Peter Dürr, Claudio Mattiussi,
and Dario Floreano
École Polytechnique Fédérale de Lausanne (EPFL),
SWITZERLAND

Genetic Representation and Evolvability of Modular Neural Controllers

Abstract–The manual design of con-
trol systems for robotic devices can be challenging. Methods
for the automatic synthesis of control systems, such as the evolution of
artificial neural networks, are thus widely used in the robotics community. However,
in many robotic tasks where multiple interdependent control problems have to be solved
simultaneously, the performance of conventional neuroevolution techniques declines. In this
paper, we identify interference between the adaptation of different parts of the control system as
one of the key challenges in the evolutionary synthesis of artificial neural networks. As modular net-
work architectures have been shown to reduce the effects of such interference, we propose a novel,
implicit modular genetic representation that allows the evolutionary algorithm to automatically
shape modular network topologies. Our experiments with plastic neural networks in a
simple maze learning task indicate that adding a modular genetic representation to a
state-of-the-art implicit neuroevolution method leads to better algorithm
performance and increases the robustness of evolved solutions
against detrimental mutations.

I. Introduction

One of the main determinants of the performance of a robotic system is the control system. Not surprisingly then, the manual design of control systems can be very challenging. An alternative to manual design is the use of Evolutionary Algorithms (EAs, [1]) for the design of robots and their control systems [2], [3]. In this context, Artificial Neural Networks (ANNs) are often used as control architectures because they can approximate arbitrary mappings from sensory inputs to actuator outputs [4], [5].

Researchers have studied the evolution of ANNs in a variety of examples such as wheeled and legged robots (e.g., [6], [7], [8], [9], [10]), swimming robots (e.g., [11], [12]), and flying robots

(e.g., [13], [14], [15]). Evolved control architectures are often surprisingly different from hand-designed systems and sometimes more efficient in terms of computational requirements [16].

However, in tasks that require solving several control problems simultaneously, there can be interference between parts of the evolving neural architecture. As mutations simultaneously affect multiple parts of the control system, the probability of changing one part of an evolved solution whithout disrupting others can be very low.

In [17], we have shown a case of such interference where a neural network was evolved to control a wheeled robot moving in a T-maze (see Fig. 1). In this experiment, the performance of the robot depended on both the ability to navigate within the maze and the ability to adapt its behavior to the changing location of a reward token (see Section III-A). A first

The proposed encoding allows the evolution of arbitrary network topologies with modular mapping from genotype to phenotype.

set of experiments revealed that it was difficult to find optimal solutions to this problem because mutations affecting the ability of the robot to adapt to the changing environment had a high probability of negatively affecting the collision-avoidance behavior. In other words, evolvability was impaired by interference between the two subproblems. However, when the effect of mutations was constrained by a hand-designed modular network structure separating the two subproblems, evolution was able to find optimal neural controllers consistently. In the literature there are other examples where researchers relied on manual decomposition of the control problem by subdividing the control architectures into a predefined number of modules, based on *a priori* knowledge of the problem (e.g., [18], [19], [20]). However, it is not always obvious how control problems should be decomposed into tractable sub-problems [21].

It has thus been suggested to allow the evolutionary algorithm to automatically shape the modularity of evolving networks. Based on the hypothesis that a modular genotype-phenotype map (see e.g., [22], [23]) may be largely responsible for the evolvability of complex biological organisms [24], researchers have implemented genetic representations for ANNs which allowed for modular mapping from genotype to phenotype (i.e. a genotype-phenotype map that translates modular genomes into modular neural networks). Experiments in a robotic cleaning task [25], [26], [27] and a visual discrimination problem [28], [29] indicated that the solutions found using modular mapping performed as well as hand-designed modular network structures and were better than non-modular networks. A limitation of these experiments is that they relied on simple, non-plastic, feed-forward network architectures and direct encoding of the network topology and weights, which restricts scalability.
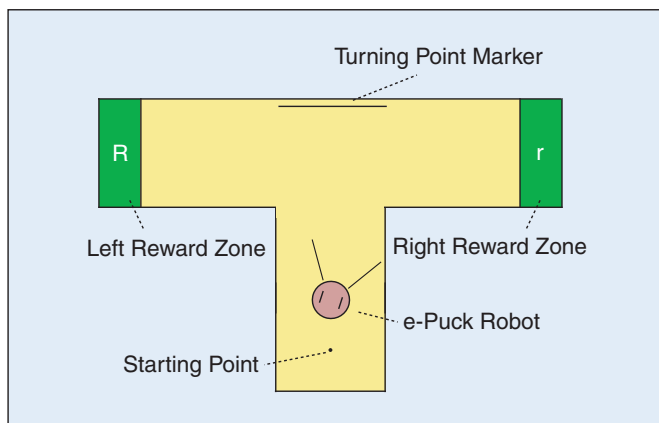
Other researchers have shown that co-evolving populations of modules [30] can lead to automatic problem decomposition in various problem domains (see e.g., [31], [32], [33], [34], [35]). However, this approach requires estimating the contribution of the subcomponents to the performance of the whole system, which can be difficult [36], [37].

In this paper, we propose an implicit encoding that allows the evolution of arbitrary network topologies with modular mapping from genotype to phenotype. We compare the performance of the proposed representation to a representation without modular mapping and analyze the evolved solutions in the robotic T-maze experiment presented in [17]. In the following we will first address the genetic representations of artificial neural networks and introduce our implicit modular representation.

II. Genetic Representation

The simplest approach to the genetic representation of ANNs is the explicit encoding of all neurons, synaptic connections and parameters of the network as a concatenated list of characters that form the genome of the evolving individuals. This approach, known as *direct encoding* [4], has the disadvantage that the length of the genome grows rapidly with the size of the network, which affects evolvability [38].

As an alternative to direct encodings, it has been suggested to mimic the developmental process of biological cells and to encode the parameters of a developmental process which constructs the network (see e.g., [39]). In [40], a developmental encoding called Cellular Encoding was successfully used to synthesize a gait controller for a six-legged walking robot. However, while developmental encodings allow for a compact representation of large networks, the design of genetic operators is difficult, because small changes in the developmental process tend to have large effects on the resulting networks. Comparison of performance in a pole-balancing problem [41], [42], [43] revealed that Cellular Encoding was outperformed by direct encodings.

