Evolving a neural network using dyadic connections

Andreas Huemer
Institute Of Creative Technologies
De Montfort University, UK
E-mail: ahuemer@dmu.ac.uk

Mario Gongora
Centre for Computational Intelligence
De Montfort University, UK
E-mail: mgongora@dmu.ac.uk

David Elizondo
Centre for Computational Intelligence
De Montfort University, UK
E-mail: elizondo@dmu.ac.uk

***Abstract*—Since machine learning has become a tool to make more efficient design of sophisticated systems, we present in this paper a novel methodology to create powerful neural network controllers for complex systems while minimising the design effort.**

Using a robot task as a case study, we have shown that using the feedback from the robot itself, the system can learn from experience, or example provided by an expert.

We present a system where the processing of the feedback is integrated entirely in the growing of a spiking neural network system. The feedback is extracted from a measurement of a reward interpretation system provided by the designer, which takes into consideration the robot actions without the need for external explicit inputs.

Starting with a small basic neural network, new connections are created. The connections are separated into artificial dendrites, which are mainly used for classification issues, and artificial axons, which are responsible for selecting appropriate actions. New neurons are then created using a special connection structure and the current reward interpretation of the robot.

We show that dyadic connections can also make an artificial neural network acting and learning faster because they reduce the total number of neurons and connections needed in the resulting neural system.

The main goal of the research is to create a novel unsupervised learning system where the designer needs to define only the interface between the robot and the neural network in addition to the feedback system which includes a calculation of a reward value depending on the performance of the robot (or task aim of the system being developed).

I. Introduction

To design controllers for robots capable of performing complex tasks, much effort has been put into improving machine intelligence techniques. Given the vast variety of possible conditions present in the real world, machine learning has been the subject of significant research to improve the responses of autonomous systems to a multitude of situations. This paper relates to a novel methodology being studied and evaluated which is capable of creating a spiking neural based controller for applications such as robotics. Our method also takes into account the aspect of online learning which is a much more intuitive approach to real world problems.

Neural networks have been applied successfully to some control problems and learning systems, however, if the robot needs to be able to adapt to completely new situations, there have to be enough adaptable components in the neural network. One approach could be to provide a neural network with enough neurons and connect them fully, and have those connections adapted with known machine learning methods. Alternatively, it has been shown that partially connected neural networks are faster to train and have better generalisation capabilities [2]. Similar effects have been found considering the number of neurons [4], where it has been shown that more is not always better.

Consequently, a novel method for creating neural networks which, among other applications, could be useful for controlling a mobile robot is to use a neural network with a minimum number of neurons and connections and grow it until it can fulfil the task performance required by the system. The results presented in this paper have been evaluated with a set of experiments where a robot or a simulation of it learns to wander around in a room and to avoid obstacles.

We have created a self building neural network which learns from experience by connecting the neurons, adapting the connections and growing new neurons depending on a feedback process that will correspond to the measurement of a perceived reward of the robot. The measurement of the reward that the robot perceives can be defined by an evaluation function of the task performance, in which case the neural network will develop itself without the need of any human intervention, creating a purely automated learning mechanism; alternatively the reward values can be fed into the system at runtime, representing feedback from a trainer (either automated or human operated).

Florian simulated a worm that was fed with positive reward when its mouth was moving towards its food source and negative reward when its mouth was moving in the other direction [3]. A neural network controlled the movements of the worm. Depending on the feedback the connections between the neurons were adapted, which finally took the mouth of the worm to the food source. A similar method will be used in the experiments of this paper. In addition to adapting the connections between the neurons the reward will also be used for growing new neurons.

We have separated the neural connections in two parts: artificial dendrites and axons. These do not only play an important role with the growth mechanism but in the basic decision mechanism for the actions as well.

Initial constraints have been used at this stage of the research to provide a reliable evaluation of the novel growth methods. For example recurrent connections and Spike Time Dependent Plasticity have been excluded, which would both increase the capabilities of the neural network but which would also increase the dynamics of the network and hence the effort of evaluating it and the certainty of the evaluation results at this initial stage.

