Evolution of Robotic Neurocontrollers with Intrinsic Noise and their Behavior in Noisy Environments

Helmut A. Mayer

***Abstract*—We report on experiments with robotic neurocontrollers with intrinsic noise evolved for a peg pushing task. The specific controller of the simulated robot is a feed–forward network with noisy weights, i.e, the weight values are perturbed by additive, normal noise. The neurocontrollers are evolved in a noise–free environment, and the best–performing networks are then tested in noisy environments, where peg movement and sensor signals are afflicted by noise. We find that the internal (robotic brain) noise is beneficial in coping with external noise, especially, in the case of noisy sensors.**

I. Introduction

Our initial motivation to investigate the effects of noise affecting the internal computations in an *Artificial Neural Network* (ANN) is based on work by Ishiguro et al. (1999), where the authors present ANNs with simulated, artificial neuromodulators (ANMs). The ANMs may be released by neurons at certain events and change the weights of the network upon reception by other neurons. With the addition of ANMs a highly dynamic system emerges, which constantly alters its internal states. In order to generate meaningful behavior of a robot controlled by such a network, all the parameters related to ANMs are determined using evolutionary computation [1].

In the latter work it has been demonstrated that a network with ANMs is able to cope with noise from the environment, e.g., imprecise (noisy) movement of objects pushed by the robot, much better than an evolved network without neuromodulators. This even to the point that only a robot with the ANM network could perform a task in a noisy environment, when the network was evolved in a noise–free environment. The respective network without ANMs failed to control the robot correctly, and could no longer solve the task.

At first sight, it may be counter–intuitive that an instable system (introduced by the ANMs) should be better suited to cope with external noise. However, we came up with the hypothesis that the ANM network intrinsically experiences a form of noise due to its constantly changing internal states. This internal noise is always present (also during evolution in a noise–free environment), and thus the network is well–prepared for external noise. It should be stressed that diffusion and reception of ANMs are not random processes, as the former follow given (partially evolved) rules [1], but even in a small ANN the dynamics are such that a human observer would attribute it to intrinsic noise.

Helmut A. Mayer is with the Department of Computer Sciences, University of Salzburg, Austria (email: helmut@cosy.sbg.ac.at).

Consequently, in order to verify our hypothesis we investigate the effects of intrinsic noise in this paper. Our networks are not equipped with NMs, but the dynamics are introduced by adding noise to all the weights in a network (Section II). We evolve these networks with intrinsic noise for the same robotic task as presented in [1] (Section IV), and compare the performance of the noisy robotic controllers in noise–free and noisy environments.

Certainly, the deliberate addition of noise to a technical system is against most engineering principles, but it is a good model for *Biological Neural Networks* facing the *Stability–Plasticity* problem. Noise is ubiquitous in BNNs as the physical and chemical states of cells change constantly even without external excitation. Then, the question arises, how the remarkable stability of biological systems can be achieved, e.g., the precise movements of a human hand. Actually, there is evidence that intrinsic noise may contribute positively to stability in BNNs (Section I-A).

A. Related Work

As outlined above the work presented in [1] and [2] demonstrates that a robotic neurocontroller with ANMs performs more robustly in the presence of noise than a conventional neurocontroller (both with feed–forward structure). It should be noted that the term *noise* in the context of robots is used in its computer science flavor, where noise is a broad abstraction for imperfect information. In the specific task used in above work the robot should learn to push a peg (in)to a light source. When the evolved robot is confronted with a peg with a diameter different from the training phase, the change in diameter is also termed as noise. Most impressing was a video demonstration at the conference [1] where the ANM robot could solve the given task confronted with a peg (an additional weight has been excentrically put on the peg) it has never seen before.

