Static versus Plastic Controllers in Evolutionary Robotics

Thijs van Lankveld [a] Guido de Croon [a] Karl Tuyls [a]

[a] *Adaptive Agents Group, MICC, Maastricht University, The Netherlands*

Abstract

Evolving a control mechanism for a physical robot is time consuming. In their work Floreano and Urzelai introduced the plastic controller. They showed that this controller can be quickly developed in simulation, and successfully transferred to a physical body. Moreover, they showed that plastic controllers perform better than static controllers. However, when comparing the performance of plastic controller to controllers with evolutionary determined weights, they used a different encoding during evolution for both controllers. This difference in encoding led to a difference in search space, which requires more time to find a good solution. The question arising then is how plastic controllers and static weight controllers compare when both use the same encoding. In this paper we address this question and compare the plastic and static weight controller on two tasks using the same encoding during evolution. The tasks are the light switching task and the gradient grey task.

1 Introduction

There are many different methods for creating robots and the programs that control them. One of the fields in which these methods are developed is Evolutionary Robotics (ER) [6]. In this field robot controllers are created through evolutionary processes. Typically, an Artificial Neural Network (ANN) serves as the controller. Since the behaviour of a robot controlled by an ANN is governed by the weights of the connections in the ANN, the evolutionary process usually optimises the weights of the neural network and the weights do not change over time during robot control. These networks are called static weight controllers, because their weights do not change during operation.

Since evolution on real robots is very time-consuming, the robot controllers are usually evolved in simulation and then transferred to the real robot for execution [2, 4]. A disadvantage of static neural networks is that their performance degrades when they are ported to a real robot. The reason for this are the differences between the simulated and the real robot, i.e., differences in real world sensing and actuation.To improve performance on the real robot, evolution sometimes has to be continued on the physical robot.

In recent work Floreano and Urzelai introduced the plastic controller, also called plastic neural network. This network is based on Hebb's theory about the adaptation of synapses [3]. Plastic controllers use one of four learning rules, to update the strength of the weights in the ANN during robot control. These learning rules are based on Hebb's theory that states that neurons that fire together, wire together. This means that when two connected neurons are active at the same time, the connection between them is strengthened. The main promise of using this controller is that it can be transferred from simulation to reality without having to re-evolve [1, 8]. Floreano and Urzelai give multiple examples of the power of plastic controllers in their articles. For instance, they show that plastic controllers can be developed faster than static weight controllers and that plastic controllers perform better than static controllers when faced with changing environments. However, in their comparison of plastic and static controllers, they used different weight encodings, resulting in different sizes of search space. As Floreano and Urzelai used populations of the same size and also the same number of generations for both controllers, the results are not directly comparable.

In this article, we investigate the following research question: 'Does an evolved plastic neural network produce better controllers for a robot than an evolved static weight network when both are using the same genome encoding?' We answer this question by comparing synapse encoded plastic and static controllers on two tasks: the light switching task that was originally used by Floreano and Urzelai, and the gradient gray task, which is a more difficult task.

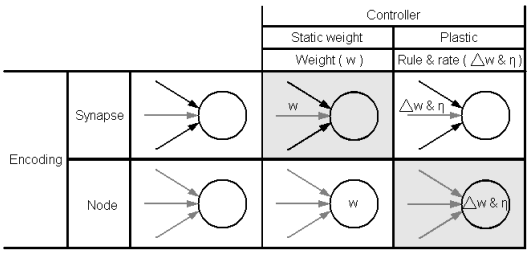


Figure 1: The different dimensions that Floreano and Urzelai compare. The different controllers are shown at the top, with the information they encode in the genome. The different encodings are shown on the left, with the difference in encoding shown graphically: synapse encoding encodes per connection; node encoding encodes per node. The comparison made by Floreano and Urzelai is shown in the grey cells.

The remainder of this paper is structured as follows. In Section 2 we motivate the studied comparison in more detail. Section 3 introduces the necessary background on the evolutionary algorithm and the studied controllers. We continue with the experimental setup in Section 4. Next we outline the results in Section 5 and end with a discussion and conclusion.

