Co-evolution of Morphology and Control for Roombots

Master Thesis Presentation

Ebru Aydın
Advisors:
Prof. Auke Jan Ijspeert
Rico Möckel
Jesse van den Kieboom
Soha Pouya
Alexander Spröwitz

Co-evolution

General Approach : Hardware , Software
Body - brain evolution

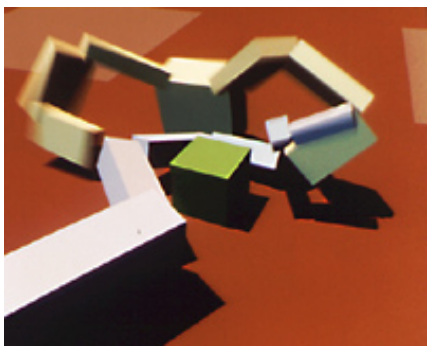
- Building Blocks
  - sticks, actuated joints, artificial neurons
- Encodings
  - direct, direct recurrent, cellular, generative

Developmental (Generative) Encoding

- The algorithm to build the robot
- Biologically inspired
- Reuse of components
- Self-similar structures
- Compact representation
- L-system grammars, tree based

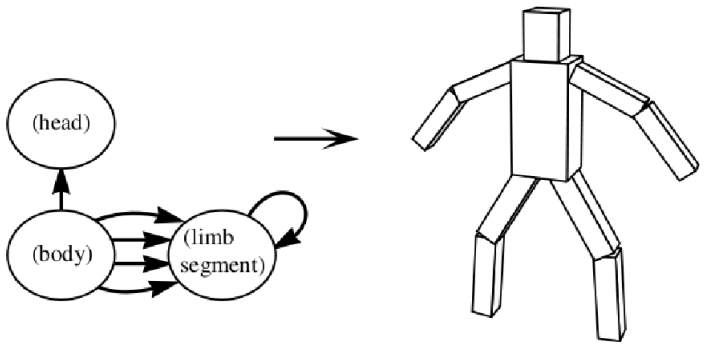
Previous work: Karl Sims

Sims 1994, Evolving 3D Morphology and Behavior



(a) Competing

(b) Swimming



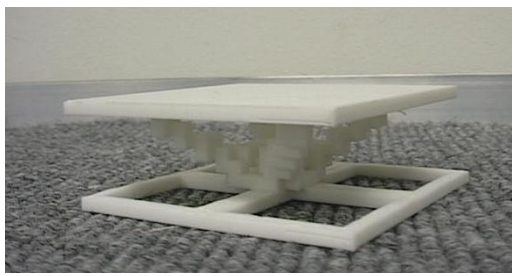
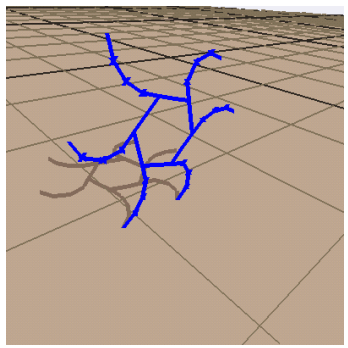
(c) Phenotype and genotype

- Neural network for controller
- Tree based encoding for morphology, self similar structures, code reuse
- A variety of locomotion styles and behavior strategies are evolved

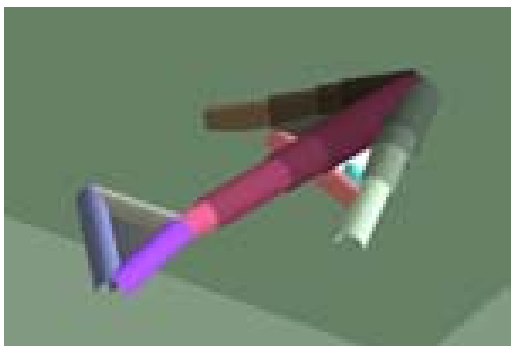
Previous work: Demo Lab

Projects from Demo Lab : Hornby, Pollack, Lipson

- Generative Representations for Design Automation
  Static structures, 2D and 3D robots are evolved.


- The Golem Project

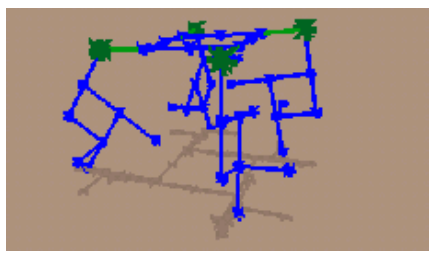



Previous work: Demo Lab

Generative encoding builds creatures with better locomotion ability then non-generative encoding

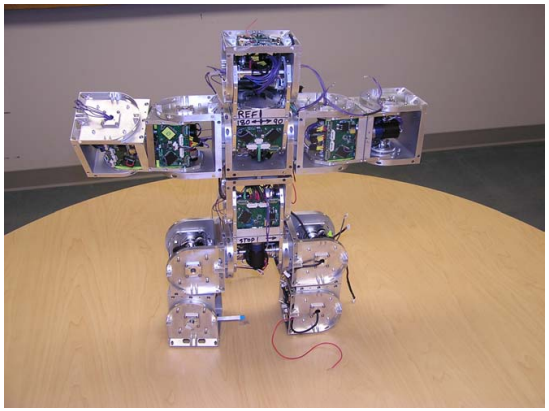
Generative encoding brings:

- Scalability through self-similar structures
- Hierarchical structure
- Compact representation

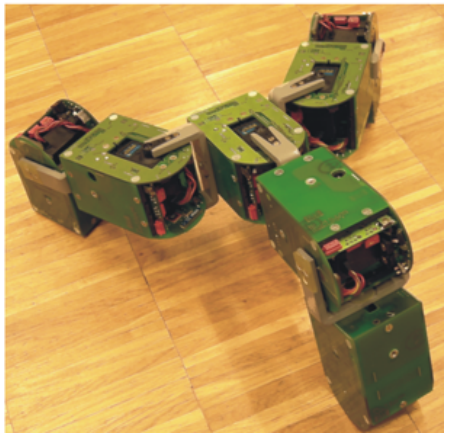


Modular robots

(k) MTRAN



(l) Superbot

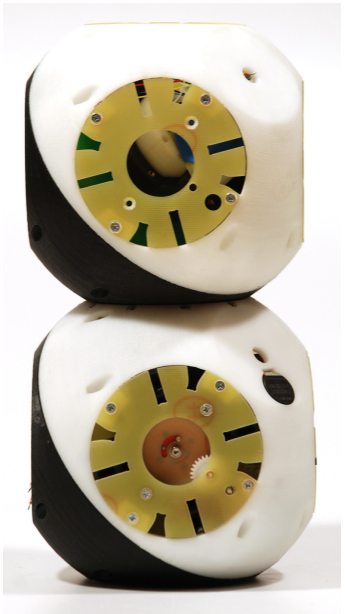


(m) YAMOR

- Detach-attach
- Self-configuration
- Adaptation

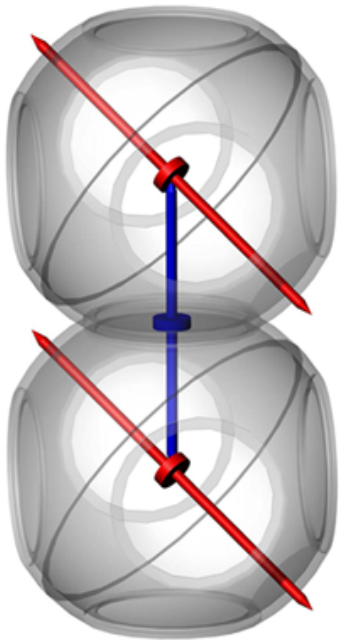
Roombots

- Self-reconfigurable
- Self sufficient
- Uniform
- $220mm \times 110mm \times 110mm$, $1.4kg$
- ACM
- 3 DOF - Continious rotation