Fernandez–Leon and Di Paolo (2008) presented experiments with noise introduced in recurrent neurocontrollers [3]. The simulated robot with two light sensors should learn to approach a light source, whose position is changed randomly during evolution. The neurocontroller is a *Continuous Time Recurrent Neural Network* (CTRNN) with two sensor neurons, two motor neurons, and two internal neurons. Internal noise is introduced by adding a uniformly distributed random number drawn from a pre–defined interval to the neurons' cell potential. Time constants and synaptic weights are

[1] IEEE Systems, Man, and Cybernetics,1999, Tokyo.

evolved using a real–valued encoding. Neurocontrollers generated with neural noise present during evolution exhibited higher robustness against neural noise in tests after evolution. Inspecting specific neurocontrollers the authors suggest that evolution finds neural systems resistant to the effects of bifurcation. Specifically, it is argued that the evolved systems either stay within the basin of an attractor, or experience bifurcations to attractors, which are functionally similar. Also, it is mentioned "that evolution may find solutions for which noise is advantageous" [3].

II. The Noisy Neurocontroller

The neurocontroller being a sensori-motor network with noisy weights is sketched in Figure 1.

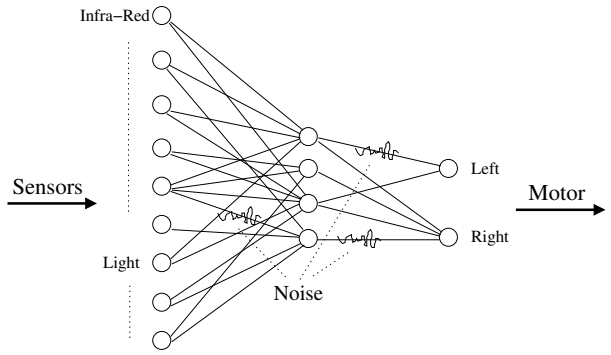

Fig. 1. The neurocontroller with exemplary noisy weights.

The only difference to a conventional feed–forward neurocontroller are the weights, which change its value due to noise with each update of sensor data. I.e., with every recall of the network, the weights are floating around their mean values. Note that in Figure 1 only a few weights are associated with noise as an example, but in the noisy neurocontroller all weights are affected by noise.

The noise model is based on additive normal noise, so the noisy weight values w' change according to $w' = \overline{w} + N(0, \sigma)$. The constant mean value $\overline{w}$ is determined by evolution, and the standard deviation $\sigma = \nu_w \ \overline{w}$ with ν_w being the weights' noise level, which is set by the experimenter so as to investigate varying degrees of noise. Note that this noise model assigns a different σ to each weight, which arguably models noise in BNNs, where more efficient synapses with larger amounts of neurotransmitters may generate more noise than those with smaller amounts. Additional biological evidence for this noise model can be gathered from (signal–dependent) motor noise, which increases approximately linearly with the amplitude of the motor command signal [4].

III. Genetic Encoding

In many approaches evolving neurocontrollers the network structure is pre–defined and the weights are evolved. Here, we also subject the number of hidden neurons, and the connectivity to evolutionary processes, hence network structure is evolved in combination with the weights. Weights, neurons, and connections are directly encoded using bit strings on three chromosomes, respectively.

This multi–chromosomal representation [5], technically, eases the addition of parameter sets (e.g., activation function parameters could be added on a fourth chromosome), and allows different encodings on different chromosomes (e.g., bits for neurons and real values for weights [6]). With the biologically motivated introduction of *Chromosome Shuffling* [5] recombination is performed on two levels. First, corresponding chromosomes of two parents are exchanged with a shuffle probability $p_s = 0.5$, then conventional crossover is performed within the single chromosomes (Figure 2).

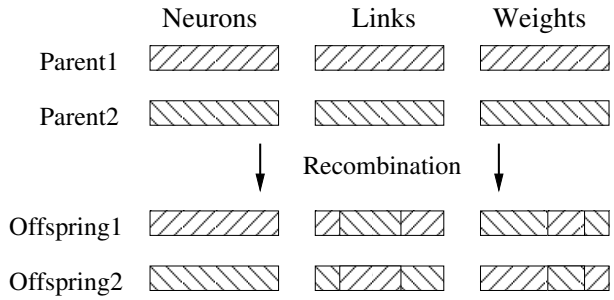

Fig. 2. Multi–chromosomal representation of neurocontroller with chromosome shuffling and 2–point crossover.

