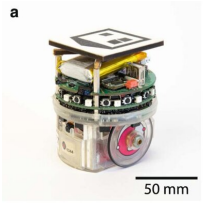
Neuro-evolutionary robotics: A gap between simulation and reality

16 July 2021



b

Input	Value	Description
$\text{prox}_{i\in\{1,\ldots,8\}}$	$[0,1]$	reading of proximity sensor i
$\text{light}_{i\in\{1,\ldots,8\}}$	$[0,1]$	reading of light sensor i
$\text{gnd}_{j\in\{1,2,3\}}$	{black, gray, white}	reading of ground sensor j
n	$[0,19]$	number of neighbouring peers perceived
V	$([0.5,20],[0,2\pi]\,\text{rad})$	range-and-bearing vector

Output	Value	Description
$v_{k\in\{l,r\}}$	$[-0.12,0.12]\,\text{m s}^{-1}$	target linear wheel velocity

Period of the control cycle: 100 ms

The robot and its reference model. a The e-puck robot in the configuration used for the experiments presented in the paper. Details are provided in “Methods”. b The reference model RM 1.1, which formally describes the programming interface through which, in the experiments presented in the paper, the control software interacts with the underlying hardware. The range-and-bearing vector points to the aggregate position of the neighboring peers perceived; its magnitude increases with the number of neighboring peers perceived and decreases with their distance. Formally, V=?nm=1(11+rm,?bm)V=?m=1n?(11+rm,?bm), where rm and ?bm are range and bearing of neighbor m, respectively. If no neighboring peer is perceived, the vector points in front of the robot and has unitary magnitude; formally, V = (1, ?0). Credit: DOI: 10.1038/s41467-021-24642-3

Neuro-evolutionary robotics is an attractive approach to realize collective behaviors for swarms of robots. Despite the large number of studies that have been devoted to it and although many methods and ideas have been proposed, empirical evaluations and comparative analyses are rare.

A publication in the journal *Nature Communications*, led by Mauro Birattari and his team at the research center IRIDIA, École Polytechnique de Bruxelles, Université Libre de Bruxelles, compares some of the most popular and advanced neuro-evolutionary methods for offline design of robot swarms.

"Concretely, these methods can enable the development of humanoid robot behavior, but to my knowledge, neuro-evolutionary robotics is not yet routinely adopted in real-world applications," explains Mauro Birattari.

All of these processes use evolutionary algorithms to generate a neural network that controls the robots, i.e., a neural network that takes sensor readings as input and outputs actuator commands. These methods use computer simulations to generate a neural network appropriate for the specific mission that the robots must accomplish. Once the neural network is generated (in simulation), it is installed on the physical robots and tested.

When comparing the different methods, the researchers observed a kind of "overfitting": the design process becomes too specialized in the simulation environment, and the neural network produced fails to "generalize" to the real world. This is a reality gap, i.e. the difference between reality and the simulator used in the design process. Although the simulator is fairly accurate, differences are inevitable.

"For example, if robots need to move back and forth between two areas, one solution that the evolutionary process might find in simulation is to produce a neural network that makes the robot move along a circular path that touches both areas. This solution is very elegant and works very efficiently in simulation. When applied to robots, this solution would fail miserably: for example, if the real diameter of (one of) the robot's wheels differs slightly from the nominal value, the radius of the trajectory will be different... the trajectory will no longer pass through the two given zones as desired and as predicted by the simulation," illustrates Mauro Birattari.

Although counter-intuitive, the solution seems to be to reduce the "power" of the design method: adopt a method that can produce a limited range of behaviors. This clearly means that researchers will have to accept that they will get worse results in

simulation. This method will not perform as well in simulation as a "powerful" method because it will not be able to exploit all the characteristics of the simulator, yet the result will be more general, less specialized to the simulator and therefore more likely to generalize well to reality. The simpler the better.

The chocolate method seems a good illustration of this idea. Chocolate is a process that researchers at the IRIDIA Center proposed a few years ago and that does not belong to neuro-evolutionary robotics but that, in a similar way to neuro-evolution, automatically generates control software for robots, under the same conditions. Chocolat operates on pre-existing software modules that are low-level behaviors (e.g., I go in the direction of the light, I stop, I move away from perceived peers...) and conditions for moving from one low-level behavior to another (e.g., I am surrounded by peers, the color of the floor I am on is black...).

Instead of playing with a very powerful neural network capable of producing a wide range of behaviors, chocolate plays with predefined building blocks that are (comparatively) much more "coarse." The working hypothesis is that by doing so, the risks of "over-fitting" will be reduced.

More information: Ken Hasselmann et al, Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms, *Nature Communications* (2021). DOI: 10.1038/s41467-021-24642-3

Provided by Université libre de Bruxelles