Genetic Algorithm (Part 3)

Alexandre Bergel
DCC - University of Chile
http://bergel.eu
09/11/2020

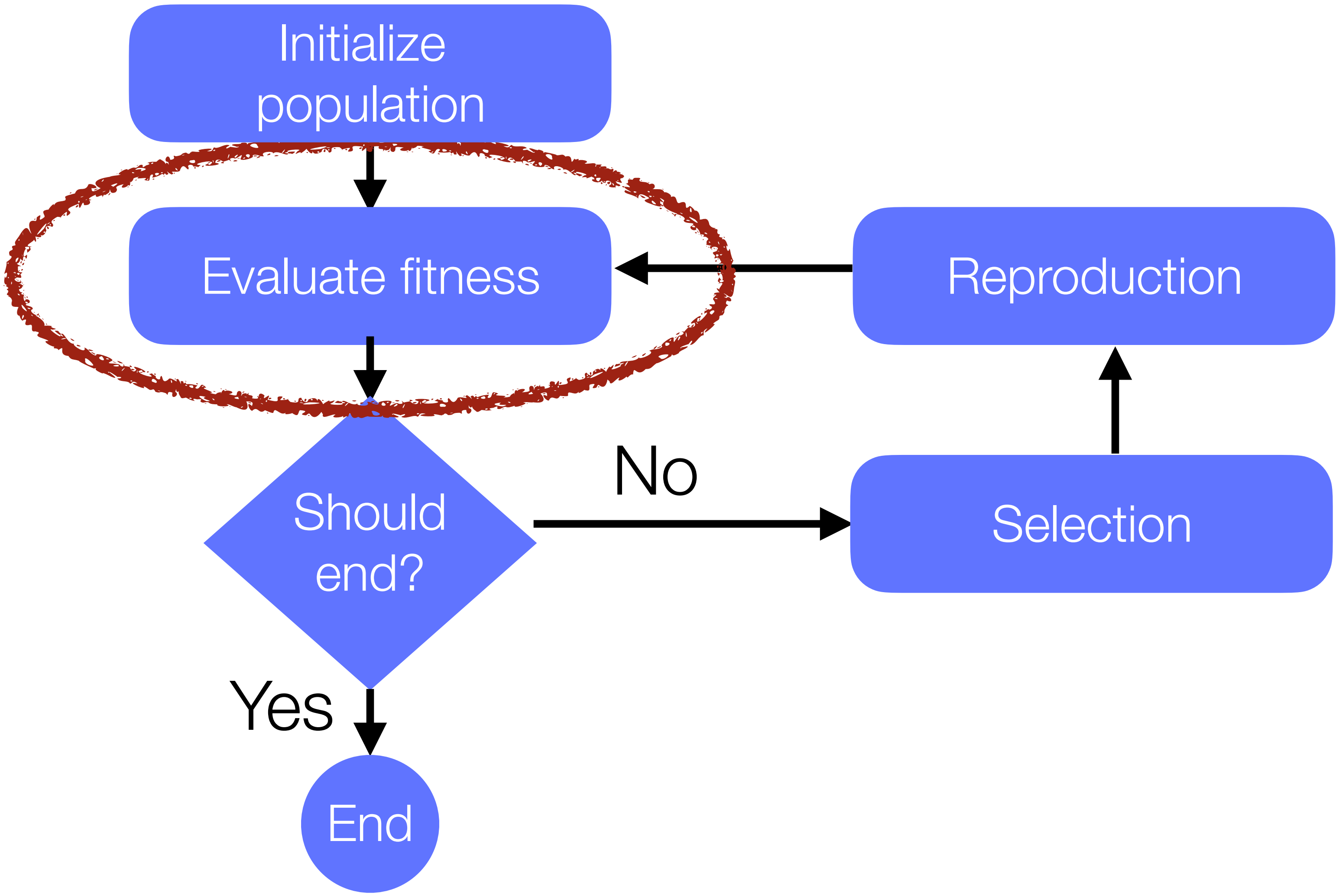
Goal of today

Cover advanced topics: *multi-objective optimization*, *Pareto front*, *elitism*

Provide complex examples involving *zoomorphic organisms*

zoomorphic | ˌzōəˈmôrfik |
adjective
having or representing animal forms or gods of animal form: *pottery decorated with anthropomorphic and zoomorphic designs*.