2 Motivation

To demonstrate the power of the plastic controller, Floreano and Urzelai [1] compared it with a controller with genetically determined weights in a task called the light-switching task. They also used different ways to encode the controllers in a genome during evolution. For the static weight controller they encoded every synaptic weight in the genome, whereas for the plastic controller they used the same learning rule and rate for every incoming connection to a node and so they encoded only one connection per node in the genome. The encoding they used for the static weight controller is called synapse encoding or direct encoding. The encoding they used for the plastic controller is called node encoding, because only one connection per node is encoded in the genome. This difference in encoding is not a requirement of the different controllers.

An important problem emerging from the comparison of Floreano and Urzelai is that not only the static weight controller and plastic controller are compared, but also node encoding and synapse encoding are compared. Both dimensions are shown in Figure 1. It is possible that when the controllers are compared using the same encoding, the results will differ from those presented by Floreano and Urzelai. More precisely, the difference in performance between the static weight and plastic controller could be caused by the difference in size of the search space. Both controllers use the same number of bits in their genome to represent one connection, i.e., 5. However node encoding and synapse encoding use a different number of bits in their genome, i.e., node encoding uses 5 bits per node, where synapse encoding uses 5 bits per connection. In controllers created using an evolutionary algorithm, the search space is proportional to the size of the genome and a larger search space requires more iterations to obtain a comparable result. More concretely, in the topology used by Floreano and Urzelai [1], node encoded networks have a genome of 60 bits, while synapse encoded networks have a genome of 720 bits. When this architecture is scaled up with 20 hidden neurons, the sizes go up to 160 and 5120 bits respectively. As Floreano and Urzelai [1] used populations of the same size and also the same number of generations for both controllers, the results are not directly comparable. The differences in performance might just as well be attributed to the size of the search space, though it must be admitted that one of the original goals of the plastic controller was reached: the controller yielded good results, even when evolved in a small population, using few generations and a compact encoding.

Note that another comparative study on plastic controllers was done by Tuci and Quinn [7], when they compared the plastic controller with a continuous time recurrent neural network (CTRNN) on a different task. Their results show that in this case the plastic controller is unable to learn a pattern and remember it, giving a less optimistic view of the plastic controller than Floreano and Urzelai show.

(a) Static weight controller genome

(b) Plastic controller genome

Figure 2: A gene in the genome: (a) a static weight controller genome and (b) a plastic controller genome. In both figures a gene is marked grey. In the case of synapse encoding, each gene determines one connection in the network.

3 Background

This section introduces the necessary background information to understand the remainder of the paper. First we discuss the evolution of the different controllers. Next we continue with an elaboration on the different controllers.

3.1 Evolutionary Algorithm

In this paper, both controllers use synapse encoding. A comparison is made between the top two cells shown in Figure 1. In the genotype of a static weight network, each gene represents the strength of a connection in the network. Each gene is normalised to [-1, 1] during development, by taking the decimal value of the last four bits and dividing this by 16. The first bit is then used to provide the sign. In the genotype of a plastic network, each gene contains three parts. The first bit represents the sign of the weight. The next two bits represent the learning rate η, chosen from $\{0.0, 0.3, 0.6, 0.9\}$. The last two bits represent the learning rule used. These differences in the genes are illustrated in Figure 2, and they are the same as used in [8]. Note that the genes differ in meaning, but not in size. As both controllers use synapse encoding, both genomes are also the same size, unlike the comparison done by Floreano and Urzelai. The genome size still differs per task, because a different number of sensors is used. After development, each ANN goes through a run in the environment. At this stage the ANN is also called an individual. During this run, the genetic algorithm computes the individual's fitness. This is a measure for how well the individual performs the task. The individuals that are allowed to reproduce are picked using the method of selection. There are different ways to select individuals for reproduction. Methods that are often used are elitism, roulette wheel, and rank selection. In elitism the individuals that performed best are selected, with no chance that individuals that didn't make it to the best few are selected. This method is straightforward, but has the disadvantage of quickly losing genetic diversity.

In this paper a selection operator akin to elitism is used, because Floreano and Urzelai also used it in their research. During selection only a number of the best individuals is selected, depending on the tasks described in Section 4. These individuals are allowed to reproduce and the offspring make up the next population. The operators used during reproduction are crossover and mutation. In this research multiple-point crossover is used. This means that there can be multiple points on the genome where the bits are crossed over. Mutation is a method where there is a small chance for each bit to flip. A flipped bit changes from 0 to 1 or vice versa. No members of the population will be copied without crossover or mutation. For a detailed elaboration of evolutionary algorithms see for instance [5].