Roombots

- Self-reconfigurable
- Self sufficient
- Uniform
- $220mm \times 110mm \times 110mm$, $1.4kg$
- ACM
- 3 DOF - Continious rotation



Co-evolution for the Roombots

Modular robots can assemble into many different structures
Developmental encoding : Self-similarity, compactness, modularity

- Co-evolution for Roombots
  - Morphological Building Blocks: Roombots and Passive elements
  - Controller : Coupled Phase Oscillators
- Encoding (L-systems), combine morphology and controller

L-systems

- Re-write systems
- Growth process of the organisms
- Formal grammars

$$G = (V, w, P)$$

$$w \in V^+$$

$$P \subset V \times V^*$$

Examples

$$V = \{a, b\}$$
$$w = a$$
$$P = a \rightarrow ab, b \rightarrow a$$

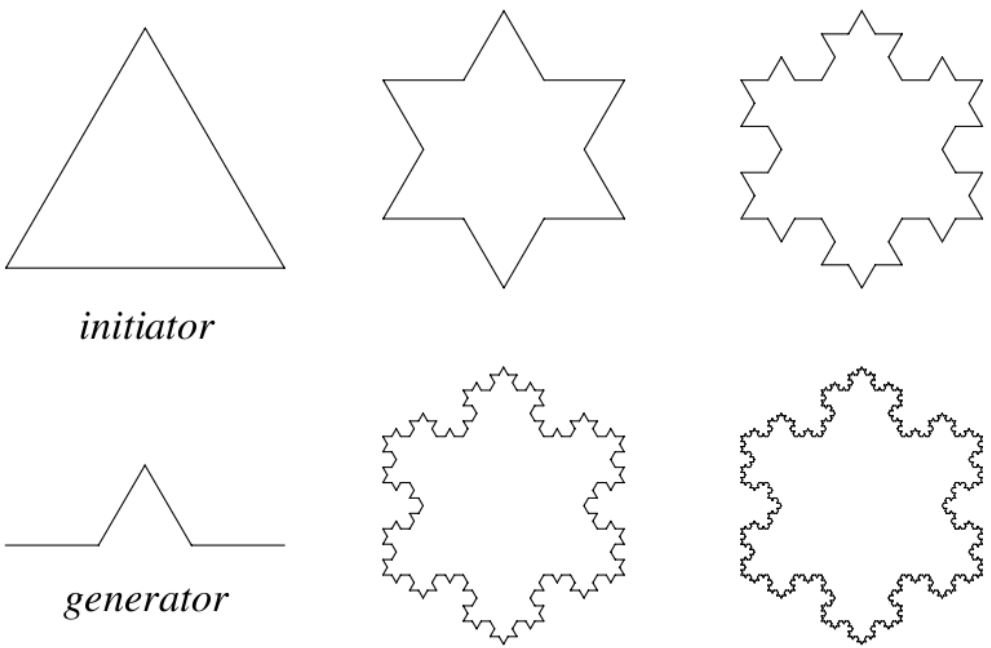
Starting from axiom, a, the following sequence is generated.

Step0 : a
Step1 : ab
Step2 : aba
Step3 : $abaab$

Examples

Turtle Rules

$$V = \{F, +, -\}$$
$$w = F - -F - -F$$
$$P = F \rightarrow F + F - -F + F$$



Examples

Branching L-systems [ ]

$$V = \{F, +, -, [,]\}$$
$$w = F$$
$$P = F \rightarrow F[+F]F[-F[+F][-F]]F$$

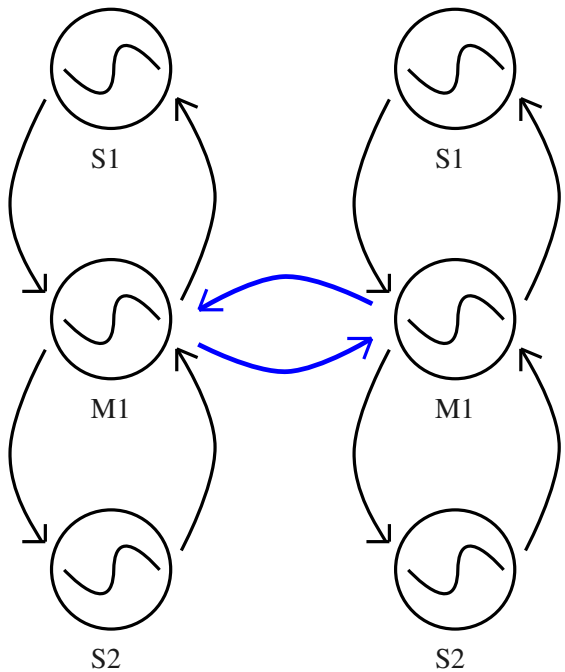
L-system for the Roombots

- Deterministic branching L-systems
- An L-system $\longrightarrow$ a robot
- Additive turtle like interpretation
- Constants $\longrightarrow$ Build commands
- Build commands :
  1. Add a new part
  2. Modify an open parameter

Controller

- CPG, rhythmic patterned output
- Limit cycle behavior $\longrightarrow$ robust
- Parameterize

servo motor $\longleftrightarrow$ **oscillator**



Controller

$$\dot{\phi}_i = 2 \cdot \pi \cdot \nu + \sum \omega_{ij} \cdot r_j \cdot \sin\left(\phi_j - \phi_i - \psi_{ij}\right)$$

$$\dot{r}_i = a(R_i - r_i)$$

$$\dot{x}_i = b(X_i - x_i)$$

Fixed parameters : ν , ω_{ij} , a , b

Open parameters : R_i , ψ_{ij} , X_i

Controller

$$\dot{\phi}_i = 2 \cdot \pi \cdot \nu + \sum \omega_{ij} \cdot r_j \cdot \sin\left(\phi_j - \phi_i - \psi_{ij}\right)$$
$$\dot{r}_i = a(R_i - r_i)$$
$$\dot{x}_i = b(X_i - x_i)$$

