Online adaptation of robots controlled by nanowire networks: A preliminary study

Paolo Baldini[1][0000−0002−3625−5414], Michele Braccini[1][0000−0002−2335−2514], and Andrea Roli[1,2][0000−0001−9891−5441]

[1] Department of Computer Science and Engineering (DISI), Campus of Cesena, *Alma Mater Studiorum* Università di Bologna, Italy
[2] European Centre for Living Technology, Venice, Italy

Abstract. The ability to adapt to changes and unexpected situations is a commonly acknowledged hallmark of autonomy and intelligence. In this work we take inspiration from biology for the definition of a robot able to continuously adapt to changes. Specifically, we define its control structure and the mechanism used to perform the adaptation. The former is based on the *Reservoir Computing* framework, on which the latter acts. The result is the design of an Online Adaptive Reservoir Computing system based on a novel memristive reservoir: the *Nanowire Network*. Finally, the robot is tested on three different tasks taking place in different arenas. The results are then discussed and compared with a baseline algorithm.

Keywords: Phenotypic plasticity · Online adaptation mechanism · Nanowire network robotics

1 Introduction

Recent technological advances have made it possible to build incredibly small robots, till the size of tens of nanometers. Nevertheless, the current smallest robots can perform only few predetermined actions, therefore they cannot attain the level of adaptivity required to accomplish complex missions. Conversely, AI robotic software has recently made tremendous advancements and has been proven capable of tackling difficult tasks with a high degree of reliability. This software, however, cannot be run onto small size robots. A viable way for filling this gap is provided by control programs based on unconventional computation, such as the ones based on artificial neural networks or models of genetic networks.

In this work we present a study on the viability of using Nanowire Networks to control robots subject to an online adaptive mechanism. This work is a first step towards the deployment of small robots capable of adapting their behavior to the environment in which they operate. This is indeed the main property required to actual autonomous embodied agents [20].

2 Nanowire networks

One of the main innovative points introduced in this work is the use of Nanowire Networks (NNs) for the control of a robot [3]. NNs are a novel kind of nanoscale electrical circuit, whose interest resides in their ability to produce a neuromorphic behavior. This is given by their self-organizing property and intrinsic structural and functional plasticity, mimicking biological networks [17,18]. The similarity resides in what is known as Hebbian Theory [12], a property that is commonly summarized as "neurons wire together if they fire together" [15]. The artificial equivalent of the synaptic strengthening is here represented by the behavior of the ion-silver-bridges that connect the wires of the network (see Figure 1 B). When subject to a voltage difference, the ions aggregate in the junction increasing its conductance. When the stimulus is removed, the network slowly returns to its stable state. In this work, we consider the voltage stimulation to be the only way to modify the internal state of the NN. This dynamics rules the network plasticity and can be seen as a short-term memory, that can be used to process spatio-temporal data [9,11,7].

NNs are a promising technology in all the context that require complex computations and have strict consumption constraints, like edge computing and robotics.

3 Control system

NNs are powerful computing devices with promising properties for robotics, but their use requires some ingenuity. We decided to design a robotic architecture shaped on their specific characteristics and, in particular, based on the Reservoir Computing (RC) framework and including an adaptive mechanism to endow the robot with phenotypic plasticity.

3.1 Reservoir computing

Reservoir Computing is currently accepted as the *de facto* framework for the use of unconventional dynamic systems for computation [19]. The working principle consists in perturbing a *reservoir* and analyze its resulting state through the use of a simple learning method, like regression or classification [13,16] (see Figure 1 A). This exploits the ability of the reservoir to re-project spatio-temporal inputs to a higher dimensionality. The result is a faster and less power-consuming training. However, despite all its benefits the use of RC in robotics is still limited [4]. Additionally, online variants of this framework have been explored only to a limited extent [1].

In this work, we address some of these missing explorations. Specifically, we propose the use of NNs as reservoir. Their non-linear dynamics allow indeed sequences of data to be evaluated differently, exploiting the intrinsic memory arising by their structural plasticity. Additionally, we suggest in combination the use of an online adaptive mechanism. The goal is to create a robot able to adaptively perform advanced operations, exploiting very simple systems: the NNs.

