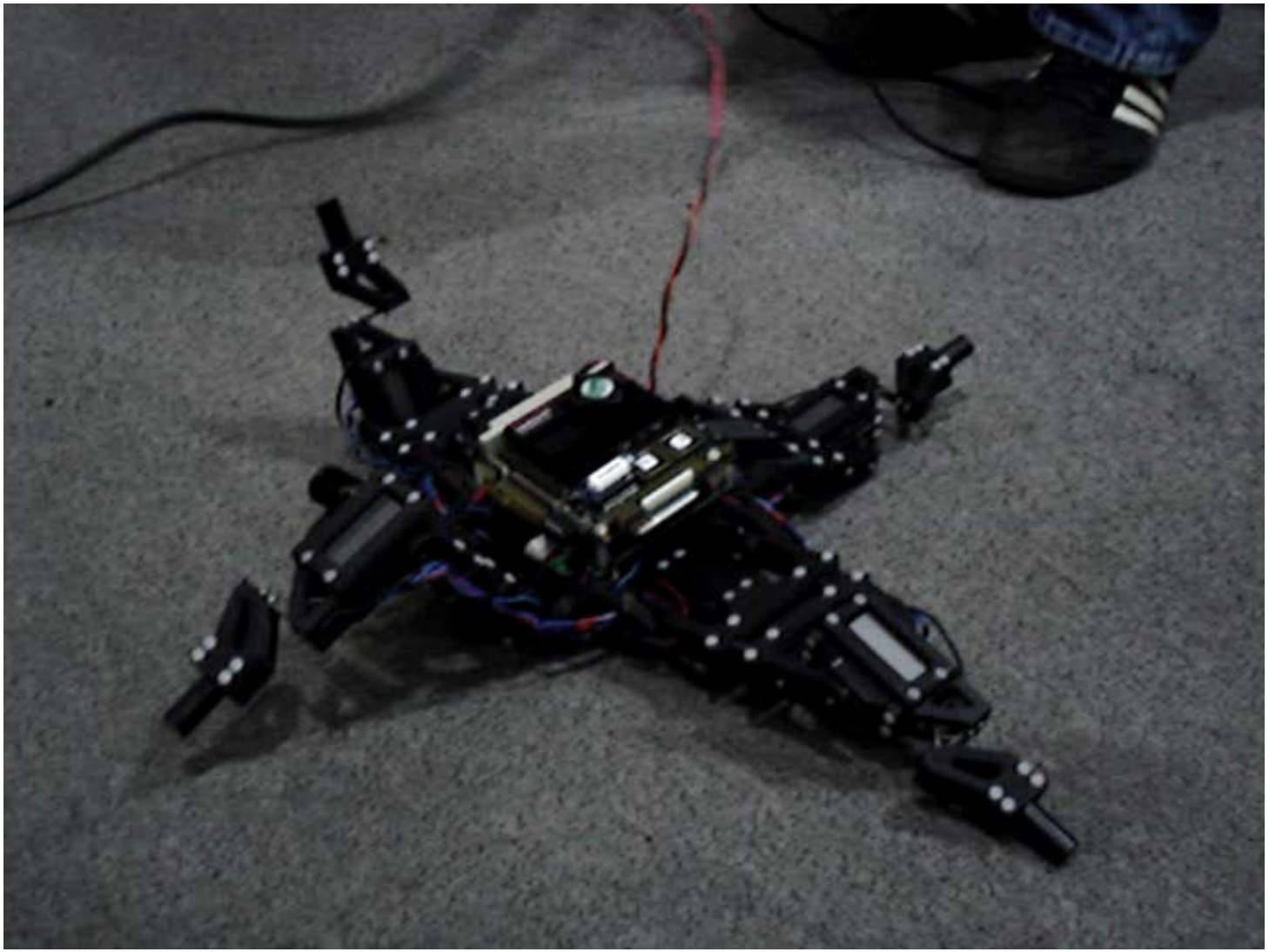
Finally



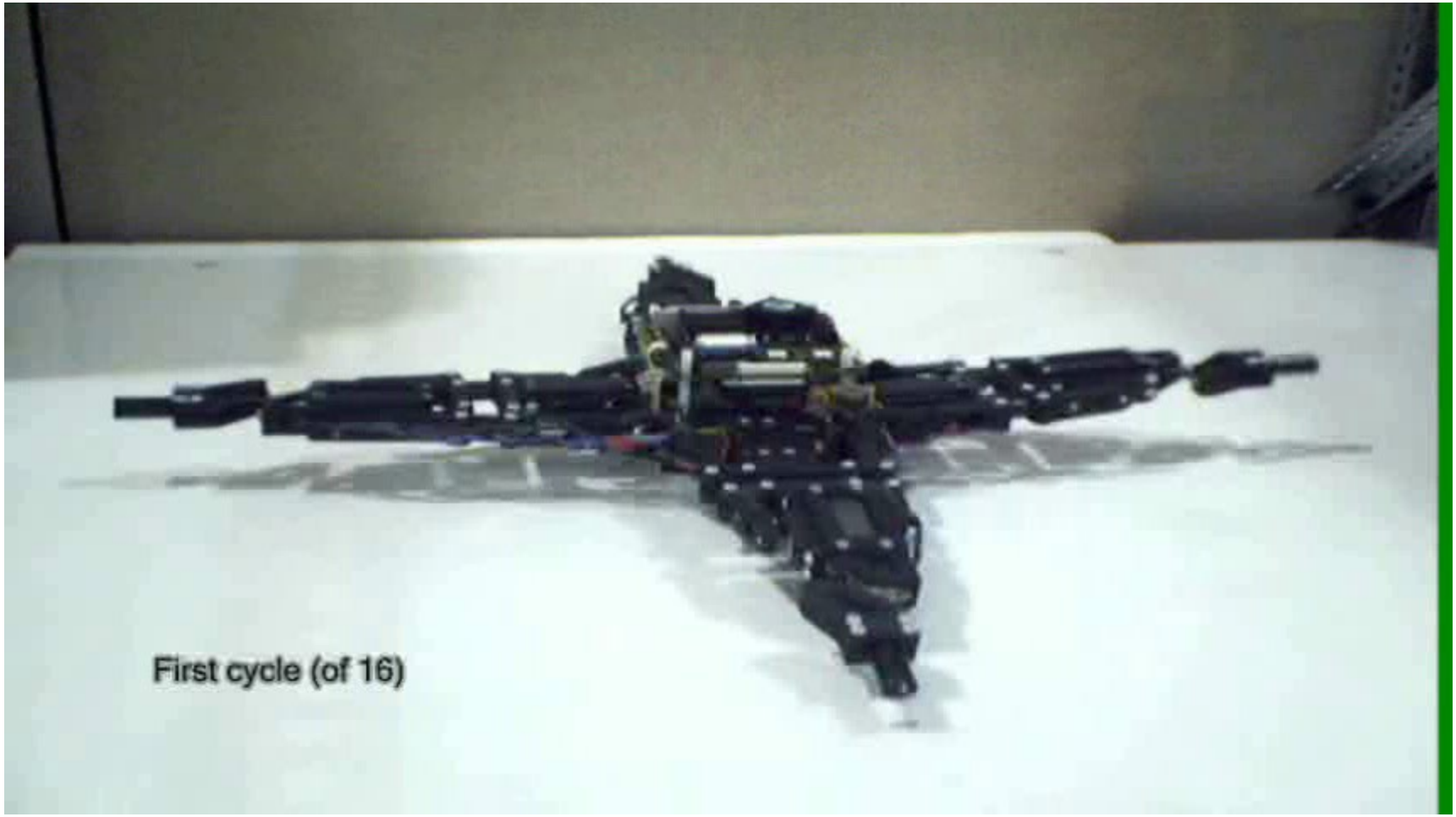

Tilt Sensors
Servo
Actuators


Self-Model synthesis
?
Exploratory Action synthesis
Target Behavior synthesis

Morphological Estimation

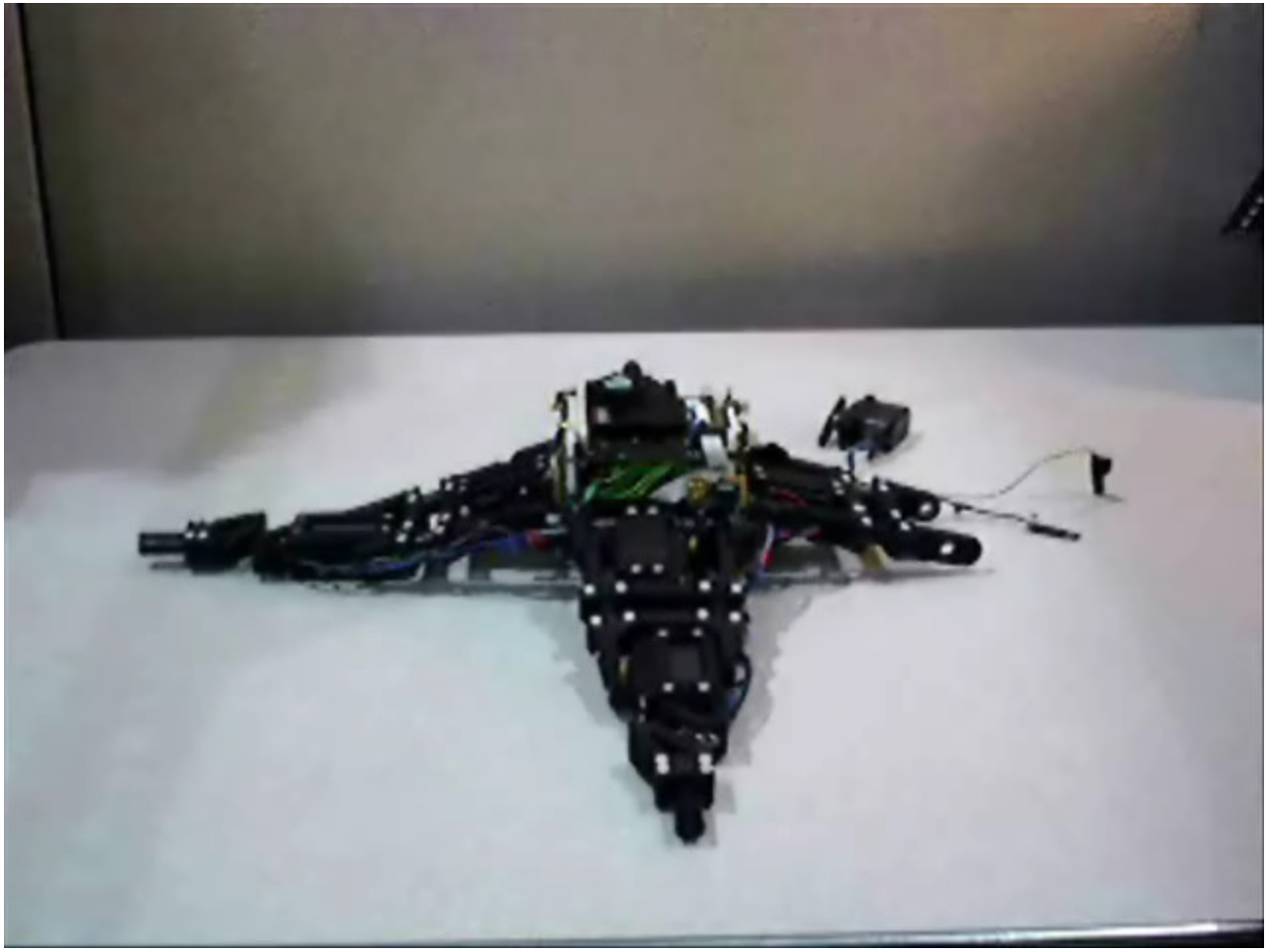


Emergent Self-Model

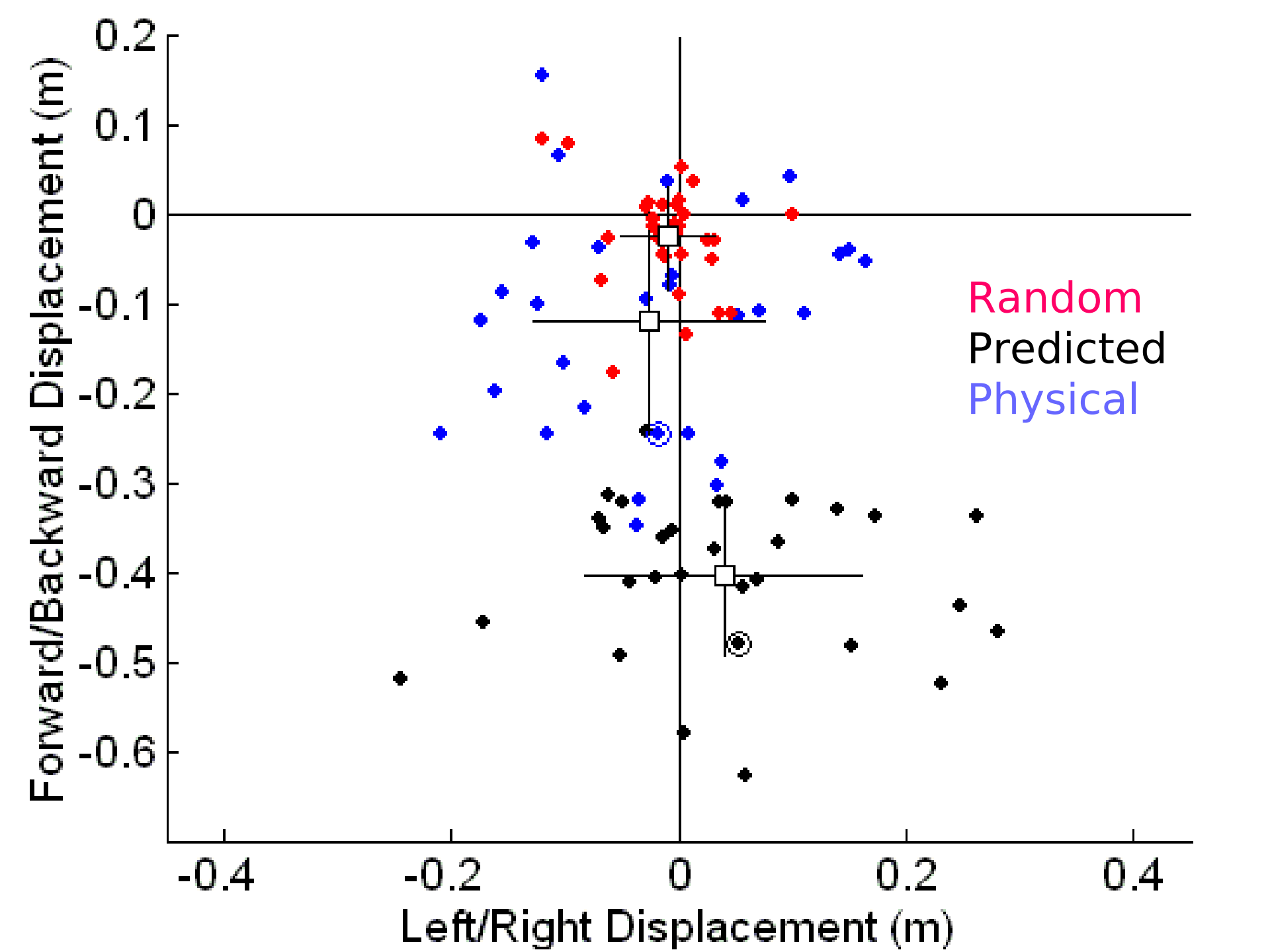


With Josh Bongard and Victor Zykov, Science 2006

Damage Recovery



With Josh Bongard and Victor Zykov, Science 2006


Random
Predicted
Physical
Forward/Backward Displacement (m)
Left/Right Displacement (m)
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.4
-0.2
0
0.2
0.4

Overview

Co-Evolution

Competitive co-evolution

- Goal: survival
- Model: predator-prey — Lotka-Volterra

$$\frac{\partial N_1}{\partial t} = N_1(r_1 - b_1 N_2), \frac{\partial N_2}{\partial t} = N_2(-r_1 + b_2 N_1)$$

- $\rightarrow$ population sizes oscillate
- Simulation: fixed population size, performance varies
- Fitness computed by turnament

  global, random, with best individuals, ...

