XLIV Jornadas de Automática 2023

https://jautomatica.es

Evolution of robot controllers for solving multiple tasks sequentially

Sendra-Arranz, R.[a,*], Gutiérrez, Á.[a]

[a]*E.T.S. Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain*

To cite this article: Sendra-Arranz, R., Gutiérrez, Á. 2023. Evolution of robot controllers for solving multiple tasks sequentially. XLIV Jornadas de Automática, 738-743. https://doi.org/10.17979/spudc.9788497498609.738

Resumen

El cambio de tarea en robótica evolutiva es un problema en cual los robots deben aprender a resolver múltiples tareas independientes y a cambiar entre ellas en el momento adecuado. Cuando los robots son controlados por redes neuronales artificiales, el cambio de tarea aumenta notablemente en complejidad debido a que el mismo modelo neuronal debe codificar multiples comportamientos. En este artículo se propone un experimento en el que robots controlados por redes neuronales recurrentes en tiempo continuo deben resolver un problema de cambio de tarea. Los parámetros y topología del controlador neuronal son evolucionados mediante una combinación del algoritmo evolutivo NeuroEvolution of Augmenting Topologies (NEAT) y aprendizaje Hebbiano. En el experimento propuesto, el grupo de robots debe resolver secuencialmente las tareas de (i) seguir lo más cerca posible una fuente de luz móvil y (ii) transportar pequeños objetos y depositarlos en una zona de almacenamiento común. El orden de ejecución de las tareas es aleatorio. Los resultados muestran que los robots son capaces de resolver correctamente las tareas propuestas y de cambiar entre ellas en el instante de tiempo adecuado.

Palabras clave: Algoritmos evolutivos, Robótica inteligente, Redes neuronales, Cambio de tarea, Sistemas multi-agente, Aprendizaje Hebbiano

Evolution of robot controllers for solving multiple tasks sequentially

Abstract

Task switching in evolutionary robotics is a problem in which robots have to learn to solve multiple independent tasks and switch among them at the correct timing. When the robots are controlled by means of an evolved Artificial Neural Network (ANN), the task switching is utterly complexified as the same neural model must encode multiple behaviors. In this paper, we addressed a task switching experiment using a group of homogeneous robots that are controlled by Continuous-Time Recurrent Neural Networks (CTRNN). The CTRNN parameters and topology are evolved using a combination of the NeuroEvolution of Augmenting Topologies (NEAT) algorithm and Hebbian learning rules. In the proposed experiment, the group of robots is evolved with the aim of solving sequentially the tasks of (i) following as closely as possible a mobile light source and (ii) transporting small objects to a common nest area. The order of the tasks is determined randomly. The results showed that the robots can successfully solve the proposed tasks and switch between them at the correct timing.

Keywords: Evolutionary algorithms, Intelligent robotics, Neural networks, Task switching, Multi-agent systems, Hebbian learning

1. Introduction

In Evolutionary Robotics (ER) (Nolfi and Floreano, 2000), the behavior of robots, typically controlled by an Artificial Neural Network (ANN), is determined by an evolutionary algorithm. The artificial evolution optimizes the parameters of the neural controller of the robot with the aim of solving a given problem or task. A highly complex and interesting problem to be addressed using ER is the solving of multiple independent tasks sequentially. This problem, usually referred as task switching or switch of labor, is specially challenging when neu-

*Autor para correspondencia: r.sendra@upm.es

ral controllers are fine-tuned using an evolutionary algorithm. In these cases, the evolved ANN must encode in its parameters not only multiple independent behaviors corresponding to the different tasks, but also the ability to switch its neural regimes to address the requested task.