3.2 Adaptive mechanism

In order to perform an online adaptation, we decided to modify the classical RC architecture substituting the training of the readout with a stochastic optimization process operating on the inputs. The idea is that it is possible to learn how to perturb the reservoir in order to induce a desired internal state. The approach consists in performing a reconnection and weighting of the input signals to different nodes of the network (see Figure 1 C). The affected connections are a stochastically chosen sub-set ranging from 10% to 40% of the total. Network nodes eligible for reconnection must differ from previous ones and be k-nodes distant from the outputs[1]. The weighting consists in attenuating or intensifying the input signal through the use of a multiplier. Its initial value is chosen with a normal distribution centered in 1, and is then adapted adding the result of a normal distribution centered in 0. This parameter allows to balance the influence of specific inputs in the computation. A possible application is when some sensors are more useful than others for a specific task, and we want to enhance their role (e.g., front sensors in collision avoidance). Alternatively, this feature may help to balance measures of different physical properties, or in different environmental conditions where the signals homogeneously change (e.g., average brightness during day or night).

The proposed strategy is a variant of a methodology used for the adaptation of robots controlled by Boolean Networks (BNs) [6]. The novelty consists in the weighting of the input signals. This modification takes advantage of the analogic working mode of the NNs, that adds both a complexity and a potential compared to BNs. As the inspiring vision of this work is that of a micro-robot, the adaptation mechanism used in the experiment is minimalistic, so as to facilitate the construction for real applications.

3.3 Robotic architecture

The robotic architecture we designed is inspired by biological organisms. Specifically, our model is shaped on the Central Nervous System (CNS), which forwards and transforms the signals from the sensors to the cortex, and from the cortex to the muscles. The final version of this architecture (see Figure 1 C) was obtained by an iterative refactoring, trying to match the technical with the biological part. The first step concerned the individuation of five macro-areas: *i.* sensing $(S_0, ..., S_n)$, *ii.* input weighting and reconnection $(\alpha_0, ..., \alpha_n)$, *iii.* reservoir, *iv.* output reconnection (β), *v.* actuation (LM, RM). Each of them is mapped into its biological correspondent (see Figure 2). The sensing apparatus (*i.*) is equivalent to the biological sensors. The transmission and processing of the input signals (*ii.*) is represented by the thalamus; its role is indeed to forward all

[1] In our experiments k is 2.

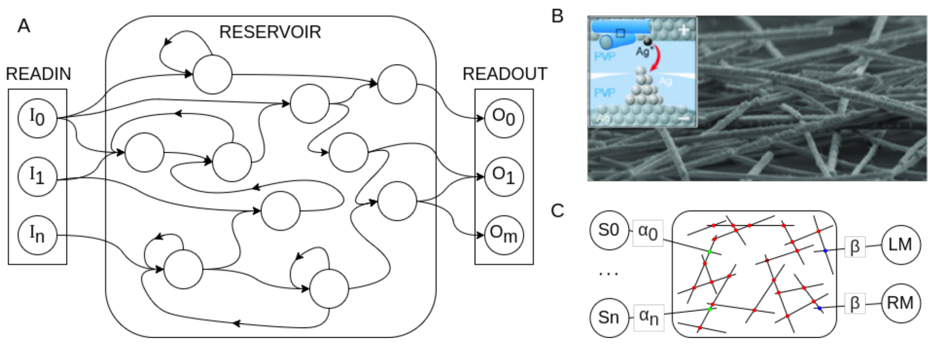

Fig. 1. A. The Echo State Network: one of the first instances of the Reservoir Computing architecture. **B.** Microscopy image of a Nanowire Network. Picture taken by [17] by courtesy of the authors. **C.** Schematic representation of the control architecture. $[S_0, ..., S_n]$ represent the robot sensors, while LM, RM represent the left and right motor. $[\alpha_0, ..., \alpha_n]$ represent the weighting factor, different for each sensor signal. β represents the influence of the motor resistance in the electrical equivalent system.

the incoming signals and to redirect feedbacks to the correct location [21]. Additionally, it controls the flow of information and transforms the signals acting like a filter [5,8]. In fact, this is a simplification since in humans the transformation of the inputs does not happen in a single point, but along the entire afferent CNS[2].