This paper is organised in the following way. Section II explains the principle of dyadic connections. In section III the basic learning mechanisms of the spiking neural network are described. The growth mechanism of the neural network is discussed in section IV, followed by results of testing the mechanism in section V. Section VI contains concluding remarks. At the end some ideas for further work are mentioned, in section VII.

II. Dyadic connections

A. Two parts for two tasks

There are different methods for connecting neurons in an artificial neural network. In the model presented in this paper spiking neurons are used, which send Boolean signals via the connections when a certain threshold potential is exceeded (a basic explanation of these can be found in [12]). For the experiments reported in this paper the threshold is kept equal in the whole neural network; one advantage of this is that all new neurons created need the same properties.

Neural networks can be used for: classification and/or action selection. We will define a classification task as of deciding which class a data vector belongs to according to a set of input features. An input pattern can for example belong to a certain class, if neuron 2 *AND* neuron 5 are active together. On the other hand a certain action is for example selected, if neuron 1 *OR* neuron 3 is active.

In a control system for a robot the classification task is needed to reduce the number of neurons that are responsible for selecting an action. Also the number of connections between neurons can be reduced by merging certain input patterns into classes. Classification is usually done by connecting several input neurons to a neuron that represents the class that all of the connected input neurons belong to. This process is often called representation and can be distributed over several layers. By combining several neurons into a single one at the next level, also in the succeeding parts of the network the number of neurons and connections can be reduced. It has been shown that less neurons and connections result in less computation effort and better development of the network [2] [4].

To be capable of dealing with classification tasks and action selection mechanisms at the same time, it is useful to separate the connections into two parts. For the model we are presenting dendrites connect axons with a postsynaptic neuron and axons connect a presynaptic neuron with a dendrite.

B. Computation

If the presynaptic neuron fires, which means that it sends a spike, its axons influence the dendrites connected to it. The influence can differ from axon to axon, depending on its weight, which is adjusted by the learning process discussed later. A single axon can be sufficient to activate a dendrite.

More issues of the separation of connections into axons and dendrites are discussed in section IV. The signals travel from a presynaptic to a postsynaptic neuron as explained by the following equations.

For all equations it is assumed that all axon weights of one dendrite sum up to 1. If weights are changed, they have to be normalised afterwards, so that the sum is 1 again.

Input of a dendrite:

$$I_d = \sum_p O_a(p) \cdot w_a(p) \tag{1}$$

where I_d is the dendrite's input. $O_a(p)$ is the output of axon p, which is 1, if the presynaptic neuron has fired and 0 otherwise. $w_a(p)$ is the weight of axon p.

Output of a dendrite:

$$O_d = \frac{1}{1 + e^{-b \cdot (I_d - \theta_d)}} \tag{2}$$

where O_d is the dendrite's output and I_d is its input. θ_d is a threshold value for the dendrite. b is an activation constant and defines the abruptness of activation.

The dendrites which belong to the postsynaptic neuron are also weighted and adapted by the learning process. Contrary to the situation in a dendrite, a neuron is only activated and fires when many or all of the excitatory dendrites are active. An excitatory dendrite has got a positive weight, while an inhibitory dendrite has got a negative weight and decreases the probability of a neuron to fire.

Similar to the axons, all excitatory dendrites of one neuron sum up to 1. The inhibitory dendrites of a neuron sum up to -1. Again, normalisation is needed after weight changes.

Input of a neuron:

$$I_j = \sum_q O_d(q) \cdot w_d(q) \tag{3}$$

where I_j is the input of the postsynaptic neuron j, $O_d(q)$ is the output of dendrite q and $w_d(q)$ is the weight of dendrite q.

Change of neuron potential:

$$P_j(t+1) = \delta \cdot P_j(t) + I_j \tag{4}$$

where the new neuron potential $P_j(t+1)$ is calculated from the potential of the last time step t, $P_j(t)$, and the current contribution by the neuron input I_j. δ is a constant between 0 and 1 for recovering the resting potential (which is 0 in this case) with time. The fact that δ will never bring the potential exactly to the resting potential, is not very important but can be avoided with a total reset when reaching a small range around the resting value.

