Dynamics of Neuronal Models in Online Neuroevolution of Robotic Controllers

Fernando Silva[1,3], Luís Correia[3], and Anders Lyhne Christensen[1,2]

[1] Instituto de Telecomunicações, Lisboa, Portugal
[2] Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
[3] LabMAg, Faculdade de Ciências, Universidade de Lisboa, Portugal
{fsilva,luis.correia}@di.fc.ul.pt, anders.christensen@iscte.pt

Abstract. In this paper, we investigate the dynamics of different neuronal models on online neuroevolution of robotic controllers in multirobot systems. We compare the performance and robustness of neural network-based controllers using summing neurons, multiplicative neurons, and a combination of the two. We perform a series of simulation-based experiments in which a group of e-puck-like robots must perform an integrated navigation and obstacle avoidance task in environments of different complexity. We show that: (i) multiplicative controllers and hybrid controllers maintain stable performance levels across tasks of different complexity, (ii) summing controllers evolve diverse behaviours that vary qualitatively during task execution, and (iii) multiplicative controllers lead to less diverse and more static behaviours that are maintained despite environmental changes. Complementary, hybrid controllers exhibit both behavioural characteristics, and display superior generalisation capabilities in simple and complex tasks.

Keywords: Evolutionary robotics, artificial neural network, evolutionary algorithm, online neuroevolution.

1 Introduction

Evolutionary computation has been widely studied in the field of robotics as a means to automate the design of robotic systems [1]. In evolutionary robotics (ER), robot controllers are typically based on artificial neural networks (ANNs) due to their capacity to tolerate noise in sensors. The parameters of the ANN, such as the connection weights, and occasionally the topology, are optimised by an evolutionary algorithm (EA), a process termed *neuroevolution* [2].

Online neuroevolution of controllers is a process of continuous adaptation that potentially gives robots the capacity to respond to changes or unforeseen circumstances by modifying their behaviour. An EA is executed on the robots themselves while they perform their task. The main components of the EA (evaluation, selection, and reproduction) are carried out autonomously by the robots without any external supervision. This way, robots may be capable of long-term self-adaptation in a completely autonomous manner.

L. Correia, L.P. Reis, and J. Cascalho (Eds.): EPIA 2013, LNAI 8154, pp. 90–101, 2013.

In a contribution by Watson *et al.* [3], the use of multirobot systems in online neuroevolution was motivated by the speed-up of evolution due to the inherent parallelism in groups of robots that evolve together in the task environment. Over the last decade, different approaches to online neuroevolution in multirobot systems have been proposed [4]. Notwithstanding, properties at the level of individual neurons have largely been left unstudied. Online neuroevolution studies have been almost exclusively based on ANNs composed of a variation of the neuronal model introduced in the 1940s by McCulloch and Pitts [5], that is, summing neurons. With advances in biology, multiplicative-like operations have been found in neurons as a means to process non-linear interactions between sensory inputs [6,7]. In machine learning, multiplicative neurons have shown to increase the computational power and storage capacities of ANNs [8,9].

In this paper, we investigate the dynamics and potential benefits of multiplicative neurons and of summing neurons, separately and combined, in online neuroevolution in multirobot systems. We conduct a simulated experiment in which a group of robots modelled after the e-puck [10] must perform an integrated navigation and obstacle avoidance task in environments of distinct complexity. The task implies an integrated set of actions, and consequently a trade-off between avoiding obstacles, maintaining speed, and forward movement. The three types of controllers are compared with respect to the speed of convergence, task performance, generalisation capabilities, complexity of solutions, and diversity of behaviours evolved. The main conclusion is that the combination of summing neurons and multiplicative neurons leads to superior generalisation capabilities, and allows robots to exhibit different behaviours that vary qualitatively or that remain static, depending on task complexity.

2 Background and Related Work

In this section, we describe the neuronal models considered, and we introduce odNEAT, the online neuroevolution algorithm used in this study. We exclusively consider discrete-time ANNs with summing or multiplicative neuronal models. Models that have more complex or explicit time-dependent dynamics, such as spiking neurons [11], are left for posterior investigation.