Several authors have addressed the problem of task switching from the perspective of evolutionary computation and ANNs. In (Capi, 2007), the author evolves a multilayer perceptron to learn to correctly switch among three different tasks. The proposed tasks are to (i) closely follow a target robot, (ii) to collect objects and (iii) to explore the environment. The evolution is accomplished using multiobjective evolutionary computation, so that the sought outcomes of the evolution are no longer point solutions but a curve of optimal solutions. This curve is called Pareto front and its elements provide the optimum balance in maximizing the fitness of all the tasks simultaneously. A similar approach is considered in (Capi et al., 2008), where the tasks to be commuted by the robot are approaching to a sound source and reaching several lights distributed along the environment. The authors of (D'Ambrosio et al., 2011) study the task switching problem in a multi-agent patrol and return task. The agents are evolved using an extension of multi-agent HyperNEAT, so that they can learn multiple behaviors and switch among them depending on the experiment requirements. Tuci et al. address in (Tuci et al., 2013) a task switching problem in a foraging experimental setup. The members of a team of 5 robots have to learn to distribute roles of patrolling the nest area and exploring and seeking food outside the nest, and perform the task switching at the precise timing. They show that CTRNN controllers evolved using an EA can successfully accomplish the experiment. In (Garattoni and Birattari, 2018), a new task sequencing strategy called TS-Swarm is proposed in the field of swarm robotics. TS-Swarm is devoted to the online discovery and execution of tasks in the correct order, which is initially unknown. The swarm system combines both purely reactive and deliberative behaviors.

In this paper, we evolved the neural controllers of a group of robots aiming at solving two different and independent tasks sequentially. The robots not only have to learn to solve both tasks but also to switch between them at the correct timing (task switching). In order to address this problem, we used Continuous-Time Recurrent Neural Networks (CTRNN) as the robot controller. A combination of evolution and learning is used in order to find the best performing CTRNN parameters for the task switching problem. Artificial evolution is accomplished by means of the NeuroEvolution of Augmenting Topologies (NEAT) algorithm (Stanley and Miikkulainen, 2002). NEAT not only evolves the parameters of the ANN but also searches the most suitable network topology. This feature is specially important in task switching and multi-tasking because the same neural architecture must simultaneously encode multiple behaviors for the requested tasks. Additionally, Hebbian learning is also used to allow the lifetime learning of agents, specifically their adaptation to the task switching requirements. In order to provide statistically significant results, we collected a sample of 50 independent simulations of the best performing individual of the NEAT population. In the light of the results, we verified that it is possible to evolve neural controllers to learn to solve two utterly different tasks simultaneously and switch between them at the correct timing.

The document is structured as follows. Sec. 2 details the task switching experiment and its constituent tasks. It also defines the fitness function to guide artificial evolution towards a suitable solution and specifies the neural controller and the optimization setup. Sec. 3 exposes the results of the task switching problem. Finally, Sec. 4 concludes the paper.

2. Methodology

2.1. The Robots and the Tasks

A group of robots is placed within a simulated 2 m × 2 m squared arena environment. The robots are simulated as minimal two-wheeled mobile robots with a set of simple sensors and actuators. The robots have cylindrical shape with about 55mm of height and a base radius of 70mm (similar to the e-puck robot (Mondada et al., 2009)). Hereafter, we denote the set of robots or agents as $\mathcal{R}$, so that each robot $r \in \mathcal{R}$ has a position $\mathbf{x}_r$ and a orientation θ_r within the arena. The arena also contains light sources $\ell \in \mathcal{L}$ of different colors and position $\mathbf{x}_\ell$, that will be used to formulate the experiment. The robots can sense the intensity of the light sources and the color through a light sensor (LS) unit. Additionally, they are equipped with a distance sensor (DS) that allows the detection of near obstacles (robots, cubes or arena walls) through a set of IR emitters and detectors positioned along the robot perimeter. The range of the light sensor is large enough to perceive the light sources from any coordinates in the arena, while the range of the distance sensor is about 80 cm. Both DS and LS are sectorized sensors, meaning that the robot can know the intensity of the sensed signal from different orientations. In this paper, we use sensors with 4 sectors positioned at θ_r, $\theta_r + \pi/2$, $\theta_r + \pi$ and $\theta_r + 3\pi/2$.

In this paper, we propose a task switching experiment in which the robots of the group will have to solve two different and independent tasks sequentially. The simulations are split into two time windows of the same length, so that all the agents in the group have to address the same task requested in the current time window. Tasks are solved sequentially and, desirably, all agents have to solve only the corresponding task at each time instant. Furthermore, the order of the tasks is randomly selected in each simulation, so that the robots do not have a priori knowledge about the correct task to be faced.