If the neuron potential reaches a certain threshold θ_j, the neuron fires and resets its potential to its resting state. In contrast to similar neuron models that are for example summarised by Katic [8], a refractory period is not implemented here.

The whole computation process for a neuron is shown in figure 1.

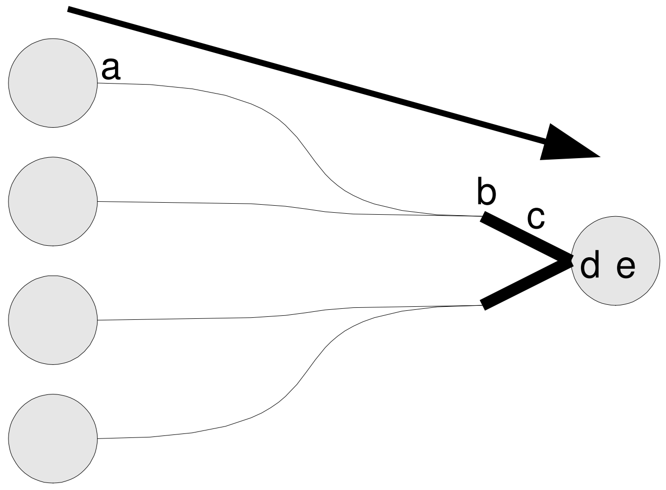

Fig. 1. A spike is produced when the presynaptic neuron fires and is sent to a dendrite (a). The dendrite sums up the weighted spikes (b, equation 1) and calculates its output (c, equation 2). The postsynaptic neuron sums up the weighted output of all of its dendrites (d, equation 3) and calculates its new potential (e, equation 4).

III. Learning from experience

A. Adapting connection weights

Learning from experience means to use past events to correct and optimise the behaviour in the future. For some applications an exact error is known and is to be minimised. In complex situations like robots acting in the real world there is no exact error value. As an alternative to an error value one or more reward values can be fed into the control system to represent the well-being of the robot, which is modified by positive ("that was good") or negative ("that was bad") feedback. The reward is to be maximised.

In the introduced model a single reward value is used that represents the general well-being of the robot. Its range is kept from -1, very bad, to 1, very good. The calculation of the reward can be varied. Usually it combines current measurements like fast movement, crashes or the energy level with residual effects of recent ones to avoid too rapid changes. For example, if the robot crashes into an object, the value for representing its well-being will be negative for a short while. A robot that moves away from an obstacle after crashing into it deserves an increase of the reward.

For the methods that are explained here it is assumed to have a meaningful global reward value $\Pi(t)$ at each time step t. This value can be added to a learning rule as an additional factor. Different authors, all of them using different neuron functions and learning functions, have shown that this surprisingly simple method can successfully be used to implement reinforcement learning in a neural network [1] [3] [7]. They do not need an external module that evaluates and changes the connections of the network after each processing step any more.

An example for adapting axons and dendrites using Activation Dependent Plasticity is shown below. Activation Dependent Plasticity is based on Hebb's ideas of strengthening connections that fire together [5]. As shown by Izhikevich reward can also be integrated into the more sophisticated Spike Time Dependent Plasticity (STDP) learning model [7].

Adaptation of an axon weight:

$$w_a(t+1) = w_a(t) + \eta_a \cdot \Pi(t) \cdot \phi_a \cdot O_d \tag{5}$$

where $w_a(t)$ and $w_a(t+1)$ are the axon weights before and after the adaptation. η_a is the learning factor for axons and O_d is the recent output of the connected dendrite. ϕ_a shows if the axon was active shortly before the postsynaptic neuron fired. For STDP this value can be the result of a function that takes into consideration the time when spikes were transmitted via the axon. In any case ϕ_a is a value from 0 to 1.

