Incremental Online Evolution and Adaptation of Neural Networks for Robot Control in Dynamic Environments

Florian Schlachter†*, Christopher S. F. Schwarzer‡*, Serge Kernbach†, Nico K. Michiels‡, and Paul Levi†
†*Institute of Parallel and Distributed Systems*
University of Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
Email: {Florian.Schlachter, Serge.Kernbach, Paul.Levi}@ipvs.uni-stuttgart.de
‡*Institute for Evolution and Ecology*
University of Tübingen, Auf der Morgenstelle 28, 72076 Tübingen, Germany
Email: {Christopher.Schwarzer,Nico.Michiels}@uni-tuebingen.de
**These authors contributed equally.*

***Abstract*—Many approaches have been developed to tackle the design complexity of modern robotic systems by using evolutionary processes. Starting with an initial solution, the evolutionary process tries to adapt to a given scenario and in the end produces an improved solution. Previous work showed that incremental evolution, a stepwise increase in the scenario difficulty, can increase the success of evolutionary adaptation. In this work, we clearly confirm this effect in the context of online evolution of neural networks. The goal of our online evolutionary approach is to produce on average good, intermediate solutions while the system is adapting. We show that also the average performance of the continuous evaluations is increased by evolving first in a simple scenario and then transitioning to a more difficult scenario.**

***Keywords*-online evolution; neural networks; robotics;**

I. Introduction and Background

In evolutionary robotics, the design of the robot controllers is driven by bio-inspired approaches [1], [2], [3]. Many of them are evolved offline on an external computer. After optimizing the controllers for a certain task, the best controllers are deployed to the robots. For the evolution of robot control, neural networks play an important role hence to their close relationship to natural systems. In several approaches it has been shown, that the evolution of neural networks can be speed up, by structural evolution of the networks. One of the early works in this field is the Generalized Acquisition of Recurrent Links (GNARL)[4]. In this work, they developed algorithms for the evolution of neural networks with recurrent links. The networks are randomly initialized (random hidden neurons and links) and evaluated. Afterwards, fifty percent of the population are allowed to create offspring (two children) for the next generation and so on. In the NeuroEvolution of Augmenting Topologies (NEAT)[5] the structural evolution starts with empty neural networks and develops over time. They also introduced a cross-over mechanism based on historic information and showed mechanisms for innovation protection (speciation). The improvements to the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) [6] extend the algorithms with a generative encoding and inclusion of sensors and output geometries [7].

Since robots operate in real world, the environment and conditions are subject to continuous changes. Through interaction and disturbances by other robots, humans or changes in the environment, control structures or functions can be obsolete or improper for the current task and need further adoption. Especially, in dynamic scenarios, the requirements to fulfil a defined task (implicitly defined in the fitness function) are subject to changes. Often this changes are hard to predict and occur randomly. One way to deal with this is a continuous process of adaptation of the robot controller to fit to the environment and requirements. This process of adaptation has to be performed on the robots during runtime, since the necessary changes are not known in advance. So the robot needs to evaluate its performance and an integrated evolutionary engine drives the evolution and thus the online adaptation. Additionally, this process can be embedded into an incremental evolution. The advantages of incremental

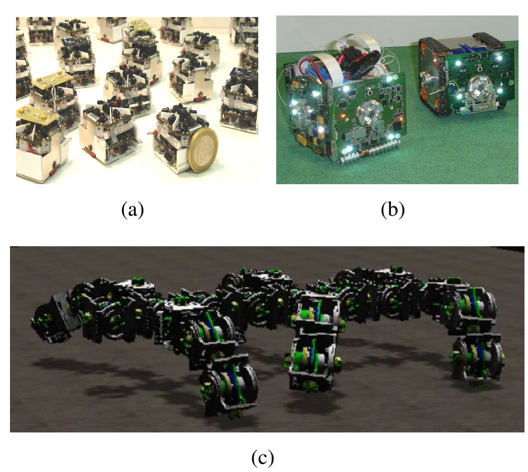


Figure 1. (a) Jasmine swarm (b) Prototypes of the Symbrion and Replicator Robots (c) Exemplary Organism in the Symbrion and Replicator Simulation.

ISBN: 978-1-61208-109-0

evolution were also proved by Gomez and Miikkulainen for a prey capture scenario [8] and by Barlow [9], where the complexity of the scenario for an aerial vehicle grows over time and the controller can develop step-wise.