With chromosome shuffling complete parameter sets, e.g., the connectivity of the network, may be exchanged between parents, but still, crossover (and mutation) may introduce local alterations. The number of hidden neurons is set to a maximal value before the start of evolution, which is able to prune some or all hidden neurons. The number of input and output neurons is fixed. All the possible links (including short-cut connections from input to output) are encoded on the links chromosome. The presence or absence of a connection is encoded by a single bit. The weights are encoded in a similar fashion, however, each weight is encoded by 8 bits mapped to the real interval $[-10, 10]$. If a neuron is pruned, all associated links and weights become irrelevant.

IV. Experimental Setup

Experiments with the *Peg Pushing Task* are executed with a cycle time $t_c = 0.1s$ in the Java simulator framework SIMMA developed at our institution. The task and most parameter settings are taken from [2], as our initial intention was to investigate, whether the performance of neurocontrollers with neuromodulators could be explained by the intrinsic noise the modulators introduce.

The cylindrical robot shown in Figure 3 is equipped with six infra-red sensors (at positions $-90°$, $-45°$, $-10°$, $10°$, $45°$, $90°$ w.r.t. the heading direction of the robot) for detection of the peg, and three light sensors (at positions $-9°$, $0°$, and $9°$ w.r.t. heading) for detection of the light source [2].

The infra-red sensor signal s_i is given by

$$s_i = \frac{1}{1 + \exp\left(0.65\ d_p - 14\ \cos\left(1.3\ \alpha\right)\right)}, \qquad (1)$$

where d_p is the distance from the sensor position to the nearest point of the peg, and α is the angle between sensor direction and center of peg direction.

The light sensor signal s_l in case of the sensor being outside the light source is given by

$$s_l = (r_l/d_l)(1 - \frac{2\ \alpha}{\delta_a}), \qquad (2)$$

where r_l is the radius of the light source (spot), d_l is the distance from the sensor to the center of the light source, α is the angle between sensor direction and center of light source direction, and δ_a is the aperture angle of the sensor. The aperture angle ($18°$ for all light sensors) characterizes the angular sensitivity. If light is outside the angular range given by δ_a, the sensor does not detect anything. If the sensor is inside the light source, s_l becomes 1.

The basic structure of the neurocontroller is a *One–Hidden Layer* network with nine input neurons (no bias, no activation function (AF)) for the sensor signals, a maximum of four hidden neurons (with bias and AF), and two output (motor) neurons (no bias, with AF). The activation function is defined to be the logistic function. The (left and right) motor values in the range of $[0.0, 1.0]$ generated by the network are linearly transformed to the range $[-0.5, 0.5]$ so as to cover both motor directions (negative values are backward).

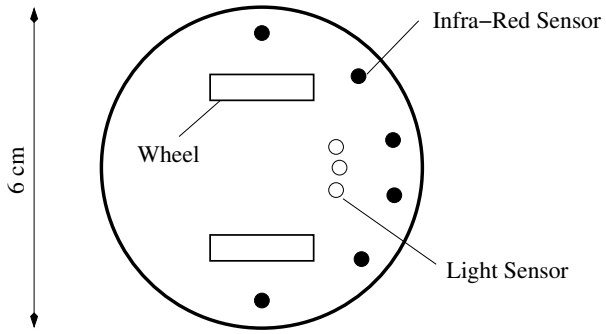

Fig. 3. The cylindrical robot.

A. The Peg Pushing Task

In this task the simulated robot is placed in a rectangular arena (1.05×0.70m) and should learn to move a peg into the center of a light source. The peg is a small circular disc with a radius $r = 2$ cm, and the light source is a circular spot ($r = 6$ cm) emitting light.

During evolution the fitness of each controller is evaluated in ten episodes of 30 seconds each. At the beginning of each episode the robot is placed at $(0.15, 0.35)$ (coordinate origin is at upper left corner of the arena) with random orientation. The peg at $(0.23, 0.35)$ is in contact with the robot, while the light source is stationary at $(0.85, 0.35)$ (Figure 4). The episode fitness (Equation 3) is given by the position of the peg as

$$f_e = (1 - \frac{d_{ps,T}}{d_{ps,0}})^2, \qquad (3)$$

with $d_{ps,T}$ being the distance between the centers of peg and light source at the end of the episode, and $d_{ps,0} = 0.62$ the very distance at the start of the episode. The final fitness of a neurocontroller, then is the mean of all episode fitness values. Note that the square function in Equation 3 (taken from [2]) does not properly discriminate between pegs pushed further away from the light source and those moved towards the light source, e.g., a fitness of $(0.2)^2$ (towards light) is equal to $(-0.2)^2$ (away from light). Still, it is sufficient to evolve well–performing robots.