Flow - chart of a genetic algorithm



Fitness function

GA searches for a solution, *without specifying an analytic way* of doing so

This is very appealing since we just need to say how to evaluate a possible solution instead of saying how to get it

However, a fitness function may be complex

It happens that often some constraints have to be specified:

- E.g., a robot needs to find the exit *and* find the shortest path
- E.g., numbers *cannot be* repeated when solving the knapsack problem

Minimization vs Maximization

For some problems, getting closer to the solution means decreasing a function. E.g.,

- Distance between a robot to the exit
- Error function

In such a case, the fitness function needs to be expressed as:

- \- objective
- N - objective
- 1 / (1 + objective)

Multi-objective optimization

How to enforce that the robot follows the shortest path?

Two components:

- distance from the exit

- length of the path

Multi-objective optimization

Constraints may be expressed in different ways:

Penalty for violation in the fitness function

Implicit via the *encoding* of the problem

Multi-objective fitness function

Implement a *repair capability* in case of an infeasible / unsound individual

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Multi-objective optimization

A fitness function F can have sub-fitness functions $f_1, f_2, \ldots, f_n$

Do the f_i have the same priority?

Multi-objective optimization

If all the f_i have the very same priority, then we could have

$$F(x) = min(f_1(x), f_2(x), \ldots, f_n(x))$$

Multi-objective optimization

If all the f_i have the very same priority, then we could have

$$F(x) = min(f_1(x), f_2(x), \ldots, f_n(x))$$

Note that we have $F(x) \in \mathbb{R}$

$F(x)$ is a number, therefore comparison is trivial

Multi-objective optimization

Alternative definition of the fitness function exit to cope with multi-objective. The Pareto optimality is a definition

If we have n fitness functions, noted $f_1, \ldots, f_n$

Then we can have $F(x) \in \mathbb{R}^n$, in that case

$$F(x_1) > F(x_2) \iff \forall i \,.\, f_i(x_1) \geq f_i(x_2) \wedge \exists i \,.\, f_i(x_1) > f_i(x_2)$$

Multi-objective optimization

Considering:

$$F(x_1) > F(x_2) \iff \forall i \,.\, f_i(x_1) \geq f_i(x_2) \wedge \exists i \,.\, f_i(x_1) > f_i(x_2)$$

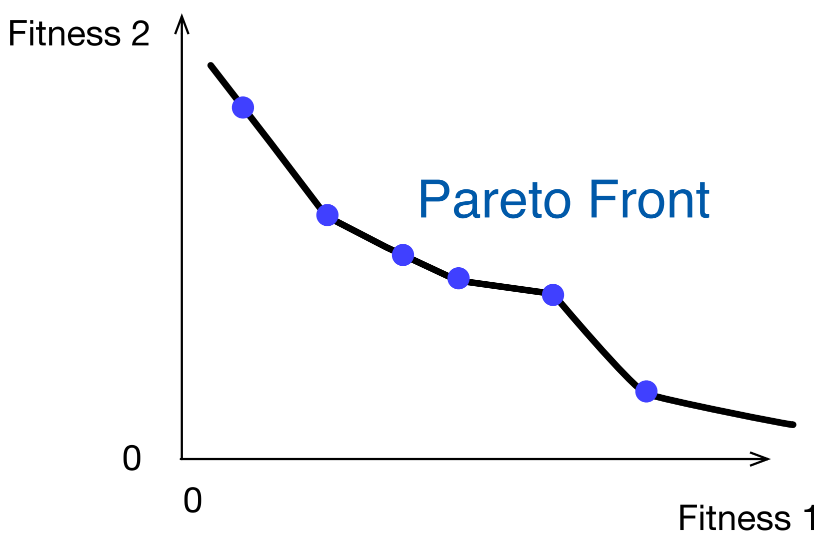
$\forall$ indicates *for all*

$\exists$ indicates *exists*

$\wedge$ indicates *logical and*

Pareto front

Searching for solutions using Pareto optimality may produce a set of solutions that are non-dominated



Multi-objective optimization

The robot has two objectives:

- Finding the exit
- Having a short path

These two objectives have not the same priority

Finding the exit is more important

Multi-objective optimization

Our robot can have the fitness $F(x) \in \mathbb{R}^2$

x represents a path

$F(x) = (d, l)$ in which d is the distance from the exit and l is the number of steps to reach the exit

We therefore wants to minimize both d and l, but we minimize l only if $d = 0$

Comparison is done as follow

If $d_1 < d_2$ then $F(x_1) > F(x_2)$

If $d_1 = d_2 \wedge l_1 < l_2$ then $F(x_1) > F(x_2)$

Else $F(x_1) < F(x_2)$

DCC
CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Bad luck?

