LETTER Communicated by Gordon Pipa

Neuronal Assembly Dynamics in Supervised and Unsupervised Learning Scenarios

Renan C. Moioli
r.moioli@sussex.ac.uk
Phil Husbands
p.husbands@sussex.ac.uk
Centre for Computational Neuroscience and Robotics, Department of Informatics, University of Sussex, Falmer, Brighton, BN1 9QH, U.K.

The dynamic formation of groups of neurons—neuronal assemblies—is believed to mediate cognitive phenomena at many levels, but their detailed operation and mechanisms of interaction are still to be uncovered. One hypothesis suggests that synchronized oscillations underpin their formation and functioning, with a focus on the temporal structure of neuronal signals. In this context, we investigate neuronal assembly dynamics in two complementary scenarios: the first, a supervised spike pattern classification task, in which noisy variations of a collection of spikes have to be correctly labeled; the second, an unsupervised, minimally cognitive evolutionary robotics tasks, in which an evolved agent has to cope with multiple, possibly conflicting, objectives. In both cases, the more traditional dynamical analysis of the system's variables is paired with information-theoretic techniques in order to get a broader picture of the ongoing interactions with and within the network. The neural network model is inspired by the Kuramoto model of coupled phase oscillators and allows one to fine-tune the network synchronization dynamics and assembly configuration. The experiments explore the computational power, redundancy, and generalization capability of neuronal circuits, demonstrating that performance depends nonlinearly on the number of assemblies and neurons in the network and showing that the framework can be exploited to generate minimally cognitive behaviors, with dynamic assembly formation accounting for varying degrees of stimuli modulation of the sensorimotor interactions.

1 Introduction

Since Hebb's seminal work on brain activity (Hebb, 1949), the transient formation of neuronal groups or assemblies is increasingly linked to cognitive processes and behavior. In fact, there is a growing consensus that ensembles of neurons, not single neurons, constitute the basic functional unit of the central nervous system in mammalians (Averbeck & Lee, 2004;

Nicolelis & Lebedev, 2009). However, labeling a certain group of neurons as constituting an assembly is a challenging task that only recently has been alleviated by more advanced recording techniques and analysis tools (Buzsaki, 2010; Lopes dos Santos, Conde-Ocazionez, Nicolelis, Ribeiro, & Tort, 2011; Canolty et al., 2012). Also, it is still unclear how neuronal groups form, organize, cooperate, and interact over time (Kopell, Kramer, Malerba, & Whittington, 2010; Kopell, Whittington, & Kramer, 2011).

One hypothesis that has gained considerable supporting experimental evidence states that groups of neurons have their functional interactions mediated by synchronized oscillations—so-called binding by synchrony (Singer, 1999; Varela, Lachaux, Rodriguez, & Martinerie, 2001; Uhlhaas et al., 2009). Because structural connectivity is relatively static at the timescale of perception and action, the central idea is that the synchronization of neuronal activity by phase locking of network oscillations is exploited to define and encode relations between spatially distributed groups of neurons, and information dynamics and computations within the network relate to the timing of individual spikes rather than their rates. Indeed, phase relationships contain a great deal of information on the temporal structure of neural signals, modulate neuron interactions, are associated with cognition, and relate to memory formation and retrieval (Izhikevich, 1999; Womelsdorf et al., 2007; Masquelier, Hugues, Deco, & Thorpe, 2009; Kayser, Montemurro, Logothetis, & Panzeri, 2009). Moreover, recent work has shown that specific topological properties of local and distant cortical areas support synchronization despite inherent axonal conduction delays, thereby providing a substrate on which neuronal codes relying on precise interspike time can unfold (Vicente, Gollo, Mirasso, Fischer, & Pipa, 2008; Pérez et al., 2011).

Based on these concepts, in this letter, we focus on a pragmatic investigation of three aspects of computations in neuronal assemblies. Given a computational task and a neural network model comprising many neurons that are organized in an arbitrary number of assemblies, (1) does increasing the number of neural assemblies improve performance? (2) does the number of neurons per assembly affect performance? and (3) can dynamic assembly reorganization alone, leaving aside other plasticity mechanisms, be exploited to solve different tasks?