2.1 Neuronal Models

In discrete-time ANNs, the classic summing unit is the most commonly used neuronal model. The summing unit model performs the computation as follows:

$$a_i = f\Big(\sum_{j=1}^{N} w_{ji} \cdot x_j + w_0\Big) \,. \tag{1}$$

where a_i is the activation level of a given neuron i, and f is the activation function applied on the weighted sum of inputs from incoming neurons x_j, plus the bias value w_0. The activation function f can take the form of, for instance, a

threshold function or a sigmoid function. However, relying exclusively on sums of weighted inputs potentially limits performance and learning capabilities when complex or non-linear interactions are considered [6,9], as in the case of robotics.

In machine learning, multiplicative neuronal models were introduced more than 20 years ago. Examples include the pi-sigma and the sigma-pi units [9], and the more general product unit neuronal model [8], which we use in this study. In the product unit model, the activation of a neuron i is computed as follows:

$$a_i = f\Big(\prod_{j=1}^{N} x_j^{w_{ji}}\Big) \,. \tag{2}$$

with notations similar to Eq. 1. The number of exponents j gives the *order* of the neuron, thus denoting the ANN as a *higher-order neural network*. The exponents are real values, in which negative exponents enable division operations.

The potential benefits of the product unit model have been widely discussed in the literature. Using gradient descent methods, Durbin and Rumelhart [8] concluded that product units have superior information and learning capabilities compared to summing units. Complementary, Schmitt [9] analysed the gains in information processing and learning capabilities of product unit neurons in terms of solution complexity and computational power. Despite the positive results, only recently the potential benefits of product units were investigated in an evolutionary robotics context. Cazenille *et al.* [7] performed the offline evolution of ANN controllers for the *coupled inverted pendulums* problem, a benchmark in modular robotics. The authors investigated the interplay between microscopic properties such as the neuronal model, and macroscopic properties such as modularity and repeating motifs in ANNs. Surprisingly, their results suggested that product units may be counter-productive when used alone. If ANNs display regularities such as modularity, then product units may lead to better fitness scores, while requiring fewer evaluations for evolving a solution.

2.2 odNEAT: An Online Neuroevolution Algorithm

odNEAT [4] is a decentralised online neuroevolution algorithm for multirobot systems. odNEAT optimises both the parameters and the topology of the ANN controllers. The algorithm starts with networks with no hidden neurons, and with each input neuron connected to every output neuron. Topologies are gradually *complexified* by adding new neurons and new connections through mutation, thus allowing odNEAT to find an appropriate degree of complexity for the task.

odNEAT implements the online evolutionary process according to a physically distributed island model. Each robot optimises an internal population of genomes through intra-island variation, and genetic information between two or more robots is exchanged through inter-island migration. In this way, each robot is potentially self-sufficient and the evolutionary process capitalises on the exchange of genetic information between multiple robots for faster adaptation.

During task execution, each robot is controlled by an ANN that represents a candidate solution to a given task. Agents maintain a virtual energy level reflecting their individual performance. The fitness value is defined as the average of the energy level, sampled at regular time intervals. When the virtual energy level of a robot reaches zero, the current controller is considered unfit for the task. A new controller is then created by selecting two parents from the repository, each one via a tournament selection of size 2. Offspring is created through crossover of the parents' genomes and mutation of the new genome. Mutation is both structural and parametric, as it adds new neurons and new connections, and optimises parameters such as connection weights and neuron bias values.

Note that odNEAT is used in this study as a representative efficient online neuroevolution algorithm that optimises ANN weights and topologies [4]. As all experiments are based on odNEAT, the main distinctions among them will be the use of summing neurons, multiplicative neurons, or a combination of the two, rather than the online neuroevolution algorithm or its particular details.

3 Methods

In this section, we define our experimental methodology, including the robot model, the two variants of the navigation and obstacle avoidance task, the experimental parameters, and how we characterise the behaviours evolved.

3.1 Robot Model and Behavioural Control

To conduct our simulated experiments, we use JBotEvolver[1], an open-source, multirobot simulation platform, and neuroevolution framework. The simulator implements 2D differential drive kinematics. In our experimental setup, the simulated robots are modelled after the e-puck [10], a small (75 mm in diameter) differential drive robot capable of moving at a maximum speed of 13 cm/s. Each robot is equipped with eight infrared sensors, for obstacle detection, and communication of, for instance, genomes at a range of up to 25 cm between sender and receiver. Each infrared sensor and each actuator are subject to noise, which is simulated by adding a random Gaussian component within $\pm 5\%$ of the sensor saturation value or actuation value. Each robot also has an internal sensor that allows it to perceive its current virtual energy level.