An *exceptionally good individual* may be created in a particular generation

However, the characteristics of that individual may be lost if

- it is not selected to be parent
- its children are worse than the parent

Elitism

One common practice is to place in a new generation the best individual from the previous generation

The fitness curve never goes down

Each generation represents a progress if any

Pharo Tools Roassal | System Debugging Windows Help

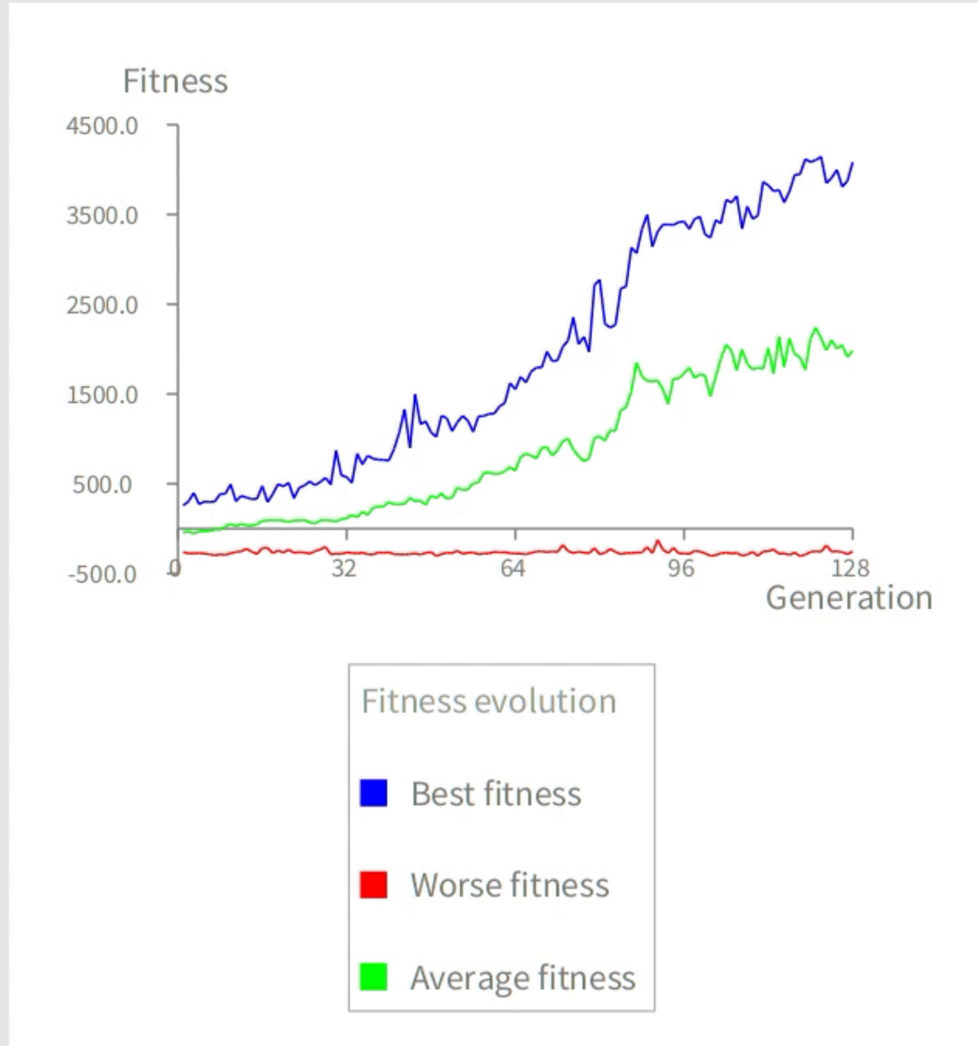
Playground

Page

```
   creature resetPosition.
   c := CWorld new.
   c addPlatform: (CPlatform new height: 20; width: 80; translateTo: 100 @
-10).
   c addPlatform: (CPlatform new height: 20; width: 80; translateTo: 400 @
-10).
   c addPlatform: (CPlatform new height: 20; width: 80; translateTo: 700 @
-10).
   c addPlatform: (CPlatform new height: 20; width: 80; translateTo: 1000 @
-10).
   c addCreature: creature.
   3000 timesRepeat: [ c beat ].
   creature position x
] value: (g logs at: 128) fittestIndividual genes.

c := CWorld new.
creature := CCreature new color: Color red; configureBall: 10.
creature materialize: g logs last fittestIndividual genes.
c addCreature: creature.

creature := CCreature new color: Color yellow darker darker; configureBall: 10.
creature materialize: (g logs at: 50) fittestIndividual genes.
c addCreature: creature.

creature := CCreature new color: Color blue darker darker; configureBall: 10.
creature materialize: (g logs at: 100) fittestIndividual genes.
c addCreature: creature.

creature := CCreature new color: Color green darker darker; configureBall: 10.
creature materialize: (g logs at: 90) fittestIndividual genes.
c addCreature: creature.

c addPlatform: (CPlatform new height: 20; width: 80; translateTo: 100 @ -10).
c addPlatform: (CPlatform new height: 20; width: 80; translateTo: 400 @ -10).
c addPlatform: (CPlatform new height: 20; width: 80; translateTo: 700 @ -10).
c addPlatform: (CPlatform new height: 20; width: 80; translateTo: 1000 @ -10).
c open
```