The NN reservoir (*iii.*) performs most of the computations of the artificial control system. Because of that, we consider this component equivalent to the cortex. The transmission of the motor commands (*iv.*) from the cortex is represented by the pyramids. As for the afferent tract, this is also a simplification of the biological world. The efferent fibers[3] are indeed redirected to the muscles in many points of the descending tract. Finally, the motors (*v.*) are mapped on the biological muscles. This choice is straightforward, since both operate as an actuation apparatus.

In a nutshell, the basic idea of the control architecture is that the sensory inputs are weighted and forwarded to specific points of the network, influencing the "reasoning" of the robot. Accordingly, motion commands are taken from specific points and used to control the actuators.

3.4 Adaptation cycle

The main goal of the adaptation mechanism is the emergence and optimization of a successful behavior, showing what is in biology known as phenotypic plasticity. This is the ability of a genotype to produce a visible response that depends on and adapts to the environmental conditions. To provide this ability

[2] The tract of the CNS going from the sensors to the cortex.

[3] Fibers of the human body carrying signals from the cortex to the muscles.

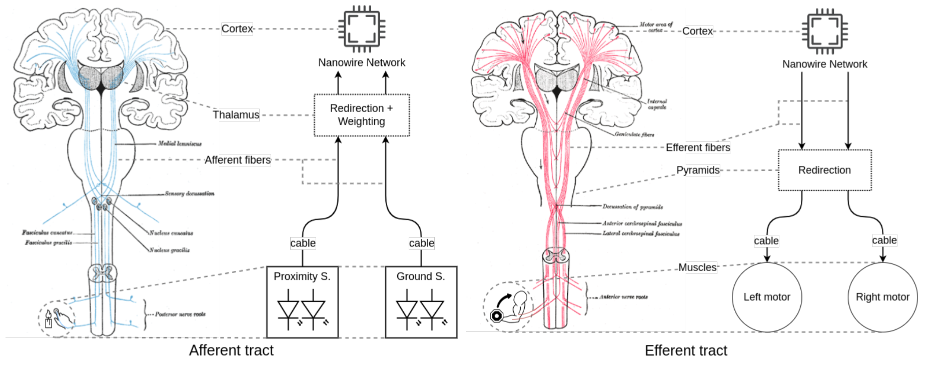

Fig. 2. Mapping between the control systems of the biological and artificial agent. On the left, the sensing path is represented. On the right, the control path is shown.

the adaptation runs continuously during robot's life, and is therefore said to work *online* (see Figure 3). The initial configuration is set at a random (i.e., a set of inward and outward connections and weights is sampled at random), and is evaluated during a fixed period of time and saved. This is then adapted by the adaptive mechanism previously described. The result is tested and, if better than the best one found, it is saved. Otherwise, the previously-found best-configuration is kept for future adaptations. The process repeats adapting the best found configuration and evaluating the quality of the adaptation. The quality is internally calculated by the robot itself in terms of a utility function, which is the agent internal driving force [2]. As the cycle continues perpetually, the robot has the opportunity to continuously improve its behavior and adapt it to possible changes in the working environment. As a consequence, each robot undergoes its own development, as typically happens in ontogenetic processes and historical processes in general [14].

During the experiments, we noticed that the optimization of some poor configurations slows down the adaptation. In order to speed up the process, we tested the use of a threshold value to early discard unsatisfactory solutions. Instead of them, new random ones are generated and evaluated.

4 Experiments

We evaluated the robotic system on three tasks taking place in three different simulated environments (see Figure 4): *i.* Collision Avoidance (CA), *ii.* Area Avoidance (AA), *iii.* T-maze (TM). The experiments run within the Webots simulator and consist in the test of 250 unique NNs. The NN-based architecture is adapted 300 times, each generating a *configuration*. Each configuration is then tested during an *epoch*. The duration of each epoch depends on the task and on the size of the arena:

- 20s for the CA,

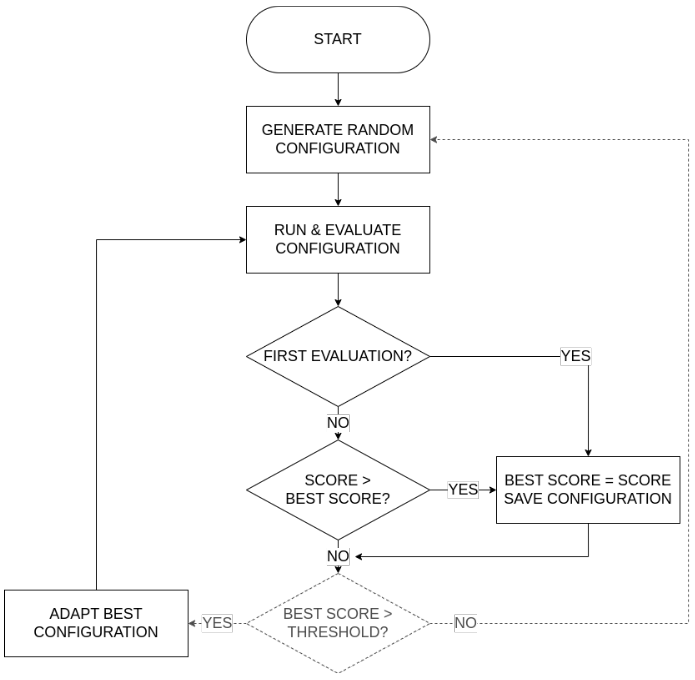

Fig. 3. The adaptation cycle that is continuously run by the robot. The gray dashed block and lines are an addition to the basic idea of adaptation.

– 100s for the AA,
– 100s for the TM.

Each assignment is defined by specific Objective Function (OF), that the robot self-evaluate and try to maximize.

The performance of this mechanism is compared with that of a stochastic mechanism, whereby the adaptation does not take place and the new configurations are generated randomly at each step. In other words, in the stochastic mechanism the best configuration is never adapted, but instead a new one is created from scratch at each epoch. This is used as a baseline to evaluate the quality of the adaptive mechanism.

4.1 Collision avoidance

In the Collision Avoidance task the robot is required to avoid collisions while going as fast as possible on a straight line. Therefore, the corresponding OF penalizes excessive turns and time spent near to obstacles:

$$fitness = (1 - \sqrt{p_{max}}) \cdot (1 - |v_l - v_r|) \cdot \frac{v_l + v_r}{2} \tag{1}$$

where:

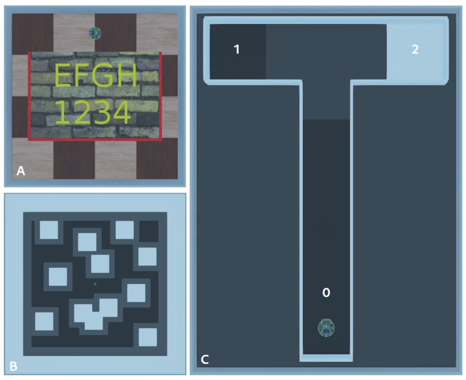

Fig. 4. The corresponding arena of each task: **A.** Collision Avoidance, **B.** Area Avoidance, **C.** T-Maze. **1** & **2** represent the T-maze end-points; **0** the starting-point).

$p_{max} \in [0, 1]$ is the normalized maximum proximity[4],
$v_l, v_r \in [0, 1]$ are respectively the normalized left and right motor speeds[5].

In Equation 1, the proximity undergoes a square root in order to increase the sensibility of the OF to collisions[6]. The goal is to reward configurations that stay far from the obstacles, reducing the influence of the trajectory and speed.

The arena consists in a central obstacle and a set of external walls (see Figure 4 A). The result is a circuit in which the robot has to run avoiding collisions. The passages on the left and right are stricter than the top and bottom ones, adding a minimal complexity to the task.

4.2 Area avoidance

The Area Avoidance task is similar to the CA one, but requires the robot to avoid virtual areas identified only by the ground color. The obstacles stop to be physical and become instead intangible. This complicates the tasks because the robot has to react faster. A slow response would indeed push the robot deeper into the forbidden area, reducing the score of future actions. In this task the system is driven by a single sensor, allowing to test how much the weighting of the signal is useful in achieving a good result. Additionally, it shows that it is possible to obtain a wandering behavior and still avoiding obstacles with a computationally limited system. The performance of the robot at a specific step in the AA task is computed with the following OF:

$$fitness = (1 - |v_l - v_r|) \cdot \frac{v_l + v_r}{2} - 100 \cdot c \qquad (2)$$

where:

[4] 0 represents a far object, 1 represents a near one.
[5] 0 represents an anticlockwise revolution, 0.5 a still state and 1 a clockwise revolution.
[6] Being the range of value in [0, 1], the square root increases the value of proximity.