The specific tasks that compose the task switching experiment are *Task A* and *Task B*:

- *Task A*: the robots have to follow as closely as possible a mobile red light source. The light source emits red light omnidirectionally so that it can be sensed by the robots' red light sensor. The trajectory of the light source is composed of two phases. Firstly, the initial position of the red light source at the beginning of the simulation is always the center of the arena. Thereafter, the light source describes a spiral trajectory so that its distance to the origin of coordinates is exponentially increased. Once a distance of 1.5m from the center of the arena is reached, the red light movement is settled to a simple circle orbit around the central coordinates (0, 0).

- *Task B*: is a transportation task in which robots have to collect small blue cubes and transport them to a common

nest area. The nest area is a circular grey ground area with a yellow light source above. The light source has a coverage range large enough so that robots can sense it from any point in the arena. This light acts as a beacon, notifying the agents the location of the nest. Both the nest ground area and the yellow light source are static. Three different grey ground areas are placed in the arena at fixed positions, albeit there is only one yellow light source randomly situated above one of them. Consequently, at each simulation trial, the correct nest area where cubes have to be transported, is the ground area underneath the yellow light source. The blue cubes can be perceived by the robots by means of a binary reading from a color sensor that detects whether the amount of blue color exceeds a fixed threshold.

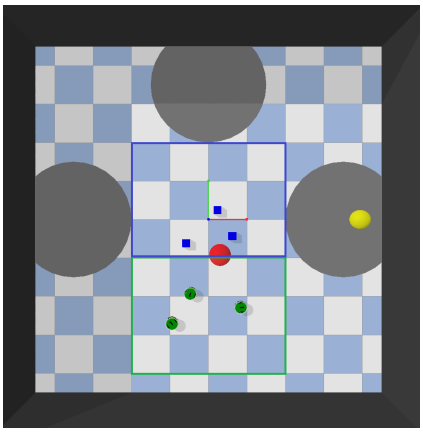


Figure 1: First frame of the Experiment 1 arena with a zenithal view.

Fig. 1 shows a zenithal view of the first time instant of the simulation. In this snapshot it can be observed three mobile robots in green, three blue cubes, three grey ground areas, and the yellow and red light sources. The green and blue rectangles respectively sketch the area where robots and blue cubes are randomly initialized.

2.2. The Agent's Controller

We use an Artificial Neural Network (ANN) as the mathematical model to control the behavior of the agents. Specifically, the ANN model is the Continuous-Time Recurrent Neural Network (CTRNN) (Beer and Gallagher, 1992). We define the neuron dynamics as in Eqs. 1 and 2.

$$\left.\begin{aligned} \tau_k \frac{\partial v_k(t)}{\partial t} &= -v_k(t) + I_k(t) \\ u_k(t) &= \sigma\left(g_k \cdot (v_k(t) + \beta_k)\right) \end{aligned}\right\} \quad (1)$$

Eq. 1 depicts the single neuron's voltage ($v_k(t)$) and activation ($u_k(t)$) dynamics. β_k, g_k and $\sigma(\cdot)$ are the neuron's bias, gain and sigmoid activation, respectively. In addition, τ_m is the neuron´s time constant. $I_k(t)$ is the total current fed to the neuron's soma which is calculated as in Eq. 2,

$$I_k(t) = \sum_{i \in \mathcal{N}_k} w_{ki} u_i(t) + \sum_{j \in \mathcal{N}_k^{\phi}} w_{kj}^{\phi} \phi_j(t) \quad (2)$$

where w_{ki} is the weight of the synapse connecting pre-synaptic neuron i with post-synaptic neuron k and w_{kj}^{ϕ} denotes the weight of the synapse between the j-th input and neuron k. $\phi_j(t)$ is the j-th input signal being fed to the CTRNN and $\mathcal{N}_k$ and $\mathcal{N}_k^{\phi}$ are the sets respectively comprising the pre-synaptic neurons and pre-synaptic inputs to neuron k.