Predator-prey

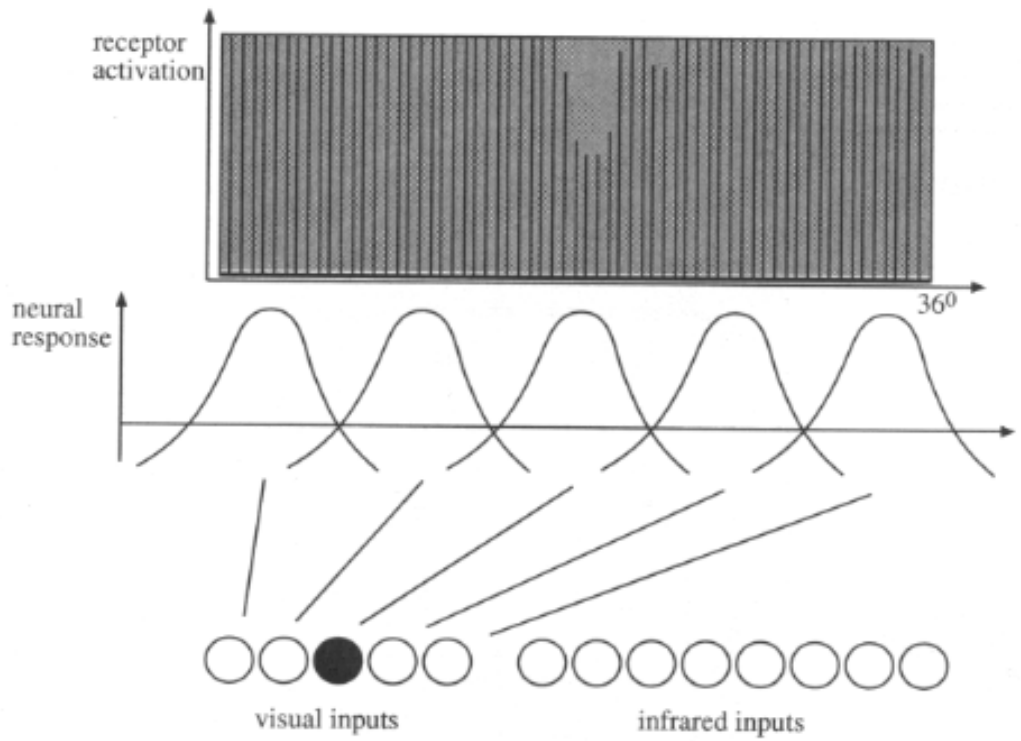
Floreano et Nolfi, 97-99

- Predator: sees; is slow

RN 8+5 $\rightarrow$ 2 recurrent

- Prey: is blind; is twice as fast

RN 8 $\rightarrow$ 2 recurrent



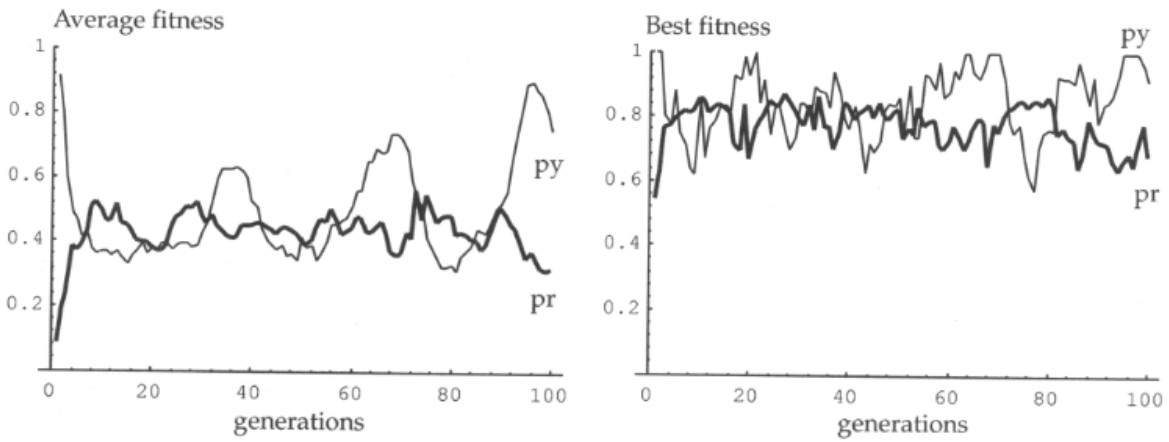
Fitness

- Round robin turnament, all predators and preys
- Stops when predator catches the prey (ad hoc sensor)
- .. or after 500 cycles, $\approx$ 50s
- performance (each) $+=$ duration of turnament

Predators must minimize performance Preys must maximize performance

Behavioral, implicit, internal/external

First results

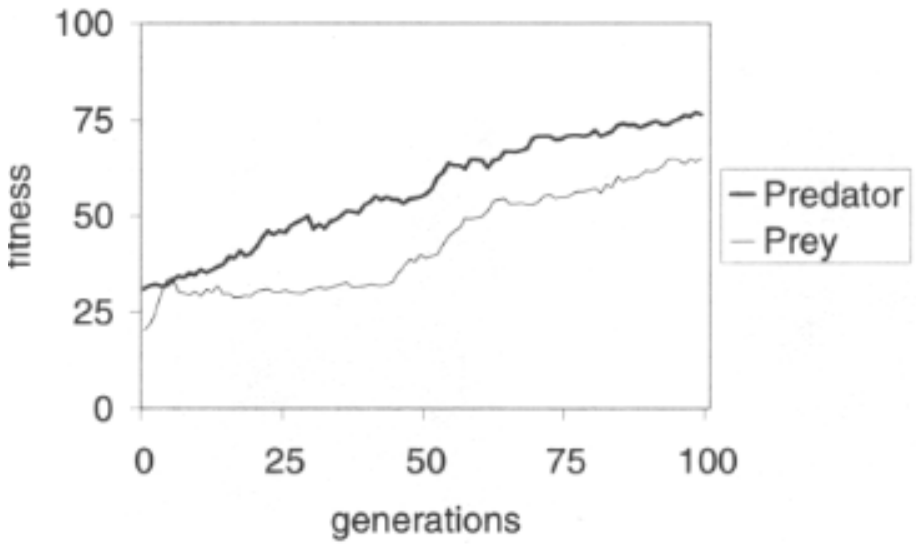


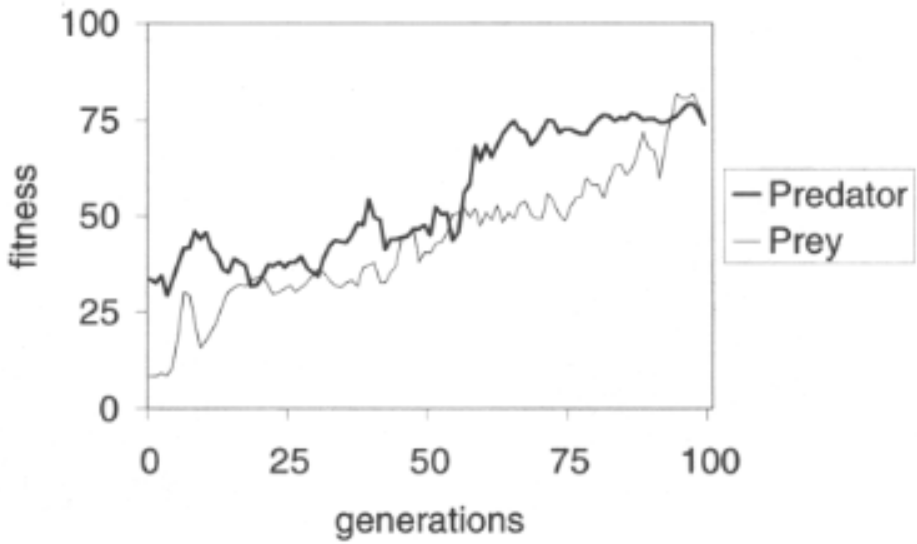
- First predators very bad
- Beware of the Red Queen ! Paredis 97
- The final best can be caught by previous best ones!

Hall of fame

Intuition

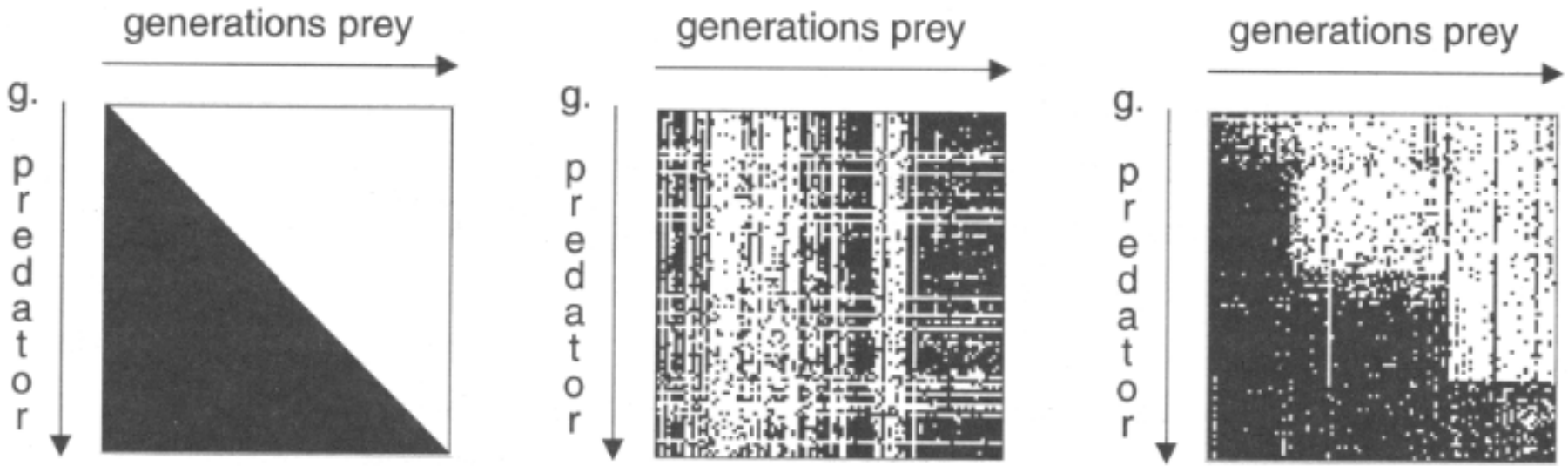
Also compete with best ancestors





Hall of fame, 2

Turnament among all individuals in all generations
Black ≡ predator wins, white ≡ prey wins



Ideal situation / Without Hall of Fame / With Hall of Fame

Final best are better than (almost) all ancestors.