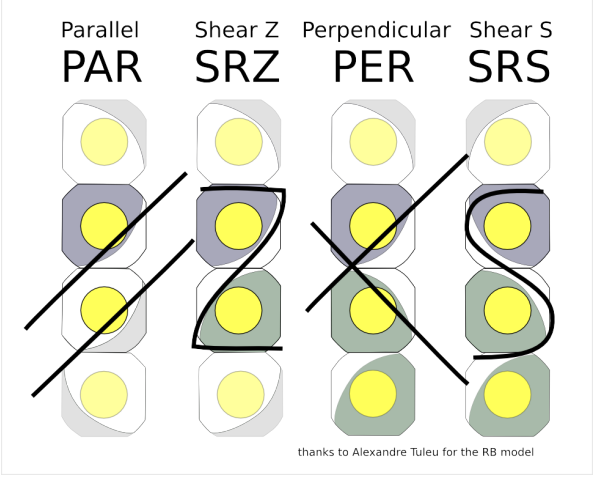
Fixed parameters : ν , ω_{ij} , a , b
Open parameters : R_i , ψ_{ij} , X_i

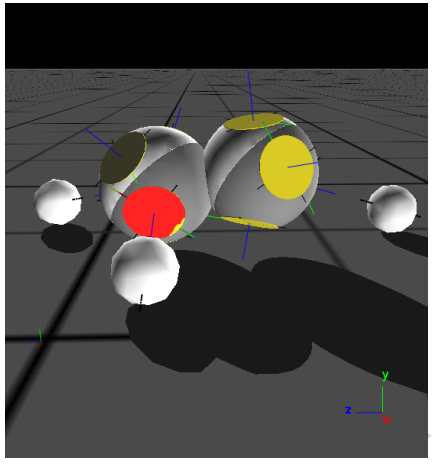
Drive functions:

$$\theta_i = r_i \cdot \sin\left(\phi_i\right) + x_i \qquad Oscillation$$
$$\theta_i = \pm\phi_i \qquad Rotation$$
$$\theta_i = x_i \qquad Locked$$

Morphology

- Roombots modules
  - Connection face
  - Connection type
  - Phase offset
- Passive elements
  , spring-damper systems
  - Connection face
  - Length





State

State : Module , Connection face , Phase bias

State update commands:

- **ADD** *CONNECTION TYPE, CONNECTION FACE*
- **CHANGE_FACE** *N*
- **CHANGE_PHASE_BIAS_BM** *N*

State

State : Module , Connection face , Phase bias

State update commands:

- **ADD** *CONNECTION TYPE, CONNECTION FACE*
- **CHANGE_FACE** *N*
- **CHANGE_PHASE_BIAS_BM** *N*

"CHANGE_VALUE N" command is interpreted as:

$$VALUE \equiv VALUE + N \pmod{S}$$

$$ACTIVE\_FACE \equiv ACTIVE\_FACE + N \pmod{4}$$

Build Commands:

- **BRANCH OPERATOR : [ ]**
- **REPETITION OPERATOR :** { } *N*
- **ADD** *CONNECTION TYPE, CONNECTION FACE*
- **ADD_PASSIVE_LEG:** *LENGTH*

Modify Control Parameters:

- **CHANGE_AMPLITUDE** *OSC N*
- **CHANGE_OFFSET** *OSC N*
- **CHANGE_DRIVE_FUNCTION** *OSC N*
- **CHANGE_PHASE_BIAS** *COUPLING N*

L-system Grammars

- Deterministic , context-free, branching
- Four rules
- Non-recursive
- The grammar encodes the build process of the robot

Table: Build Commands' Abbreviations

Command	**Abbreviation**
ADD	ADD
CHANGE_FACE	CF
CHANGE_PHASE_BIAS_BM	CPBBM
CHANGE_PHASE_BIAS	CPB
CHANGE_AMPLITUDE	CA
CHANGE_OFFSET	CO
CHANGE_DRIVE_FUNCTION	CDF

An example grammar

R1 → CPB(s1::m1 90) CA(m1 1.4) CO(s2 150) CPB(m1::s2 220) CO(m1 290)
R2 → ADD(C1Y PER) CDF(s1 3) CDF(s2 3) R1 CA(s2 1.4)
R3 → CPB(s1::m1 100) R2 CA(m1 2.5) 1 x { CA(m1 2.4) }
R4 → CDF(m1 2) CO(s2 340) CPBBM(100) R2 R2 R3 CF(2)

Rewrite steps

step0 : R4

Rewrite steps

step0 : R4

step1 : CDF(m1 2) CO(s2 340) CPBBM(100) R2 R2 R3 CF(2)

Rewrite steps

step0 : R4

step1 : CDF(m1 2) CO(s2 340) CPBBM(100) R2 R2 R3 CF(2)

step2 : CDF(m1 2) CO(s2 340) CPBBM(100) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) R1 CA(s2 1.4) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) R1 CA(s2 1.4) CPB(s1::m1 100) R2 CA(m1 2.5) 1 x { CA(m1 2.4) } CF(2)

Rewrite steps

step0 : R4

step1 : CDF(m1 2) CO(s2 340) CPBBM(100) R2 R2 R3 CF(2)

step2 : CDF(m1 2) CO(s2 340) CPBBM(100) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) R1 CA(s2 1.4) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) R1 CA(s2 1.4) CPB(s1::m1 100) R2 CA(m1 2.5) 1 x { CA(m1 2.4) } CF(2)

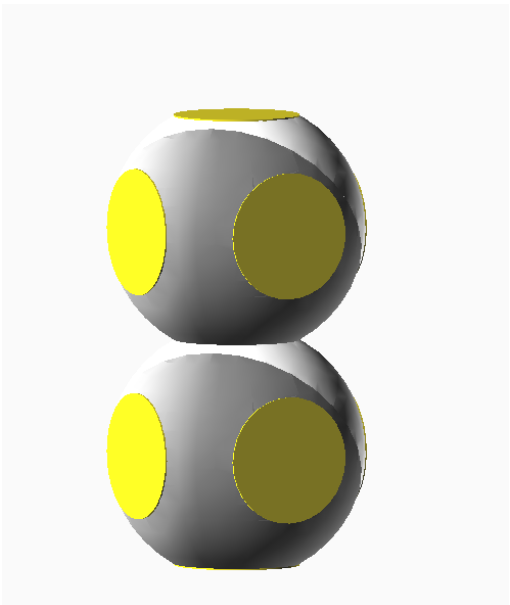
step3 : CDF(m1 2) CO(s2 340) CPBBM(100) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) CPB(s1::m1 90) CA(m1 1.4) CO(s2 150) CPB(m1::s2 220) CO(m1 290) CA(s2 1.4) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) CPB(s1::m1 90) CA(m1 1.4) CO(s2 150) CPB(m1::s2 220) CO(m1 290) CA(s2 1.4) CPB(s1::m1 100) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) R1 CA(s2 1.4) CA(m1 2.5) 1 x { CA(m1 2.4) } CF(2)