The main goal of our work with evolutionary robotics is to create a system that is capable of adapting controllers online with the necessary complexity for controlling symbiotic robotic organisms [10]. This is a major part of the grand challenges of the Symbrion and Replicator projects (www.symbrion.eu and www.replicators.eu) both for swarm robots and artificial organisms like depicted in Figure 1.

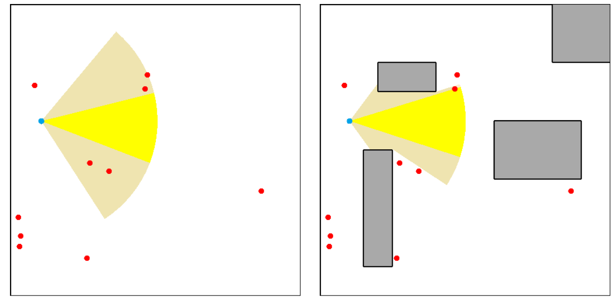
The paper is organized in the following way. In Section II we introduce our approach for evolutionary design of robot controllers and enlighten the different aspects of ongoing work and performed experiments. In the following Section III we present the results of the applied experiments and their impact to our work and finally we conclude the paper in Section IV.

II. Experimental Setup and Implementation

A. Arena and Experimental Setup

We evaluated our approach for simulated online evolution in a multi-agent simulation framework that uses a 2D physics engine to simulate a virtual environment. The robot is modelled as an agent in a two dimensional square arena with a size of 500x500 units that is surrounded by impassable walls. Within this arena, there are always 10 red points that symbolize energy sources for the robot, power cubes. These power cubes are static physical objects and pose an obstacle for the robot of the same size as the robot. If the robot is in close proximity of a power cube, it gains one reward point every 50 simulation ticks. After a power cube has dispensed 10 reward points, it is removed and a new, fully charged power cube is placed on a random position in the arena. The sequence of random positions for power cubes is the same for each run. Two arena setups are used; one completely empty and one with four large impassable boxes in a fixed configuration as seen in Figure 2. This particular configuration was chosen to provide a more complex scenario with more obstacles and a differently structured environment. In the empty arena the robot has a red power cube in sight most of the time and can trail a path from cube to cube without having to actively explore the arena. We also considered a maze layout with many thin wall segments scattered in the arena but this promoted simple wall following behaviours which was more simple to adapt to than the empty arena.

The simulated robot is equipped with seven virtual sensors: three sensors to detect the red power cubes in a field of vision with a range of 200 units; three distance sensors with a range of 100 units, in the same layout as the red sensors; one sensor that detects if a power cube is in immediate vicinity. The orientation and location of the red sensors is exemplary shown in Figure 2(a). The yellow cone is the



(a) (b)

Figure 2. The two arenas used in the simulated experiment runs: the empty arena **(a)** and the arena with boxes **(b)**. Black lines are impassable walls, red circles are simulated power cubes and the blue circle is the simulated robot. Shown is the initial configuration of the robot and red points which was used for each run. Also shown are the three fields of vision of the sensors of the robot which detect the power cubes.

middle sensor, while the light yellow coloured ranges are the left, respectively right sensor. The blue circle represents the robot, the red circles are collectible power resources. The range of the sensors is limited by the walls and obstacles in the arena 2(b). The robot has two actuators that simulate a differential drive with two wheels.

B. Neural Network and Control

The robot is controlled by an artificial neural network with recurrent connections and no restriction on network topology. This allows us to find good solutions regarding the complexity of the neural net. The decision of how many hidden layers, connections and neurons are necessary is shifted from human design to an evolutionary automated process. Doing so, the evolution of the neural net can find an optimal balance of the number of neurons and their connectivity. Design decisions made by humans can have no influence to the ability to adapt or can hinder the development of the neural nets by weak start configurations. The network itself performs one update step at each simulation time tick. It has eight input neurons (seven sensors plus bias neuron) and two output neurons. All inputs are mapped to values from 0 to 1, the output neurons provide values from -1 to 1. The values of the two output neuron values is transformed with Equation 1, which gives two positive values l' and r'. These modified actuator values are interpreted as a change to bearing b and linear velocity v as seen in Equations 2 and 3. Afterwards, the output is multiplied with a constant factor to scale the values to the simulation and set the velocity or change the bearing directly without simulation of inertia.

$$o' = \frac{o+1}{2} \tag{1}$$

$$b = r' - l' \tag{2}$$

$$v = r' + l' \tag{3}$$

The described actuator mapping smoothens the fitness landscape for a completely undifferentiated start network because output values of 0 produce a straight forward movement. Note that with this mapping, the network needs to output -1 on both output neurons to come to a full halt and it is impossible to drive backwards. Different actuator mappings had a large impact on the performance of the evolutionary process during preliminary experiments. This particular mapping was chosen as a compromise between a challenge for the evolutionary adaptation and to allow a smooth evolutionary start with an undifferentiated initial network.