We approach these questions employing a neural network model based on the Kuramoto model of coupled phase oscillators (Kuramoto, 1984). It has been extensively studied in the statistical physics literature, with recent applications in a biological context due to its relatively simple and abstract mathematical formulation yet complex activity that can be exploited to clarify fundamental mechanisms of neuro-oscillatory phenomena without making too many a priori assumptions (Ermentrout & Kleinfeld, 2001; Cumin & Unsworth, 2007; Kitzbichler, Smith, Christensen, & Bullmore, 2009; Breakspear, Heitmann, & Daffertshofer, 2010; Moioli, Vargas, & Husbands, 2012). The model explicitly captures the phase dynamics of units that have intrinsic spontaneous oscillatory (spiking) activity and once connected can generate emergent rhythmic patterns. The correspondence between coupled phase oscillators and neuronal models is grounded in the phase reduction approach (Ermentrout & Kopell, 1986), according to which analysis of neuronal synchronization phenomena based on complex models can be greatly simplified by using phase models.

However, in addition to modeling constraints (described in section 2), the original Kuramoto model has limited spectral complexity compared to that of more biologically plausible neuronal models (Bhowmik & Shanahan, 2012). For this reason, recent extensions have been formulated to enhance its suitability to study a variety of neurobiological phenomena, incorporating, for example, spatially embedded couplings, transmission delays, and more complex phase response curves (Breakspear et al., 2010; Wildie & Shanahan, 2012). Nevertheless, it is possible to represent neurons as simple phase oscillators and model the spiking of individual cells, and results can still be of relevance. Indeed, this is exactly the objective: to avoid physiologically precise models that could make the analyses laborious and instead use a model that despite all the simplifications still presents complex and relevant spatiotemporal activity. One particular extension, presented in Orosz, Moehlis, and Ashwin (2009), allows one to fine-tune the synchronization regime, the number of assemblies, and the number of neurons per assembly, thus suiting our study, while also avoiding any problems in obtaining phase information (an issue in other models that consider frequency and amplitude dynamics; Pikovsky, Rosenblum, & Kurths, 2001). Hence, the extended Kuramoto model is highly relevant, at a certain level of abstraction, to modeling neural mechanisms underlying adaptive and cognitive behaviors and is used in the studies presented here.

The experiments were set up to encompass supervised and unsupervised learning scenarios (Dayan & Abbott, 2001). In supervised learning, there is an explicit target or supervisory signal mapping each set of inputs to expected outputs. In unsupervised learning, the system exploits the statistical structure of the set of inputs and operates as a self-organized, goal-oriented process. Although the latter is regarded as being more common in the brain, evidence suggests that both learning paradigms overlap and may be implemented by the same set of mechanisms (Knudsen, 1994; Dayan, 1999).

The first experiment, a supervised learning scenario, follows a method described in Maass, Legenstein, and Bertschinger (2005) and Legenstein and Maass (2007) to assess computational performance in generic neuronal microcircuits. More specifically, we analyze the computational power and generalization capability of neuronal networks with diverse assembly configurations in a generic spike pattern classification task. The method is especially suited to our goals because it proposes a measure to test the computational capability of neural microcircuits that is not exclusive to the task investigated here but to all computational tasks that need to have in common only which properties of the circuit input are relevant to the target outputs. In networks with the same number of neurons, we show that the performance of architectures constituted by many assemblies (and fewer neurons per assembly) is higher than the ones with fewer assemblies (and more neurons per assembly). We also show that in networks of varied size, performance saturates as soon as a given number of assemblies is formed, and the addition of neurons in each assembly does not influence performance in the classification task. In both scenarios, an analysis of redundancy and synergy, based on concepts of information theory, supports and provides further insights into the properties of the system.

The pattern classification task mentioned above may reflect or mimic some of the computations that are actually carried on in a real-world cognitive scenario; nevertheless, it does not capture the main task of cognition, which is the guidance of action. As Engel, Fries, and Singer (2001) pointed out "The criterion for judging the success of cognitive operations is not the correct' representation of environmental features, but the generation of actions that are optimally adapted to particular situations." Therefore, in the second experiment, an unsupervised learning scenario, we investigate evolved embodied cognitive behaviors in a simulated robotic agent. Following an evolutionary robotics approach, we show that the same network architecture of experiment 1 can be used as a control system for a simulated robotic agent engaged in a minimally cognitive task and that assembly reconfiguration can account for good performance in multiple, possibly conflicting tasks. The analysis is centered on both the system's variables dynamics, illustrating the interplay between dynamic assembly formation and the action being displayed by the robot, and the information dynamics between some components of the system, which complements the former analysis by quantifying and emphasizing the nonlinear relationships that are present in the brain-body-environment interactions.