Rewrite Steps

step4 : CDF(m1 2) CO(s2 340) CPBBM(100) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) CPB(s1::m1 90) CA(m1 1.4) CO(s2 150) CPB(m1::s2 220) CO(m1 290) CA(s2 1.4) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) CPB(s1::m1 90) CA(m1 1.4) CO(s2 150) CPB(m1::s2 220) CO(m1 290) CA(s2 1.4) CPB(s1::m1 100) ADD(C1Y PER) CDF(s1 3) CDF(s2 3) CPB(s1::m1 90) CA(m1 1.4) CO(s2 150) CPB(m1::s2 220) CO(m1 290) CA(s2 1.4) CA(m1 2.5) 1 x { CA(m1 2.4) } CF(2)

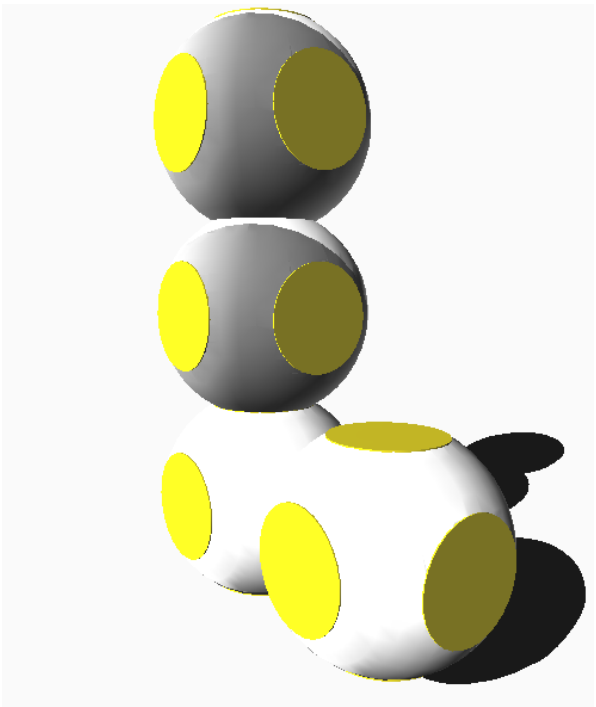
Interpretation of the string

CDF(m1 2) CO(s2 340) CPBBM(100) ...



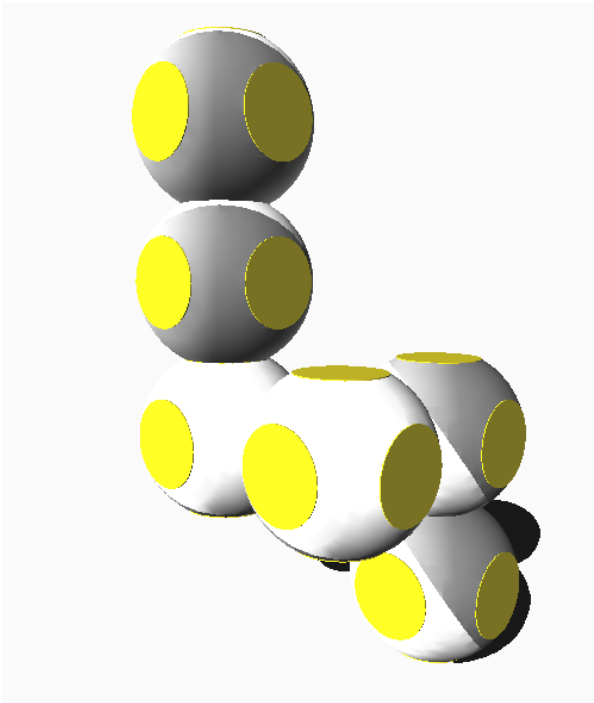
- CDF(m1 2) Rotation ($-\phi_i$)
- CO(s2 340)
- CPBBM(100) Update state

Interpretation of the string



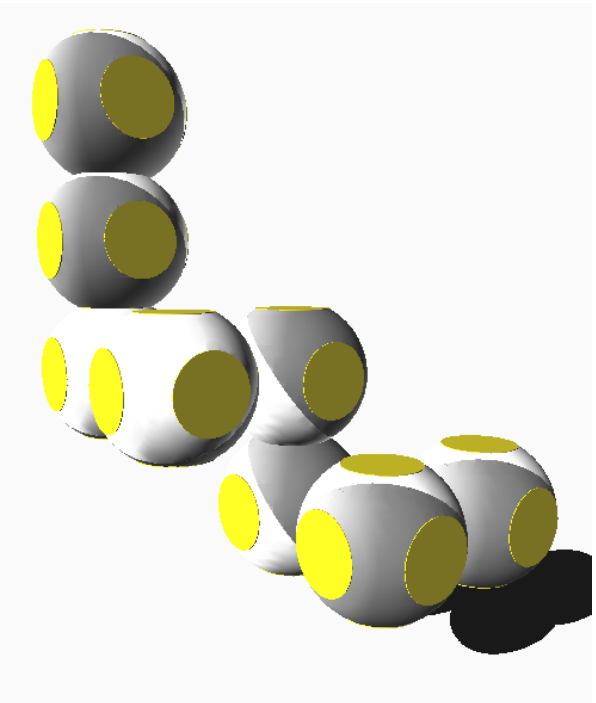
... ADD(C1Y PER)
CDF(s1 3) CDF(s2 3) CPB(s1::m1 90)
CA(m1 1.4) CO(s2 150) CPB(m1::s2
220) CO(m1 290) CA(s2 1.4) ...

Interpretation of the string



...
ADD(C1Y PER) CDF(s1 3) CDF(s2 3) CPB(s1::m1 90) CA(m1 1.4) CO(s2 150) CPB(m1::s2 220) CO(m1 290) CA(s2 1.4) CPB(s1::m1 100) ...