C. Evolutionary Algorithm

For the evolutionary process, we use a genome that encodes the neural network in a structure similar to NEAT [5]. The genome is a list of connection genes and each gene encodes one neural link with source neuron id, destination neuron id and link weight. In the mutation operator, each gene changes its weight with probability 0.2 by applying a uniform random change from -0.2 to 0.2, capped in the range -1 to 1. Additionally, with a probability of 0.4, one structural change is made: Deleting a link, creating a new random link with weight 0 or creating a new hidden neuron by inserting it into an existing link. There is no mutation to delete hidden neurons, however hidden neurons are automatically removed if they are unconnected. For crossover, we are also faced with the problem of finding a suitable mechanism to avoid known problems of recombination of neural networks. The original NEAT approach [5] and likewise the rtNEAT extension [11] is not directly transferable due to a missing supervisor to track the innovations for crossover and due to the small number of robots for speciation and innovation protection.

For the evolutionary engine, we use an evolutionary algorithm based on the $(\mu+1)$ algorithm [12] with a random parent selection and elitism survivor selection scheme and no cross-over operators. The algorithm maintains a population of ten genomes. For each evaluation, one genome is uniform randomly picked to produce one mutated offspring which is evaluated next. After evaluation, the worst in the population is replaced if the evaluatee is better.

To evaluate the performance of the individual robot controller we tried to find an implicit fitness value. Since we can not create new robot offspring, the possibility to measure the performance by reproduction rate is limited. Alternatively, a virtual life energy or power resource can be used. Within the scenario the robots are able to collect power resources. Finally, the robots with a high rate of collected cubes have automatically a high fitness. This includes implicitly the ability of collision avoidance. The robots have to avoid obstacles and drive on optimal paths in order to keep a high movement speed. In case of collision or suboptimal paths, the robot is slowed down or fails to collect the resources.

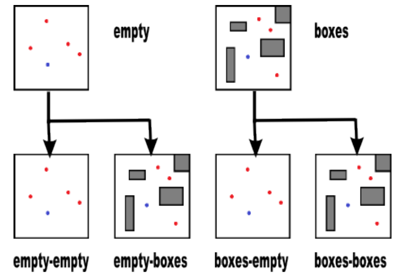


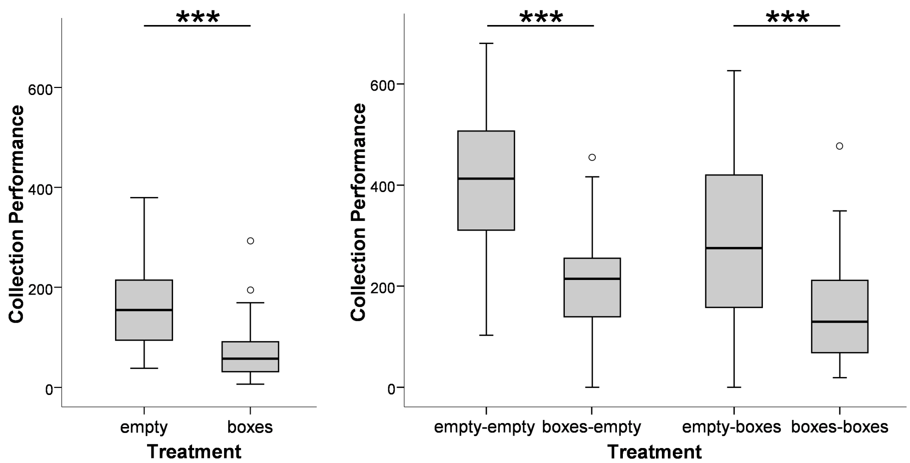
Figure 3. The six different treatments of the experimental setup. In the first two treatments, *empty* and *boxes*, a basic population of undifferentiated networks is evolved to adapt to their respective arena. In the second set of treatments the evolved populations are redeployed and evolved in the other arena (treatments *empty-boxes* and *boxes-empty*) or in the same arena again (treatments *empty-empty* and *boxes-boxes*).