Each robot is controlled by an ANN synthesised by odNEAT. The ANN's connection weights $\in$ [-10,10]. The input layer consists of 17 neurons: (i) eight for robot detection, (ii) eight for wall detection, and (iii) one neuron for the virtual energy level sensor. The output layer contains two neurons, one for each wheel.

3.2 Integrated Navigation and Obstacle Avoidance

Navigation and obstacle avoidance is a classic task in evolutionary robotics, and an essential feature for autonomous robots operating in real-world environments.

[1] `https://code.google.com/p/jbotevolver/`

Robots have to simultaneously move as straight as possible, maximise wheel speed, and avoid obstacles. The task implies an integrated set of actions, and consequently a trade-off between avoiding obstacles in sensor range and maintaining speed and forward movement. Navigation and obstacle avoidance is typically conducted in single robot experiments. In multirobot experiments, each robot poses as an additional, moving obstacle for the remaining group.

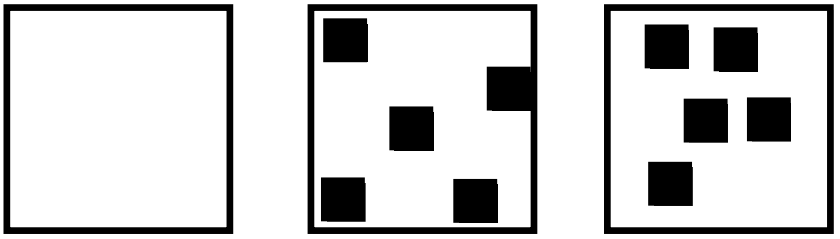
In our experiments, a group of five robots operates in a square arena surrounded by walls. The size of the arena was chosen to be 3 x 3 meters. Initially, robots are placed in random positions. During simulation, the virtual energy level E is updated every 100 ms according to the following equation:

$$\frac{\Delta E}{\Delta t} = f_{norm}(V \cdot (1 - \sqrt{\Delta v}) \cdot (1 - d_r) \cdot (1 - d_w)) \; . \tag{3}$$

where V is the sum of rotation speeds of the two wheels, with $0 \leq V \leq 1$. $\Delta v \in [0, 1]$ is the normalised algebraic difference between the signed speed values of the wheels (positive in one direction, negative in the other). d_r and d_w are the highest activation values of the infrared sensors for robot detection and for wall detection, respectively. d_r and d_w are normalised to a value between 0 (no robot/wall in sight) and 1 (collision with a robot or wall). f_{norm} maps from the domain $[0, 1]$ into $[-1, 1]$.

Experimental Setup. We conducted experiments in two different environments to assess performance. The first environment is a plain arena, in which the only obstacles are the robots and the walls that confine the arena. The second environment is an arena with five additional obstacles with dimensions of 0.5 x 0.5 meters. The additional obstacles are of the same material as the walls, and intuitively increase the difficulty of the task by reducing the area for navigation. In each evolutionary run conducted in the second environment, obstacles are placed at random locations. The environments are illustrated in Fig. 1.

We performed three sets of evolutionary experiments in each environment, characterised by different neuronal models. In one set of experiments, neurons



(a) Plain environment (b) Complex environment — example 1 (c) Complex environment — example 2

Fig. 1. The two types of environments used to evolve controllers. Each of the arenas measures 3 x 3 meters. The dark areas denote physical obstacles, while the white areas denote the arena surface on which robots can navigate.

added through structural mutation are multiplicative product units. In the second set of experiments, neurons introduced are summing units. In the third set of experiments, each new neuron has an equal probability of 0.5, sampled from a uniform distribution, of being either a multiplicative or summing neuron. Multiplicative product units may therefore be combined with summing units, and introduce the ability for computing weighted sums of products, and vice-versa.

For each experimental configuration, we performed 30 independent evolutionary runs. Each run lasts for 100 hours of simulated time. The virtual energy level of robots is limited to the range [0,100] energy units. When the energy level reaches zero, a new controller is generated and assigned the default energy value of 50 units. Other parameters are the same as in [4].