As a consequence of approaching different learning paradigms, the analyses for the two experiments use distinct but appropriate tools. However, it is important to stress that these experiments are conceptually connected by the emphasis on neuronal assembly dynamics and its impact on task performance. The methods employed to explore supervised learning tasks struggle to operate in unsupervised scenarios because the former rely on coordinated, time-specific perturbations and measurements, with a focus on precise classifications, while the latter are mainly concerned with the behavior of the evolved robots. Notwithstanding, the first experiment provides insights into the system's dynamics, which contribute to the comprehension of the more elaborated second experiment. In this sense, the supervised and unsupervised learning tasks and the respective methods of investigation do not contradict but rather reinforce the flexibility of the framework in addressing diverse learning problems.

This letter is organized as follows. Section 2 presents the neural network model, including the rationale behind neural network models using coupled phase oscillators and the extension to the Kuramoto model, which facilitates the study of assembly dynamics, and a brief introduction to information theory, which is the basis of some analysis carried on in the experiments; sections 3 and 4 contain task-specific analysis methods and the results of the first and second experiment, respectively. The letter concludes in section 5 by highlighting the main contributions and giving a general discussion of the results obtained.

2 Theoretical Background

2.1 Neural Network Model. Neural network models based on the dynamics of voltage-dependent membrane currents (among which the Hodgkin-Huxley model is perhaps the best known) can be described by a single phase variable θ provided that the neurons are assumed to spike periodically when isolated, their firing rates are limited to a narrow range, and the coupling between them is weak (Hansel, Mato, & Meunier, 1995). In fact, many neural oscillatory phenomena can be captured and analyzed by studying the dynamics of coupled phase oscillators (Izhikevich, 2007), provided that the above conditions hold. In this sense, the Kuramoto model (Kuramoto, 1984) of coupled phase oscillators has been shown to be a useful tool in studying oscillatory phenomena in a broad range of fields, from semiconductor physics to fireflies' blinking pattern. The model is described by equation 2.1:

$$\dot{\theta}_n = \omega_n + \frac{K}{N}\sum_{m=1}^{N} g(\theta_n - \theta_m), \quad n = 1, \ldots, N, \tag{2.1}$$

where θ_n is the phase of the nth oscillator, ω_n is the natural frequency of the nth oscillator, K is the coupling factor between the nodes of the network, $g(\theta_n - \theta_m) = sin(\theta_n - \theta_m)$ represents the interaction between nodes, and N is the total number of oscillators.

The phase interaction function (PIF) g assumes the mutual influence among the symmetrically coupled oscillators to be periodic, that is, $g_{nm}(x + 2\pi) = g_{nm}(x)$; it can thus be expanded into a Fourier series. The Kuramoto model considers only the first term of this series, but when g incorporates more complex interactions between the nodes rather than the first harmonic only, the model displays a more complex spatiotemporal behavior and the synchronization patterns observed are closer to the ones measured in real brains (Hansel et al., 1995; Breakspear et al., 2010).

In particular, Ashwin, Wordsworth, and Townley (2007) and Wordsworth and Ashwin (2008) showed, when adopting a specific g, that the model is able to display heteroclinic cycles, a fundamental mechanism of cognition according to some authors (Ashwin & Timme, 2005; Rabinovich, Afraimovich, Bick, & Varona, 2012). Additionally, Orosz and collaborators

(2009) demonstrated how to design g so that the network organizes itself in an arbitrary number of stable clusters with a given phase relationship between clusters. These clusters, which emerge as an attractor of the system, remain stable up to a certain level of perturbations, applied in the form of inputs, above which a reorganization occurs, maintaining the same number of assemblies but with different membership configurations. Therefore, considering the aims of our study, this latter extension will be used in the subsequent experiments and is used as an abstract representation of interactions between spiking neurons. Equation 2.2 describes the model for N oscillators (Orosz et al., 2009):

$$\dot{\theta}_n = \omega_n + \frac{K}{N}\sum_{m=1}^{N} g(\theta_n - \theta_m) + \epsilon_n I_n(t), \quad n = 1, \ldots, N, \tag{2.2}$$

where $I_n(t)$ is an input scaled by a factor ϵ_n, and the PIF $g(\gamma)$ has the form of equation 2.3:

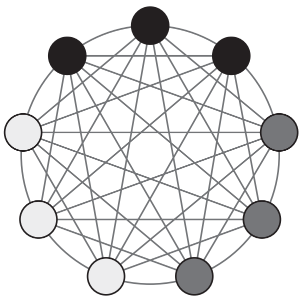
$$g(\gamma) = f_M(\gamma) + f_M(\gamma - \xi) \tag{2.3}$$

where $f_M(\gamma) = -2\tanh(M\sin(\gamma/2))\,\mathrm{sech}^2(M\sin(\gamma/2))\cos(\gamma/2)$ and $\xi = 2\pi/M$.