a GAEngine

View Logs Raw Meta

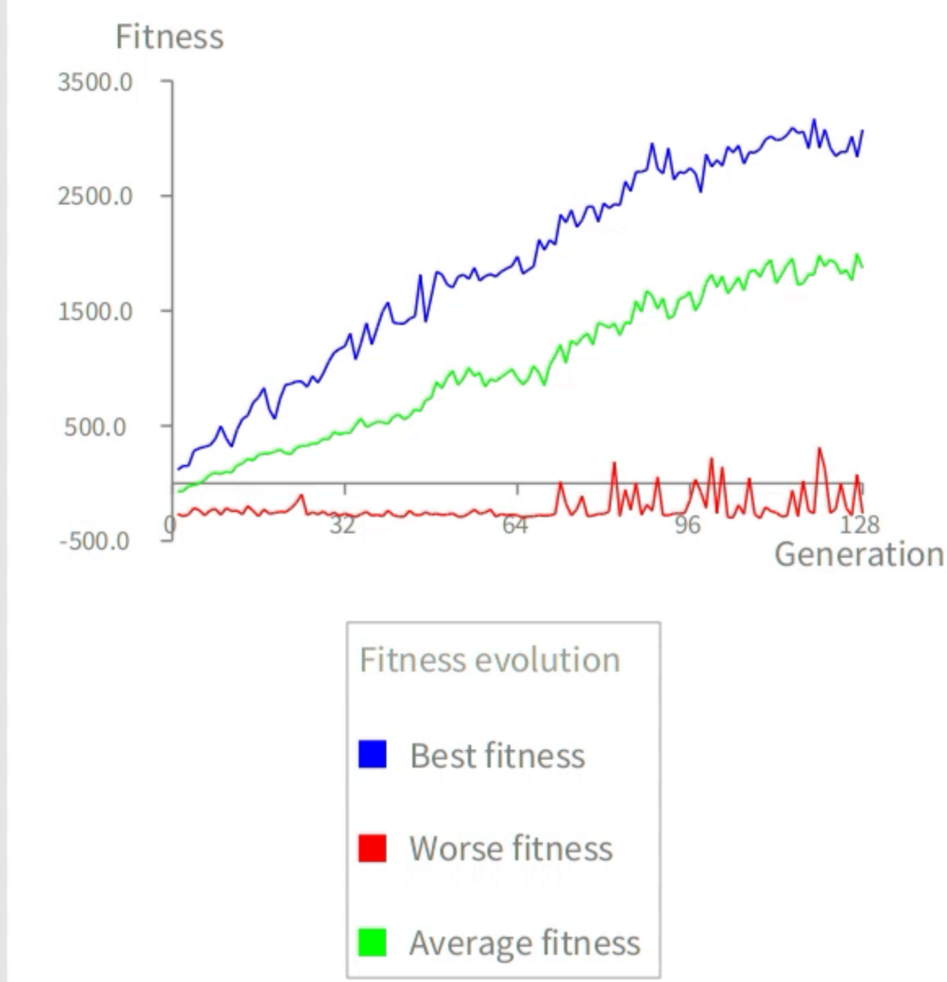


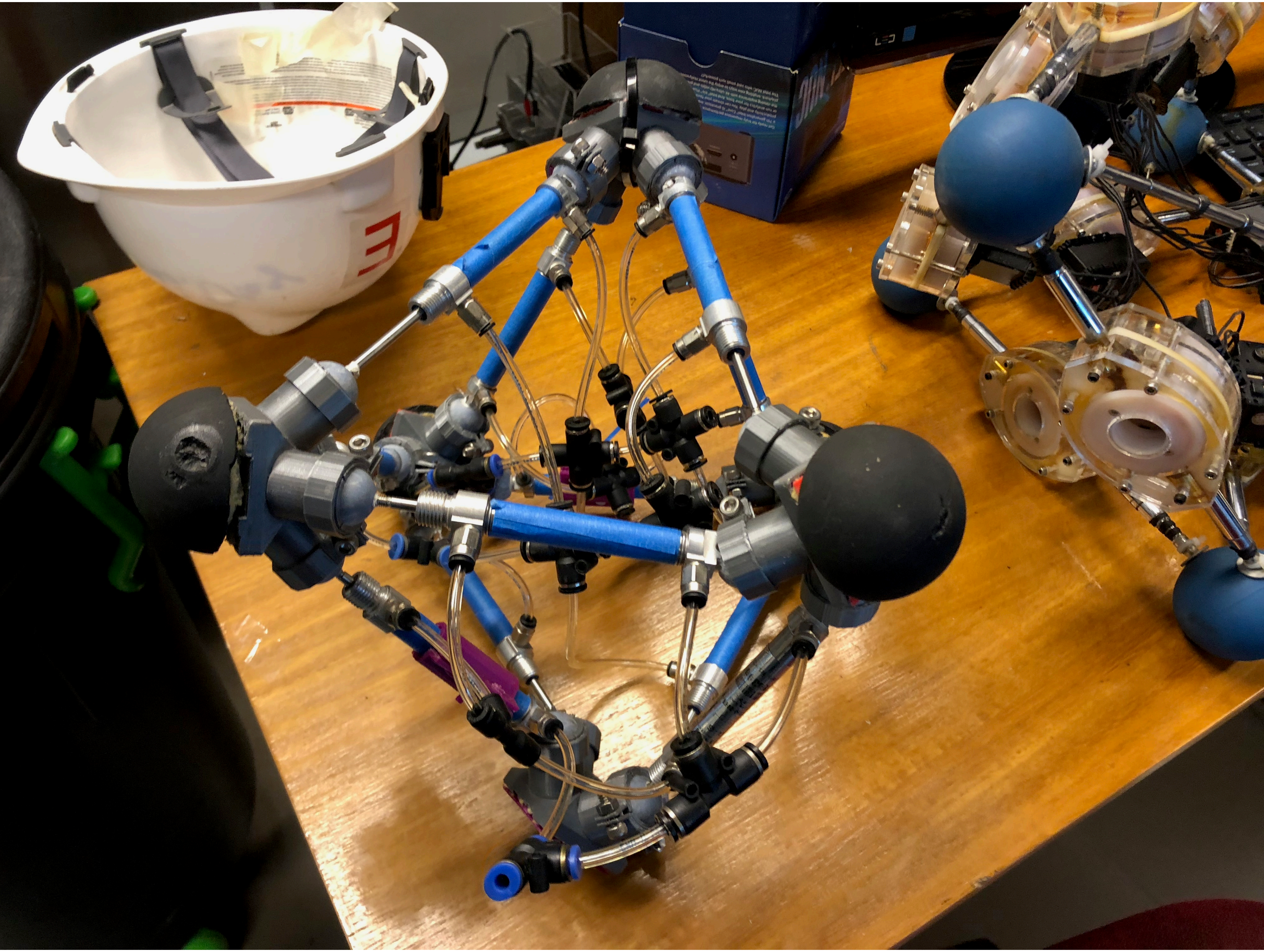
Playground

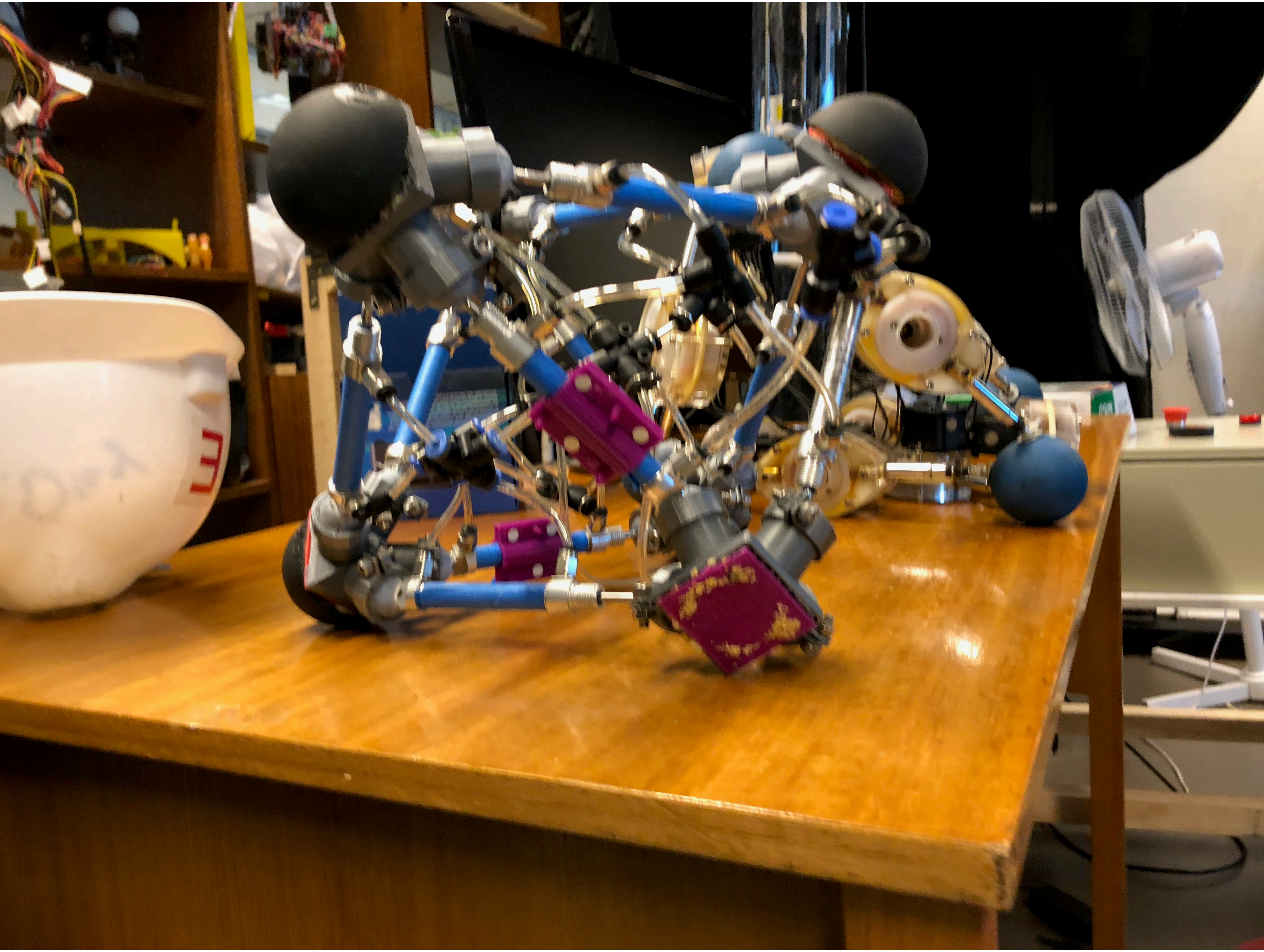
Page | a GAEngine

View | Logs | Raw | Meta

```
g selection: (GATournamentSelection new).
g mutationRate: 0.02.
g endForMaxNumberOfGeneration: 128.
g populationSize: 100.
g numberOfGenes: numberOfMuscles * 5.
g createGeneBlock: [ :r :index | mg valueForIndex: index ].
g fitnessBlock: [ :genes |