3.2 Controllers

The type of topology used in this work is a fully connected recurrent, discrete time neural network. In this topology a node gets as input the previous activation of every node. Some nodes are classified as input nodes. These also receive input from one sensor. Which sensors are used is defined by the task. Finally there are two nodes that are classified as output nodes. The activation of these nodes after propagation is sent to the motors. This is the same topology that Urzelai and Floreano used in their research on plastic controllers [8].

A static weight controller is a controller with weights that do not change during a run. This means that in artificial evolution, the weights are genetically determined. In a static weight network, a bias neuron is used.

Unlike static weight networks, the weights of a plastic controller are not genetically determined. Instead the genome stores the way the weights change during a run. This change in weight is according to one of

four learning rules. These rules are chosen by Floreano, because of neurophysiologic findings presented by Willshaw and Dayan [10]. The weights are constrained to the range of $[0, 1]$ by a self limiting mechanism. This ensures that the weights cannot grow indefinitely, but it also means that it must be explicitly stated if a connection is excitatory or inhibitory. At the start of each run the weights are randomly initialised in the range of $[0, 0.1]$. During the run, the weights of the plastic controller are updated according to the following formula.

$$w_{ij}^t = w_{ij}^{t-1} + \eta \Delta w_{ij}$$

Here $0 < \eta < 1$ is the learning rate, w_{ij}^t is the weight between node i and node j at time t and Δw_{ij} is its learning rule. This rule is one of the four defined learning rules:

Plain Hebb	$\Delta w_{ij} = (1 - w_{ij}) x_i y_j$	
Post-synaptic	$\Delta w_{ij} = w_{ij}(x_i - 1) y_j + (1 - w_{ij}) x_i y_j$	
Pre-synaptic	$\Delta w_{ij} = w_{ij} x_i (y_j - 1) + (1 - w_{ij}) x_i y_j$	
Covariance	$\Delta w_{ij} = (1 - w_{ij}) F(x_i, y_j)$	if $F(x_i, y_j) > 0$
	$\Delta w_{ij} = w_{ij} F(x_i, y_j)$	otherwise
	$F(x_i, y_j) = arctan(4(1 - \lvert x_i - y_j \rvert) - 2)$	

Here x_i is the activation of the sending node, y_j the activation of the receiving node and w_{ij} the current weight of the connection.

Each rule follows Hebb's theory that nodes that fire together, wire together. However, each rule differs in the way this is done. The plain Hebb rule is the most basic learning rule, increasing the weight when both nodes have an activation above 0. The post-synaptic rule does the same, but also decreases the weight when the post-synaptic node is active, while the pre-synaptic node is not. The pre-synaptic rule works in a comparable way, but decreases the weight when the pre-synaptic node is active, while the post-synaptic node is not. The covariance rule increases or decreases the weight according to the similarities in activation of both nodes. When both nodes have comparable activations, the weight is increased, but when the activations of the nodes differ greatly the weight is decreased. An important side note to this process is that the sign of the weights is not changed by the learning rules. This way an excitatory connection stays excitatory and an inhibitory connection stays inhibitory. This is done by using the learning rules on the absolute value of each weight, discounting the sign during the learning process. Besides having weights that are updated on-line, a plastic controller is comparable to a controller with static weights.

Unlike a static weight network, a plastic controller does not use a bias node. The way a plastic neural network is encoded differs from the encoding of a static weight network, because different aspects must be saved in the genome. For more details we refer to [8]. Note that learning is done once the task starts, and the tasks have time constraints. This means that fast learning is encouraged.

4 Experimental Setup

4.1 Light Switching Task

The light switching task is the task used by Floreano and Urzelai to show the power of the plastic controller [1, 8, 9]. Where possible, the parameters used in this task are derived from their publications. In the light switching task the robot must switch on the light on one side of the environment and then stand under the light on the opposite side of the environment. The environment is an area of 600mm by 400mm. On the right side of the environment is a black mark, 50mm broad, in the centre of the wall. Under this mark is a semicircle with a radius of 100mm which is also black. On the left side of the environment is a light attached to the centre of the wall with under it a semicircle of 100mm that is grey. Note that the semicircles on the floor are no input to the controller; they are merely used to determine when the light is switched on (dark grey) and when the controller will be awarded fitness (light grey). This setup is shown in Figure 3.