$\Pi(t)$ is the current reward. If it is positive, the strength of the axon will increase. A negative value will decrease the strength of the axon.

Adaptation of a dendrite weight:

$$w_d(t+1) = w_d(t) + \eta_d \cdot \Pi(t) \cdot \phi_d \tag{6}$$

where $w_d(t)$ and $w_d(t+1)$ are the dendrite weights before and after the adaptation. η_d is the learning factor for dendrites and ϕ_d is the activity value of the dendrite. The activation of the postsynaptic neuron is not given explicitly. This factor is always 1 in the proposed model, because the function is only called when the neuron has fired. ϕ_d is the equivalent of ϕ_a, but ϕ_d represents the activity of the dendrite.

With this function, active excitatory dendrites are strengthened and active inhibitory dendrites are weakened, if the current reward $\Pi(t)$ is positive. Otherwise excitatory dendrites are weakened and inhibitory dendrites are strengthened.

B. Delayed feedback

An important issue to consider when dealing with feedback from the environment and resulting rewards is delayed feedback. When weights are adapted and as discussed later also neurons are created based on the current reward, this may seem the wrong time at the first glance. Typically, the feedback is received *after* the responsible action is executed. Actually, the time difference can vary significantly.

There are two methods that can make the discussed methods efficient anyway:

- Feedback is not fed directly into the neural network but just changes the current reward value, which also contains residual effects of past feedback. This avoids fast changes of the reward value, which would be difficult to assign to a certain neuron activity pattern.
- In spiking neural networks, there is no single event that is responsible for an action, but a continuous flow of spikes. The input pattern, and hence the spiking pattern, usually does not change rapidly if a certain feedback is received. Figure 2 shows an example situation for this issue.

Of course there remain situations that make it difficult to assign the feedback correctly, for example if there is a big time difference between action and feedback, or if there are

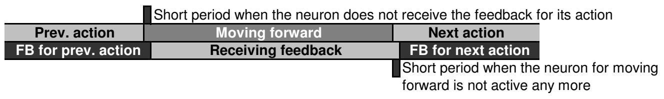
Short period when the neuron does not receive the feedback for its action
Prev. action
Moving forward
Next action
FB for prev. action
Receiving feedback
FB for next action
Short period when the neuron for moving forward is not active any more

Fig. 2. Example for delayed feedback

many competing actions or feedback values at the same time. However, even humans do not always arrive at the correct conclusions. They can deal with very complex relations but not with all.

In further work, a method will be introduced that may enable a robot to deal with delayed feedback in a better way, or may even be used to predict feedback. The method will be refined through further experimentation and research.

IV. Neural network growth

The neural network to be grown to create a robot control system initially has no links from the input to the output. The developer only defines the input neurons and how they are fed with signals to produce spikes, the output neurons and how their signals are used, and how the global reward is calculated. An example for how this is done is explained in section V.

If a non-input neuron has got no predecessors (neurons, which it gets spikes from), it creates a new excitatory dendrite and connects it to any neuron. In the experiments that are discussed later a predecessor is looked for that is positioned above the postsynaptic neuron in a layered network structure. Excitatory dendrites can also look for new presynaptic neurons every now and then and connect them with weak strength (low weights). That way a new connection does not abruptly change the established behaviour.

The method to grow new axons, which are the connections between presynaptic neurons and dendrites, can only be used for the action selection task. To classify different input patterns a method that creates new neurons is presented. Liu, Buller and Joachimczak have already shown that correlations between certain input patterns and a certain reward can be stored by creating new neurons [9] [10].

In the model proposed here, if the current reward is positive, a neuron that was active recently should be active again in similar situations, because, if a certain action was responsible for positive reward, it may be successful again. In section III delayed feedback was discussed. To avoid wrong correlations between feedback and neuron activity, a neuron will only create a connection to a new neuron in the following way, if it was active for some time already:

- All axons with enough influence on a neuron that was active before receiving positive feedback are redirected to a new neuron. The influence depends on the axon weights and the recent activity of the presynaptic neurons.
- The redirected connections are no longer just axons to one dendrite but are all connected to their own dendrite at the new neuron. This stores the combination of input signals.
- The old axons need not be removed completely, but most of their strength will be moved to a new axon that is connected to the new neuron.
- The process, which is illustrated in figure 3, is repeated for all dendrites of a neuron.