More recently, it has been suggested to use representations inspired by the principles of genetic regulatory networks [44]. In genetic regulatory networks, the interaction between genes is not explicitly encoded in the genome, but follows implicitly from the physical and chemical environment in which the genome is immersed. Based on an abstraction of this process, the synaptic connections between neurons of an ANN can be encoded implicitly in an artificial genome (see Section II-A). This *implicit encoding*, which has been used not only in the domain of neuroevolution [45], [46], [47], but also in the design of electronic circuits [48] and the reverse engineering of genetic regulatory networks [49], [50], shares to some degree the capability of developmental encoding to provide a compact representation, while at the same time allowing for simple, biologically inspired mutation and recombination operators [38].

In [45], an implicit encoding called Analog Genetic Encoding (AGE, [48]) was evaluated in the same pole-balancing problem mentioned above. The performance of AGE was equal

to the performance of the best direct encoding, and it outperformed Cellular Encoding.

However, the implicit representations that have been suggested so far do not feature a modular mapping from genotype to phenotype. In the following, we introduce a novel implicit encoding based on AGE which allows a modular genotype-phenotype mapping. While we focus on the synthesis of neural controllers here, the presented approach applies to the synthesis of any type of analog network [38].

A. Analog Genetic Encoding

Analog Genetic Encoding consists of a digital genome in the form of a string g of characters drawn from a finite genetic alphabet (here, the 26 characters of the ASCII uppercase alphabet were used). In order to decode the network, the genome is scanned for short strings, so-called tokens, which separate coding parts from non-coding parts of the genome (see Fig. 2). Different tokens are associated with different types of neurons. For each coding part indicated by a particular, predefined token in the genome, the corresponding type of neuron is inserted into the network.

The network topology can then be constructed based on the sequences in the coding parts. The strength of the synaptic connection between two neurons w_{12} is determined by an interaction map $I(s_1, s_2)$, which takes sequences s_1 and s_2 from the coding parts associated with the two neurons as arguments and produces a numeric value for the synaptic weight $w_{12} = I(s_1, s_2)$. In the experiments reported below, an interaction map based on logarithmic mapping of the local alignment score [51] of the two sequences s_1 and s_2 was used (for more details see [48], [45]).

In summary, the decoding involves the identification of neurons in the genome (indicated by the respective tokens) and the subsequent application of the interaction map to compute the synaptic weights between all neurons in the network (note that the synaptic weight between two neurons can be zero).

As mutation operators, we used operators that affect individual characters of the genome with a certain probability (character substitution, character insertion, character deletion), operators which affect randomly selected fragments of genome (genome fragment duplication, genome fragment transposition, and genome fragment

The implementation of a modular mapping from genotype to phenotype in the case of an implicit encoding such as AGE is straight-forward.

deletion) and an operator that inserts a neuron of random type with a random coding sequence (neuron insertion, see also [45]).

B. Modular Mapping

The implementation of a modular mapping from genotype to phenotype in the case of an implicit encoding such as AGE is straight-forward. Instead of storing one string of characters g per individual, the modular genome $g^* = g_1, ..., g_n$ is composed of n strings, the so-called modules of the genome. At the time of decoding, the modules of the genome are treated individually and are separately decoded into modules of the neural network (see Fig. 3). As a consequence, there are no synaptic connections between neurons encoded in different modules of the genome. A particular type of input neurons or
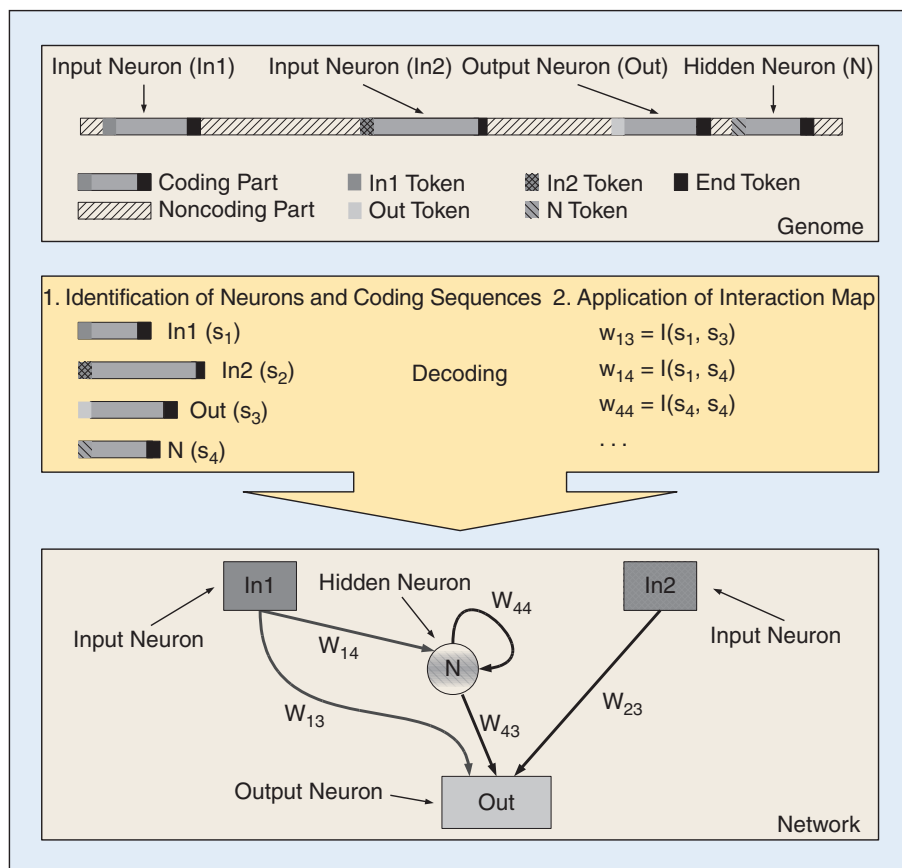

FIGURE 2 Implicit representation of a neural network. In the implicit representation used here, the genome contains coding and non-coding parts. The coding parts, which comprise a string of genetic characters delimited by tokens, represent the neurons of the network. Each type of neuron in the network is associated with a different, predefined token. The strength of the synaptic connections between neurons is encoded implicitly in the coding sequences of the two respective neurons by means of an interaction map I which takes sequences from the coding parts associated to the two neurons as arguments. For example, the synaptic weight between input neuron In1 and output neuron Out $w_{13} = I(s_1, s_3)$ is a function of the string s_1 associated with In1 and the string s_3 associated with Out.

to alter the number of modules present in the genome, which potentially allows for the evolutionary algorithm to adapt the number of genetic modules to the requirements of the problem.

output neurons can appear in more than one module of the genome. In the case of input neurons, the respective input signals are simply fed to all modules which contain the corresponding input neuron. In the case of multiple output neurons of the same type, the output neurons Out_i of all n modules present in the genome are directly connected to the respective output of the neural network with a weight of $w_{Out_i Out} = 1/n$.