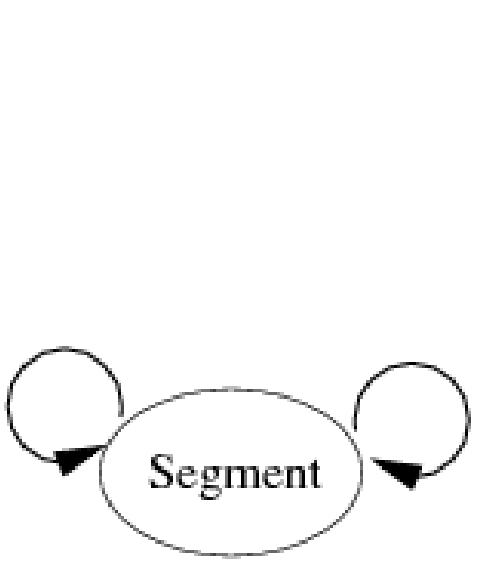
Carl Sims

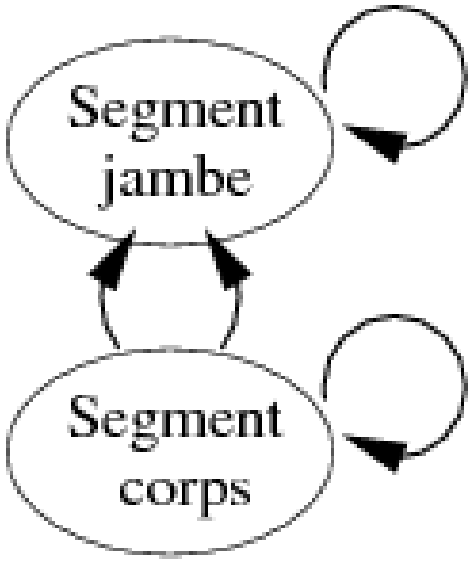
Goal

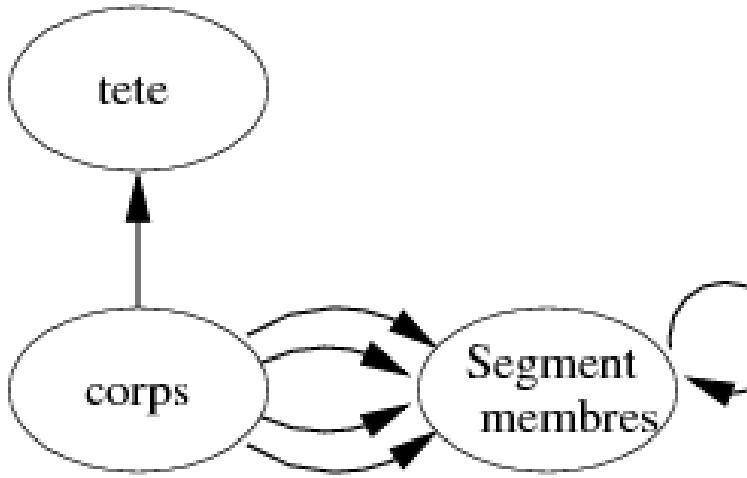
- Evolve both morphology and controller
- using a grammar (oriented graph)
- Heavy computational cost

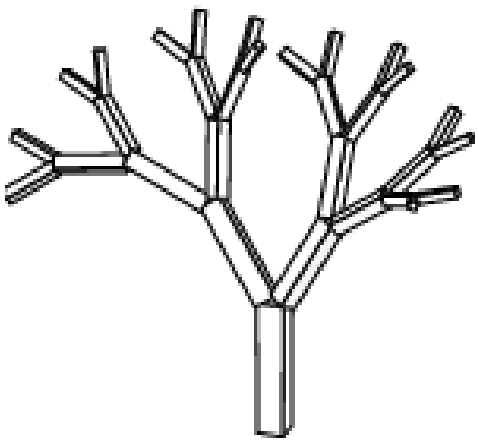
  simulation, several days on Connection Machine – 65000 proc.
- Evolving locomotion (walk, swim, jump)
- and competitive co-evolution (catch an object)

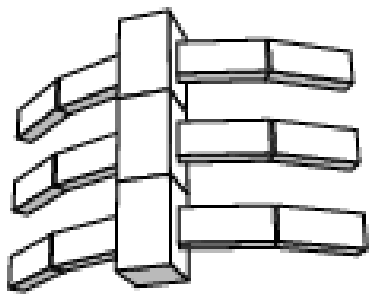
The creatures, Karl Sims

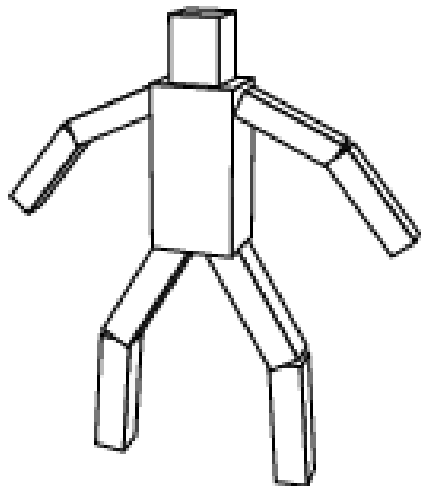










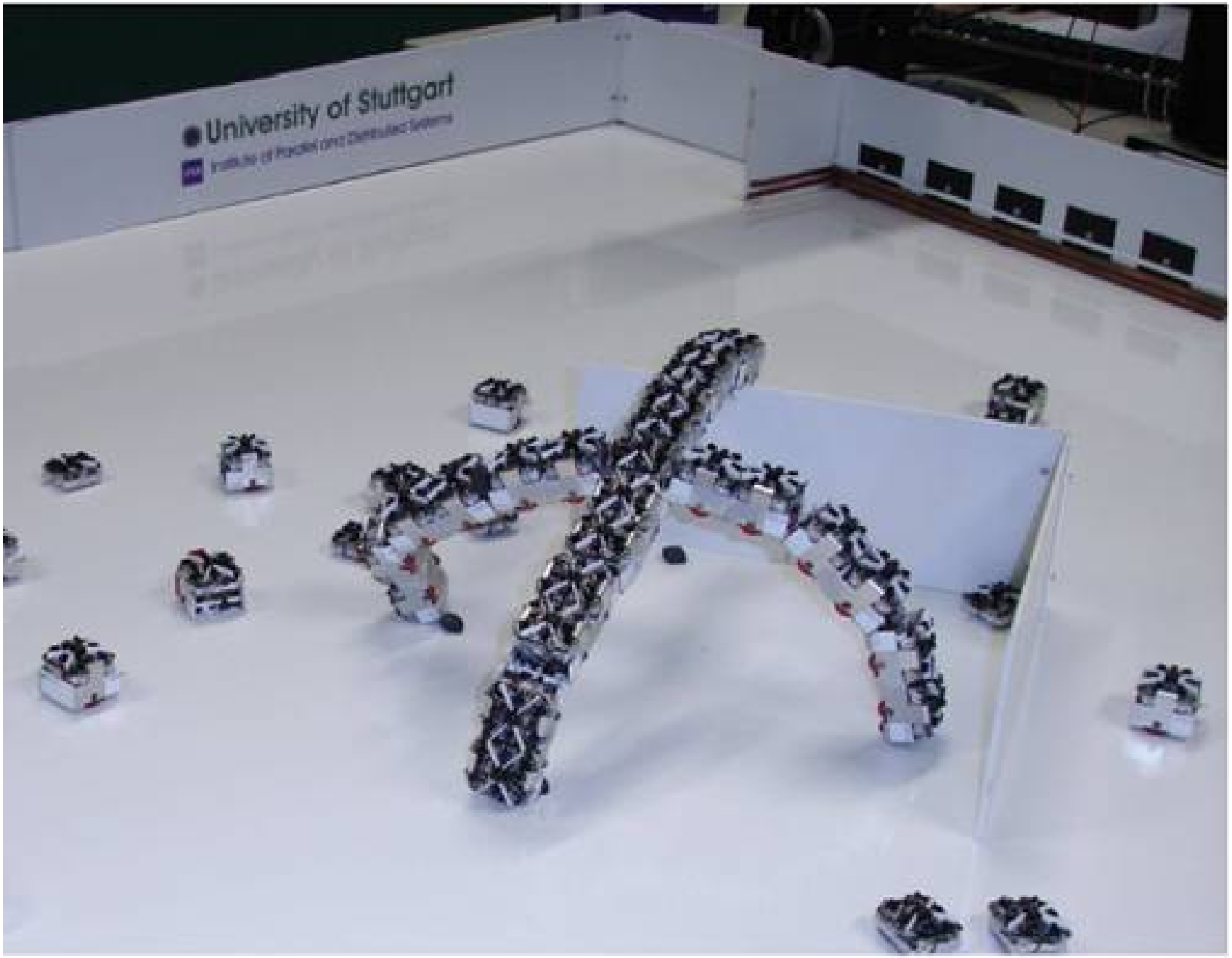


more ?

http://www.youtube.com/watch?v=JBgG_VSP7f8

Overview

Contexte



I. Getting motivated. Internal rewards

Delarboulas et al., PPSN 2010

Requirements

1. No simulation
2. On-board training
   - Frugal (computation, memory)
   - No ground truth
3. Providing "interesting results"

"Human – robot communication"

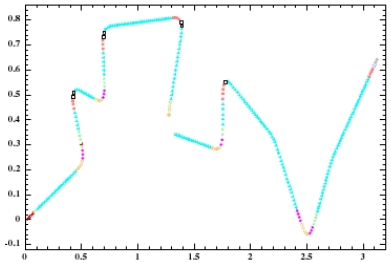
Goal: self-driven Robots : **Defining instincts**

Starting from (almost) nothing

Robot $\equiv$ a data stream

$$t \rightarrow x[t] = (sensor[t], motor[t])$$

$$\text{Trajectory} = \{x[t], t = 1 \dots T\}$$



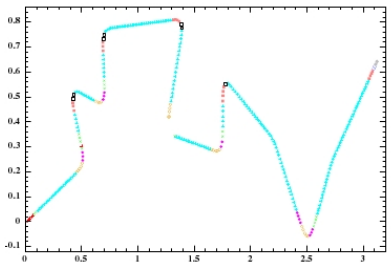
Robot trajectory

Starting from (almost) nothing

Robot $\equiv$ a data stream

$t \rightarrow x[t] = (sensor[t], motor[t])$

Trajectory $= \{x[t], t = 1 \ldots T\}$



Robot trajectory

Computing the quantity of information of the stream

Given $x_1, \ldots x_n$, visited with frequency $p_1 \ldots p_n$,

$$Entropy(trajectory) = -\sum_{i=1}^{n} p_i \log p_i$$

Conjecture

Controller quality $\propto$ Quantity of information of the stream

Building sensori-motor states

Avoiding trivial solutions...