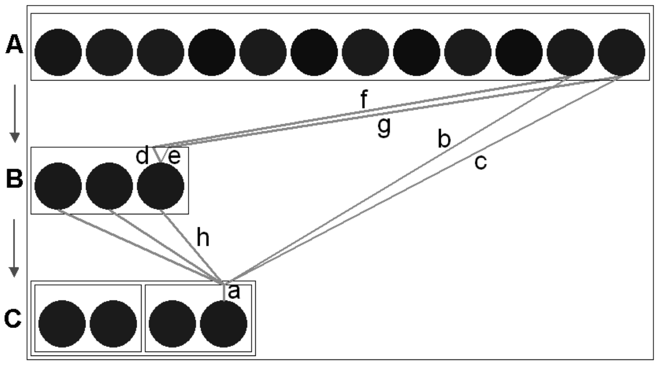

Fig. 3. The excitatory dendrite a is connected to two neurons of the input layer A via the axons b and c. Both were active when there was a significant positive reward. A new neuron was created in the hidden layer B that connects the same input neurons by two dendrites (d, e) and one axon for both dendrites (f, g). Then the new neuron was connected to dendrite a (axon h) of the neuron in the output layer C.

For a negative reward the process of creating a new neuron is similar, but the new neuron is not connected by a new axon but by a new inhibitory dendrite. In the future a similar input pattern will then inhibit the neuron that was active before receiving negative feedback. Bad actions will be suppressed that way. A new neuron that inhibits another one is shown in figure 4.

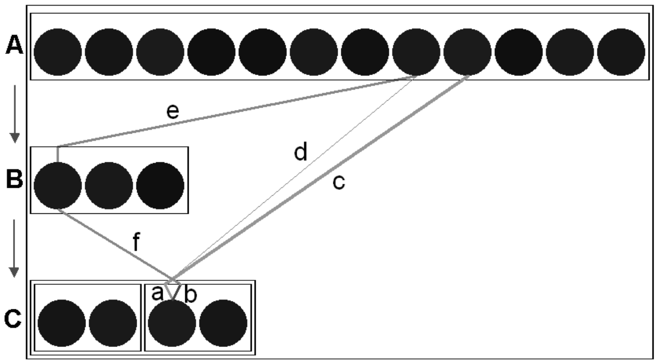

Fig. 4. Axon d, which connects an input neuron to the excitatory dendrite a, was active when there was a significant negative reward. Axon c was inactive. A new neuron was created with a connection to the active input neuron (e). For the output neuron a new inhibitory dendrite was created (b) and connected to the new neuron by axon f.

V. Experiments

A. Setup

The methods for growing a spiking neural network from a minimal initial network were tested with a simulation of a Pioneer Peoplebot which moves using differential steering, as depicted in figure 5. The initial neural network consists of 12 input neurons, 4 output neurons, and no connections as indicated by layers A and C in figures 3 and 4.

The input neurons are fed by values from 6 sonar sensors: 4 in the front and 2 in the rear as shown in figure 5. The distance value is converted in a way that one input neuron fires often for big distances and a second one fires often for small distances.