In addition to the mutation operators discussed above (which affect the individual modules), module duplication and module deletion can affect randomly selected modules with a certain probability. This provides a possibility

III. Experimental Method

A. T-Maze Setup

In the experiments reported below, we studied the evolution of a wheeled robot in a T-maze (see Fig. 1). In this setup (see also [17]), an e-puck robot ([52], see Fig. 4) had to collect rewards located at both ends of the maze. At the end of each arm of the T-maze, there was either a high reward (with a value of 10) or a low reward (with a value of 1). Starting from the bottom of the T-maze and facing in the direction of the turning point with a random angle $\gamma_s \in [-\pi/4, \pi/4]$, the robot had to collect one of the rewards by driving into the end-zone of either the left or the right arm. When the robot reached either end zone, it was awarded with the respective reward and repositioned at the bottom of the maze. The robot was controlled by an evolved neural network (see Fig. 5) that was connected to the sensory inputs and a motor output. Data from the two infrared distance sensors in the front of the robot were merged and normalized into one sensory input $IR = IR1 - IR2 \in [-1, 1]$. A turning point marker was placed in the middle of the front wall of maze, and a camera sensor $Cam \in \{0, 1\}$ indicated if the turning-point marker was in the field of view of the robot's linear camera. The robot was also equipped with a floor-color detector to sense the ends of the maze $End \in \{0, 1\}$ and the size of the collected reward at the maze end $Reward \in [0, 1]$. The motor output $Out \in [-1, 1]$ was used to control the two motors of the robot. If the absolute value of the output $|Out|$ was smaller than a threshold value $Out_t = 0.3$ the robot drove straight ahead. If the output was smaller than the threshold value ($Out < -Out_t$) the robot rotated counterclockwise and if the output was larger than the threshold ($Out > Out_t$), the robot rotated clockwise. The robot was evaluated in four independent trials of limited duration $t_{end} = 300s$ (from the start, an optimal robot needed around
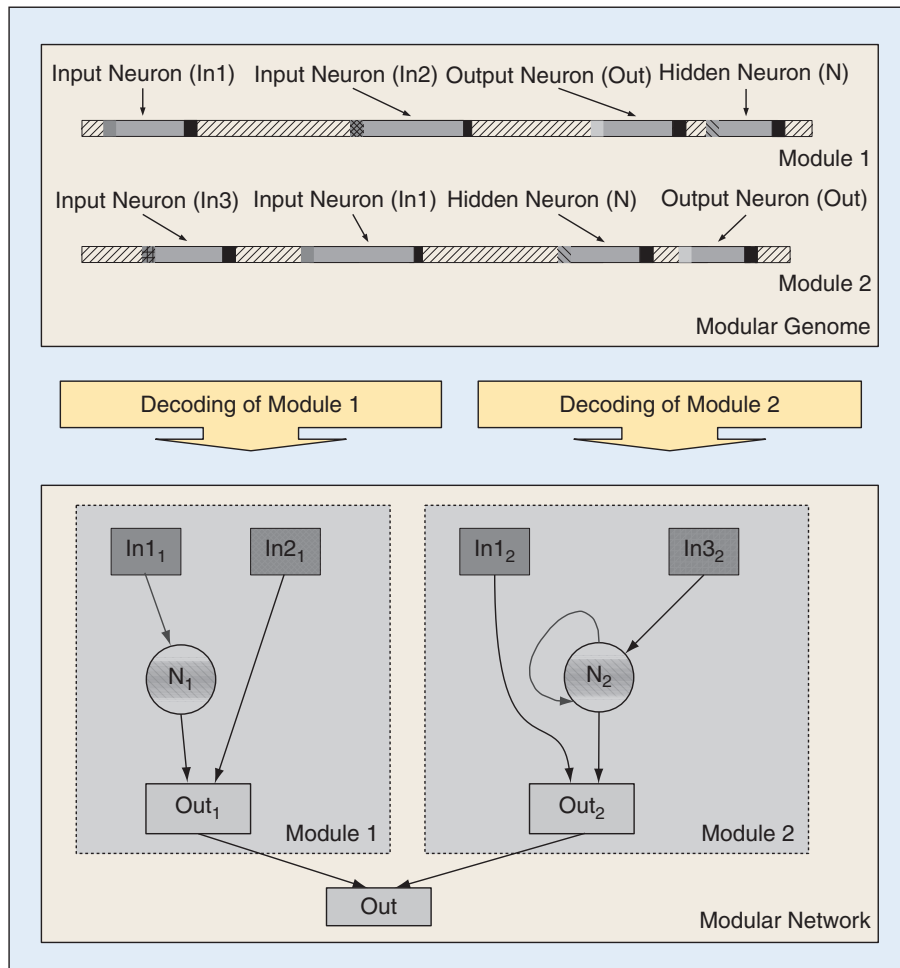

FIGURE 3 Decoding of a modular genome into a modular network. In this example, a network with three inputs and one output is encoded in a genome with two separate modules. Module 1 contains an input neuron $In1_1$, an input neuron $In2_1$, one hidden neuron N_1 and an output neuron Out_1. Module 2 contains an input neuron $In1_2$, an input neuron $In3_2$, a hidden neuron N_2, and an output neuron Out_2. The decoded network is split into two separate modules. The neurons encoded in the modules of the genome are assigned to the respective modules of the decoded network. Connections between the neurons of each module of the network are implicitly decoded from the respective module of the genome. At the evaluation of the network, the output is calculated as the average of the outputs of all output neurons from all modules of the network (in this example $Out = (Out_1 + Out_2)/2$).

4.3s to reach either end of the maze). In each trial, the location of the rewards was swapped after a random amount of time, uniformly drawn from the interval $t_{swap} \in [125s, 175s]$. At the beginning of the first and third trial, the high reward was located in the right end of the maze and the low reward was located in the left end of the maze. At the beginning of the second and fourth trial, the high reward was located in the left end of the maze and the low reward was located in the right end of the maze. The evolutionary experiments were conducted using a physics-based simulation of the robot and its environment [53].

B. Neural Network

At every point in time t, the activation x_i of each neuron i was computed as

$$x_i(t) = \sum w_{ij} \sigma(x_j(t-1)), \quad (1)$$

where w_{ij} is the synaptic weight between neuron j and neuron i and $\sigma(x)$ is a sigmoid activation function

$$\sigma(x) = \frac{1}{1 + e^{-\gamma x}} \quad (2)$$

with slope parameter γ.