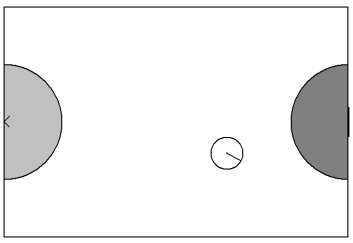


Figure 3: Light switching environment

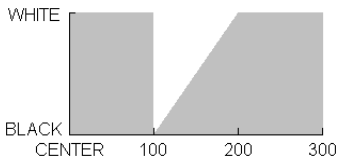


Figure 4: The colour of the floor in the gradient grey task, compared to the distance from the centre of the environment.

The type of topology used in this task is a fully recurrent, discrete time neural network . The fitness function used in this task is entirely dependent on the time spent in the light grey semicircle while the light is on. As the task is started with the light off, the robot will first have to switch the light on, but doing so does not award any fitness. The fitness function is as follows.

$f(x) = 1$ if the light is on and the centre of the robot is over the light grey area
$f(x) = 0$ otherwise

$$F = (\Sigma_{c=1}^{C} f(c))/C$$

Here f is the fitness awarded each cycle; F is the overall fitness awarded to the controller; c is the current cycle; and C is the total number of cycles the robot lives. This last term keeps the fitness between 0 and 1.

Each controller is evaluated for 10 epochs of 500 cycles. At the start of each epoch the robot is placed on a randomly determined location within the environment, facing a random direction. The robot uses eight infra red proximity sensors (pooled into four pairs), eight ambient light sensors (pooled into three groups) and a camera with three pixels as input. This input pattern follows [8].

Each input neuron receives the average input values of one group with 5% noise added. Two motors form the output, and no hidden nodes are used. In a fully connected recurrent network, this leads to a total of 144 connections to optimise. If a robot hits a wall it is slowed down, but this does not end the epoch. The evolution is run for 10 replications of 200 generations. Each generation consists of a population of 100 controllers. After each generation, the 20 best controllers are selected and copied 5 times. During the reproduction process the offspring are subjected to crossover with a probability of 0.2 and mutation with a probability of 0.05.

4.2 Gradient Gray Task

In the gradient grey task the robot must stand still as close as possible to the centre of the room. The environment is an octagonal room with a distance between opposing walls of about 600mm. The floor within 100mm of the centre of the room is white, but after this the floor is black, gradually moving toward white when moving outwards. The tone of grey is dependent on the distance from the centre as shown in Figure 4. The robot will start each epoch on one of three fixed locations near the wall. Which of these locations the robot starts from, is determined randomly. The type of topology used in this task is a fully recurrent, discrete time neural network. The fitness function used in this task is dependent on the distance of the robot to the centre of the room. This leads to the following fitness function.

$f(x) = 1 - (r/R)$ if the robot is within the white centre circle
$f(x) = 0$ otherwise

$$F = (\Sigma_{c=1}^{C} f(c))/C$$

Here f is the fitness awarded each cycle; r is the distance between the centre of the robot and the centre of the room; R is the radius of the inner circle, so 100; F is the overall fitness awarded to the controller; c is the current cycle; and C is the maximum number of cycles the robot can live. This last term keeps the fitness between 0 and 1 and because an epoch is ended as soon as the robot hits anything, a robot that crashes into a wall or obstacle will never get a fitness of 1.

Each controller is evaluated for 10 epochs of 300 life cycles. The robot starts each epoch on one of the predetermined locations within the environment, facing to the right. Which of the locations the robot starts on is randomly determined. The robot uses eight infrared sensors and the ground sensor as input with an added 5% noise. Two motors form the output. The task is repeated twice, once without hidden neurons,

and once with two hidden neurons. Without hidden nodes, this topology has a total of 121 connections to optimise; with hidden nodes, this increases to 169 connections.

The evolution is run for 10 replications of 200 generations. Each generation consists of a population of 100 controllers. After each generation, the 20 best controllers are selected and copied 5 times. During the reproduction process the offspring are subjected to crossover with a probability of 0.2 and mutation with a probability of 0.05. These probabilities have been selected after a number of evolutionary runs, because they produce fit robots in a small number of generations.