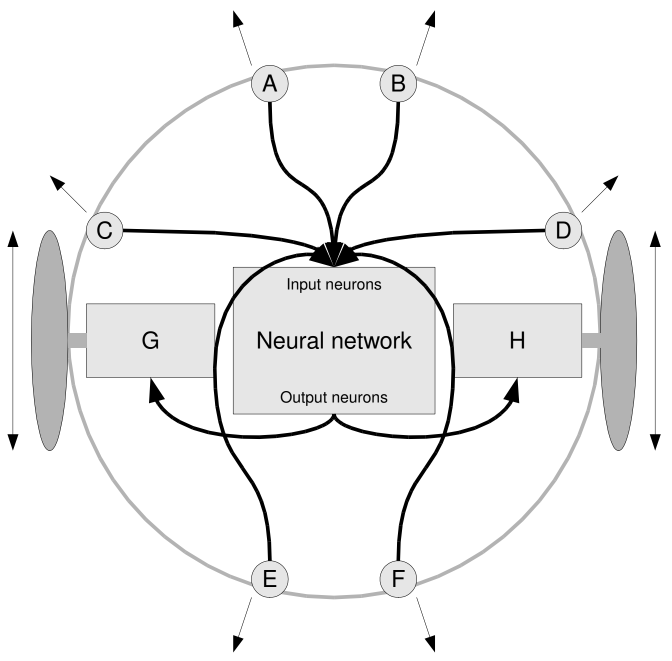

Fig. 5. Robot interface with sonar sensors A to F and motors G and H

The more frequently the first output neuron fires, the faster the left motor tries to turn forward. The more frequently the second output neuron fires, the faster the same motor tries to turn backward. The final speed is calculated by the difference between both neurons. The right motor is driven in a similar way by the two remaining output neurons.

The mechanisms discussed earlier were used for the robot to learn to wander around randomly in a simulated office while not crashing into obstacles. The global reward value is updated at each time step. When a bumper of the robot fires the reward is decreased significantly. Otherwise the reward is increased the farther the robot goes straight forward. Backward movement increases the robot's reward only if the reward was negative before (to reward recovering behaviour), but decreases the robot's reward in other cases.

The neurons are stored in a nested layer structure. The top layer contains the input neurons and the bottom layer contains the output neurons. New layers can be created in-between for new neurons. Currently, the network is evaluated as a strict feed forward network, which means there are no connections to the same layer (for example no local inhibition) or upwards (no recurrent connections).

For all experiments Activation Dependent Plasticity is used. That means actions are selected based on co-activation of certain neurons without considering the exact spike times. This is sufficient for experiments without competing actions that can be executed synchronously and without input patterns where the temporal course is relevant. Accordingly, also the learning and growing mechanisms are not based on exact spike times to make evaluation easier. Advantages of spike time dependent processes have for example been investigated by Izhikevich [6] or van Leeuwen [11]. It is possible to include some of those processes in more sophisticated future experiments.

B. Results

Initial experiments using the novel neural network growth model showed that certain circumstances can produce "neuron chains" (see figure 6). A new neuron that stores an input pattern that seems to be responsible for a certain reward is created. If that assignment is correct, the new neuron will again generate a reason to produce another new neuron because its output can also be assigned to a certain reward.

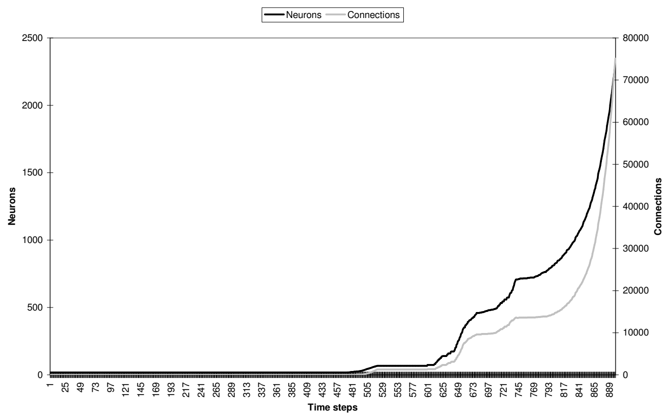

Fig. 6. The number of neurons and connections is growing so fast that the time for one step increases enormously. One step is finished after calculating the function of all neurons of the network once.

It is not possible to create a neuron only, if a combination of input signals is stored, because also single active connections can deliver important information. If that connection is not redirected to a new neuron, it may later be redirected to another neuron, but the input pattern memorised in that other neuron may be less useful than memorising an input pattern coming from a single connection.