D. Experiments

In one treatment, we let the robots evolve in an empty arena for 100 evaluations (*empty* treatment). Afterwards we placed these controllers in the same arena for another 100 evaluations (*empty-empty* treatment). Additionally, we placed the same controllers in the arena with obstacles (*empty-boxes* treatment). The motivation of changing the arena is to simulate unforeseen changes in the environment. A preevolved controller is suddenly faced with a new situation. In the empty arena, a controller implicitly avoids obstacles, as long as it can see any red power cube to follow. The chance to see power cubes is minimized in the second arena and the controller has to advance the ability for exploration. The fitness function was always the same. Each robot was awarded for collecting the power cubes. Possible collisions are implicitly punished by slowing down the robot.

The initial population of treatments *empty* and *boxes* is a genome for a perceptron without hidden neurons. There are links from each input neuron to each output neuron and each links' initial weight is 0. At each run, the robot is placed in the same starting position with 10 power cubes placed in the same initial configuration. Each run lasts 100 evaluations and each evaluation is done for 5000 simulation ticks. No changes to the arena and agent states is done in between the evaluations to simulate online evolution. Specifically, the robot remains in its position as well as the power cubes.

An overview of the experimental setup is given in Figure 3. After the 100^{th} evaluation in treatments *empty* and *boxes*, the final population of each run is stored. These evolved populations are used as starting populations for a second set of treatments. The evolved populations are put into a different arena in treatments *empty-boxes* and *boxes-empty* or put into the same arena again in treatments *empty-empty* and *boxes-boxes*. The runs in this second set of treatments last again 100 evaluations. For each treatment of the second



(a) (b)

Figure 4. The collection performance of different treatments. The middle bar marks the median, the box marks the lower and upper quartile and the whisker the minimum and maximum values of the 40 replicates. **(a)** Shown is the summed score of the last 10 evaluations of 100. The performance is lower in the arena with boxes (*Wilcoxon Rank Sums test* $n = 40$ $z = -5.2$ $p < 0.0001$) indicating that it is more difficult to collect points than in the empty arena. **(b)** The end performance after 100 more evaluations of the final populations of treatments *empty* and *boxes* in different arenas. The treatments that evolved first in the empty arena show a better final performance both in the empty arena (*Wilcoxon Rank Sums test* $n = 40$ $z = 5.6$ $p < 0.0001$) and in the box arena (*Wilcoxon Rank Sums test* $n = 40$ $z = -3.9$ $p < 0.0001$).

set one different, evolved population of the first set was used for each run. The evolved genomes were not mixed between populations and each evolved genome was only used once per treatment.

III. Results

The performance of the evolutionary process was measured by summing the collected score in a window of 10 consecutive evaluations. After the first set of treatments of simulated online evolution over 100 evaluations in the empty and boxes arenas, the performance of the last 10 evaluations is shown in Figure 4(a). The evolved controllers were able to collect significantly more power cubes in the empty arena than in the boxes arena. This shows that the robot collects power cubes slower in the arena with boxes. This arena is more difficult, likely because the robots' sensors are blocked by the boxes and because the robot has to manoeuvre more to drive around the boxes.

After the populations have evolved for 100 more evaluations in the second set of treatments, a general increase in collection performance is seen compared to the first set (Figure 4(b)). The evolutionary process did not fully adapt in the first 100 evaluations and the additional time allowed a further optimisation. Surprisingly, the treatments that were first in the empty arena perform better both in the same arena and in the different arena. It was expected that treatments perform better when they evolved the entire time in one arena rather than when the arena was switched in the middle. This can explain that treatment *empty-empty* performs better than *boxes-empty*. However, it is surprising that treatment *empty-boxes* performs better than treatment *boxes-boxes*. Generally, the arena where the population spent their first 100 evaluations in had a much larger impact on the final performance than the arena switch.

In Figure 5 the collection performance in time windows of 10 evaluations is shown over the course of both treatment sets. The values for the second treatment set are appended to the first treatment set to show the continuous development. There are small peaks in all treatments at evaluation 10 and 110 which must be an artefact of the starting phase of the runs. Presumably, in the random positions of the power cubes there are positions that are easier to collect and these are harvested first. In the initial configuration of power cubes there seems to be a high ratio of those "easy" cubes. After the initial phase, the number of "easy" cubes on the field is lower since they are continuously collected faster than the more difficult ones.