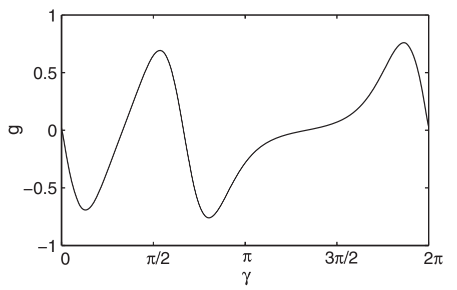
This PIF is obtained by a suitable choice of g and its derivatives to ensure that a system with N oscillators will present M stable assemblies separated equally in phase, with oscillators grouped according to their initial phases (which will dictate their position in the attraction basin determined by the total number of assemblies and parameter M). Assembly membership—which oscillator belongs to which assembly—can be changed if one applies an input to a given oscillator with a minimum magnitude and length. These will depend on the number of oscillators and assemblies (parameter M) of the network. Nevertheless, small perturbations still affect the overall behavior of the system. Figure 1 illustrates the main properties of the model.

The network is composed of nine fully connected neuronal oscillators with unitary couple (without loss of generality, the PIF is assumed to capture any effect due to larger or smaller couplings). The initial phases are uniformly distributed in the interval $[0, 2\pi)$, and the oscillators organize in $M = 3$ different (but with equal number of members) assemblies after a settling period (see Figure 1c). As the focus is on neuronal assembly in terms of phase relationships, we set the natural frequency w_n of all neurons to 1. In Figure 1d, the raster plot shows the neuronal spikes that occur every time the phase of each oscillator reaches a given threshold (0 in the example, but any other marker is acceptable). Notice from both figures the formation of three assemblies of three neurons each.

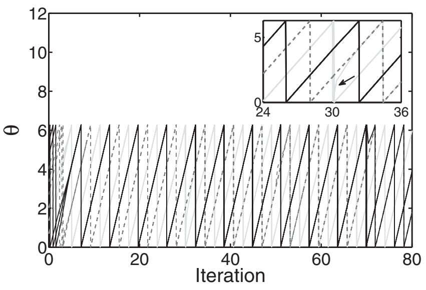
After a settling period, the system stabilizes in M assemblies and presents a periodic firing behavior. However, inputs to one or more neurons can change the network dynamics in two ways: it can modulate the ongoing



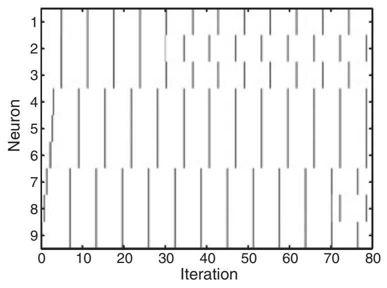
(a)



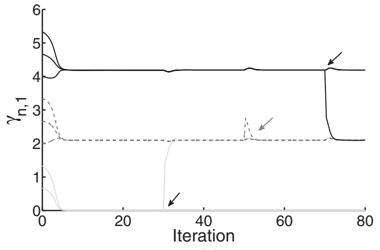
(b)



(c)



(d)



(e)

Figure 1: Model simulation using the PIF described by equation 2.3, with parameters $N = 9$, $M = 3$, $K = 1$, and $w_n = 1$. Oscillators form three clusters, and inputs to a given oscillator cause a transition to a different cluster, if the magnitude is high enough, or a modulation of the network behavior, if the input is small enough. (a) Network topology model. (b) PIF diagram (see equation 2.3). (c) Phase dynamics of each oscillator. The initial phases are uniformly distributed in $[0, 2\pi)$, and as the simulation progresses, the oscillators form $M = 3$ assemblies (assembly membership is represented by different gray tones in the plot). The small plot shows the moment ($t = 30$; see the black arrow) one oscillator moves from one assembly (solid light gray line) to another (dashed light gray line). (d) Raster plot showing the neuronal spikes that occur every time the phase of each oscillator reaches 0. (e) Effects of inputs on the system's dynamics, portrayed as the phase difference $\gamma_{n,1}$ of each node n to node 1: inputs can cause an oscillator to change assemblies (black arrows) or modulate its ongoing activity within the same assembly (gray arrow).