A first successful measure to avoid chains of neurons that are actually responsible for the same input-output mapping was to consider the age of the connections (see figure 7). Young axons that seem to be responsible for a lot of reward will not lead to a new neuron. Finding the best circumstances for creating new neurons will be an important matter for further research.

With this later adaptation the robot was able to learn to wander around and to turn away from obstacles. Figure 8 shows an example run in which the robot perceived more reward when its experience increased. Table I shows some results of a test sample of 50 simulation runs, each run starting with the initial neural network without connections and stopping after 20000 time steps. The speed values are measured in internal units of the simulation. The number of

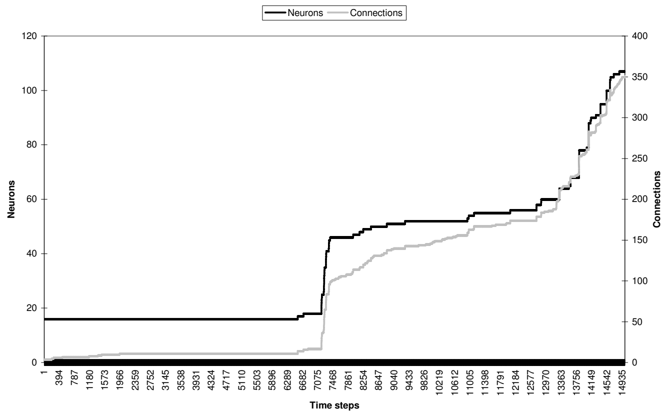

Fig. 7. The robot started at the same place under the same conditions as with figure 6. The mechanisms for growing the neural network were the same. The only difference was that a connection was not considered for the growth algorithm if it was younger than 6000 time steps.

inhibitory axons is ident to the number of inhibitory dendrites in all cases.

There is some potential for improving the methods for the robot to learn to recover if it crashes into an object. The different parameters of the neural network have to be adjusted and tested to render the strengths and weaknesses of the proposed robot control system more precisely.

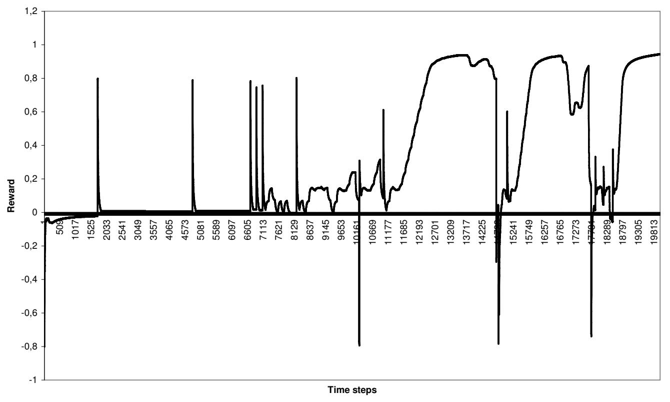

Fig. 8. At the beginning the robot did not perceive very much reward. After some random movements the robot learned how to increase positive reward. Smaller reward at later stages shows that the robot slowed down near obstacles. The negative amplitudes show that not all obstacles could be avoided.

TABLE I
RESULTS OF 50 SIMULATION RUNS

	Minimum	Maximum	Average
Total reward	-164.96	7412.83	3160.72
Average reward	-0.01	0.37	0.16
Maximum speed	395.00	1303.00	970.38
Average speed	4.09	388.24	190.48
Crashes	0.00	16.00	3.36
Neurons	16.00	32.00	20.62
Excitatory axons	14.00	126.00	34.46
Excitatory dendrites	4.00	96.00	18.98
Inhibitory axons	4.00	6.00	4.12

VI. CONCLUSIONS

We have presented our successful tests for a novel model capable of growing an artificial neural network that keeps the effort to design a control system at a minimum. With this methodology, the main effort to design a system controller is to define the inputs, the outputs and the mechanism to quantify a perception of "well-being" feedback depending on the performance of the system. In addition, this methods make it possible to simply add or remove any sensors or actuators, and the neural network will adapt to the new situation autonomously and automatically.