Interpretation of the string



... ADD(C1Y PER) CDF(s1 3) CDF(s2 3) CPB(s1::m1 90) CA(m1 1.4) CO(s2 150) CPB(m1::s2 220) CO(m1 290) CA(s2 1.4) CA(m1 2.5) CA(m1 2.4)

Evolutionary Algorithm

The L-system grammars are evolved.

a- **Mutation**

- Change parameter of a build command
- Add rule to rewrite rule
- Remove rule from rewrite rule

b- **Crossover :** Use an index, copy the rules from parents:

r1 | r2 r3 r4

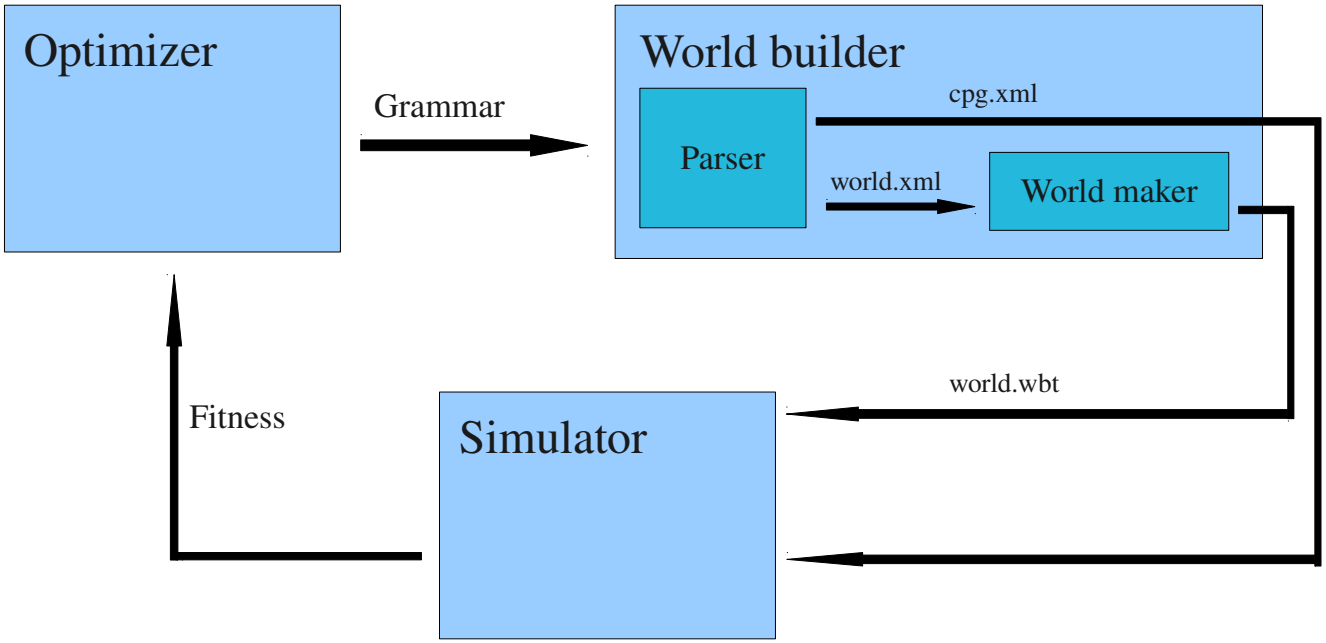
Evolutionary Algorithm

c- **Selection :** Roulette wheel

$$fitness = \frac{speed^a}{\dot{impact}^b \cdot torque^c}$$

- Evaluation in simulation
- Straight line motion
- Positive change of the impact, safe
- Energy efficiency