While it is possible to evolve recurrent neural networks which display learning behavior without plastic synapses (see e.g., [54], [55]), it is generally believed that synaptic plasticity is one of the basic principles that allows for learning and memory in biological nervous systems [56]. Different models of synaptic plasticity have been shown to contribute to the evolvability of artificial neural networks in tasks where learning or memory is required (see e.g., [5]). In the experiments described below, we used a heterosynaptic plasticity model [46], [17], [57]. This model is based on neuromodulation of the synaptic plasticity between two neurons by a special type of neuron, the so-called modulatory neuron, which connects to the post-synaptic neuron (see Fig. 6). The synaptic weight $w_{12}(t)$ between a pre-synaptic neuron N1 with the activation $x(t)$ and a post-synaptic neuron N2 with activation $y(t)$ at time t is

$$w_{12}(t) = w_{12}(t-\delta t) + \Delta w_{12}(t), \quad (3)$$

where the initial weight $w_{12}(t=0)$ is derived from the decoding of the genome and the plastic change $\Delta w_{12}(t)$ for a time step δt is

$$\Delta w_{12}(t) = m_{12}(t)\eta[Ax(t)y(t) + Bx(t) + Cy(t) + D]. \quad (4)$$

The neuromodulatory factor m_{xy} was computed as the output of the modulatory neuron, weighted by a neuromodulatory weight w_m and η, A, B, C, and D were constant factors encoded with the modulatory neuron.
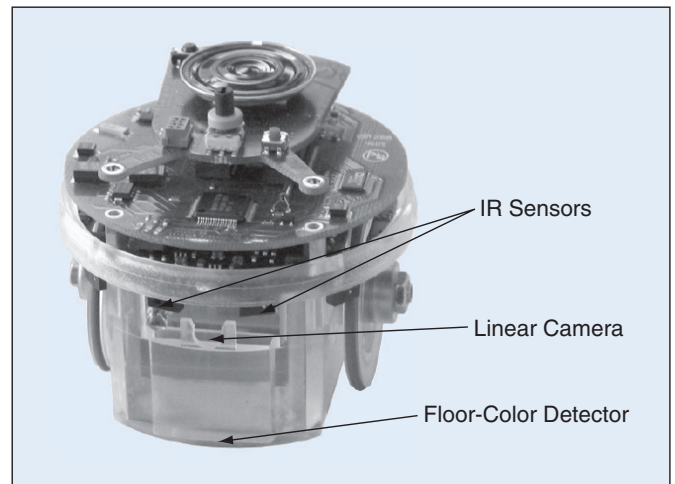

FIGURE 4 The e-puck robot was equipped with two infrared sensors in the front, a linear camera, and a floor color detector which fed the inputs of an evolved artificial neural network (see Fig. 5). Image reprinted from [52].
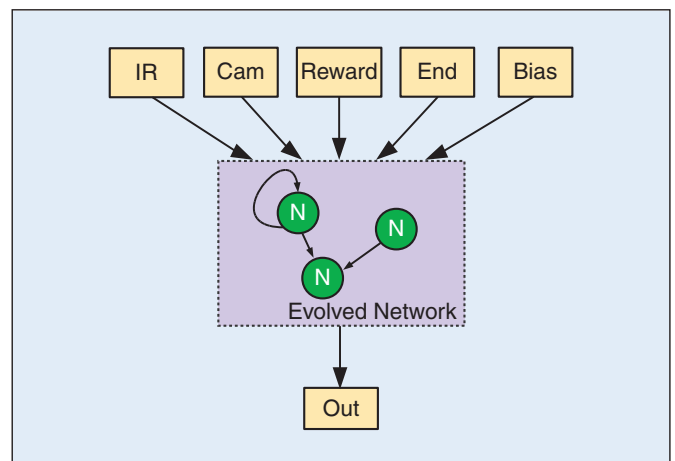

FIGURE 5 The evolved artificial neural network was connected to inputs from the infrared sensors (IR), the linear camera (Cam), a reward sensor (Reward), a maze-end sensor (End), and a constant bias unit (Bias). The output of the network was used to control the motors of the robot.
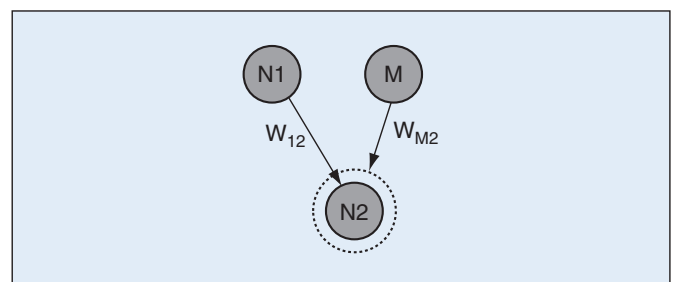

FIGURE 6 The heterosynaptic plasticity model used in the experiments. The plasticity of the synaptic weight $w_{12}(t)$ between a pre-synaptic neuron N1 and a post-synaptic neuron N2 is modulated by the weighted activity of a modulatory neuron M connected to the post-synaptic neuron [46].

C. Evolutionary Algorithm

A simple, generational genetic algorithm with tournament selection was used [1]. The neural networks were

The use of a modular representation results in a significant evolutionary advantage with respect to a non-modular representation.

genetically encoded using AGE both with non-modular mapping and with the proposed modular mapping (as described in Section II). Numerical parameters of the neurons were encoded using a self-adaptive representation for real-valued parameters based on variable length strings called Center of Mass Encoding (CoME, see [58]) with search intervals of $\gamma \in [0.5, 5]$, $\eta \in [-10, 10]$, $\{A, B, C, D\} \in [-1, 1]$. The parameters of the genetic algorithm were identical for both the experiments with the non-modular mapping and with the modular mapping (see Table 1) and have been chosen heuristically, according to earlier experiments [48], [45]. In both sets of experiments, the initial population consisted of 1,000 randomly generated networks with one genetic module.