We have shown that a robot controller can be created autonomously without the need for various stages such as learning phase, execution phase, and feedback evaluation phase. The complete process is integrated in a single stage robust methodology for the creation of the neural network; which is capable of learning from experience in a continuous way when running.

A consequence of the fact that all processes except the calculation of the reward are executed in the neurons is a very efficient system if the neurons can be executed in parallel; making a very fast practical alternative for creating an artificial intelligence system.

VII. FURTHER WORK

The different parameters that define the speed of adapting connection weights and the way of creating new neurons and connections have to be investigated further to evaluate our novel methodology for creating controllers for concurrent tasks. These investigations will lead us to find an elaborate but still very basic "artificial brain" model that enables a system to achieve a sophisticated level compared to other artificial intelligence models by learning from experience efficiently.

When the basic methods are investigated in detail, some extensions can be added like Spike Time Dependent Plasticity or a feedback prediction mechanism. Initial ideas for both enhancements were discussed in this paper. Those improvements would help the controlled systems to deal with more complex situations, especially when timing considerations are important.

Also an improvement of the network architecture can help to increase the control capabilities. For example, local inhibition can increase the contrast between different input patterns and make the controller act faster and more accurately. Recurrent connections may also be important for contrasting issues, but they may also keep the behaviour of a system more stable, which means it does not swing between different actions too often. Recurrent connections may bring even more advantages with them, but they can also make a neural network very difficult to analyse and evaluate.

As mentioned in section III assigning delayed feedback more efficiently or even predicting feedback will be an interesting research issue for future work. The idea is that a neuron that receives positive or negative reward very often when it is active will probably receive the same reward also in the future. Predicting reward could actually be one reason for producing reward. This earlier reward may now be correlated to the activity of another neuron. That neuron could again produce reward when predicting it. By the recursive process reward could potentially be predicted progressively earlier.

REFERENCES

[1] E. Daucé and F. Henry. Hebbian learning in large recurrent neural networks. Technical report, Movement and Perception Lab, Marseille, 2006.

[2] D. Elizondo, E. Fiesler, and J. Korczak. Non-ontogenetic sparse neural networks. In *International Conference on Neural Networks 1995, IEEE*, volume 26, pages 290–295, 1995.

[3] R. V. Florian. A reinforcement learning algorithm for spiking neural networks. In *Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 299–306, 2005.

[4] G. Gómez, M. Lungarella, P. Eggenberger Hotz, K. Matsushita, and R. Pfeifer. Simulating development in a real robot: On the concurrent increase of sensory, motor, and neural complexity. In *Proceedings of the Fourth International Workshop on Epigenetic Robotics*, pages 119–122, 2004.

[5] D. O. Hebb. *The Organization of Behaviour: A Neuropsychological Approach*. John Wiley & Sons, New York, 1949.

[6] E. M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, 18:245–282, 2006.

[7] E. M. Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 10:1093–1102, 2007.

[8] D. Katic. Leaky-integrate-and-fire und spike response modell. Technical report, Institut für Technische Informatik, Universität Karlsruhe, 2006.

[9] J. Liu and A. Buller. Self-development of motor abilities resulting from the growth of a neural network reinforced by pleasure and tension. In *Proceedings of the 4th International Conference on Development and Learning 2005*, pages 121–125, 2005.

[10] J. Liu, A. Buller, and M. Joachimczak. Self-motivated learning agent: Skill-development in a growing network mediated by pleasure and tensions. *Transactions of the Institute of Systems, Control and Information Engineers*, 19(5):169–176, 2006.

[11] M. van Leeuwen. Spike timing dependent structural plasticity in a single model neuron. Master's thesis, Intelligent Systems Group, Institute for Information and Computing Sciences, Utrecht University, 2004.

[12] J. Vreeken. Spiking neural networks, an introduction. Technical report, Intelligent Systems Group, Institute for Information and Computing Sciences, Utrecht University, 2003.