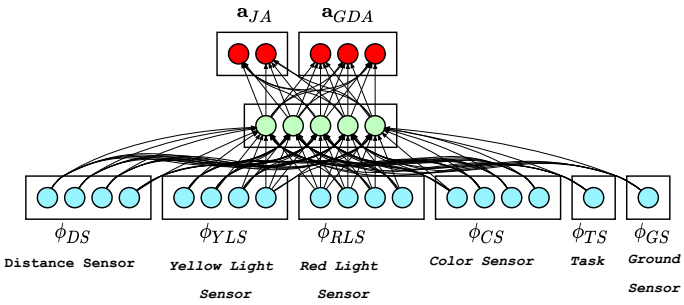


Figure 2: Initial CTRNN architecture of the task switching experiment. The blue circles are the input nodes, the green circles represent the hidden neurons and the red dots are the motor neurons.

Fig. 2 shows the CTRNN architecture that determines the robot's behavior at the beginning of evolution. It is composed by a single hidden layer with 5 neurons (in green), a set of input layers that receive the readings from the robot sensors (in blue) and two output layers that define the robot actions (in red). The input layers correspond to the readings from the IR distance sensor (ϕ_{DS}), the red light sensor (ϕ_{RLS}), the yellow light sensor (ϕ_{YLS}), the color sensor to detect the blue cubes (ϕ_{CS}), the ground sensor to detect ground areas underneath the robot (ϕ_{GS}), and a binary input that encodes the task to be executed (ϕ_{TS}). On the contrary, the output layers are respectively devoted to the angular velocity control of the motors of the two wheels of the robots ($\mathbf{a}_{JA}$), and to handle the transportation of cubes ($\mathbf{a}_{GDA}$). The action $\mathbf{a}_{GDA}$ (where *GDA* stands for Grab and Drop Actuator) has three categorical actions for either grabbing a nearby cube, dropping an already grabbed cube or doing nothing, each corresponding to one of the three neurons in the output layer. The categorical action is obtained using the softmax activation function.

2.3. Optimization Techniques

The evolutionary algorithm used in the proposed experiments to optimize the topology and parameters of the neural controller is the NeuroEvolution of Augmenting Topologies (NEAT) algorithm (Stanley and Miikkulainen, 2002). NEAT not only evolves the parameters of the neurons and synapses, but also the topology of the network itself. It starts the evolution process with a CTRNN topology of minimal complexity (see Fig. 2) and adds new neurons and synapses as evolution marches. In addition to the evolutionary algorithm, the synapses of the neural networks are also subject to an online learning process during the evaluation of the NEAT individuals. This learning process is accomplished through Hebbian learning (Hebb, 1949), so that the neural models can dynamically adapt their parameters and learn from experience at runtime by interacting with the environment. We specifically apply the generalized ABCD Hebbian learning rule (see for in-

stance (Risi et al., 2010; Najarro and Risi, 2020)). The learning rule of a single synapse is specified in Eq. 3,

$$\frac{\partial w_{ij}(t)}{\partial t} = \eta \left(a_{ij}\, u_i(t)\, u_j(t) + b_{ij}\, u_i(t) + c_{ij}\, u_j(t) + d_{ij} \right) \quad (3)$$

where the ABCD Hebbian rule applies a polynomial transformation of the pre-synaptic ($u_i(t)$) and post-synaptic ($u_j(t)$) neuron' activities. a_{ij} is a parameter that represents the importance of the correlation between neuron activities and b_{ij} and c_{ij} impose how u_i and u_j individually affect the learning. Additionally, d_{ij} is an offset term that defines the weight adaptation under no neuronal activity. Lastly, η is a learning rate that is common to all the synapses and defines the length of the learning steps.

It should be mentioned that the learned weights at the end of the simulations are not transmitted to the next generation. The learning experience has an indirect impact to the evolution process in form of the resulting fitness score achieved in the episode. The parameter search spaces considered by NEAT are $w_{ij} \in [-5, 5]$, $\beta_i \in [-2, 2]$, $g_i \in [0.05, 5]$ and $\tau_i \in [0.3, 32]$, for any neurons i and j. Besides, the parameters of the Hebbian learning rules are all constrained to the interval $[-2, 2]$.