TABLE 1 The parameters of the genetic algorithm and encoding used in the experiments.

|	
POPULATION SIZE	1,000
TOURNAMENT SIZE	2
ELITE SIZE	1
RECOMBINATION PROBABILITY	0.1
CHARACTER SUBSTITUTION PROBABILITY	0.001
CHARACTER INSERTION PROBABILITY	0.001
CHARACTER DELETION PROBABILITY	0.001
GENOME FRAGMENT DUPLICATION PROBABILITY	0.01
GENOME FRAGMENT TRANSPOSITION PROBABILITY	0.01
GENOME FRAGMENT DELETION PROBABILITY	0.01
MODULE DUPLICATION PROBABILITY	0.01
MODULE DELETION PROBABILITY	0.01
GENOME DUPLICATION PROBABILITY	0.001
NEURON INSERTION PROBABILITY	0.02
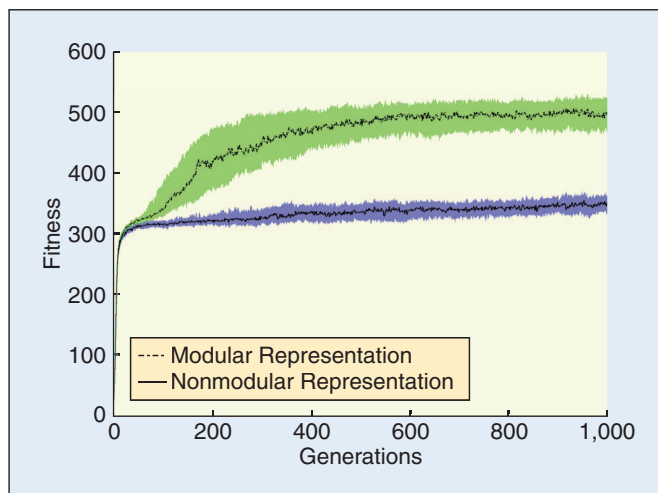
FIGURE 7 Median of the average population fitness of the 40 independent replicates with the non-modular representation and with the modular representation. Shaded areas range from the lower quartile to the upper quartile of the respective distribution.

We conducted 40 independent replicates over 1,000 generations both with the non–modular representation and with the modular representation. Fitness was defined as the sum of the values of all collected rewards divided by the number of trials.

IV. Results

A. Non-Modular vs. Modular Representation

The average population fitness in the 40 replicates per condition indicates that the use of a modular representation results in a significant evolutionary advantage with respect to a non-modular representation (see Fig. 7). At the last generation, the best solutions found in the 40 replicates with the non-modular representation had significantly higher fitness than the best solutions found by the replicates with the non–modular representation (Wilcoxon ranksum test, $p < 0.0001$, see Fig. 8).

The networks in the initial population consisted of only one module. The modular mapping allowed the algorithm to adaptively change the number of modular subnetworks. As can be observed in Fig. 9, the number of modules initially increased up to an average of three modules around generation 200 followed by a decrease to two modules. This is consistent with earlier observations that AGE implicitly tends to converge to networks with a minimal number of elements after an initial exploration phase characterized by the generation of larger networks [60], [45]. In the final generation, the number of genetic modules which contained
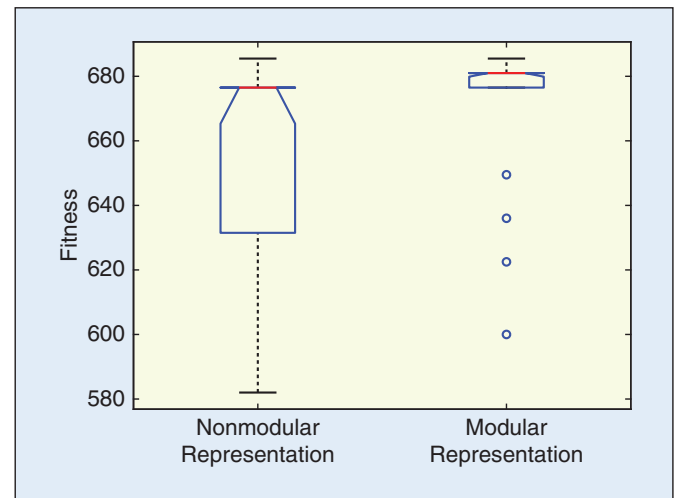

FIGURE 8 Boxplot of the fitness of the best solutions found by the 40 independent replicates with the non-modular representation and with the modular representation at the last generation. The red line in each box is the median, the borders of the box represent the upper and the lower quartile. The whiskers outside the box represent the minimum and maximum values obtained, except when there are outliers which are shown as small circles. We define outliers as data points which differ more than 1.5 times the interquartile range from the border of the box. The notches permit the assessment of the significance of the differences of the medians. When the notches of two boxes do not overlap, the corresponding medians are significantly different at (approximately) the 95% confidence level [59].

at least one input neuron and an output neuron n^* was on average $\overline{n^*} = 1.95 \pm 0.20$ (note that the hand-designed network in [17] had two modules).

Closer examination revealed that the possibility of a modular separation was used in a large majority of the replicates in experiments with the modular mapping (80% of the resulting networks of the last generation had more than one module). The structure of the networks also indicated an association between network modules and subtasks of the control system. The robot had to avoid crashing into the walls of the maze, a problem which could be solved relying on the input from the infrared sensors. At the same time, the robot had to choose the left or the right arm of the maze and possibly adapt its strategy based on the size of the collected rewards. This task could be solved using inputs from the reward sensor, the maze end sensor and the camera. From the networks which were composed of multiple network modules, 75% had a module containing the infrared sensor input and a different module containing the reward sensor (see Fig. 10 for an example).

B. Effects of the Modular Representation

It has been hypothesized that the reason for the higher evolvability of modular representations is that they change the effects of mutations [29], [61]. In order to analyze the effects of mutations, we subjected the networks of the last generation to mutations and compared the fitness of 1,000 mutant networks to the fitness of the original networks. The difference in fitness can be used as a measure of robustness to mutations. Our results indicate that the networks evolved with modular mapping were more robust to mutations (see Fig. 11). The mean effect of a mutation on fitness was significantly smaller in the networks evolved with the modular mapping than in the networks evolved with the non–modular mapping (Wilcoxon ranksum test, $p < 0.0002$).

The networks evolved with the modular mapping displayed a significantly lower ratio of pleiotropic mutations than the networks evolved with the non-modular mapping.