3.3 Characterisation of Behavioural Diversity

Ultimately, online neuroevolution of controllers synthesises the *behavioural* control of robots. To characterise and compare behaviours evolved, we use a generic Hamming distance-based behavioural metric based on the mapping between sensors and actuators. This measure has shown to be, at least, as efficient as domain-dependent behavioural metrics [12]. The behaviour metric is based on the set of sensor readings and actuation values ϑ normalised into [0,1], as follows:

$$\vartheta = \Big[\{\mathbf{s}(t), \mathbf{a}(t)\}, t \in [0, T]\Big] \; . \tag{4}$$

where $\mathbf{s}(t)$ and $\mathbf{a}(t)$ are the sensor readings and actuation values at time t, respectively, and T is the number of observations. The binary version ϑ_{bin} of ϑ is computed as follows:

$$\vartheta_{bin} = \Big[\vartheta_{bin}(t), t \in [0, T]\Big] = \Big[\{\mathbf{s}_{bin}(t), \mathbf{a}_{bin}(t)\}, t \in [0, T]\Big] \; . \tag{5}$$

where each $s_{bin,i}(t) \in s_{bin}(t)$ is defined as 1 if $s_i(t) > 0.5$ and 0 otherwise, and each $a_{bin,i}(t) \in a_{bin}(t)$ is defined as 1 if $a_i(t) > 0.5$ and 0 otherwise. The Hamming distance between two behaviours is then computed as follows:

$$\sigma(\vartheta_1, \vartheta_2) = \sum_{t=0}^{T} h(\vartheta_{1,bin}(t), \vartheta_{2,bin}(t)) \; . \tag{6}$$

$$h(\vartheta_1, \vartheta_2) = \sum_{i=1}^{len(\vartheta_1)} 1 - \delta(\vartheta_1[i], \vartheta_2[i]) \; . \tag{7}$$

where $len(\vartheta_1) = len(\vartheta_2)$ denotes the length of the binary sequences ϑ_1 and ϑ_2, and $\delta(i, j)$ is the Kronecker delta defined as $\delta(i, j) = 1$ if i = j, and 0 otherwise. We further extend the generic Hamming distance between sensor readings and actuation values to capture the *intra-behaviour* distance as follows:

$$\sigma(\vartheta_{bin}) = \sum_{t=1}^{T} h(\vartheta_{bin}(t-1), \vartheta_{bin}(t)) \; . \tag{8}$$

$\sigma(\vartheta_{bin})$ captures the differences between consecutive observations of the relation between sensor readings and actuation values, thereby approximating to what extent the behaviour of a robot varies during task execution.

4 Experimental Results

In this section, we present and discuss the experimental results. We use the Mann-Whitney test to compute statistical significance of differences between sets of results because it is a non-parametric test, and therefore no strong assumptions need to be made about the underlying distributions. Success rates are compared using Fisher's exact test, a non-parametric test suitable for this purpose [13]. Statistical dependence between two variables is computed using the non-parametric Spearman's rank correlation coefficient.

4.1 Comparison of Performance

We start by comparing the performance of the three controller models. We focus on three aspects: (i) the number of evaluations, i.e., the number of controllers tested by each robot before a solution to the task is found, (ii) the complexity of solutions evolved, and (iii) their generalisation capabilities.

The number of evaluations for the three types of controllers is listed in Table 1. In the plain environment, neuroevolution of hybrid controllers required fewer evaluations to synthesise solutions for the task. Differences between hybrid controllers and multiplicative controllers are statistically significant ($\rho < 0.001$, Mann-Whitney). Differences between other types of controllers are not significant. In the complex environment, solutions are synthesised faster when summing controllers are evolved. Multiplicative controllers, once again, require more evaluations to evolve solutions. Differences are significant with respect to summing controllers ($\rho < 0.001$, Mann-Whitney), and to hybrid controllers ($\rho < 0.01$). Differences between summing controllers and hybrid controllers are not significant, although summing controllers converged to a solution in fewer evaluations on average. Interestingly, both hybrid controllers and multiplicative controllers are affected by the increase in task complexity. Summing controllers, on the other hand, require an approximate number of evaluations in the two tasks.

Table 1. Comparison of the number of evaluations between the three types of controllers considered in the two tasks (average ± std. dev.)

Controller	Plain environment	Complex environment
Summing	23.63 ± 19.16	22.61 ± 18.68
Multiplicative	26.08 ± 21.10	32.02 ± 24.93
Hybrid	21.54 ± 20.53	27.81 ± 29.16

By analysing the complexity of solutions evolved[2], we observed simple neural topologies for solving each task, as listed in Table 2. Overall, multiplicative controllers present the least complex topologies, in equality with summing controllers in the complex environment. Despite solving the tasks with less structure, the number of evaluations to synthesise suitable multiplicative neurons is higher, as discussed previously and shown in Table 1. This is due to multiplicative neurons requiring a finer-grain adjustment of parameters. Compared to summing neurons, multiplicative controllers required more adjustments of connection weights through mutation and, therefore, a higher number of evaluations.