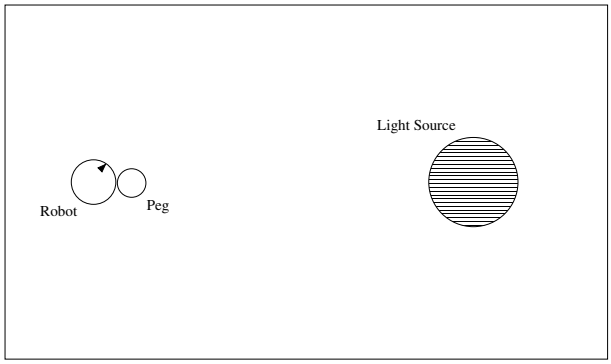

Fig. 4. The peg pushing task at the begin of an epsiode.

A single evolutionary run is set to 1,000 generations with a population size of 50 individuals (neurocontrollers). Each chromosome (Section III) experiences 2–point crossover with a crossover rate $p_c = 0.6$. Using binary tournament selection the parent generation is completely replaced by its children. In different runs we increase the noise level ν_w of the neurocontroller in the range of $[0.0 - 1.0]$ in steps of 0.05. Each run with a specific noise level is repeated 20 times. It is important to state that all evolutionary runs are executed in a noise–free environment, i.e., only the weights of the neurocontroller are affected by noise.

The best neurocontroller of each run is then tested in the presence of noisy pegs and noisy sensors with a noise level ν in the range of $[0.0 - 0.5]$ in steps of 0.05. The performance is measured exactly as during evolution (same setup in ten episodes), i.e., we again obtain fitness values, but this time for noisy neurocontrollers in noisy environments.

V. Results

First, we present test results for the best robots of each of 20 runs, when the evolved neurocontrollers with intrinsic noise face a noisy peg.

A. Peg Noise

Peg noise is introduced by adding random numbers drawn from a normal distribution $N(0, \sigma)$ to the components of the 2D peg position in each simulation step. The standard deviation $\sigma = \nu_p * r_p$, where ν_p is the peg noise in the range

of $[0, 0.5]$, and r_p is the radius of the peg. Though, this is not an exact physical approach, it generally simulates irregular peg movements due to imperfect robot-, peg-, and ground surface.

In Figure 5 the average test fitness of the best 20 neurocontrollers with (small) noise levels $\nu_w = 0.05 - 0.15$ (brain noise) is shown for various degrees of peg noise. The performance of the controllers without intrinsic noise ($\nu_w = 0.0$) serves as base line in the following figures [2].

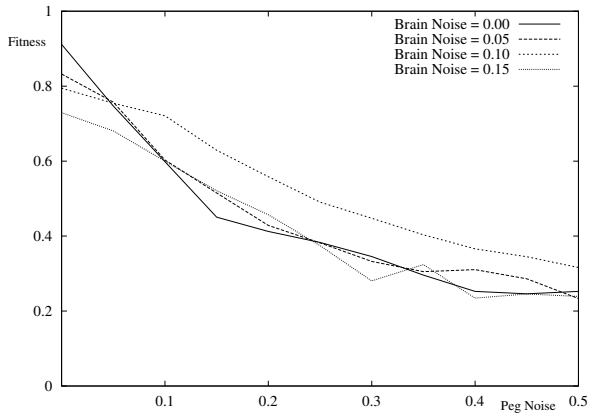

Fig. 5. Average test fitness of best neurocontrollers with little intrinsic noise in the presence of peg noise.

With the exception of the controllers with $\nu_w = 0.10$ the performance is practically identical to the noise–free reference controllers. The same picture appears with increasing brain noise, however the performance in case of a noiseless peg drops (Figure 6).