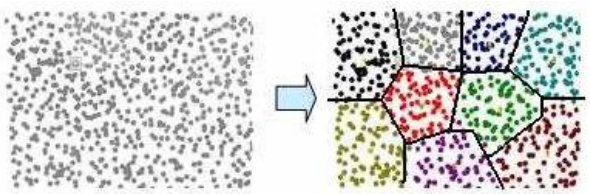
If sensors and motors are continuous / high dimensional

- then all vectors $x[t]$ are different
- then $\forall i, p_i = 1/T$; $Entropy = \log T$

... requires generalization

From the sensori-motor stream sequence of points in $\mathbb{R}^d$

to clusters **sensori-motor states**



Trajectory $\rightarrow$

$x_1 x_2 x_3 x_1 \ldots$

Clusters in sensori-motor space ($\mathbb{R}^2$)

Clustering

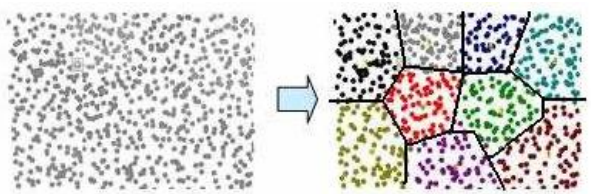
k-Means

1. Draw k points $x[t_i]$
2. Define a partition $\mathcal{C}$ in k subsets C_i *Voronoï cells*

$$C_i = \{x / d(x, x[t_i]) < d(x, x[t_j]), j \neq i\}$$

ϵ-Means

1. Init : $\mathcal{C} = \{\}$ *Initial site list*
2. For $t = 1$ to T *loop on trajectory*
   - If $d(x[t], \mathcal{C}) > \epsilon$, $\mathcal{C} \leftarrow \mathcal{C} \cup \{x[t]\}$



Curiosity Instinct

Search space

- Neural Net, 1 hidden layer.

Definition

- Controller F + environment $\rightarrow$ Trajectory
- Apply Clustering on Trajectory
- For each C_i, compute its frequency p_i

$$\mathcal{F}(F) = -\sum_{i=1}^{n} p_i * \log(p_i)$$

Curiosity instinct: Maximizing Controller IQ

Properties

- Penalizes inaction: a single state $\rightarrow$ entropy $= 0$
- Robust w.r.t. sensor noise (outliers count for very little)
- Computable online, on-board (use ϵ-clustering)
- Evolvable onboard

Limitations: does not work if

- Environment too poor
  (in desert, a single state $\rightarrow$ entropy $= 0$)
- Environment too rich
  (if all states are distinct, $\mathit{Fitness}(\text{controller}) = \log T$)

both under and over-stimulation are counter-effective.

From curiosity to discovery

Intuition

- An individual learns sensori-motor states ($x[t_i]$ center of C_i)
- The SMSs can be transmitted to offspring
- giving the offspring an access to "history"
- The offspring can try to "make something different"

fitness(offspring) = Entropy(Trajectory(ancestors $\bigcup$ offspring))

NB: does not require to keep the trajectory of all ancestors.
One only needs to store $\{C_i, n_i\}$

From curiosity to discovery

Cultural evolution transmits genome + "culture"

1. parent = (controller genome, $(C_1, n_1), \ldots (C_K, n_K)$)
2. Perturb parent controller $\rightarrow$ offspring controller
3. Run the offspring controller and record $x[1], \ldots x[T]$
4. Run ϵ-clustering variant.

$$Fitness(offspring) = -\sum_{i=1}^{\ell} p_i \log p_i$$

ϵ-clustering variant

Algorithm

1. Init : $\mathcal{C} = \{(C_1, n_1), \ldots (C_K, n_K))\}$ *Initial site list*
2. For $t = 1$ to T *loop on trajectory*
   - If $d(x[t], \mathcal{C}) > \epsilon$, $\mathcal{C} \leftarrow \mathcal{C} \cup \{x[t]\}$
3. Define $p_i = n_i / \sum_j n_j$

$$Fitness(offspring) = -\sum_{i=1}^{\ell} p_i \log p_i$$

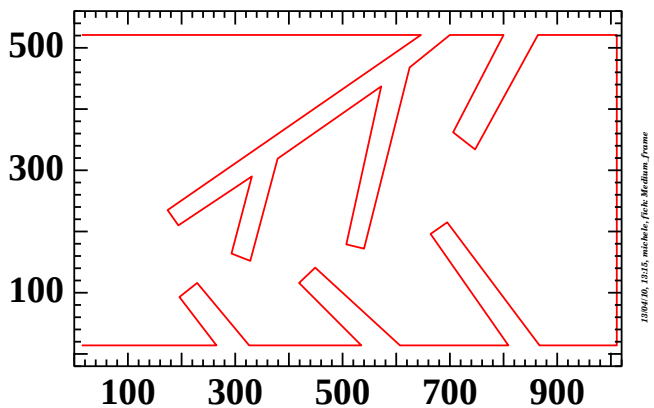
Validation

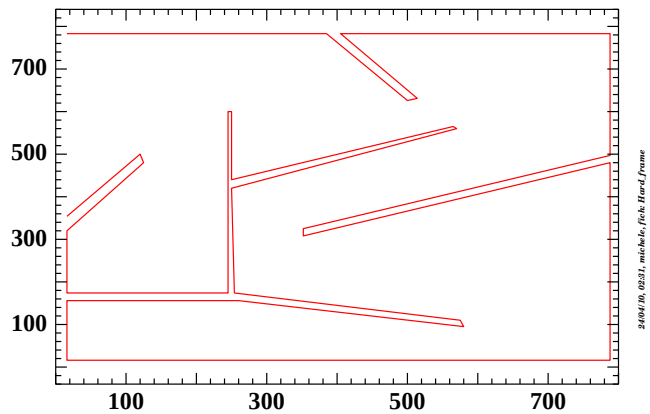
Experimental setting

Robot = Cortex M3, 8 infra-red sensors, 2 motors.

Controller space = ML Perceptron, 10 hidden neurons.

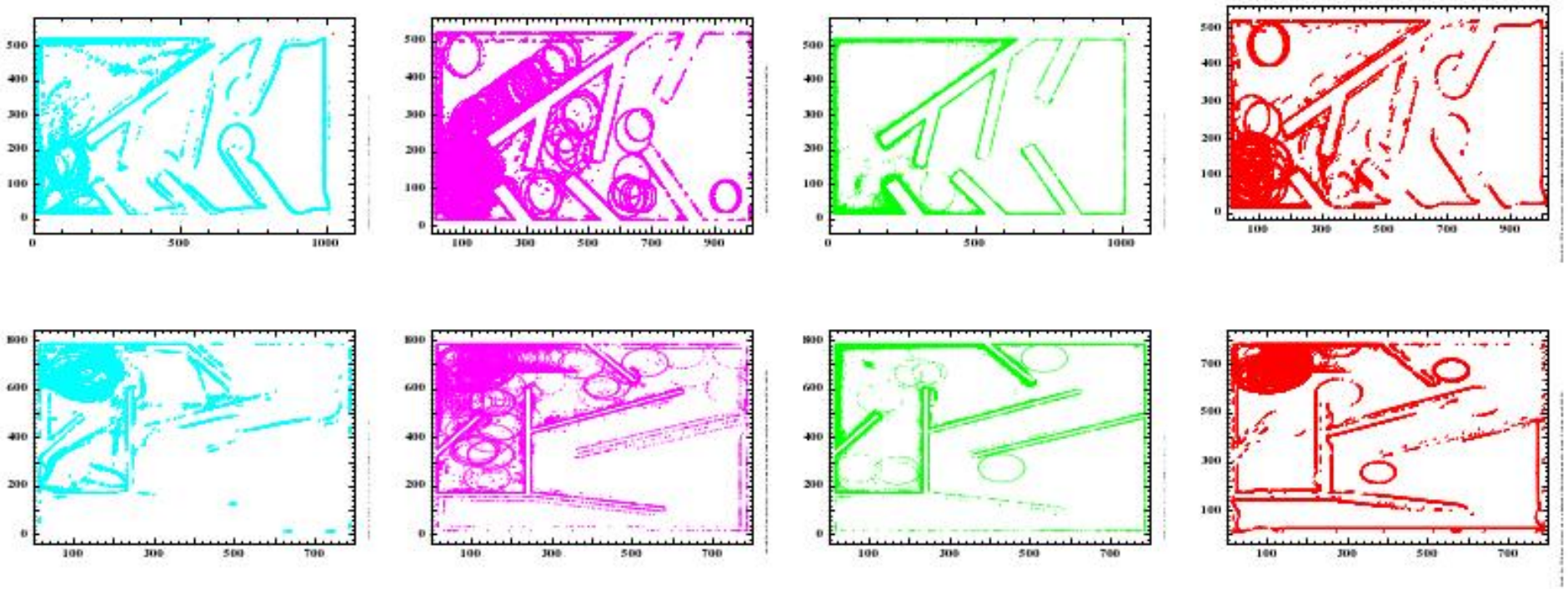
Medium and Hard Arenas





Validation, 2

Plot points in hard arena visited 10 times or more by the 100 best individuals.