Table 1 gathers the hyper-parameters used in the evolution process of NEAT. These values are mainly obtained through heuristic search, trial and error and also considering the experimental setup proposed by the authors of NEAT in their experiments (see (Stanley and Miikkulainen, 2002)). The population size is set to 300 and the fitness is evaluated 10 times. Provided that λ_i is the size of species i, the selection operator is the tournament selection with n_{sel} genotypes selected in each species as parents. Besides, the probability of node mutation is lower than the probability of adding new connections in order to avoid undesired fast topology growth. The constants c_1 and c_2 are fixed to the same value because the same importance is given to both excess and disjoint gene differences. The similarity threshold δ_t was probably the most challenging hyper-parameter to be adjusted. We found $\delta_t = 0.3$ to be a nice trade-off for the used CTRNNs. η is the learning rate of the Hebbian learning rules, which is the same for all the synapses.

Variable	Value	Description
λ	300	Population size.
N_E	10	Number of genotype evaluations.
n_{sel}	$0.3\,\lambda_i$	Num. of parents selected for crossover in species i.
p_{mw}	0.1	Probability to modify gene parameters.
p_{mc}	0.15	Probability to add a new connection.
p_{mn}	0.05	Probability to add a new neuron.
σ_w	0.1	Std. dev. of the gaussian parameter mutation.
c_1, c_2	1	Importance of the disjoint and excess terms for the genotype similarity.
c_3	0.6	Importance of the parameter's distance for the genotype similarity.
δ_t	0.3	Similarity threshold of speciation.
E	2	Number of elites per species
η	$5 \cdot 10^{-3}$	Learning rate of the Hebbian learning rules.

Table 1: Best found evolution hyper-parameters for the experiment.

2.4. Fitness Function

The overall fitness function of the task switching experiment is the combination of the fitness scores corresponding to the two proposed tasks. The combination of these fitness scores, shown in Eq. 4, is performed through a geometric mean.

$$F_{tot} = \sqrt{F_A\, F_B} \quad (4)$$

Thus, the total fitness score should be highest when the robots perform proficiently in both tasks. The values of F_A and F_B are the fitness scores corresponding to *Task A* and *Task B*, respectively.

Firstly, the light pursuit fitness function, considering a group of R robots, is defined in Eq. 5,

$$F_A = \frac{1}{(T_E/2)\,R} \sum_{t=0}^{T_E/2} \sum_{r \in \mathcal{R}} \max \left\{ 1 - \frac{\|\mathbf{x}_\ell(t) - \mathbf{x}_r(t))\|_2}{\rho_\ell(t)},\, 0 \right\} \quad (5)$$

where $\mathbf{x}_\ell$ and $\mathbf{x}_r$ are the positions of the red light source and the robots, respectively. The fitness score increases linearly as the distance from robot r to the red light is diminished. Nonetheless, for those robots whose distance to the red light is larger than a threshold ρ_ℓ, the fitness rise will be zero. With the aim of smoothing the abrupt discontinuity in the fitness function during the task switching period, the distance threshold ρ_ℓ is changed dynamically as the simulation time marches. Specifically, it starts with a value of 1.5 and is decreased linearly with time up to a steady value of 0.5, as established in Eq. 6.

$$\rho_\ell(t) = \max \left\{ 1.5 - \frac{t}{100},\, 0.5 \right\} \quad (6)$$

On the contrary, the fitness function to quantify the performance of the robots in the object transportation task is computed as follows. As mentioned in previous sections, the task of transporting cubes consists in carrying blue cubes distributed along the arena and gathering them into the correct ground area. There are three ground areas in total, and the one with a yellow light above is the nest where the robots have to transport the cubes. We denote this correct ground area as g, its center coordinates as $\mathbf{x}_g$ and its radius as ρ_g. With this notation in mind, Eq. 7 computes the number of cubes correctly placed in the nest g at the end of the task at $t_{end} = T_E/2$,

$$n_c = \mathcal{H} \left(\rho_g - \|\mathbf{x}_c(t_{end}) - \mathbf{x}_g\|_2 \right) \quad (7)$$

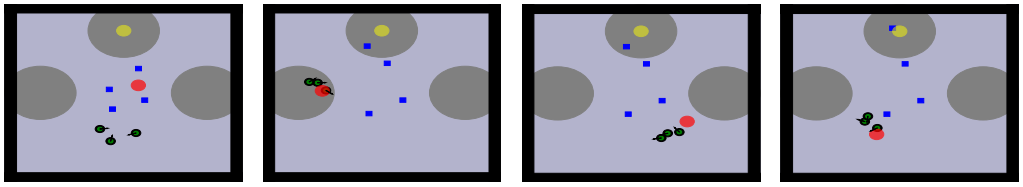
where $\mathbf{x}_c$ denotes the position of cube c, and $\mathcal{H}$ is the Heaviside function.