Complementary, hybrid controllers present the most complex topologies. The crossover operator manipulates topological structure involving different neural dynamics, i.e., summing and multiplicative neurons. We analysed the effects of evolutionary operators and found a more accentuated decrease in fitness scores when hybrid controllers are recombined. This effect is progressively eliminated as new neurons and new connections are added to the network. Nonetheless, despite differences in terms of neural complexity and number of evaluations, each experimental configuration lead to the evolution of high-scoring controllers. The average fitness score of solutions varies from 91.31 to 94.59 in the plain environment, and from 91.24 to 95.49 in the complex environment. Differences in fitness scores are not statistically significant across all comparisons.

Table 2. Neural complexity of solutions evolved. Neurons and connections added through evolution (average ± std. dev.).

	Plain environment		Complex environment	
Controller	Neurons	Connections	Neurons	Connections
Summing	3.14 ± 0.35	6.59 ± 0.90	1.17 ± 0.41	2.57 ± 1.03
Multiplicative	2.16 ± 0.40	4.62 ± 0.94	1.21 ± 0.42	2.84 ± 1.09
Hybrid	3.16 ± 0.39	6.56 ± 1.13	3.11 ± 0.32	6.47 ± 0.79

Testing for Generalisation. To analyse the generalisation capabilities of the different types of controllers, we conducted a series of generalisation tests. For each evolutionary run conducted previously, we restarted the task 100 times using the controllers evolved and not allowing further evolution. Each task restart is a generalisation test serving to assess if robots can continuously operate after redeployment, and for evaluating the ability to operate in conditions not experienced during the evolution phase. A group of robots passes a generalisation test if it continues to solve the task, i.e., if the virtual energy level of none of the robots reaches zero. Each generalisation test has the same duration as the evolutionary phase, 100 hours of simulated time. In the complex environment, the five obstacles are placed in random locations in each test.

[2] The complete set of networks evolved is available at `http://dx.doi.org/10.6084/m9.figshare.705842`

Table 3. Generalisation performance of each controller model. The table lists the average generalisation capabilities of each group of five robots, and the total of generalisation tests solved successfully.

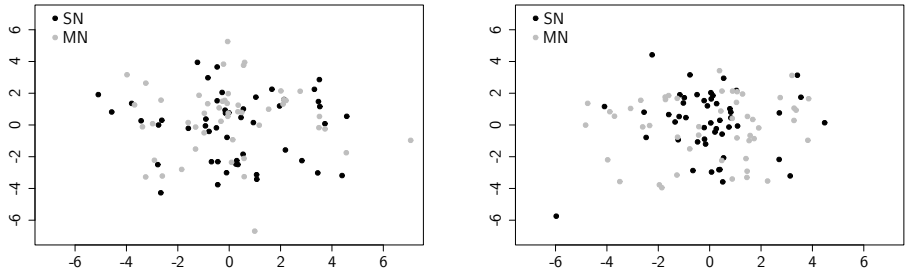
	Plain environment		Complex environment	
Controller	Generalisation (%)	Succ. tests	Generalisation (%)	Succ. tests
Summing	73.40 ± 23.33	2202/3000	45.30 ± 30.36	1359/3000
Multiplicative	85.87 ± 13.07	2576/3000	57.03 ± 26.09	1711/3000
Hybrid	84.10 ± 24.24	2523/3000	80.60 ± 14.77	2418/3000

In Table 3, we show the generalisation performance of each controller model. In the plain environment, multiplicative controllers outperform summing controllers by 12% as they solve 374 tests more. Differences in successful and unsuccessful generalisation tests are statistically significant ($\rho < 1 \cdot 10^{-4}$, Fisher's exact test). The hybrid controllers display high generalisation performance, similar to multiplicative controllers. Differences between these two controllers are not statistically significant. In the complex task, multiplicative controllers also outperform summing controllers by approximately 12% due to solving 352 tests more ($\rho < 1 \cdot 10^{-4}$, Fisher's exact test). More importantly, hybrid controllers significantly outperform both summing controllers and multiplicative controllers ($\rho < 1 \cdot 10^{-4}$, Fisher's exact test), and maintain approximate generalisation levels across the two tasks. In this way, the higher number of evaluations to synthesise hybrid controllers for complex tasks is compensated for by the increase in generalisation performance caused by the interplay between summing neurons and multiplicative neurons in the same neural architecture.