5 Results

5.1 Light Switching

In the light switching task the robot should switch on the light and then move to and stay under the light. After evolution, the static weight controllers and plastic controllers are compared, both on fitness and on behaviour. The fitness of both controllers is shown in Figure 5.

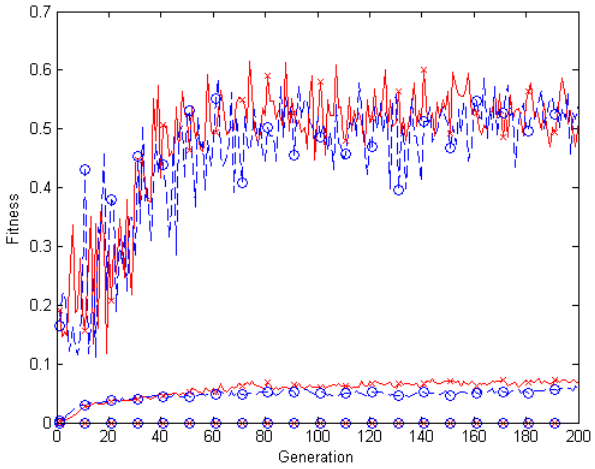


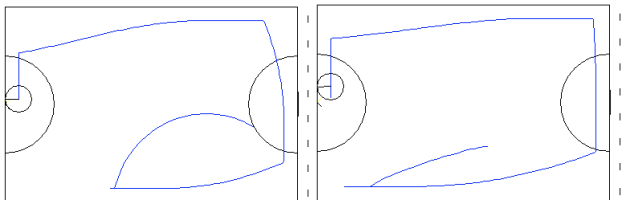
Figure 5: The fitness of the two controllers in the light switching task. The x-axis represents the generation during evolution and the y-axis represents the fitness. The top lines show the best fitness of all ten replications at each generation. The middle lines show the average fitness at each generation, taken over all ten replications. The lower lines show the worst fitness of all ten replications at each generation. The best, average, and worst fitnesses of the static weight controller are represented by the red closed lines marked with an x. The best, average, and worst fitnesses of the plastic controller are represented by the blue dashed lines marked with an o.

As this figure shows, the difference between static weight controllers and plastic controllers is minimal. This is also seen when the behaviour of both controllers is compared. The behaviour of both controllers is shown in Figure 6.

This experiment was also done with a darker environment to see if there is more difference in behaviour between static weight controllers and plastic controllers when the environment is changed slightly. The walls are turned a shade of grey that is 30% less bright than white. This coloring of the walls was not presented to the robots during evolution. When the best evolved controllers were put in this environment, they again showed comparable behaviour (Figure not included).

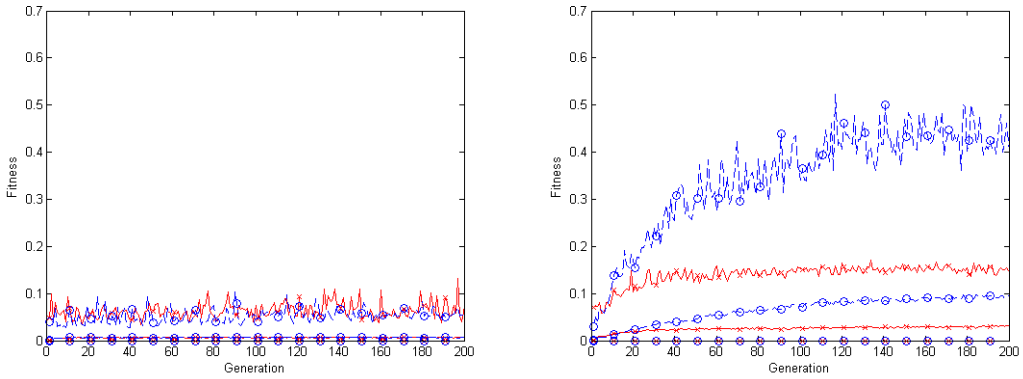
5.2 Gradient Grey

In the gradient grey task the robot should remain as close as possible to the center of the room. After evolution, the static weight controllers and plastic controllers are compared, both on fitness and on behaviour. The fitness of both controllers is shown in Figure 7a.