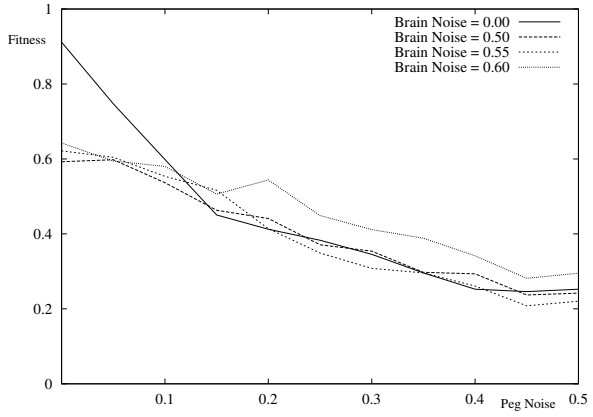

Fig. 6. Average test fitness of best neurocontrollers with medium intrinsic noise in the presence of peg noise.

The latter trend continues with even larger noise levels, as can be seen in Figure 7.

Even with large intrinsic noise levels the test fitness does not drop for a peg noise with $\nu_p \geq 0.15$, when compared to the noise-free reference controllers. Note that the networks with $\nu_w = 1.0$ on average are even better than the base case, however this may not be statistically significant. Considering that the latter noise level for a weight $w = 5.0$ makes it float somewhere between 0 and 10, this is a quite astonishing result. Even more so are those for sensor noise presented in the following section.

[2]Due to space requirements we only present a subset of all recorded results.

B. Sensor Noise

In case of noisy sensors all nine sensors (six infra–red and three ambient light sensors) of the robot are afflicted by noise. The noise–free sensor signal s is disturbed by additive normal noise according to $s' = s + N(0, \sigma)$ with $\sigma = \nu_s\ s$. Again, σ is dependent on the current sensor value s and a noise level ν_s generating the noisy sensor signal s'.

In Figure 8 the behavior of the best evolved robots with sensor noise and small intrinsic noise levels is depicted. Basically, fitness improves with noise in the neurocontroller, when compared to the base case without intrinsic noise. Though, increasing sensor noise degrades the robot's fitness, the degradation is alleviated by intrinsic noise.

A similar picture can be observed with medium (Figure 9) and heavy intrinsic noise (Figure 10), but at these higher noise levels the robot is nearly immune to sensor noise, as degradation is rather small. In fact, the fitness of the robotic brains with more noise is higher at a sensor noise level of 0.5 than those with less intrinsic noise. Loosely spoken, sensor noise is absorbed by intrinsic noise. E.g., with a brain noise (level) of 1.0 the robots with noise–free sensors have a mean fitness of 0.56, while the same robots with a sensor noise level of 0.5 average a fitness of 0.49 (Figure 10). Hence, the performance drops only slightly, when all the sensor signals fluctuate roughly in the range of $\pm 50\%$.

At low levels of sensor noise the robots with less intrinsic noise exhibit better fitness, as all neurocontrollers have been evolved using noise–free sensors. In simpler words, if there is no sensor noise, the intrinsic noise cannot be compensated (and vice versa).

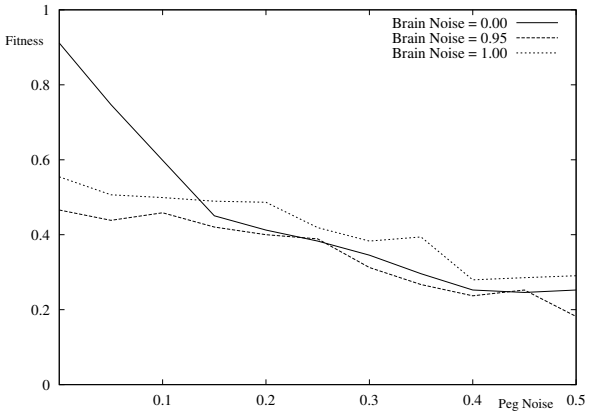

Fig. 7. Average test fitness of best neurocontrollers with heavy intrinsic noise in the presence of peg noise.

C. Robot Behavior

When studying the behavior of the robot using the best evolved neurocontrollers one can observe similarities and differences among the various robots. Though, robot, peg, and light have fixed (and identical) start positions throughout evolution, most evolved robots can cope with random start positions of all three objects in the arena. This is very likely based on the fact that during pushing the robot often loses the peg, hence it has to re–orientate in order to push the peg again in the right direction.