Thereafter, using the previously computed value of n_c (cubes correctly collected), the overall fitness function of the task is displayed in Eq. 8,

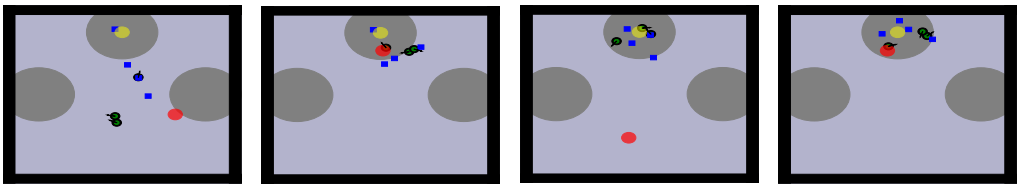
$$F_B = \max \left\{ \frac{1}{N_{cubes}^2} \sum_{c \in C} n_c + d_c,\, 0 \right\} \quad (8)$$

where C is the set of blue cubes in the arena and N_{cubes} denotes the total number of cubes. In addition, for every cube c, d_c rises the fitness score when the cubes are moved towards the nest (see Eq. 9).

$$d_c = \begin{cases} 1, & \text{if } \|\mathbf{x}_c(t_{end}) - \mathbf{x}_g\|_2 < \|\mathbf{x}_c(0) - \mathbf{x}_g\|_2 \\ 0, & \text{Otherwise} \end{cases} \quad (9)$$

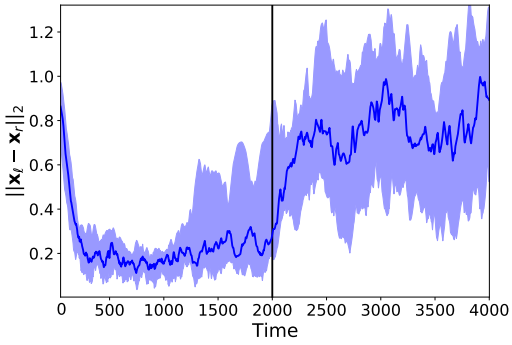


(a) $t = 1$ (b) $t = 500$ (c) $t = 1500$ (d) $t = 2000$

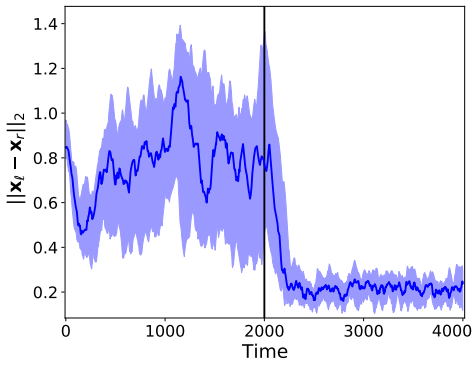


(e) $t = 2500$ (f) $t = 3000$ (g) $t = 3500$ (h) $t = 4000$

Figure 3: Frames of a simulation of the task switching experiment under the task order A, B. The switch of tasks is produced at time step 2000. The robots are represented using green balls with black contour and with a line denoting its heading orientation. The cubes are the blue squares, the red and yellow lights are painted as circles of the corresponding color and the large grey circles are the ground areas.



(a) Task Order: A ⟶ B



(b) Task Order: B ⟶ A

Figure 4: Temporal evolution of the Euclidean distance between robots and the red light source under different orderings of tasks. In (a) Task A is presented first and in (b) Task B is the starting task. In both cases the switch of task is produced at time instant 2000, which is represented with a vertical line. The dark blue curves represent the median value of the distance using the robots positions of 50 independent simulations. The contours of the blue shadows illustrate the first (lower shadow) and third (upper shadow) quartiles.