In order to quantify the incidence of pleiotropic mutations, i.e. mutations which simultaneously affect different characters, we measured two behavioral characters during fitness
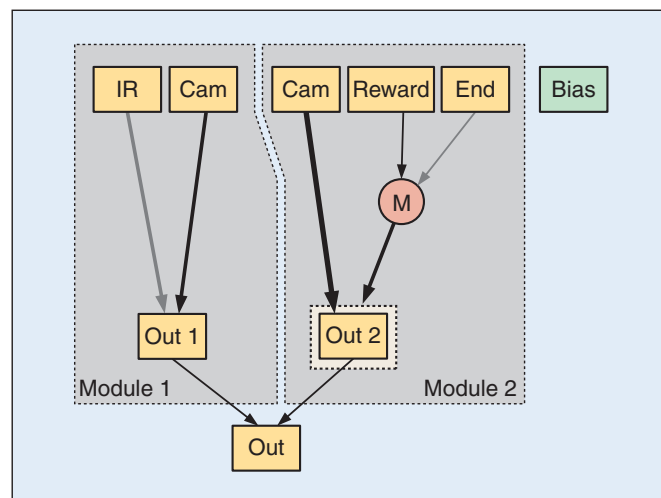

FIGURE 10 An evolved network with the modular mapping. Black arrows indicate excitatory synapses, gray arrows indicate inhibitory synapses. The network is split into two modules. Module 1 does not have plastic synapses and implements a simple collision avoidance behavior with a tendency to turn to the right when facing the turning point marker. The synapse linking the camera input to the output of Module 2 is affected by plasticity gated by a modulatory neuron M which allows adapting network behavior depending on the signals from reward and maze end sensors. The bias unit is not connected to the network.
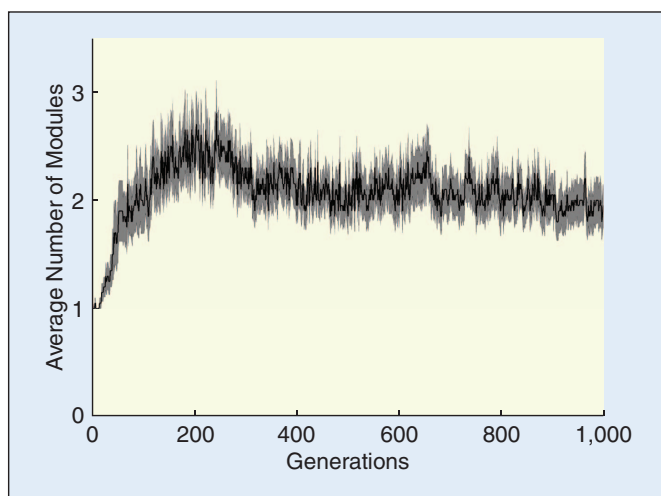

FIGURE 9 Average number of network modules used in the solutions of the 40 independent replicates with the modular mapping. The shaded area indicates the standard error of the distribution. Starting from one module in the initial population, the number of modules could change under the influence of mutations and selection.
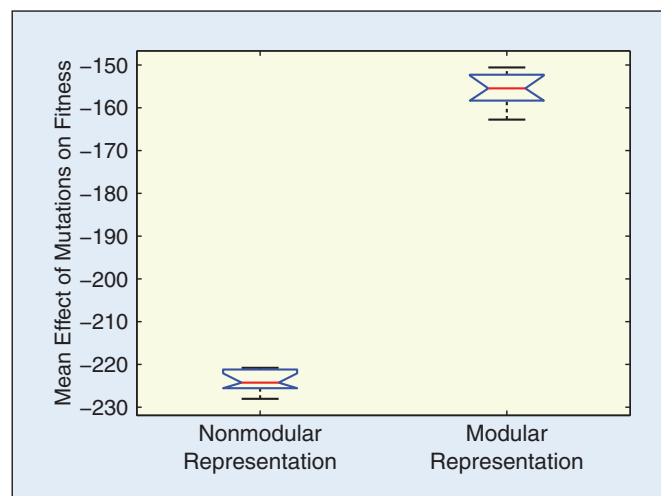

FIGURE 11 Boxplot of the mean effect of mutations on fitness. For each condition, the networks of the last generation were subjected to 1,000 mutations and the fitness of the resulting networks was compared to the fitness of the original networks. Higher values equal higher robustness to mutations. For the details of the boxplot format see the caption of Fig. 8.

> **Adding the possibility of modular mapping from genotype to phenotype to an implicit genetic encoding for artificial neural networks consistently led to improved algorithm performance.**
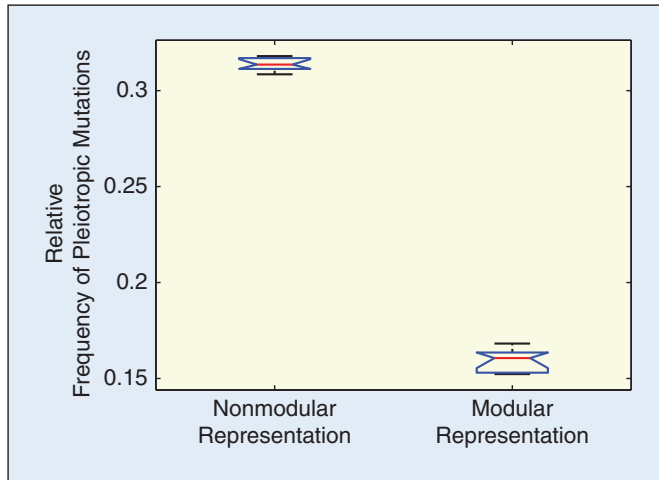

FIGURE 12 Boxplot of the relative frequency of pleiotropic mutations in the networks of the last generation for both conditions. Pleiotropic mutations were defined as mutations which simultaneously affect the navigation efficiency and the adaptivity of a robot by more than 5%. For the details of the boxplot format see the caption of Fig. 8.

evaluation: A) The *navigation efficiency* of a controller can be quantified by the total number of collected rewards (irrespective of the reward value). B) The *adaptivity* of a controller can be measured by calculating the ratio of the number of high rewards that the robot collected and the total number of collected rewards (fitness is the product of navigation efficiency times adaptivity).

We measured the incidence of pleiotropic mutations by calculating the relative frequency of mutations that simultaneously affect both behavioral traits for the networks of the last generation in the two experimental conditions (see Fig. 12). A mutation was defined as pleiotropic if it simultaneously changed both behavioral characters by more than 5%. The networks that evolved with the modular mapping displayed a significantly lower ratio of pleiotropic mutations than the networks that evolved with the non-modular mapping (Wilcoxon ranksum test, $p < 0.0002$).

These findings highlight an important difference between modular representations and other methods which potentially mitigate the effects of pleiotropic mutations. For example, there are a number of mechanisms in the literature which limit the effect of detrimental mutations by reducing selection pressure (e.g., spatial selection operators [62], speciation [43], island models [63]). Other strategies aim at changing the impact of mutations on individual behavior (e.g., memetic algorithms [64], [65]). However, unlike the presented approach based on a modular mapping from genotype to phenotype, these methods do not allow automatic evolutionary control of the effect of mutations[1].