The generalisation performance of hybrid controllers is a factor particularly important in the case of online evolution. Results obtained indicate that hybrid controllers can adapt more efficiently than multiplicative and summing controllers alone to contexts not explicitly encountered during evolution, thus avoiding the continuation of the evolutionary process. In this way, hybrid controllers revealed to be advantageous as they are high-scoring controllers with superior generalisation capabilities that require a competitive number of evaluations.

4.2 Analysis of Genotypic and Behavioural Search Space

To unveil the neuroevolution dynamics of summing neurons and of multiplicative neurons, we compared how the evolutionary search proceeds through the high-dimensional genotypic search space. To visualise the intermediate genotypes produced when using the two neuronal models, and how they traverse the search space *with respect to each other*, we use *Sammon's nonlinear mapping* [14]. Sammon's mapping performs a point mapping of high-dimensional data to two-dimensional spaces, such that the structure of the data is approximately preserved. The distance in the high-dimensional space δ_{ij} between two



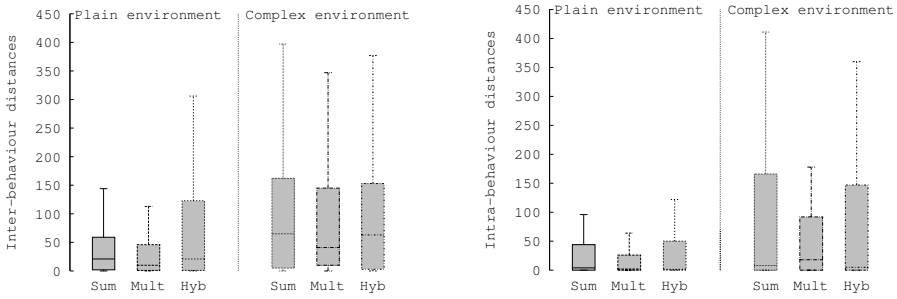
(a) Plain environment (b) Complex environment

Fig. 2. Combined Sammon's mapping of intermediate genotypes. Genotypes representing networks using summing neurons are marked in black, and genotypes representing networks with multiplicative neurons are marked in gray.

genotypes i and j is based on genomic distance as used in odNEAT. The distance between two points in the two-dimensional space is their Euclidean distance.

In Fig 2, we show the Sammon's mapping for the plain environment and for the complex environment. In order to obtain a clearer visualisation, and a consistent and representative selection of genotypes, we map the 50 most genetically different genotypes, which in turn represent the 50 most diverse topologies with respect to each experimental configuration. Generally speaking, there is a balanced exploration of the search space, as different controller models discover similar regions of the genotypic search space. Genotypes representing hybrid controllers also explore similar regions of the search space (data not shown). The progress throughout the evolutionary process is therefore similar in the three controller models, indicating that the fitness landscape is not significantly altered by the use of different neuronal models. In this way, the dynamics of neuronal models may lead to the evolution of different *behaviours*, which in turn would account for differences in performance. To verify this hypothesis, we analysed the behaviour space explored by solutions evolved. We used the two generic Hamming distance-based behaviour metrics described in Sect. 3.3. For each solution, we analysed the behaviour during 10,000 seconds of simulated time with a sampling rate of 10 seconds, resulting in a behaviour vector of length 1000.

In Fig. 3, we show the distribution of inter-behaviour distances and the distribution of intra-behaviour distances. In all experimental configurations, there is a very strong monotonic correlation between the novelty of the behaviour in terms of average inter-behaviour distance and the degree of intra-behaviour variance ($\rho > 0.98$, Spearman's correlation). The more different the behaviour is compared to remaining behaviours, the more it varies during task execution. Complementary, more common behaviours present a lower intra-behaviour variance during task execution. Despite behavioural differences, there is no clear correlation between the inter- and intra-behaviour distances and fitness scores ($-0.09 < \rho < 0.34$), as both types of behaviours lead to high performance.