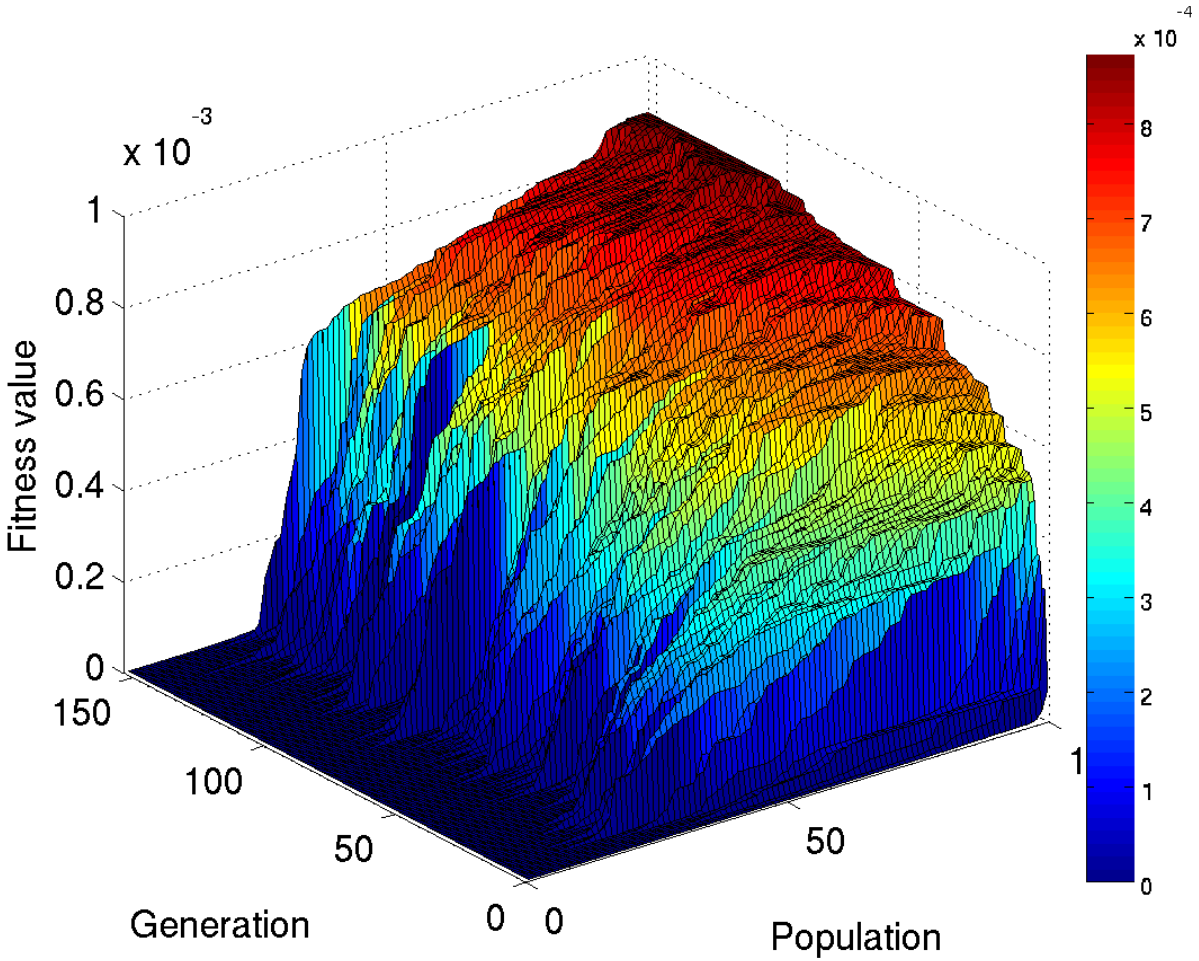
Implementation



Experiments

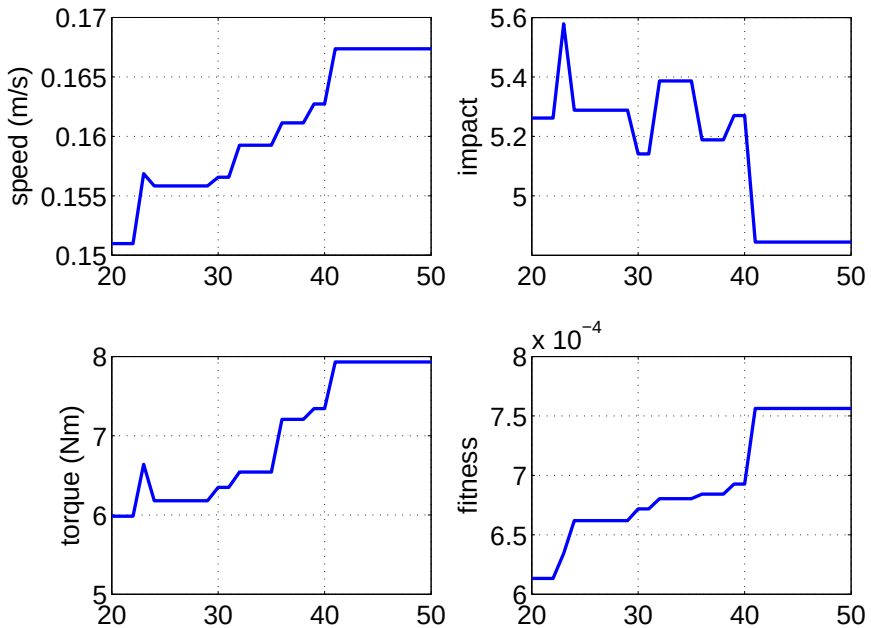
- Two sets of experiments
  - Roombots
  - Roombots and passive legs
- Generation limit 500^{th}
- Modules : 2 to 8
- Deterministic simulation environment
- Distributed over 40 computers
- 3 to 5 hours

An example Run

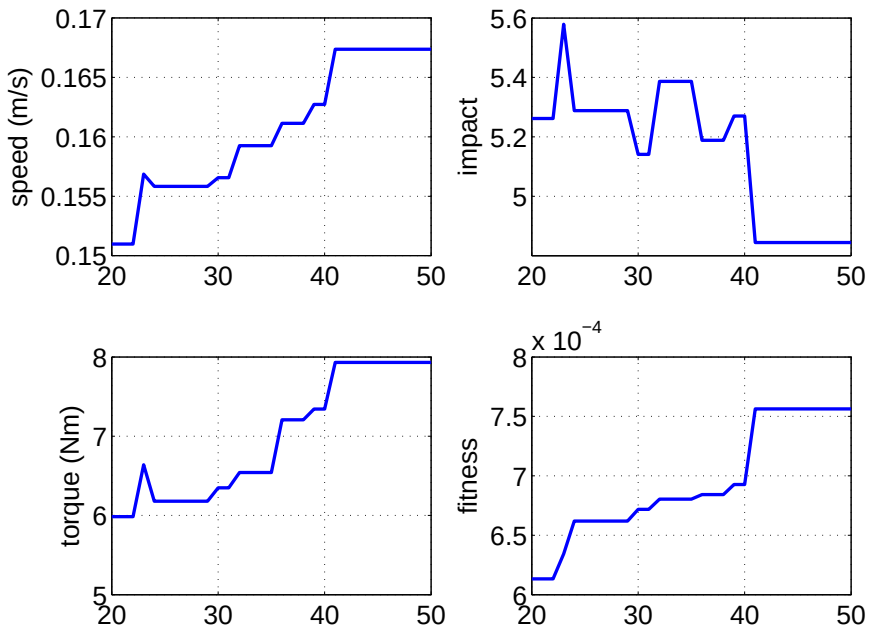


An example Run

Best fitness changed at 7 points between 20^{th} and 50^{th} generations.

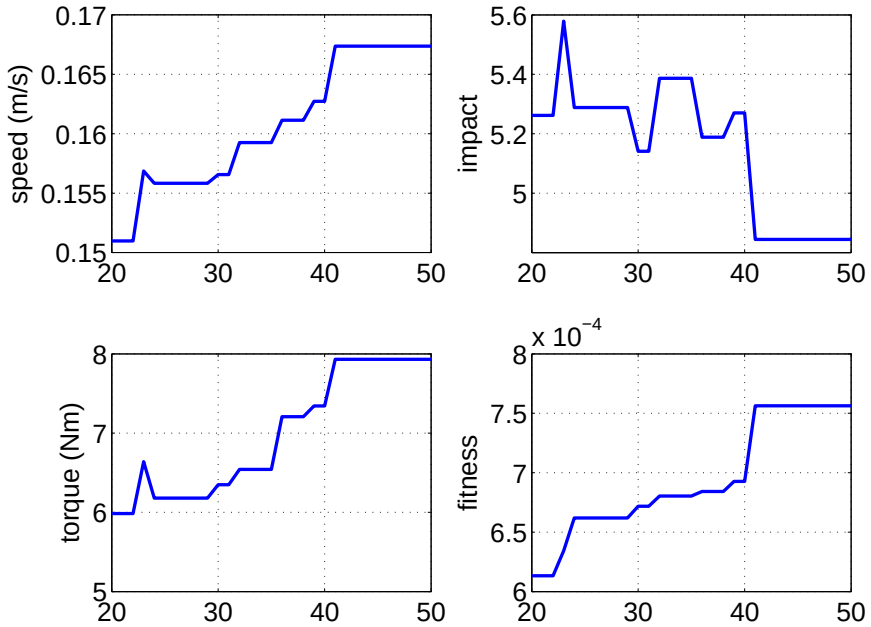


An example Run



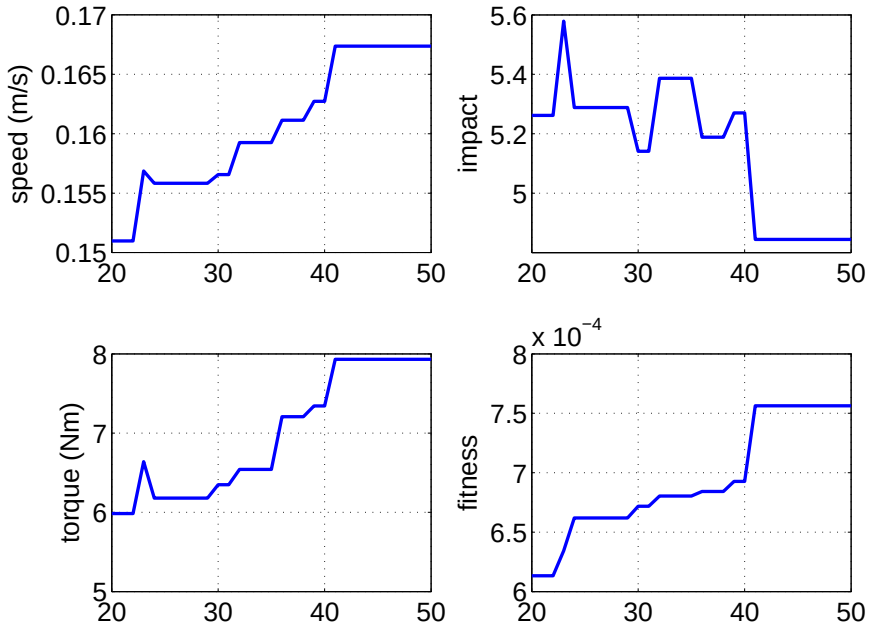
23^{rd}- Mutation (parameter modification), best of the previous generation

An example Run



24^{th}- Crossover of two average individual

An example Run



30^{th}- Crossover, same morphologies but different controllers

Evaluation of the results

Many different locomotion strategies are evolved
Videos ...