Nolfi & Floreano | Lehman & Stanley | Curiosity | Discovery

PPSN 2010

Partial conclusions

Entropy-minimization

- computable on-board;

  no need of prior knowledge/ground truth
- yields "interesting" behavior
- needs stimulating environment

See also

- Robust Intrinsic Motivation

  Baranes & Oudeyer 05,07; Oudeyer, NIPS 2012

Overview

Reinforcement Learning and Rewards

Sutton Barto 1998

Prior knowledge in RL

- In the form of a **Reward** function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- Find **Policy** π Maximizing $\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}\big(s_t, \pi(s_t)\big)\Big]$

Reinforcement Learning and Rewards

Sutton Barto 1998

Prior knowledge in RL

- In the form of a **Reward** function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- Find **Policy** π Maximizing $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \pi(s_t))\right]$

Bottlenecks

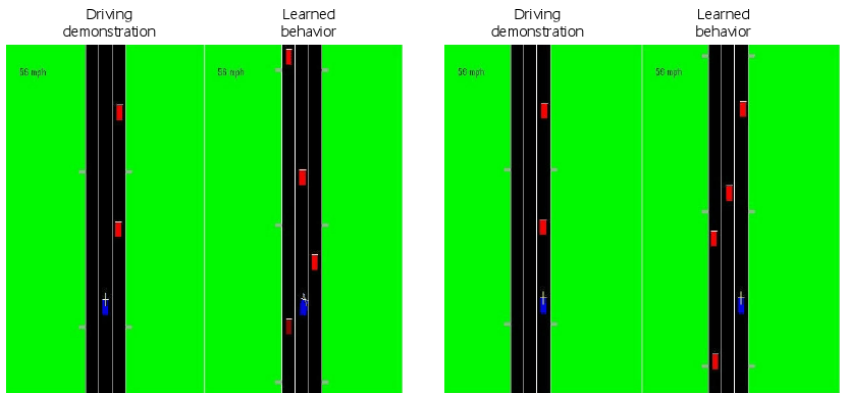
- Rewards $\equiv$ ground truth challenges in-situ
- In a swarm context $\mathcal{R}$ can be
  - Centralized: $\mathcal{R} : (\mathcal{S} \times \mathcal{A}) \times \cdots \times (\mathcal{S} \times \mathcal{A}) \mapsto \mathbb{R}$
    (global vision, tractability issues)
  - Decentralized: $\mathcal{R}_1 : (\mathcal{S} \times \mathcal{A}) \mapsto \mathbb{R}, \ldots, \mathcal{R}_N : (\mathcal{S} \times \mathcal{A}) \mapsto \mathbb{R}$
    - Tractable: Every robot optimize its own reward
    - Trials and Errors process to tune it

Inverse Reinforcement Learning?

Ng Russell 00, Abbeel Ng 04, Kolter et al. 07, ..

Prior knowledge in Inverse Reinforcement Learning

- Expert demonstrates a **good behavior** $\{s_t, a_t, s_{t+1}\}$



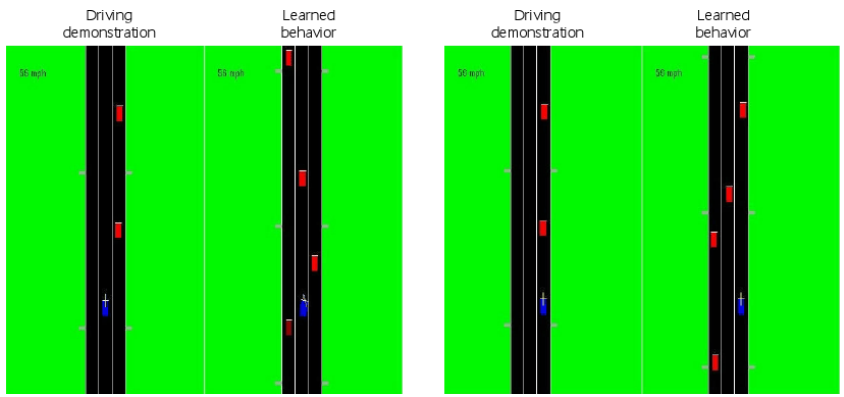
Abbeel & Ng 04

Inverse Reinforcement Learning?

Ng Russell 00, Abbeel Ng 04, Kolter et al. 07, ..

Prior knowledge in Inverse Reinforcement Learning

- Expert demonstrates a **good behavior** $\{s_t, a_t, s_{t+1}\}$



Abbeel & Ng 04

- From this, learn a reward function $\mathcal{R}$

$$\forall a \neq a_t, Action\_Value(s_t, a_t) \geq Action\_Value(s_t, a)$$

- **Then apply standard RL!**

What if no idea about a good behavior

Alan Winfield & Wenguo Liu 08

each point is a robot

Preference-based Policy Learning

Step 1: use expert's feedback to learn the goal (PPL)

Akrour et al. 2011

- **Prior knowledge:** pairwise preferences over behaviors
- Expert become a *critic* instead of a *performer*
- **Iterate**
  - **Agents:** Demonstrate a behavior
  - Expert: Compare behavior with previous ones (better/worse)
  - **Agents:** Optimize expert preferences model + exploration term

Preference-based Policy Learning

Step 1: use expert's feedback to learn the goal (PPL)

Akrour et al. 2011

- **Prior knowledge:** pairwise preferences over behaviors
- Expert become a *critic* instead of a *performer*
- **Iterate**
  - **Agents:** Demonstrate a behavior
  - Expert: Compare behavior with previous ones (better/worse)
  - **Agents:** Optimize expert preferences model + exploration term

Step 2: reduce expert's burden (APRIL)

Akrour et al. 2012

- A **hundred** of demonstrations to find a satisfying π in our exp.
- How can we reduce "Expert Sample Complexity"?

Preference-based Policy Learning

Step 1: use expert's feedback to learn the goal (PPL)

Akrour et al. 2011

- **Prior knowledge:** pairwise preferences over behaviors
- Expert become a *critic* instead of a *performer*
- **Iterate**
  - **Agents:** Demonstrate a behavior
  - Expert: Compare behavior with previous ones (better/worse)
  - **Agents:** Optimize expert preferences model + exploration term

Step 2: reduce expert's burden (APRIL)

Akrour et al. 2012

- A **hundred** of demonstrations to find a satisfying π in our exp.
- How can we reduce "Expert Sample Complexity"?
- Active Learning!?

Step 1. Preference-based Policy Learning

1. Demonstrate two policies
2. Ask the user her preference
3. Train a preference model $\mathcal{J}_t$ SVM ranking
4. Self-train: find a policy π maximizing $\mathcal{J}_t$
5. ... $+\alpha_t$ Novelty adaptive exploration wrt archive
6. Demonstrate π, iterate

- α_t increases when success

Which space ?

Environment helps!

- Parametric Representation
  policy π in $\mathbb{R}^D$ NN weight vector
- Behavioral Representation
  $\pi \rightarrow$ trajectory $\rightarrow$ histogram of sensorimotor states $\mathbb{R}^d$

Comments

- Expert interested in robot behavior (not in NN weights)
- Mapping $\mathbb{R}^D \mapsto \mathbb{R}^d$ non Lipschitz
  small variations in $\mathbb{R}^D \rightarrow$ *large variations in* $\mathbb{R}^d$

$\rightarrow$ **Learn the expert's preference model in** $\mathbb{R}^d$

Modelling the expert's preferences

Akrour et al., 2011

A system of values V

- For i-th sensorimotor state, a weight $v[i]$
- Map π onto its sms histogram $p_\pi[i]$ $\quad 1 \dots d$

$$V(\pi) = \langle v, p_\pi \rangle$$

Rank-based learning

Joachims 05

Given $\pi^{(1)} \prec \dots \prec \pi^{(k)}$, minimize

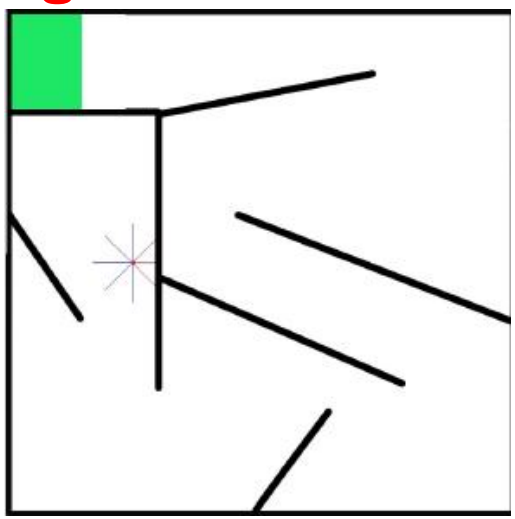
$$\frac{1}{2}||w||^2$$

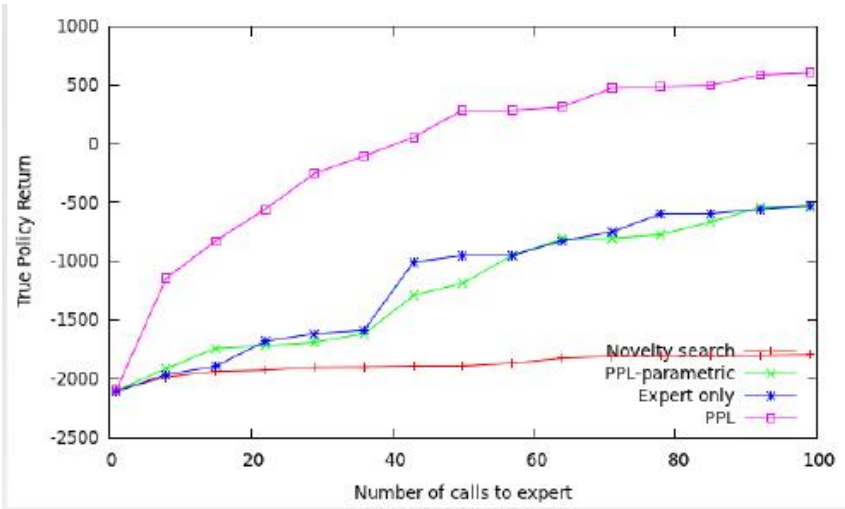
subject to

$$\langle w, p_\pi^\ell \rangle < \langle w, p_\pi^{\ell+1} \rangle + 1 \quad \ell = 1 \dots k-1$$