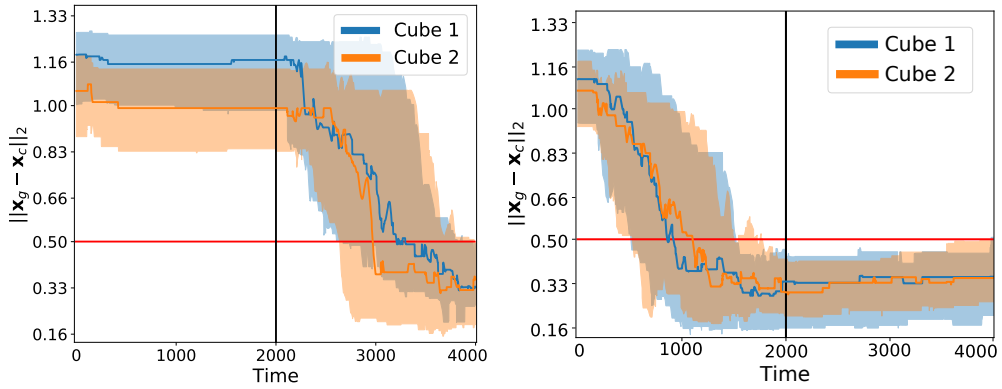
Notice that if the cube has been displaced in the direction opposite to the ground area g, then $d_c = 0$. The motivation behind including the summation of distances $\sum_{c \in C} d_c$ is to smooth the fitness function and prevent deceptive solutions due to premature convergence.

3. Results

Fig. 3 collects different frames of a simulation of the task switching experiment using the best evolved genotype. In the shown simulation, the robots should start solving Task A (red light pursuit) and conclude with Task B (cube transportation). The task switching is requested at simulation time instant 2000. In Figs. 3a-d, the agents successfully follow the red light source. In contrast, from Fig. 3e until the ending of the simulation, the robots ignore the red light source and aim at transporting the blue cubes to the ground area in the north. The last frame illustrates that all the cubes have been suitably collected in the correct ground area.

Focusing now on the behavior of the robots when Task A is addressed, Fig. 4 illustrates the distance between robots and the targeted red light source, as simulation cycles are elapsed. In order to build the plots, 50 independent simulation episodes are collected. The dark blue curve represents the median value, among the 50 sample simulations, of the Euclidean distance between the robot and the red light. Additionally, the contours of the blue shadows depict the first and third quartiles. In Fig. 4a, the task order is A, B and in Fig. 4b the ordering is B, A. The switch of tasks is produced at $t = 2000$ and it is represented as a vertical line. In these two plots, it can be observed that the Euclidean distance drastically decreases when the requested task is A. Furthermore, it reaches a steady state of about 0.2 m of distance to the light, in median value. The distance variation when robots solve Task A is, generally, remarkably reduced. Regarding the time slots corresponding to Task B, the distance to the light is larger and has much more variability because robots ignore the red light in order to collect blue cubes. Both plots expose that, at $t = 2000$, when the task switching occurs, the robots almost instantaneously realize that the task to be solved has changed and they correctly modify their behavior.

In contrast, Fig. 5 depicts the behavior of the robots when Task B is requested. It shows a scenario with 2 cubes. For each cube, the figure displays the Euclidean distance between the cube's position and the center of the nest where the objects have to be stored. The correct nest is the one underneath the yellow light source acting as beacon. The orange and blue dark



(a) Task Order: A ⟶ B

(b) Task Order: B ⟶ A

Figure 5: Temporal evolution of the Euclidean distance between each instantiated cube and nest ground area, under different orderings of tasks. In (a) Task A is presented first and in (b) Task B is the starting task. In both cases the switch of task is produced at time instant 2000, which is represented with a vertical line. The horizontal red line indicates the radius of the ground area. The dark curves represent the median value of the distance using the cubes and nest positions of 50 independent simulations. The contours of the shadows illustrate the first (lower shadow) and third (upper shadow) quartiles.