Roombots and Passive Elements Experiments

- Two types
  - Between modules (2 connection faces)
  - Attached to one module (1 connection face) (*)
- Telescopic motion
- Spring-damper systems
- Modules : 2 to 4
- Passive Legs : 2 to 6
- The aim is to explore effects of the passive compliant elements on the fitness criteria

Videos...

Comparison of the results

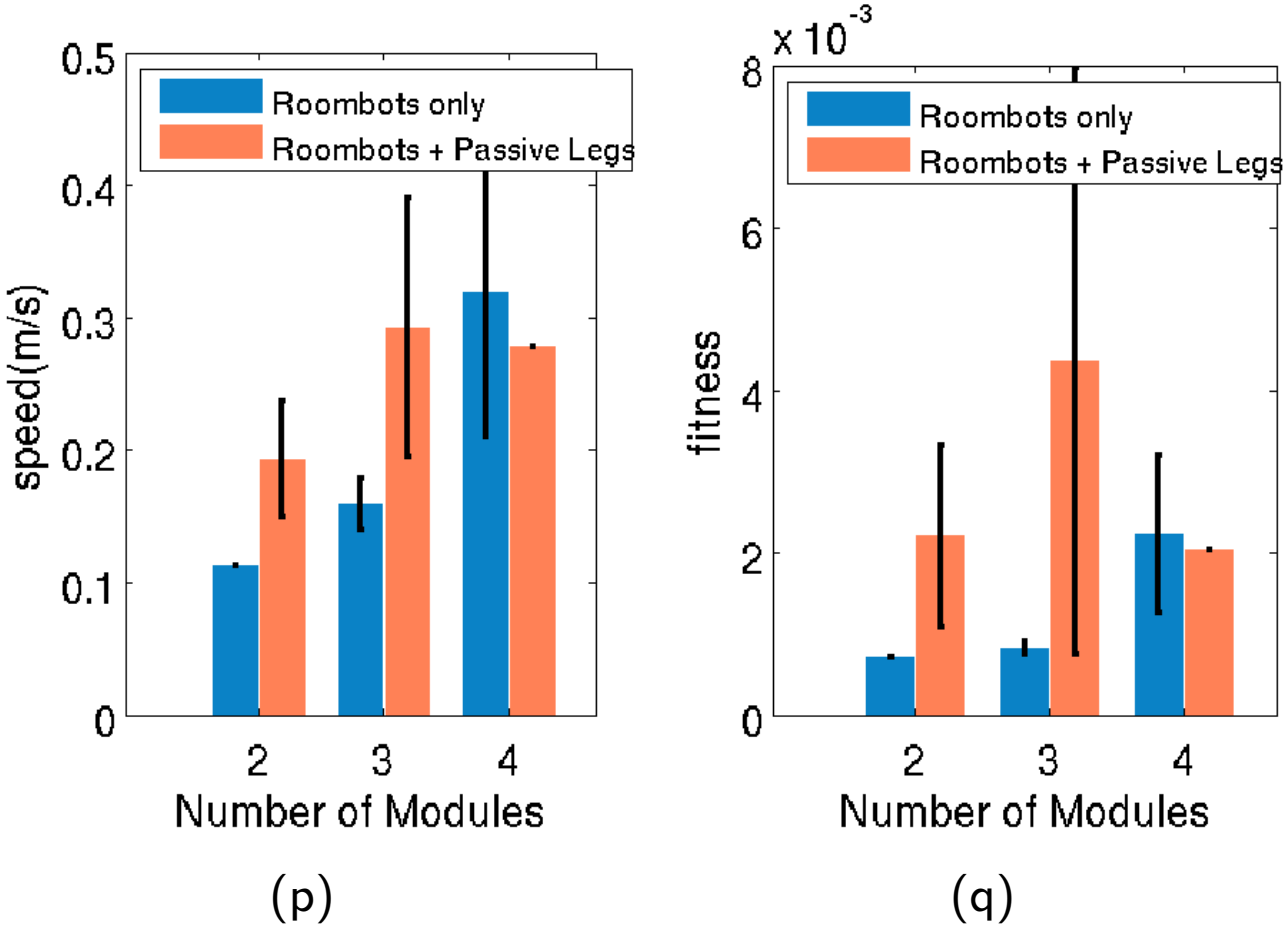


Figure: The average speeds and fitness values according to number of modules of the robots.