Validation

Getting out of a maze





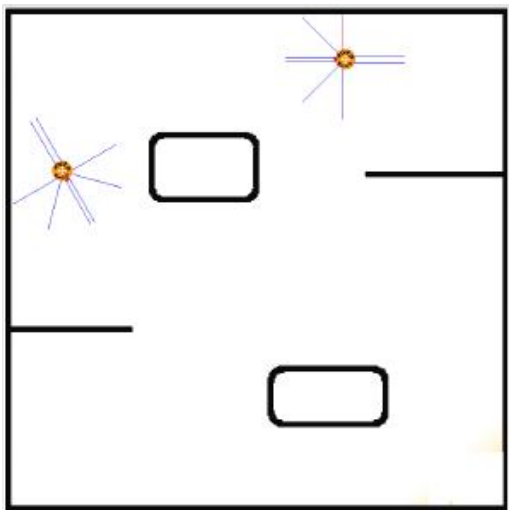
Comments

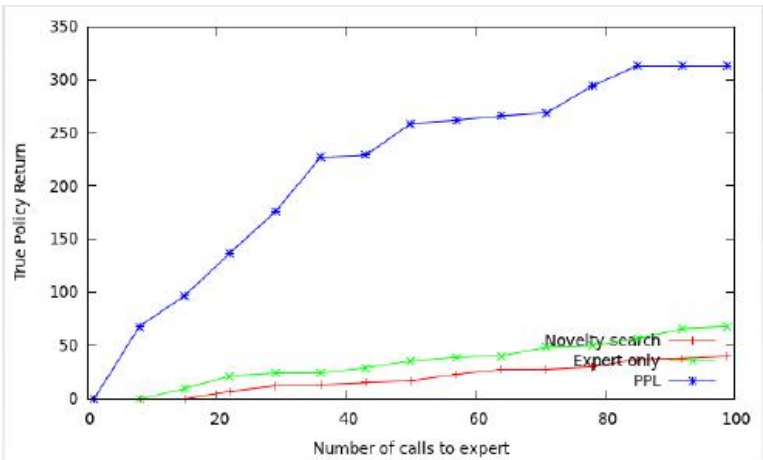
- PPL_d reaches the goal after 39 interactions (saves 3/4 interactions)
- PPL_D inefficient; Novelty search (Stanley 2010) inefficient.

Validation, 2

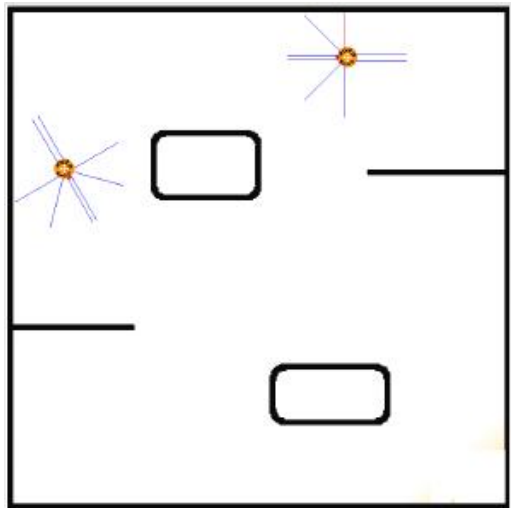
Coordinated exploration of an arena

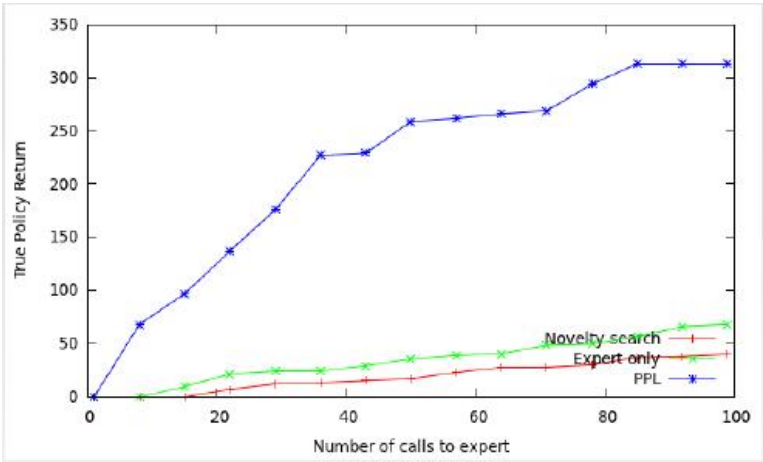
Two independent robots, operated with same controller; goal is to maximize the number of zones simultaneously visited by both robots.





Validation, cont'd





Comments

- More challenging goal
  no visual primitive (see other robot, see an obstacle
- PPL_d efficient (saves 9/10 interactions)
- PPL_D inefficient; Novelty search (Stanley 2010) very inefficient (large search space).

Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose $\mathbf{u} = \arg\max J_w(u)$?

Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose $\mathbf{u} = \arg\max J_w(u)$?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose $\mathbf{u} = \arg\max J_w(u)$?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Proposal

Select $\mathbf{u}$ maximizing *Expected Utility of Selection* (**EUS**) of candidate $\mathbf{u}$ w.r.t $\mathcal{U}_t$ Viappiani & Boutilier 10

Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose $\mathbf{u} = \arg\max J_w(u)$?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Proposal

Select $\mathbf{u}$ maximizing *Expected Utility of Selection* (**EUS**) of candidate $\mathbf{u}$ w.r.t $\mathcal{U}_t$ Viappiani & Boutilier 10

$$EUS(u;\mathcal{U}_t) = \mathbb{E}_w[\max(\langle w,u\rangle, \langle w,u_t^*\rangle)]$$

Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose $\mathbf{u} = \arg\max J_w(u)$?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Proposal

Select $\mathbf{u}$ maximizing *Expected Utility of Selection* (**EUS**) of candidate $\mathbf{u}$ w.r.t $\mathcal{U}_t$ Viappiani & Boutilier 10

$$EUS(u;\mathcal{U}_t) = \mathbb{E}_w[\max(\langle w,u\rangle, \langle w,u_t^*\rangle)]$$

$u \succ u^*$

$$\mathbb{E}_{w\in W+}[\langle w,u\rangle]$$

Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose $\mathbf{u} = \arg\max J_w(u)$?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Proposal

Select $\mathbf{u}$ maximizing *Expected Utility of Selection* (**EUS**) of candidate $\mathbf{u}$ w.r.t $\mathcal{U}_t$ Viappiani & Boutilier 10

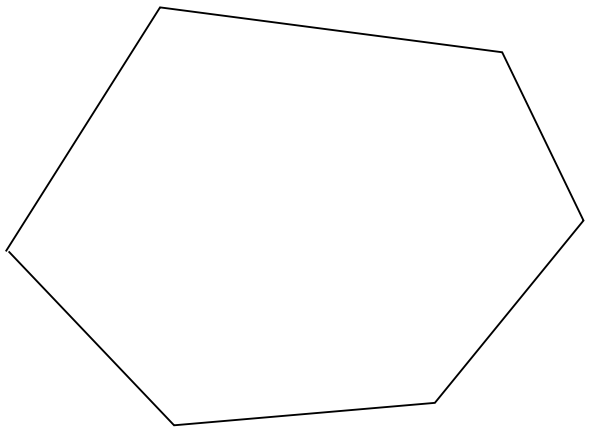
$$EUS(u;\mathcal{U}_t) = \mathbb{E}_w[\max(\langle w,u\rangle, \langle w,u^*_t\rangle)]$$

$u \succ u^*$ $\qquad$ $u \prec u^*$

$$\mathbb{E}_{w\in W+}[\langle w,u\rangle] \quad + \quad \mathbb{E}_{w\in W-}[\langle w,u^*_t\rangle]$$

Implementation

EUS **Intractable** (in practice, $dim(\mathbf{u}) > 1000$)

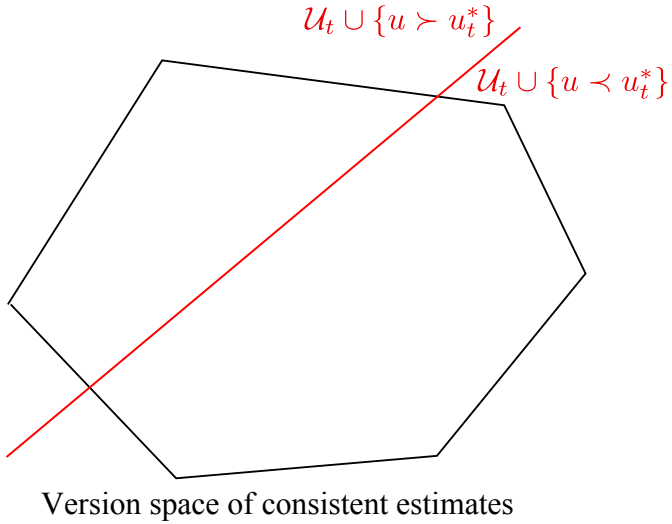


Version space of consistent estimates

- All preference constraints define a version space

Implementation

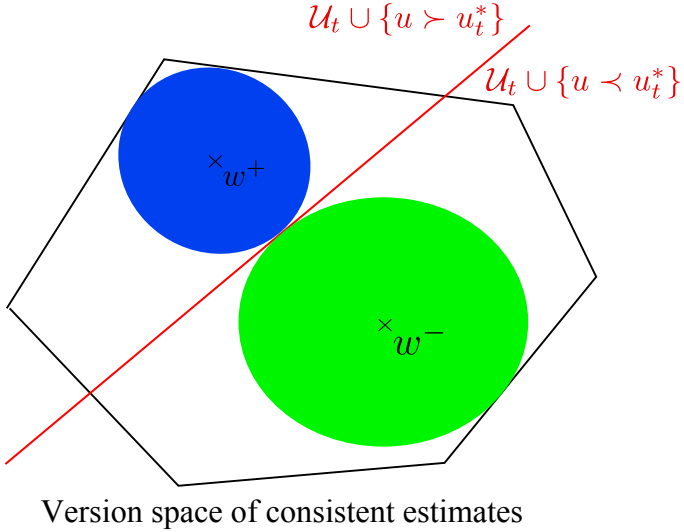
EUS **Intractable** (in practice, $dim(\mathbf{u}) > 1000$)