It can be seen in the graph that the performance is continuously increasing over time which shows the adaptive nature of the evolutionary process. The maximum score in one of the 10-evaluations windows is 703 in the empty arena and 517 in the boxes arena and thus we assume that the average performance will further increase after the 200 evaluations in our setup. In this graph it is of note that the *boxes* treatment seems almost stagnant and only after more evaluations in the *boxes-boxes* treatment a significant

 ISBN: 978-1-61208-109-0

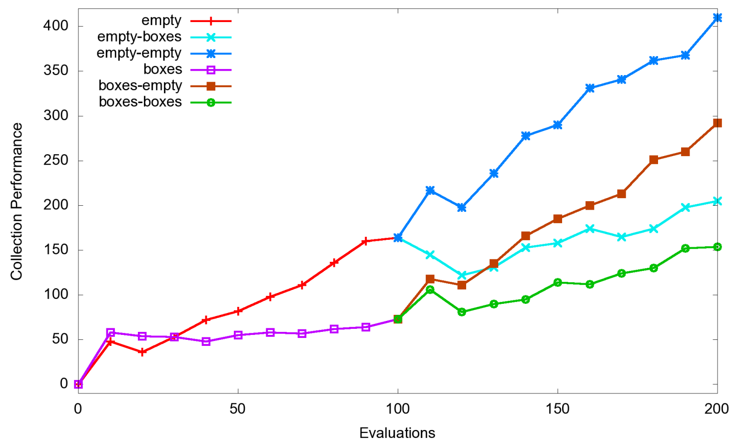


Figure 5. The development of the collection performance over time. Each data point is the summed collection score of a window of 10 evaluations, averaged over 40 runs. After 100 evaluations, the populations were stored and restarted on the same or a different arena. The peaks at 10 and 110 evaluations are an artefact from the initial placement of red points at the start of the runs. The treatment *empty-boxes* is able to maintain some of its advantage of the empty arena in the more difficult box arena. It performs better than the *boxes-boxes* treatment, which had spent more time evolving in this arena.

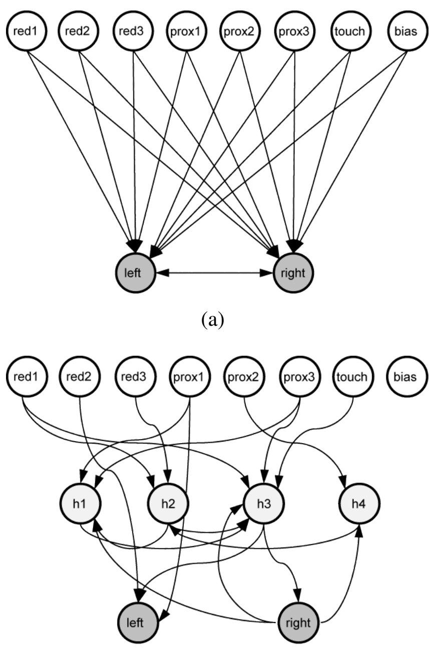
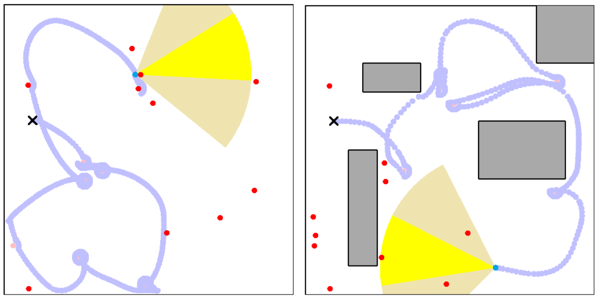


Figure 6. Two exemplary neural networks from the experiments. **(a)** The initial network with all input neurons connected to the output neurons but with a link weight of zero. **(b)** An evolved, successful neural net for the boxes arena.

upwards slope can be seen. This might explain the bad performance of the treatments that started in the boxes arena because the initial population of undifferentiated networks seems to be very unsuited to evolve efficiently in the boxes arena. In the empty arena on the other hand, the initial population evolves quickly as seen in the much steeper slope of the *empty* treatment. At the switching point of the second treatment set after 100 evaluations, the runs seem to quickly adapt to their new surroundings. The performance growth of the *empty-boxes* and *empty-empty* treatments, as well as *boxes-empty* and *boxes-boxes* treatments are almost the same. In particular, the *boxes-empty* treatment increases its performance faster after the switch from the boxes to the empty arena. Although it is difficult to see due to the aforementioned artefact peaks, the arena switch did not incur a large immediate reduction of performance. The performance of the *empty-boxes* treatment did drop after the switch, but it did not drop below that of the *boxes-boxes* treatment. It seems like the neural networks of the empty arena evolved faster and produced more flexible control structures. These networks had the plasticity to perform well or even better in a different arena compared to the population of networks that were “native” to this arena.