Comparison of the results

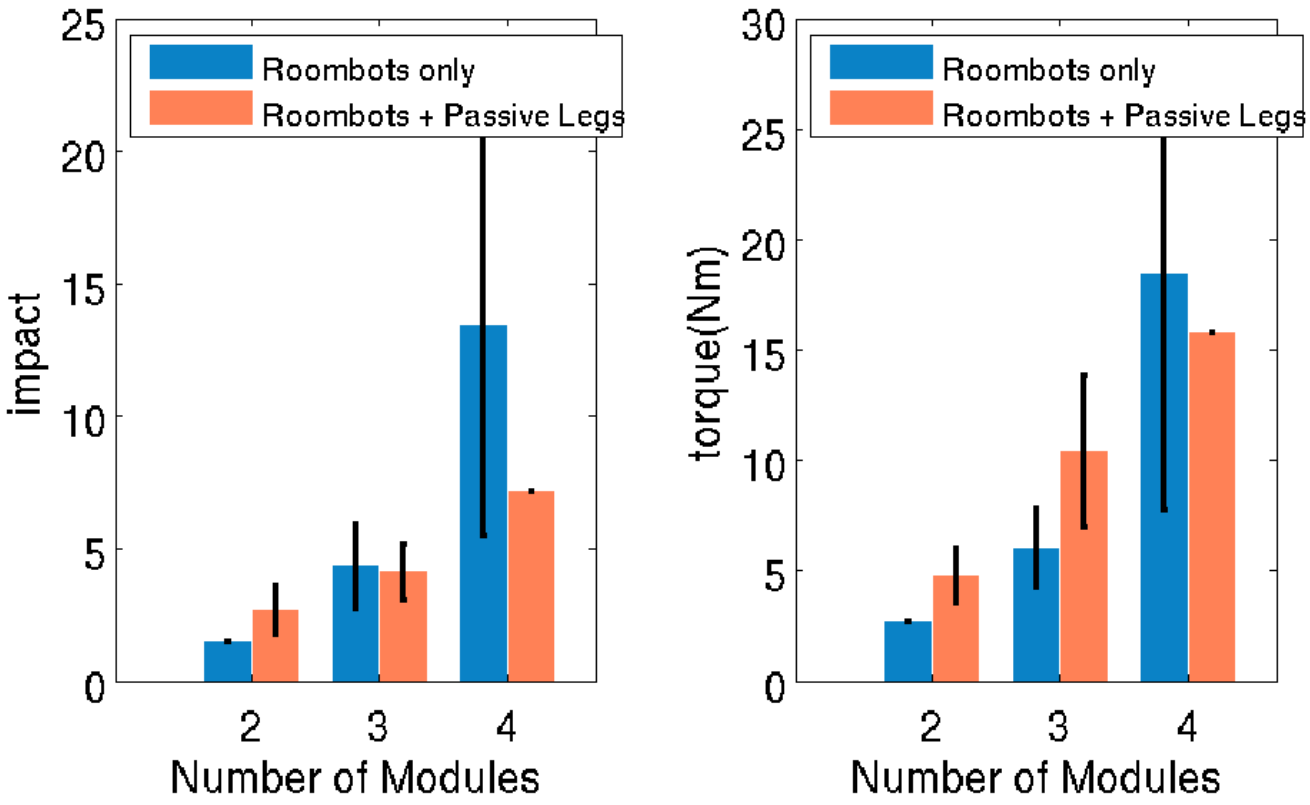


Figure: The average impact and torque values according to number of modules of the robots.

Passive Elements Between Modules

- Rotational and linear
- Unstable behavior
- Unrealistic speeds (up to $2m/s$)
- Robots disconnected
- Fast turn, gyroscope threshold did not work
- Reason could not be found
- ACM very stiff, each module is a robot
- A better model is needed

Conclusion

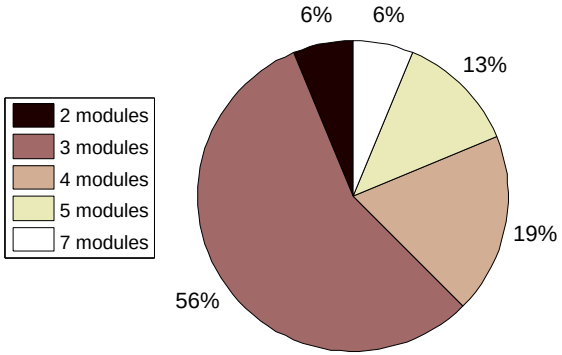
- Developmental process to evolve the morphology and controllers of the modular robots
- L-system grammars encoded the build process are evolved
- A variety of robots with different locomotion strategies are evolved
- The passive legs enable faster gaits by exploiting the energy stored at the legs
- The system can be used to test other types of passive elements
- The fitness criteria can be adapted for other tasks

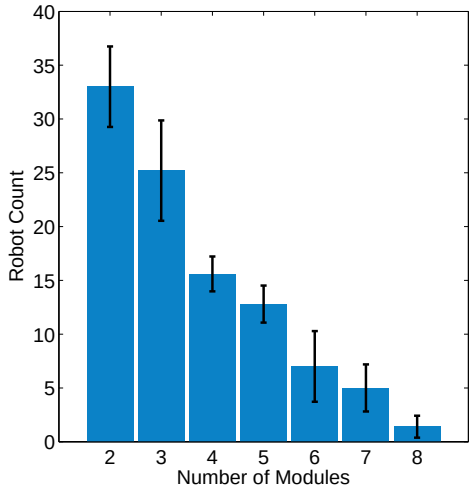
Future Work

- Sensory feedback
- Other tasks
- Unstable passive elements

Thank you for listening!
Questions...

Evaluation of the results





- Distribution of the results according to the number of modules of the fittest robot.
- The robot counts according to their modules in the initial population. Each generation is composed of 100 individuals

Evaluation of the results

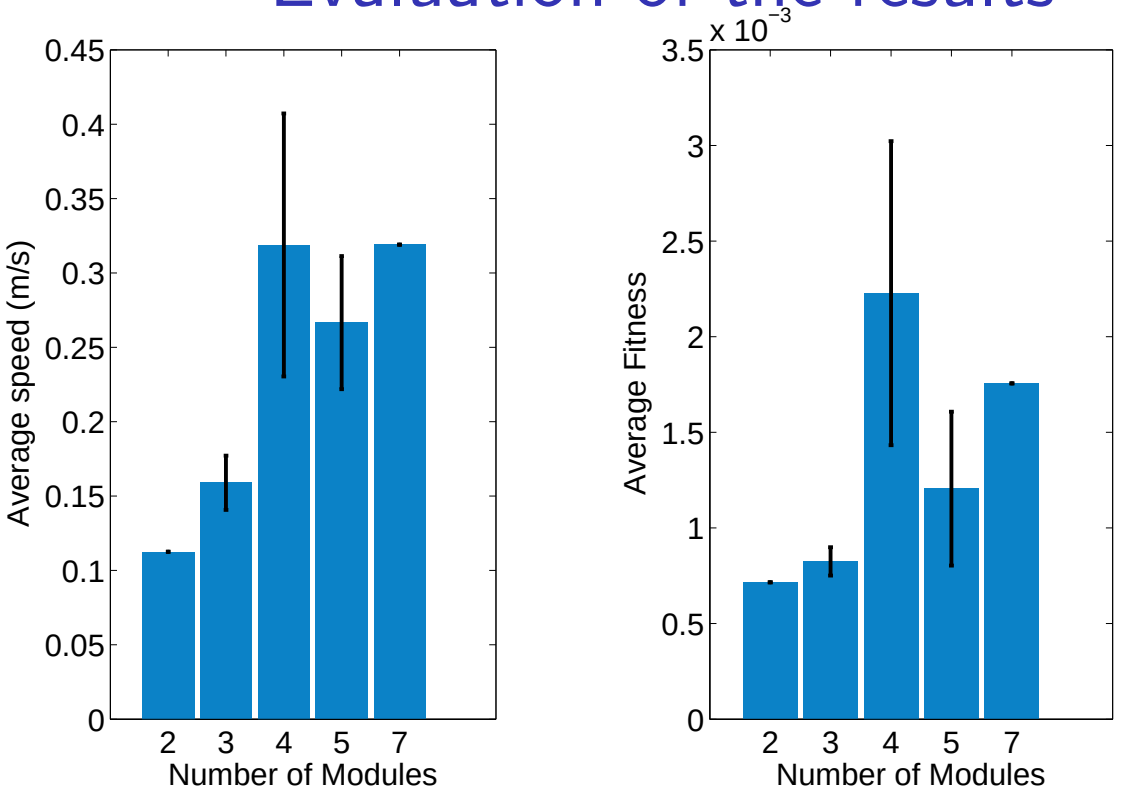


Figure: These results are grouped by the number of the modules of the robot.