(a) Static weight controller (b) Plastic controller

Figure 6: (a) Static weight controller (b) Plastic controller: The behaviour of the two controllers in the light switching task. Each figure shows the path of the robot in the environment, shown by the blue line. The circle with the line in it shows the end position of the robot and its orientation.

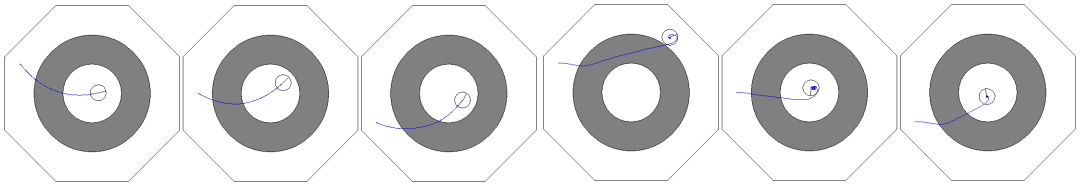


(a) Without hidden neurons (b) With hidden neurons

Figure 7: (a) Without hidden neurons (b) With hidden neurons: The x-axis represents the generation during evolution and the y-axis represents the fitness. The top lines show the best fitness of all ten replications at each generation. The middle lines show the average fitness at each generation, taken over all ten replications. The lower lines show the worst fitness of all ten replications at each generation. The best, average, and worst fitnesses of the static weight controller are represented by the red closed lines marked with an x. The best, average, and worst fitnesses of the plastic controller are represented by the blue dashed lines marked with an o.

As this figure shows, the difference between static weight controllers and plastic controllers is minimal, as neither seems to be able to perform the task with a high fitness. This was also seen in the behaviour of both controllers (figure not included). To test if the task could not be completed with a good performance, the experiment was repeated with another topology. In this topology, two fully connected recurrent hidden neurons are added. The fitness of both controllers after this experiment is shown in Figure 7b. The fitness function in this experiment is the same as that used in the regular gradient grey task. This time there is a clear difference between the static weight controller and the plastic controller. This is also seen when the behaviour of both controllers is compared. The behaviour of both controllers is shown in Figure 8.

This figure shows the behaviour of the best evolved individuals of both controllers with two hidden nodes. The static weight controller shows comparable behaviour to the behaviour of the same controller without the hidden neurons. This controller is still unable to find the centre of the environment. However the plastic controller shows a different behaviour. While still unable to find the centre of the environment from the upper start position, both other start positions result in the robot circling close to the centre. To see if the best fitnesses are an isolated occurence, or part of a pattern, the standard deviations of the best fitnesses of the ten replications is computed. These standard deviations are shown in Figure 9. This figure shows that the standard deviation of the plastic controller increases over the generations, but not as much as the mean of the best fitness of the plastic controller.



(a) Static weight controller (b) Plastic controller

Figure 8: The behaviour of the two controllers in the gradient grey task with hidden neurons: (a) the static weight controller and (b) the plastic controller. Each sub figure shows the path of the robot in the environment, starting from a different position. The path is shown by the blue line. The circle with the line in it shows the end position of the robot and its orientation.

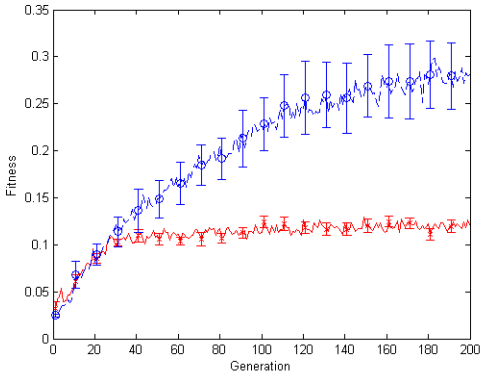


Figure 9: The average best fitness of the two controllers and their standard deviation over the ten replications of the experiment. The x-axis represents the generation during evolution and the y-axis represents the fitness. The lines show the average of the best fitnesses from the different replications. The bars at some points along the lines show the interval of -standard deviation to +standard deviation around the mean. The means and standard deviations of the static weight controller are represented by the red closed line marked with an x. The means and standard deviations of the plastic controller are represented by the blue dashed line marked with an o.