$c \in \{0,1\}$ is 1 if the robot is on the illegal area, 0 otherwise,
$v_l, v_r \in [0,1]$ are respectively the normalized left and right motor speeds[7].

In Equation 2, hovering an illegal area is associated with a strong penalty. This is independent of the direction of the robot or its speed. The result is that the adaptation leads to a behavior that mostly avoid the illegal areas, while the quality is tuned by the speed and direction of the movement.

The arena consists of few illegal areas that the robot cannot hover (white blocks in Figure 4 B). Additionally, it is limited by the presence of virtual and physical borders, assigning a deserter robot negative scores while preventing it from going too distant. The goal is to allow every configuration to go back to a legal area, also if it starts in a disadvantageous position. Finally, every illegal area is surrounded by a neutral zone that informs the robot that is approaching a forbidden space.

4.3 T-Maze

In the T-Maze task the robot is required to reach the correct end-point of a T-shaped maze (see 1 & 2 in Figure 4 C). The goal destination depends on the color of the floor at the start of the run (see 0 in Figure 4 C): right for black, left for white. This requires the adaptive mechanism to exploit the NN plasticity to somehow memorize this initial input, and behave accordingly also when the signal stop to be perceived. In order to test the behavior, the robot is periodically kidnapped and placed back at the start of the maze. The OF for the TM task is the following:

$$fitness = 2 \cdot \overline{\alpha} \cdot \overline{\beta} - \alpha \tag{3}$$

where:

$\alpha \in \{0,1\}$ is 1 if the ground color is the same as the starting one,
$\beta \in \{0,1\}$ is 1 if the ground is gray.

The Equation 3 highlights the presence of a gray color. This is the color of the ground outside the starting and ending points. In this region, the fitness of the robot does not change. Instead, the performance is penalized if the robot remains in the starting area or if it reaches the wrong end point. If the robot reaches the correct end point, its score is increased by 2 at each step of the permanence. This helps in slightly reducing the simulation time. One aspect not considered in Equation 3 is the speed of the robot. The design of the task in combination with the epoch duration implicitly requires the robot to not be slow: in the opposite case, it would not have enough time to reach the end point.

5 Results

The results of the experiments are compared considering performance increment and distribution. The value at each time-step represents the average of the best

[7] 0 represents an anticlockwise revolution, 0.5 a still state and 1 a clockwise revolution.

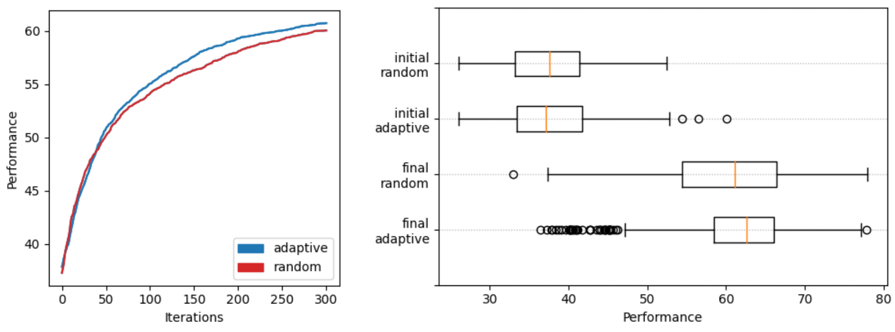

Fig. 5. CA results in terms of average fitness along the iterations (left) and distribution of the best results achieved in the 250 replicas of each experiment (right).

score obtained by each robot since the start of the simulation. This is therefore never decreasing, and is a good compromise to represent the ability to adapt [6].

The adaptive mechanism always shows higher performance than the random one. The difference between the results of the two approaches seems to be strictly related to the task that we are considering. In the CA, the results gap is not impressive (see Figure 5). Indeed, although the adaptation leads to a slightly better performance, the random mechanism still allows to obtain good results. We can explain this behavior with the low complexity of the task and the high correlation of the signals from the sensors. Their disposition around the robot cause many of them to perceive the same obstacles. This allows the system to be intrinsically more resistant to faults, but also to perform well with just few sensors correctly connected. Additionally, in the given task the importance of the sensors is not homogeneous, with the front ones being more useful than the back ones.