(a) Inter-behaviour distances (b) Intra-behaviour distances

Fig. 3. Distribution of: (a) inter-behaviour distances between behaviours evolved for each task, by each type of controllers, and (b) intra-behaviour distances

Compared to multiplicative controllers, robots using summing controllers synthesise more diverse behaviours in the two environments ($\rho < 0.001$, Mann-Whitney) with higher intra-behaviour variance. Multiplicative controllers lead to less diverse behaviours with lower intra-behaviour variance, which maintain the same qualitative behaviour despite environmental changes. Summing controllers, on the other hand, adopt qualitatively different behaviours, especially in more complex tasks. Hybrid controllers appear to exhibit both behavioural characteristics. In the plain environment, robots evolved a greater diversity of robust behaviours with the higher intra-behaviour variance. With the increase of task complexity, robots exhibit intermediate behaviours in terms of inter- and intra-behaviour distances. In other words, hybrid controllers appear to capitalise on the tendency exhibited by multiplicative controllers to maintain the same qualitative behaviour, and on the ability of summing controllers to modify the behaviour of robots during task execution.

5 Conclusions and Future Work

In this paper, we investigated the effects of multiplicative neurons in online neuroevolution of robotic controllers. We compared the dynamics of multiplicative neurons, separately and when combined with summing neurons, in terms of speed of convergence, complexity of solutions evolved, task performance, generalisation capabilities, and diversity of behaviours evolved.

We showed that solutions are synthesised faster when summing neurons are used alone. We also showed that neural controllers using a combination of summing neurons and multiplications provide competitive results in terms of speed of convergence. In complex environments, the hybrid controllers rely on larger topologies, with one or two hidden neurons more. Nonetheless, these controllers exhibit significantly superior generalisation capabilities, and appear to balance between maintaining the same behaviour and modifying the behaviour depending on task requirements and complexity. An analysis of neural activation

and structural patterns could potentially unveil differences in neural dynamics and the decision making mechanisms underlying the robot's behaviour.

The immediate follow-up work includes the study of macroscopic properties such as modularity, structural regularity and hierarchy, and investigating if these properties can facilitate online neuroevolution for real-world complex tasks.

Acknowledgments. This work was partly supported by the Fundação para a Ciência e a Tecnologia (FCT) under the grants PEst-OE/EEI/LA0008/2013 and SFRH/BD/89573/2012.

References

1. Floreano, D., Keller, L.: Evolution of adaptive behaviour by means of Darwinian selection. PLoS Biology 8(1), e1000292 (2010)
2. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evolutionary Intelligence 1(1), 47–62 (2008)
3. Watson, R., Ficici, S., Pollack, J.: Embodied evolution: Distributing an evolutionary algorithm in a population of robots. Robotics and Autonomous Systems 39(1), 1–18 (2002)
4. Silva, F., Urbano, P., Oliveira, S., Christensen, A.L.: odNEAT: An algorithm for distributed online, onboard evolution of robot behaviours. In: 13th International Conference on Simulation & Synthesis of Living Systems, pp. 251–258. MIT Press, Cambridge (2012)
5. McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology 5(4), 115–133 (1943)
6. Koch, C.: Biophysics of computation: information processing in single neurons. Oxford Univ. Press, Oxford (2004)
7. Cazenille, L., Bredeche, N., Hamann, H., Stradner, J.: Impact of neuron models and network structure on evolving modular robot neural network controllers. In: 14th Genetic and Evolutionary Computation Conference, pp. 89–96. ACM Press, New York (2012)
8. Durbin, R., Rumelhart, D.E.: Product units: A computationally powerful and biologically plausible extension to backpropagation networks. Neural Computation 1(1), 133–142 (1989)
9. Schmitt, M.: On the complexity of computing and learning with multiplicative neural networks. Neural Computation 14(2), 241–301 (2002)
10. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: 9th Conference on Autonomous Robot Systems and Competitions, IPCB, Castelo Branco, Portugal, pp. 59–65 (2009)
11. Floreano, D., Schoeni, N., Caprari, G., Blynel, J.: Evolutionary bits 'n' spikes. In: 8th International Conference on Simulation & Synthesis of Living Systems, pp. 335–344. MIT Press, Cambridge (2003)
12. Mouret, J., Doncieux, S.: Encouraging behavioral diversity in evolutionary robotics: An empirical study. Evolutionary Computation 20(1), 91–133 (2012)
13. Fisher, R.: Statistical Methods for Research Workers. Oliver & Boyd, Edinburgh (1925)
14. Sammon Jr., J.: A nonlinear mapping for data structure analysis. IEEE Transactions on Computers C-18(5), 401–409 (1969)