Basically, all robots exhibit three different modes of action, namely, i) locating the peg by rotating and moving towards the peg as soon as it is detected (locating action), ii) pushing the peg towards the light source (pushing action), and iii) circling around the light source once the peg is inside the light source (protecting action). Actually, only the pushing action is directly assessed by the fitness function (Equation 3), while locating is the consequence of occasional peg loss, and protecting is a by–product, which cannot be explained in terms of fitness. As soon as the peg is inside the light source, the only thing the robot should learn is to not touch the peg, again. Thus, moving away from the light source would also be a valid option, but none of all the observed robots chose this opportunity.

Various techniques for peg pushing emerged in different evolutionary runs. Without intrinsic noise some robots try to push the peg in a rather straight manner towards the light source. Though, this seems to be very efficient, it often results in peg loss, hence the robot has to turn around and reposition itself with respect to peg and light. A different pushing style is created by robots with rather irregular movements. These robots push the peg a short distance to the left, and then to the right (or vice versa), which in sum leads to a rather straight movement of the peg while reducing the risk of losing control of the peg.

The latter technique is essentially used by all robots with intrinsic noise, which in general leads to "shaky" movements. However, these irregular movements are used in a favorable way to push the peg efficiently. As a consequence, even robots with heavy brain noise are able to move the peg on a rather straight line, though their movements would suggest else. This kind of pushing action also helps in dealing with peg and sensor noise (also for robots without intrinsic noise).

Still, the most interesting result of above experiments is the huge performance difference between peg and sensor noise. With peg noise the performance decreases comparably for all levels of brain nose, while the latter seems to make the robots immune to sensor noise. This is even more remarkable, as with peg noise only the peg is a source of noise, but with sensor noise all nine sensors are noisy. A possible explanation to that phenomenon is that sensor noise yields a blurred image of the scene (all sensors are equally noisy), while peg noise produces a distorted one. Sensors close to the peg fully transmit the noisy peg moves, while the more distant sensors produce nearly the same signal regardless of the noise level of the peg. In this same metaphor brain noise also generates a blurred image, but cannot deskew a distorted image. In other words brain noise prepares the robot in dealing with a blurred image, as it is not really relevant, if the blurred image stems from internal or external noise.

VI. Summary

We have presented experiments with robotic neurocontrollers evolved for a task, where the robot should push a peg (in)to a light source. The specific interest in this work was the investigation of the effects of noise inside the neurocontrollers. In a simulator we evolved neurocontrollers with varying degrees of intrinsic noise in a noise–free environment, i.e., only the weights of the network were made to be noisy. After evolution the controllers have been tested in noisy environments by adding noise to the peg movements or to all the nine sensor signals. With peg noise the noisy neurocontrollers showed only slight differences to their noise–free counterparts, hence intrinsic noise did not degrade performance.

When confronted with sensor noise the noisy neurocontrollers clearly outperformed the robotic brains without

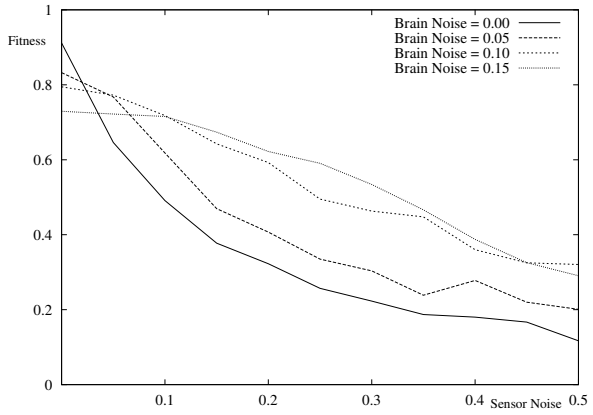

Fig. 8. Average test fitness of best neurocontrollers with little intrinsic noise in the presence of sensor noise.