- All preference constraints define a version space
- A candidate behavior **u** splits the VS in two

Implementation

EUS **Intractable** (in practice, $dim(\mathbf{u}) > 1000$)

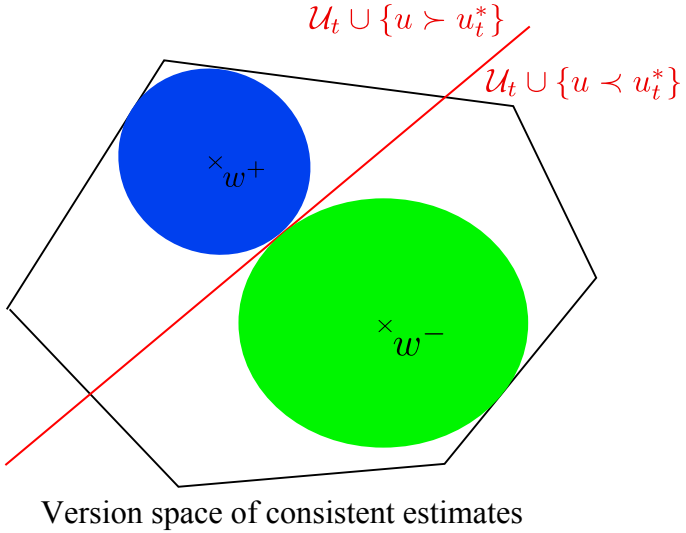


Version space of consistent estimates

- All preference constraints define a version space
- A candidate behavior $\mathbf{u}$ splits the VS in two
- w^+ and w^- solution of the associated ranking problem

Implementation

EUS **Intractable** (in practice, $dim(\mathbf{u}) > 1000$)



Version space of consistent estimates

- All preference constraints define a version space
- A candidate behavior $\mathbf{u}$ splits the VS in two
- w^+ and w^- solution of the associated ranking problem

Approximated Expected Utility of Selection

$$AEUS(\mathbf{u}; \mathcal{U}_t) = \frac{\langle w^+, \mathbf{u} \rangle}{F^+} + \frac{\langle w^-, \mathbf{u}_t^* \rangle}{F^-}$$

Policy selection criteria

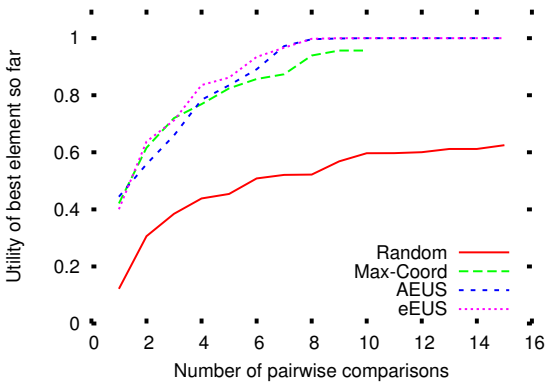
$$\pi_t = \arg\max_{\pi} \mathbb{E}_{\mathbf{u} \sim \pi}[AEUS(u)]$$

APRIL Algorithm

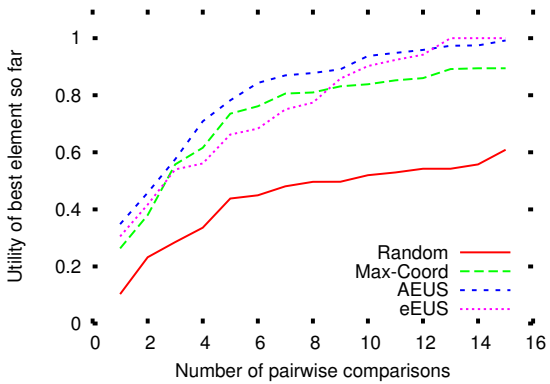
- $\pi_0 \leftarrow$ *random*
- $\mathbf{u}_0 =$ demonstration of π_0
- Archive $\mathcal{U}_0 = \{\mathbf{u}_0\}$
- FOR$t = 0 \rightarrow T$ (while **Expert** cooperates)
  - **(R)** Select $\pi_{t+1} = \arg\max\{\mathbb{E}_{u\sim\pi}[AEUS(\mathbf{u};\mathcal{U}_t)]\}$
  - **(R)** Demonstrate $\mathbf{u}_{t+1}$ from policy π_{t+1} to the expert
  - **(E)** Expert ranks $\mathbf{u}_{t+1}$ and archive $\mathcal{U}_t$ is updated.

  ENDFOR

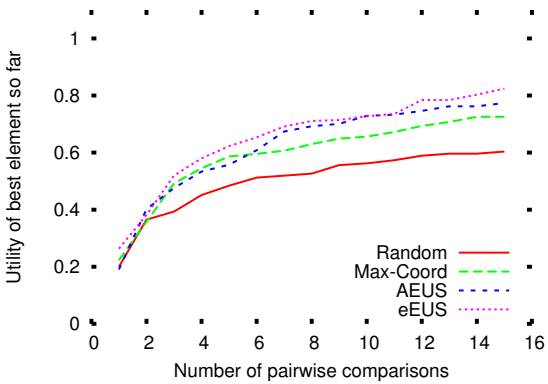
Experimental Validation of *AEUS*



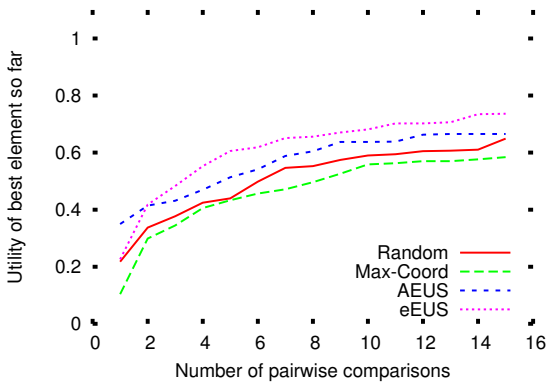
$d = 10$



$d = 20$



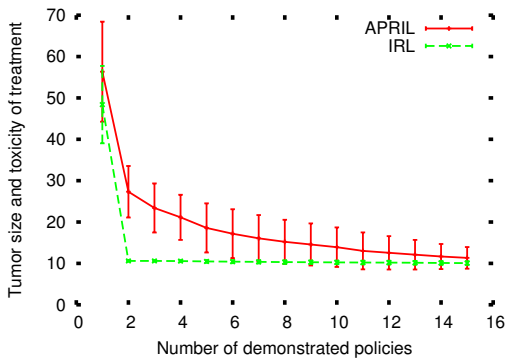
$d = 50$



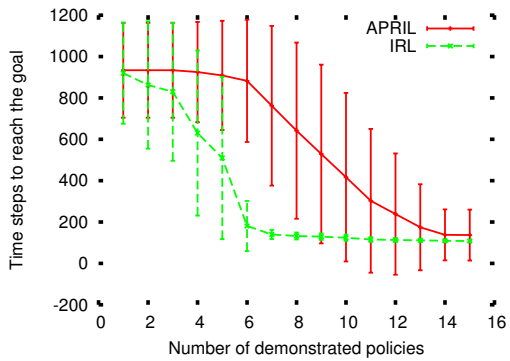
$d = 100$

- Sample $w* \in d$-dimensional L_2-unit-sphere
- $S = \{\mathbf{u}_1, \ldots \mathbf{u}_{1000}\}$ sampled unif. from L_1-unit-sphere
- Find $\arg\max_{u \in S} \langle w^*, u \rangle$ using minimal number of pairwise comparisons
- Compare *AEUS* with *SEUS* (SEUS = sample 10,000 w in the VS to approx. EUS)
- **Result: *AEUS* matches closely *SEUS* !**

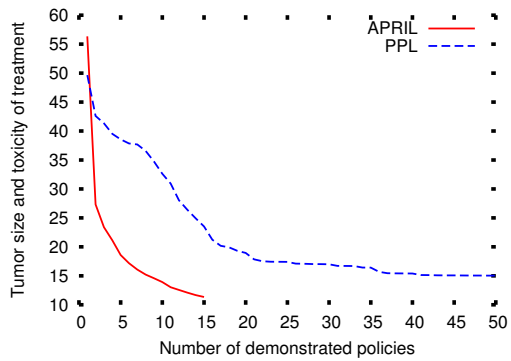
Policy Learning Tasks



APRIL vs IRL on Cancer problem



APRIL vs IRL on Mountain Car



APRIL vs PPL on Cancer problem

APRIL vs IRL

- Two RL benchmarks: Mountain Car and Cancer Treatment
- What's the cost of not having a demonstration as input?
- **15 pairwise comparisons!**

APRIL vs PPL

- Huge gain compared to **non-active** variant

Conclusions and Perspective

Learning a Policy

- Can make it with few bits of **external** information

Weakness

- Computational heavy (succession of optimization problems)

Next

- Better analysis of the *AEUS* approximation
- Multiple instance ranking