   creature := CCreature new configureBall: numberOfNodes.
   creature materialize: genes.
   creature resetPosition.
   c := CWorld new.
   c addCreature: creature.
   1 to: 25 by: 3 do: [ :x |
      c addPlatform: (CPlatform new height: 20; width: 80; translateTo: x *
100 @ -10).
      c addPlatform: (CPlatform new height: 20; width: 80; translateTo: x *
100 + 50 @ -30).
      c addPlatform: (CPlatform new height: 20; width: 80; translateTo: x *
100 + 100 @ -50).
      c addPlatform: (CPlatform new height: 20; width: 80; translateTo: x *
100 + 150 @ -70).
   ].
   c addCreature: creature.
   3000 timesRepeat: [ c beat ].
   creature position x
].
g run.

creature := CCreature new configureBall: 10.
creature materialize: g result.
c := CWorld new.
1 to: 25 by: 3 do: [ :x |
   c addPlatform: (CPlatform new height: 20; width: 80; translateTo: x * 100
@ -10).
   c addPlatform: (CPlatform new height: 20; width: 80; translateTo: x * 100
+ 50 @ -30).
   c addPlatform: (CPlatform new height: 20; width: 80; translateTo: x * 100
+ 100 @ -50).
   c addPlatform: (CPlatform new height: 20; width: 80; translateTo: x * 100
+ 150 @ -70).
].
c addCreature: creature.
c open
```





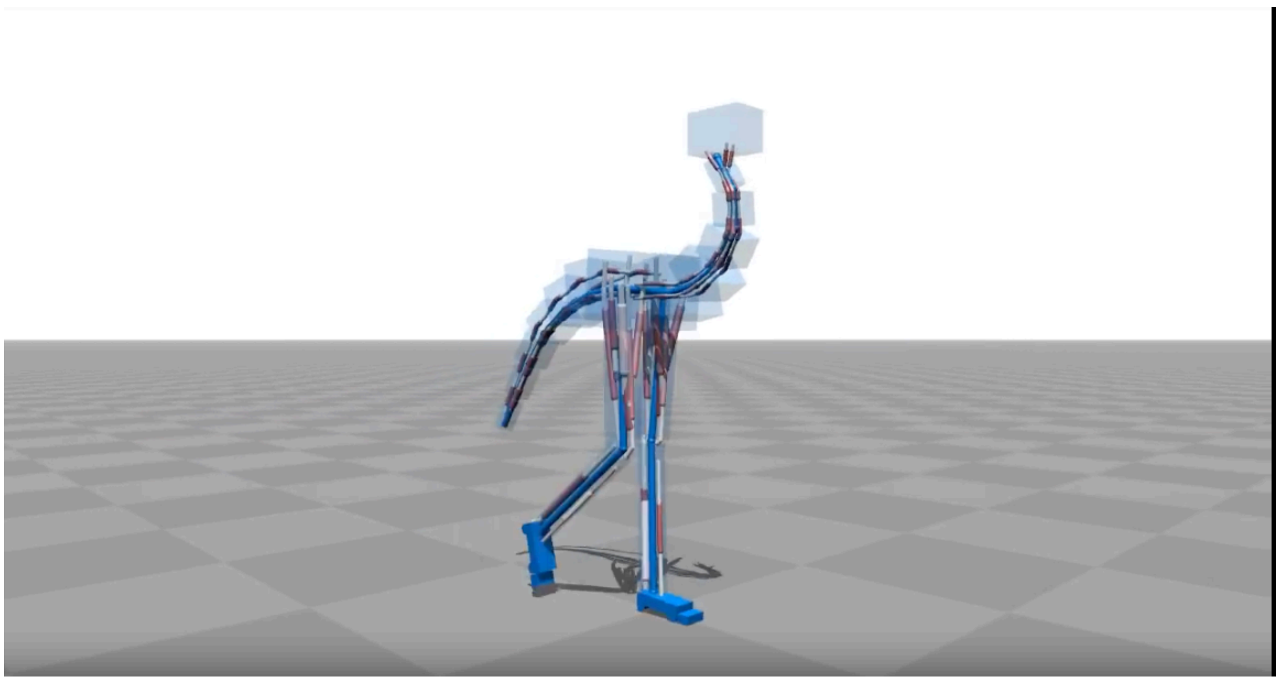


Evolving Virtual Creatures With Genetic Algorithms

https://www.youtube.com/watch?v=bBt0imn77Zg

Flexible Muscle-Based Locomotion for Bipedal Creatures

https://www.youtube.com/watch?v=pgaEE27nsQw



dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

/ DCCUCHILE