When we take a look at the evolving neural networks, we can clearly see, the structural grow of the networks. Figure 6 depicts the initial network and a exemplary neural network after 200 steps in the empty-box scenario. The top row are the three camera sensors for the red pixels (red1, red2. red3), the proximity sensors (prox1, prox2, prox3), the sensor for touching a food source and an additional bias neuron (not used by this net). The nodes h1, h2, h3, h4 are the evolved hidden neurons and left and right describe the motor output. In Figure 7(a) shows an exemplary run of a robot in the empty arena and Figure 7(b) a more advanced controller in the box arena. In both figures, the view of the sensors and the path of the robot is shown. The cross marks the starting point.

ISBN: 978-1-61208-109-0



(a) (b)

Figure 7. Paths of evolved successful controllers for the empty **(a)** and box **(b)** arenas. The cross marks the starting point. The duration of the trace is the same as the evaluation time of the experiment (5000 ticks). Note the tight circling around red points, which is the commonly evolved strategy to stay close to a power cube until it is completely harvested. Only very late controllers evolved that halted in front of the cubes.

IV. Conclusions

In this paper, we showed the feasibility and advantages of structural online evolution combined with a stepwise increase of the scenario difficulty. We showed ways for structural online and onboard evolution and performed experiments with promising success. It is obvious, that future more complex tasks need a big amount of hidden neurons and recurrent links. The proposed system gives a design tool and automatism at hand, to unburden the developers from the decision of structure and number of neurons.

Regarding incremental evolution, we showed that artificial evolution has different speeds of adaptation depending on the scenario and the initial population. With a given initial evolutionary population and a given target scenario there is a set of intermediate evolutionary scenarios with relaxed difficulty where evolutionary speed is higher than in the target scenario. As seen in our experiments, with the initial population of undifferentiated networks and the target scenario of the boxes arena, the fastest adaptation to the boxes arena was achieved by first evolving in the empty arena and later transitioning to the boxes arena. In this case, the empty arena acted like a relaxed scenario with reduced difficulty than the boxes arena. Skills and structures are quickly evolved in relaxed environments that still give an advantage in different and more difficult environments.

For future work, we want to extend the scenarios with additional robots and a non-supervised mechanism for crossover, so that evolved controllers can be transferred to less evolved robots. Even so, the focus shifts to the transition from robot swarms to artificial organisms and their actual control.

Acknowledgment

The authors would like to thank Nadine Timmermeyer from the University of Tübingen for her help. The "SYMBRION" project is funded by the European Commission within the work programme "Future and Emergent Technologies Proactive" under the grant agreement no. 216342. The "REPLICATOR" project is funded within the work programme "Cognitive Systems, Interaction, Robotics" under the grant agreement no. 216240. Additionally, we want to thank all members of projects for fruitful discussions.

References

[1] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press, 2000.

[2] D. Floreano and C. Mattiussi, *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press, 2008.

[3] D. Floreano and L. Keller, "Evolution of adaptive behaviour in robots by means of darwinian selection," *PLoS Biol*, vol. 8, no. 1, January 2010. [Online]. Available: http://dx.doi.org/10.1371/journal.pbio.1000292

[4] P. J. Angeline, G. M. Saunders, and J. P. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, January 1994.

[5] K. O. Stanley and R. Miikkulainen, "Evolving neural network through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[6] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artif. Life*, vol. 15, no. 2, pp. 185–212, 2009.

[7] D. B. D'Ambrosio and K. O. Stanley, "A novel generative encoding for exploiting neural network sensor and output geometry," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007, pp. 974–981.

[8] F. Gomez and R. Miikkulainen, "Incremental evolution of complex general behavior," *Adaptive Behavior*, vol. 5, pp. 5–317, 1996.

[9] G. J. Barlow, C. K. Oh, and E. Grant, "Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming," in *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS)*, Singapore, December 2004, pp. 688–693.

[10] P. Levi and S. Kernbach, Eds., *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Springer-Verlag, 2010.

[11] K. Stanley, B. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 6, pp. 653 – 668, dec. 2005.

[12] A. Eiben, E. Haasdijk, and N. Bredeche, "Embodied, online, on-board evolution for autonmous robotics," in *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, P. Levi and S. Kernbach, Eds. Springer-Verlag, 2010, pp. 367–388.

ISBN: 978-1-61208-109-0