When we consider the AA task the improvement is more evident, especially in the score distribution (see Figure 6). This is due to the finer tuning of the weighting parameter, that helps to exploit all the information contained in the single input signal. Nevertheless, at the same time, the presence of a single sensor reduces the complexity of the connection, allowing also the random mechanism to perform well. This cause the effectiveness of the adaptation to be limited to the optimization of the input weight.

Finally, the TM task sees the highest improvement in performance (see Figure 7). This is due to the increasing complexity, requiring a finer tuning. The ability to change direction is indeed strictly related to the way the system stores information in the NN. In order to make it influential, the mechanism has to accurately balance the stimulation from the ground sensor. This strongly suggests that the adaptation might become more useful as the complexity of the task increase, eventually becoming essential.

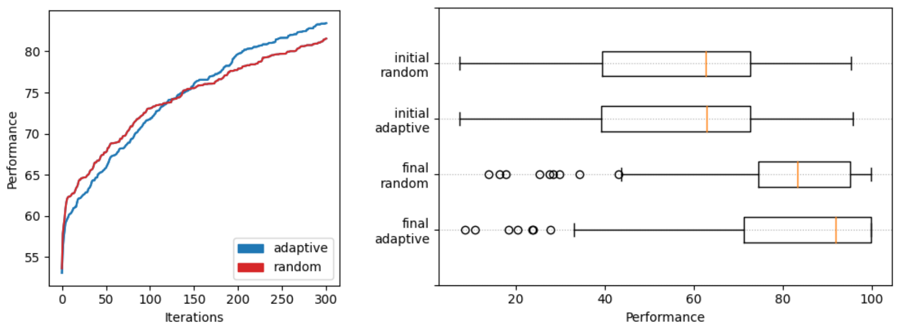

Fig. 6. AA results in terms of average fitness along the iterations (left) and distribution of the best results achieved in the 250 replicas of each experiment (right).

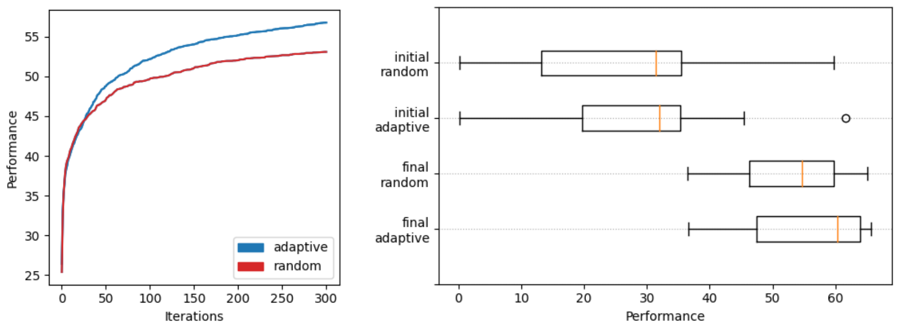

Fig. 7. TM results in terms of average fitness along the iterations (left) and distribution of the best results achieved in the 250 replicas of each experiment (right).

6 Discussion

The results presented in the previous section highlight a consistent gap between the scores obtained by the two compared mechanisms. The adaptation produces indeed overall better performances. Nevertheless, it is also evident that this difference is not impressive. One explanation is found in the amount of network nodes and epochs. Due to the complexity of the simulation, the tested NNs only contain a small amount of wires and junctions (see Table 1). Those are clearly much less than the number of epochs. This means that during the adaptation each sensor can possibly connect with any network node. Together with the intrinsic correlation of some sensor signals[8], and with their different influence[9], an effective solution may be found properly connecting just few sensors. In the full set of possible permutations of the input nodes, the amount of adequate

[8] Neighbor sensors often perceive same or similar information.

[9] Not all the sensors have the same utility in a task. For example, back sensors are usually less useful than front ones in the CA.

configurations is therefore fairly high. This means that the number of trials needed to obtain a good behavior is relatively small.

	Wires	Junctions
AA	68	120
CA	166	381
TM	166	381

Table 1. Average number of wires and junctions in the used NNs. The difference between the AA task and the others is due to the creation density.