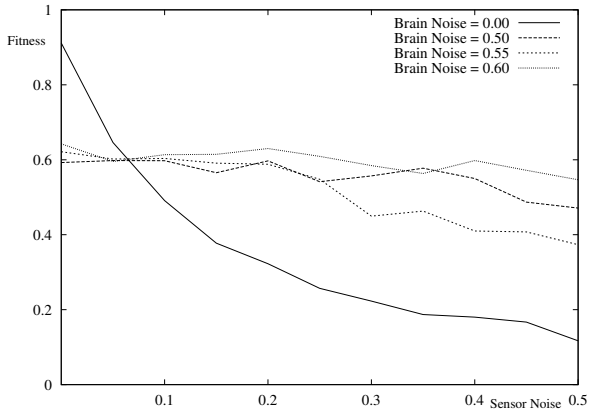

Fig. 9. Average test fitness of best neurocontrollers with medium intrinsic noise in the presence of sensor noise.

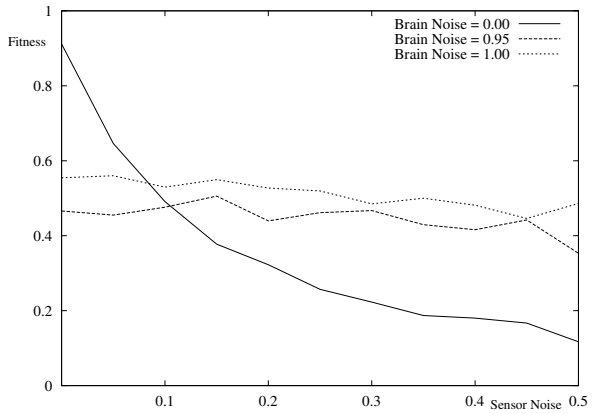

Fig. 10. Average test fitness of best neurocontrollers with heavy intrinsic noise in the presence of sensor noise.

internal noise. From these results we believe that noise in a neurocontroller prepares the robot for imprecise information in general, as it constantly has to cope with the internal noise. Consequently, the robot can deal with external noise more easily, as there is no fundamental difference between external and internal noise, i.e., the robot does not even "know", where the noise comes from. If these arguments also apply to biological neural networks, the inevitable noise in biological systems may contribute positively to the processing of noisy sensory input.

In future work we would like to intensify our collaborations with neurobiologists to potentially improve the simulation of the noise sources. Among further research topics are the effects of motor noise and a theoretical analysis of noisy artificial neural networks.

REFERENCES

[1] A. Ishiguro, S. Tokura, T. Kondo, Y. Uchikawa, and P. Eggenberger, "Reduction of the Gap between Simulated and Real Environments in Evolutionary Robotics: A Dynamically–Rearranging Neural Network Approach," in *IEEE Systems, Man, and Cybernetics Conference*. IEEE, October 1999, pp. III – 239–244.

[2] P. Eggenberger, A. Ishiguro, S. Tokura, T. Kondo, and Y. Uchikawa, "Toward Seamless Transfer from Simulated to Real Worlds: A Dynamically–Rearranging Neural Network Approach," in *Advances in Robot Learning*. Springer Berlin/Heidelberg, 2000, vol. 1812, pp. 44–60.

[3] J. A. Fernandez-Leon and E. A. Di Paolo, "Neural Noise Induces the Evolution of Robust Behaviour by Avoiding Non-functional Bifurcations," in *10th International Conference on Simulation of Adaptive Behavior, LNCS 5040*. Berlin Heidelberg: Springer, July 2008, pp. 32–41.

[4] D. E. Meyer, R. A. Abrams, S. Kornblum, C. E. Wright, and J. E. K. Smith, "Optimality in Human Motor Performance: Ideal Control of Rapid Aimed Movements," *Psychological Review*, vol. 95, no. 3, pp. 340–370, 1988.

[5] H. A. Mayer and M. Spitzlinger, "Multi–Chromosomal Representations and Chromosome Shuffling in Evolutionary Algorithms," in *2003 Congress on Evolutionary Computation*. IEEE, December 2003, pp. 1145–1149.

[6] A. Mayer and H. A. Mayer, "Multi–Chromosomal Representations in Neuroevolution," in *Proceedings of the Second IASTED International Conference on Computational Intelligence*. ACTA Press, November 2006.