curves indicate the temporal evolution of the median value of the previously mentioned distance using 50 independent simulation trials. The contours of the shadows highlight the temporal evolution of the first and third quartiles. Fig. 5a shows the results when the task order is A, B while Fig. 5b depicts the distance when Task B is firstly requested. The vertical black line marks the time instant when the task switching is produced and the horizontal red line indicates the radius of the nest ground area. Therefore, when the distance between cubes and the nest is below this threshold, it means that the corresponding cube has been properly stored inside the correct nest. It can be appreciated that, at the last cycle of the corresponding time slot (4000 in Fig. 5a and 2000 in Fig. 5b), almost all the cubes are correctly placed inside of the nest. The rate at which robots displace the cubes is also remarkably steady along the 50 simulations. Furthermore, within the time window corresponding to Task A, the cubes are generally static because the robots are accomplishing the red light pursuit.

4. Conclusions

In this paper, we addressed a task switching problem in which robots have to solve multiple tasks sequentially and switch among them at the correct timing. The agents are controlled by CTRNN models that are optimized using a combination of artificial evolution and learning. The evolutionary algorithm used is the NeuroEvolution of Augmenting Topologies (NEAT) and the learning is produced during the simulation run time using Hebbian learning rules. The specific tasks that have to be solved by the robots sequentially are the pursuit of a light source and the transportation of small cubes to a common nest area. After the optimization process, a statistically significant sample of multiple independent simulations are collected and analysed. The results showed that the robots can successfully solve both of the requested tasks and correctly switch between them at the correct timing.

Acknowledgments

This work has been supported by Grant PID2020-112502RB-C41 funded by MCIN/AEI/10.13039/501100011033. R. Sendra-Arranz's acknowledges support from the predoctoral grant from the "Programa Propio I+D+i" financed by the Universidad Politécnica de Madrid. The authors gratefully acknowledge the Universidad Politécnica de Madrid for providing computing resources on Magerit Supercomputer.

References

Beer, R. D., Gallagher, J. C., 1992. Evolving dynamical neural networks for adaptive behavior. Adaptive Behavior 1 (1), 91–122.
DOI: 10.1177/105971239200100105

Capi, G., 2007. Robot task switching in complex environments. In: 2007 IEEE/ASME international conference on advanced intelligent mechatronics. pp. 1–6.
DOI: 10.1109/AIM.2007.4412489

Capi, G., Pojani, G., Kaneko, S.-I., 2008. Evolution of Task Switching Behaviors in Real Mobile Robots. In: 2008 3rd International Conference on Innovative Computing Information and Control. pp. 495–495.
DOI: 10.1109/ICICIC.2008.261

D'Ambrosio, D. B., Lehman, J., Risi, S., Stanley, K. O., 2011. Task switching in multirobot learning through indirect encoding. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2802–2809.
DOI: 10.1109/IROS.2011.6094509

Garattoni, L., Birattari, M., 2018. Autonomous task sequencing in a robot swarm. Science Robotics 3 (20), eaat0430.
DOI: 10.1126/scirobotics.aat0430

Hebb, D., 1949. The organization of behavior: A neuropsychological theory. New York: John Wiley, Science Editions.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., Martinoli, A., 2009. The e-puck, a robot designed for education in engineering. Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions 1, 59–65.

Najarro, E., Risi, S., 2020. Meta-learning through hebbian plasticity in random networks. In: Advances in Neural Information Processing Systems. Vol. 33. Curran Associates, Inc., pp. 20719–20731.

Nolfi, S., Floreano, D., 2000. Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. MIT press.

Risi, S., Hughes, C. E., Stanley, K. O., 2010. Evolving plastic neural networks with novelty search. Adaptive Behavior 18 (6), 470–491.
DOI: 10.1177/1059712310379923

Stanley, K. O., Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. Evolutionary Computation 10 (2), 99–127.
DOI: 10.1162/106365602320169811

Tuci, E., Mitavskiy, B., Francesca, G., 2013. On the evolution of self-organised role-allocation and role-switching behaviour in swarm robotics: a case study. MIT Press, pp. 379–386.
DOI: 10.1162/978-0-262-31709-2-ch055