The previous consideration is not completely satisfying, in that it would still suggest better performances of the adaptive method already from the initial epochs. The improvement is instead visible only after some tens of iterations. Therefore, another point to consider is the working mode of the NNs. Their relations with electrical circuits and the presence of the stimulation makes the choice of the connection node generally less important than in other network based systems (e.g., Boolean networks). Indeed, there is no risk for the input to disappear when reconnecting to neighbor nodes[10], but instead its influence on the actuators will only slightly change. Because of this working principle, the most important point in connecting to a NN seems to be related to the balance between the signals. This overall reduces the complexity of the wiring, but also limits the amount of complex behaviors that we can characterize from the network.

The result is that an adaptive reconnection, although useful, might not perform dramatically well compared to a random approach. It helps mainly when the complexity of the task is high or the size of the network grows, and when a more tuned balance between the input signals is needed. This idea is supported by the results obtained in the various tasks, seeing the performance gap increasing according to the complexity.

7 Conclusion

The goal of this work is to allow robots endowed with NNs to present the artificial correspondent of phenotypic plasticity. To achieve this goal we designed a control architecture based on NN and inspired by human CNS. The resulting system has been tested on three different tasks and environments. The results show that the robot is able to adapt its behavior to different tasks, attaining good performance. Additionally, they show that the adaptive mechanism allows to successfully exploit the intrinsic memory of the NNs. We conjecture that the limited advantage of an adaptive mechanism over a random one can be attributed to the complexity of the task, the size of the NN, and its working mode. Therefore, we plan

[10] Neighbor nodes in the graph representation of the NN. Two spatially near wires might indeed not be connected.

to test larger networks in harder tasks and environments. Other explorations may consider the use of local search techniques, exploiting a heuristic bias in the choice of the connections. As long term goal, we aim to design hardware adaptive mechanisms for the creation of microscopic robots, for example through the use of technologies like self-assembling wires [10].

References

1. Antonik, P., et al.: FPGA implementation of reservoir computing with online learning. In: 24th Belgian-Dutch Conference on Machine Learning (2015)
2. Ashby, W.: Design for a brain: The origin of adaptive behaviour. Butler & Tanner Ltd., second edn. (1954)
3. Baldini, P.: Online adaptation of robots controlled by nanowire networks. Master's thesis, University of Bologna (2022)
4. Baldini, P.: Reservoir computing in robotics: a review (2022)
5. Basso, M., et al.: Cortical function: a view from the thalamus. Neuron (2005)
6. Braccini, M., Roli, A., Barbieri, E., Kauffman, S.: On the criticality of adaptive boolean network robots. Entropy **24**(10), 1368:1–1368:21 (2022)
7. Christensen, D., et al.: 2022 roadmap on neuromorphic computing and engineering. Neuromorphic Computing and Engineering (2022)
8. Connelly, W., et al.: The thalamus as a low pass filter: filtering at the cellular level does not equate with filtering at the network level. Frontiers in neural circuits (2016)
9. Demis, E., et al.: Nanoarchitectonic atomic switch networks for unconventional computing. Japanese Journal of Applied Physics (2016)
10. Dueweke, M., et al.: Self-assembling electrical connections based on the principle of minimum resistance. Physical Review E (1996)
11. Fu, K., et al.: Reservoir computing with neuromemristive nanowire networks. In: 2020 International Joint Conference on Neural Networks (IJCNN) (2020)
12. Hebb, D.: The organization of behavior: A neuropsychological theory. Psychology Press (2005)
13. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report (2001)
14. Longo, G.: How future depends on past and rare events in systems of life. Foundations of Science **23**(3), 443–474 (2018)
15. Löwel, S., Singer, W.: Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. Science (1992)
16. Maass, W., et al.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural computation (2002)
17. Milano, G., et al.: Brain-inspired structural plasticity through reweighting and rewiring in multi-terminal self-organizing memristive nanowire networks. Advanced Intelligent Systems (2020)
18. Milano, G., et al.: Connectome of memristive nanowire networks through graph theory. Neural Networks (2022)
19. Nakajima, K., Fischer, I.: Reservoir Computing. Springer (2021)
20. Pfeifer, R., Scheier, C.: Understanding intelligence. The MIT Press (2001)
21. Sommer, M.: The role of the thalamus in motor control. Current opinion in neurobiology (2003)