5.3 Discussion

The light switching task requires the robot to switch on a light and then move to it. This task shows comparable results when both static weight controller and plastic controller use synapse encoding as shown in Figure 5. The difference in maximum fitness is about 5%. However, the static weight controller does take longer to achieve this fitness. The fitness graph of the static weight controller flattens after about 60 generations, while the plastic controller only takes about 40 generations for its fitness graph to flatten. Nevertheless, the behaviour of both controllers is comparable. Both follow the wall, until they find a light source as shown in Figure 6. The second task that the controllers were compared on is the gradient grey task. In this task two concentric circles are placed on the floor of the environment: a small white circle and a larger circle with a gradient that gradually goes from black at the edge of the small circle to white at its own edge. The robot should drive over the gradient into the small circle and remain as close to the centre as possible. The results presented in Figure 7a show that both static weight controller and plastic controller have a comparable maximum fitness. Though the static weight controller gets a maximum fitness that is about 30% higher than the plastic controller, the fitness lines of both controllers are very erratic. The best results seem to be more random occurrences than a directed search, as the fitness does not improve greatly after the first generation. The behaviour of both static weight controller and plastic controller are comparable. Both move towards the centre, but neither is able to stop near the centre. A possible reason for this is the lack of hidden neurons in the topology. To test this hypothesis, the experiment is repeated with a topology with two hidden nodes. These results show a significant difference in fitness between the static weight controller

and the plastic controller as seen in Figure 7b, and Figure 9. Though the fitness graph for the static weight controller is less erratic, the maximum fitness is comparable to the maximum fitness it achieved without hidden nodes. However, the plastic controller does show a great increase in performance. The fitness graph shows a steady increase in fitness for about 120 generations and the maximum fitness is about three times higher than the maximum fitness of the static weight controller. The behaviours of the controllers also show the difference as seen in Figure 8. The static weight controller shows comparable behaviour to the earlier experiment. However, the plastic controller is now able to find the centre and stay near it. This shows that neural plasticity combined with recurrent hidden nodes is sufficient to accomplish the task, while either one separately is not.

6 Conclusion

We conclude that the performance difference between plastic and static controllers depends on the type of task. In the case of the light-switching task, plastic controllers do not outperform static controllers, when both have synapse encoding. Changing the wall-color to grey did not change the performances. Although we did not test the difference in the controllers on a real robot, these results suggest that the difference in performance reported in [8] might be due to a difference in behaviour on the original task.

In the case of the more difficult gradient gray task, the plastic controller outperforms the static controller when hidden neurons are added. The combination of plastic weights and recurrent hidden neurons seems to form an internal state powerful enough to judge distance travelled without changing input.

Our investigation leads to believe that the original performance differences in the research of Floreano and Urzelai are partly due to the differences in the size of the search space. If the plastic controller can perform well on more difficult tasks with node encoding, then this forms another advantage of the plastic controller. Whether this is the case could be the subject of further research.

References

[1] Dario Floreano and Joseba Urzelai. Evolution of neural controllers with adaptive synapses and compact genetic encoding. In Dario Floreano, Jean-Daniel Nicoud, and Francesco Mondada, editors, *ECAL*, volume 1674 of *Lecture Notes in Computer Science*, pages 183–194. Springer Verlag, 1999.

[2] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jokobi. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20:205–224, 1997.

[3] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, 1949.

[4] O. Miglino, H. H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434, 1995.

[5] M. Mitchell. *An introduction to Genetic Algorithms*. MIT Press, second edition, 1996.

[6] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, 2000.

[7] E. Tuci and M. Quinn. Behavioural plasticity in autonomous agents: a comparison between two types of controller. In *Proceedings of The Second European Workshop on Evolutionary Robotics EvoROB2003*, pages 661–672, 2003.

[8] J. Urzelai and D. Floreano. Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evolutionary Computation*, 9(4):495–524, 2001.

[9] J. Urzelai and D. Floreano. Evolution of plastic control networks. *Autonomous Robots*, 11(3):311–317, 2001.

[10] D. Willshaw and P. Dayan. Optimal plasticity from matrix memories: What goes up must come down. *Neural Computation*, pages 85–93, 1990.