V. Conclusion

Modular mapping from genotype to phenotype is widely recognized as a cornerstone of evolvability in biological organisms [24]. The results of our experiments show that adding the possibility of modular mapping from genotype to phenotype to an implicit genetic encoding for artificial neural networks consistently led to improved algorithm performance. The hypothesis that modular solutions allow for a higher evolvability in cases where multiple control problems have to be solved simultaneously is corroborated by the automatic decomposition of the control architecture in functionally separate modules.

Further analysis revealed that the presented modular representation also led to improved robustness of the networks to detrimental mutations. In particular, the modular mapping allowed for a lower incidence of pleiotropic mutations. This is in line with the hypothesis that modularity in biological organisms evolved by limiting pleiotropic effects of mutations [22].

VI. Acknowledgments

References

[1] T. Bäck. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford Univ. Press.

[2] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* Cambridge, MA: MIT Press, 2004.

[3] K. A. De Jong, "Evolving intelligent agents: A 50 year quest," *IEEE Comput. Intell. Mag.*, vol. 3, no. 1, pp. 12–17, 2008.

[4] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, pp. 1423–1447, 1999.

[5] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evol. Intell.*, vol. 1, no. 1, pp. 47–62, Mar. 2008.

[6] J. Gallagher, R. Beer, and K. Espenschied, "Application of evolved locomotion controllers to a hexapod robot," *Robot. Auton. Syst.*, 1996.

[7] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Trans. Syst., Man, Cybern.*, vol. 26, no. 3, pp. 396–407, 1996.

[8] S. Baluja, "Evolution of an artificial neural network based autonomous land vehicle controller," *IEEE Trans. Syst., Man, Cybern.*, 1996.

[9] S. Nolfi and D. Parisi, "Evolving non-trivial behaviors on real robots: A garbage collecting robot," *Robot. Auton. Syst.*, 1997.

[10] F. Gruau and K. Quatramaran, "Cellular encoding for interactive evolutionary robotics," in *Proc. 4th European Conf. Artificial Life*, 1997.

[11] A. Ijspeert and J. Kodjabachian, "Evolution and development of a central pattern generator for the swimming of a lamprey," *Artif. Life*, 1999.

[12] B. V. Haller, A. Ijspeert, and D. Floreano, "Co-evolution of structures and controllers for Neubot underwater modular robots," *Adv. Artif. Life*, 2005.

[13] J. Zufferey, D. Floreano, and M.V. Leeuwen, "Evolving vision-based flying robots," in *Proc. 2nd Int. Workshop Biologically Motivated Computer Vision (BMCV'2002)*, 2002, pp. 592–600.

---

[1]Of course, it is possible to combine one or multiples of these mechanisms with the suggested modular implicit genetic representation which could further increase performance.

[14] F. Gomez and R. Miikkulainen, "Active guidance for a finless rocket using neuro-evolution," in *Proc. Genetic and Evolutionary Computation Conf.*, 2003.

[15] Y. Sit and R. Miikkulainen, "Learning basic navigation for personal satellite assistant using neuroevolution," in *Proc. 2005 Conf. Genetic and Evolutionary Computation*, 2005.

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

[17] P. Dürr, C. Mattiussi, A. Soltoggio, and D. Floreano, "Evolvability of neuromodulated learning for robots," in *Proc. 2008 ECSIS Symp. Learning and Adaptive Behavior in Robotic Systems*, 2008, pp. 41–46.

[18] R. De Nardi, J. Togelius, O. E. Holl, and S. M. Lucas, "Neural networks for helicopter control: Why modularity matters," in *Proc. IEEE Congr. Evolutionary Computation*, 2006.

[19] S. Nolfi. (1997). *Using emergent modularity to develop control systems for mobile robots*.

[20] J. Reeder, R. Miguez, J. Sparks, M. Georgiopoulos, and G. Anagnostopoulos. (2008, Dec.). Interactively evolved modular neural networks for game agent control. *Proc. 2008 IEEE Symp. Computational Intelligence and Games*, pp. 167–174.

[21] A. Christensen and M. Dorigo. (2006). *Incremental evolution of robot controllers for a highly integrated task. From Animals to Animats 9 (Lecture Notes in Computer Science)*. Berlin: Springer-Verlag 4095, pp. 473–484.

[22] G. P. Wagner. (1996, Feb.). Homologues, natural kinds and the evolution of modularity. *Integr. Comp. Biol.* 36(1), pp. 36–43.

[23] G. P. Wagner and L. Altenberg, "Perspective: Complex adaptations and the evolution of evolvability," *Evolution*, vol. 50, no. 3, pp. 967–976, 1996.

[24] T. F. Hansen, "Is modularity necessary for evolvability? Remarks on the relationship between pleiotropy and evolvability," *Biosystems*, vol. 69, no. 2–3, pp. 83–94, 2003.

[25] R. Calabretta, S. Nolfi, D. Parisi, and G. P. Wagner, *Emergence of Functional Modularity in Robots*. Cambridge, MA: MIT Press, 1998.

[26] R. Calabretta, S. Nolfi, D. Parisi, and G. P. Wagner, "A case study of the evolution of modularity: Towards a bridge between evolutionary biology, artificial life, neuro- and cognitive science," in *Proc. 6th Int. Conf. Artificial Life*, 1998, pp. 275–284.

[27] R. Calabretta, S. Nolfi, D. Parisi, and G. P. Wagner, "Duplication of modules facilitates the evolution of functional specialization," *Artif. Life*, vol. 6, no. 1, pp. 69–84, 2000.

[28] A. Di Ferdinando, R. Calabretta, and D. Parisi, "Evolving modular architectures for neural networks," in *Proc. 6th Neural Computation and Psychology Workshop: Evolution, Learning, and Development*, R. French and J. Sougné, Eds. London: Springer-Verlag, 2001, pp. 253–262.

[29] R. Calabretta, A. Di Fernando, G. P. Wagner, and D. Parisi, "What does it take to evolve behaviorally complex organisms?," *BioSystems*, vol. 69, pp. 245–262, 2002.

[30] M. A. Potter and K. A. De Jong. (2000, Jan.). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evol. Comput.* 8(1), pp. 1–29.

[31] D. Moriarty and R. Miikkulainen. (1997). Forming neural networks through efficient and adaptive coevolution. *Evol. Comput.* 5(4), pp. 373–399.

[32] J. Reisinger, K. O. Stanley, and R. Miikkulainen, "Evolving reusable neural modules," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO-2004)*. Berlin: Springer-Verlag, 2004, pp. 69–81.

[33] V. R. Khare, X. Yao, B. Sendhoff, Y. Jin, and H. Wersing, "Co-evolutionary modular neural networks for automatic problem decomposition," in *Proc. 2005 IEEE Congr. Evolutionary Computation, CEC 2005*. Edinburgh, U.K.: IEEE Press, 2005, pp. 2691–2698.

[34] V. Khare, X. Yao, and B. Sendhoff. (2006). Multi-network evolutionary systems and automatic decomposition of complex problems. *Int. J. General Syst.* 35(3), pp. 259–274.

[35] X. Yao and M. M. Islam, "Evolving artificial neural network ensembles," *IEEE Comput. Intell. Mag.*, vol. 3, no. 1, pp. 31–42, 2008.

[36] N. García-Pedrajas, C. Hervás-Martínez, and J. Muñoz Pérez. (2002, Dec.). Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). *Neural Netw.* 15(10), pp. 1259–1278.

[37] N. García-Pedrajas, C. Hervás-Martínez, and D. Ortiz-Boyer. (2005, June). Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Trans. Evol. Comput.* 9(3), pp. 271–302.

[38] C. Mattiussi, D. Marbach, P. Dürr, and D. Floreano. (2008). The age of analog networks. *AI Mag.* 29(3), pp. 63–76.

[39] H. Kitano, "Designing neural networks by genetic algorithms using graph generation system," *Complex Syst. J.*, vol. 4, pp. 461–476, 1990.

[40] F. Gruau. (1995). Automatic definition of modular neural networks. *Adaptive Behav.* 3(2), pp. 151–183.

[41] F. Gruau, D. Whitley, and L. Pyeatt, "A comparison between cellular encoding and direct encoding for genetic neural networks," in *Proc. 1st Annu. Conf. Genetic Programming 1996*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. CA, Stanford University, 1996, pp. 81–89.

[42] F. J. Gomez and R. Miikkulainen, "Solving non-markovian control tasks with neuro-evolution," in *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI)*, 1999, pp. 1356–1361.

[43] K. O. Stanley and R. Miikkulainen. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.* 10(2), pp. 99–127.

[44] J. Bongard. (2002). Evolving modular genetic regulatory networks. *Proc. 2002 Congr. Evolutionary Computation*, vol. 2, pp. 1872–1877.

[45] P. Dürr, C. Mattiussi, and D. Floreano. (2006). *Neuroevolution with analog genetic encoding. Parallel Problem Solving from Nature—PPSN iX (LNCS)* 9, pp. 671–680.

[46] A. Soltoggio, P. Dürr, C. Mattiussi, and D. Floreano, "Evolving neuromodulatory topologies for reinforcement learning-like problems," in *Proc. IEEE Congr. Evolutionary Computation (CEC)*, 2007.

[47] J. Reisinger and R. Miikkulainen, "Acquiring evolvability through adaptive representations," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO 2007)*, 2007.

[48] C. Mattiussi and D. Floreano, "Analog genetic encoding for the evolution of circuits and networks," *IEEE Trans. Evol. Comput.*, 2007.

[49] D. Marbach, C. Mattiussi, and D. Floreano. (2007). Bio-mimetic evolutionary reverse engineering of genetic regulatory networks. *Proc. 5th European Conf. Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO 2007)*, pp. 155–165.

[50] D. Marbach, C. Mattiussi, and D. Floreano. (2009). Replaying the evolutionary tape: Bio-mimetic reverse engineering of gene networks. *Ann. New York Acad. Sci.* 1158, pp. 234–245.

[51] D. Gusfield. (1997). *Algorithms on Strings, Trees, and Sequences*. Cambridge, U.K., Cambridge Univ. Press.

[52] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proc. 9th Conf. Autonomous Robot Systems and Competitions*, 2006, pp. 59–65.

[53] S. Magnenat, M. Waibel, and A. Beyeler. (2009). *Enki—An open source fast 2D robot simulator*.

[54] B. M. Yamauchi and R. D. Beer, "Sequential behavior and learning in evolved dynamical neural networks," *Adaptive Behav.*, vol. 2, pp. 219–246, 1994.

[55] J. Blynel, "Evolving reinforcement learning-like abilities for robots," in *Evolvable Systems: From Biology to Hardware*. 2003, pp. 320–331.

[56] Y.-T. Wang, "Synaptic plasticity in learning and memory," *Int. J. Develop. Neurosci.*, vol. 24, no. 8, pp. 492–493, 2006.

[57] A. Soltoggio, J. A. Bullinaria, C. Mattiussi, P. Dürr, and D. Floreano, "Evolutionary advantages of neuromodulated plasticity in dynamic, reward- based scenarios," in *Proc. 11th Int. Conf. Artificial Life (Alife XI)*, 2008, S. Bullock, J. Noble, R. Watson, and M. A. Bedau, Eds. 2008, pp. 569–576.

[58] C. Mattiussi, P. Dürr, and D. Floreano. (2007). Center of mass encoding: A self-adaptive representation with adjustable redundancy for real-valued parameters. *Proc. Genetic and Evolutionary Computation Conf.*

[59] R. Mcgill, J. W. Tukey, and W. A. Larsen, "Variations of box plots," *The Amer. Statist.*, vol. 32, no. 1, pp. 12–16, 1978.

[60] C. Mattiussi. (2005). *Evolutionary synthesis of analog networks*. Ph.D. dissertation, Lausanne, Switzerland.

[61] V. Kvasnicka and J. Pospíchal. (2002, Jan.). Emergence of modularity in genotype-phenotype mappings. *Artif. Life* 8(4), pp. 295–310.

[62] J. Sarma and K. De Jong, "An analysis of the effects of neightborhood size and shape on local selection algorithms," in *Parallel Problem Solving from Nature—PPSN IV*. 1996, pp. 236–244.

[63] D. Whitley, S. Rana, and R. B. Heckendorn. (1998). The island model genetic algorithm: On separability, population size and convergence. *J. Comput. Inform. Technol.* 7, pp. 33–47.

[64] P. Moscato. (1989). *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms*.

[65] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong. (2006). Classification of adaptive memetic algorithms: A comparative study. *IEEE Trans. Syst., Man, Cybern. B* 36(1), pp. 141–152.