activity in all assemblies without changing their organization, or it can cause the assemblies to rearrange. Figure 1e illustrates the effects (see the caption for simulation parameters). At the beginning of the simulation, the initial phase values of each neuron will determine to which assembly each neuron will be associated. The number of assemblies (parameter M) determine the size of the attraction basin and hence the necessary input amplitude and length to cause a given node to switch assemblies. In the example, the phase of an oscillator has to be perturbed by an absolute value greater than $\pi/3$ to change to a different stable cluster. At time $t = 30$, an input of sufficient duration and magnitude is applied to one neuron, causing it to "jump" and take part in a different assembly. At iteration $t = 50$, an input of the same duration but smaller amplitude than the one at $t = 30$ perturbs the overall dynamics of the network but does not result in a change in assembly membership. Finally, at iteration 70, an input of the same duration but opposite magnitude as the first causes the related neuron to jump to another assembly. Notice, in the insert plot of Figure 1c and in the raster plot in Figure 1d, the changes in phase dynamics and spiking activity due to different forms of inputs.

2.2 Information Theory. Information theory provides a framework for quantifying and emphasizing the nonlinear relationships between variables of the system, hence its suitability in biology and robotics studies (Rieke, Warland, van Steveninck, & Bialek, 1997; Lungarella & Sporns, 2006). According to the standard definition, information is not an absolute value obtained from a measurement but rather a relative estimation of how much one can still improve on the current knowledge about a variable.

Commonly, transmitter-receiver modeling involves random variables, and the inherent uncertainty in trying to describe them is termed entropy (Shannon, 1948; Cover & Thomas, 1991). It is an intuitive notion of a measure of information, described by equation 2.4:

$$H(X) = -\sum_{x \in A} p(x) \log p(x), \tag{2.4}$$

where X is a discrete random variable defined for an alphabet A of symbols and probability mass function $p(x)$.

In experiments 1 and 2, we present different measures of information, based on the concept of entropy, to gain further knowledge on the relationship of input spike trains, neuronal responses, and motor behavior.

3 Experiment 1

Maass et al. (2005) proposed a method to evaluate the computational power and generalization capability of neuronal microcircuits independent of the network setup. In this first experiment, the model described in section 2 is used to analyze the computational performance of networks structured in various assembly sizes with diverse numbers of neurons per assembly. In the following analysis, different network configurations are obtained varying the value of M (see equation 2.2) and the initial phase of each neuron.

3.1 Methods

3.1.1 Classification Tasks, Computational Power, and Generalization Capability. Maass et al. (2005) proposed the linear separation property as a quantitative measure for evaluating the computational power of a neuronal microcircuit. The premises are that the microcircuit consists of a pool of highly recurrently connected neurons and that the information encoded in their activity can be extracted by linear readout neurons able to learn by synaptic plasticity, with no influence from readout units to the microcircuit. Although these simplifying assumptions have as impact on the biological relevance of the results, they are still valid in the face of the many uncertainties regarding electrochemical interactions in the brain and the nature of neural coding. In fact, the literature on brain-machine interface (BMI) studies has been able to show that a relatively simple linear readout unit from a reduced number of neurons is able to extract the relevant neuronal activity that relates to the action being performed (Lebedev et al., 2005). Also, Buzsaki (2010) argues that cell assembly activity can be better understood from a "reader" perspective, able to produce outputs given the ongoing activity.

Consider the model described by equation 2.2. Let us call the n size vector $\theta(t_0)$ the system state at time t_0. Now consider a neuronal microcircuit C and m different inputs $u_1, \ldots, u_m$ that are functions of time. One can build an $n \times m$ matrix M in which each column consists of the states $\theta_{u_i}(t_0)$; that is, each column consists of the phase value of each node n at time t_0 after the system has been perturbed by an input stream u_i. The rank $r \leq m$ of matrix M can then be considered as a measure of the computational power of circuit C. Based on linear algebra, the rationale is as follows: if M has rank m, a linear readout unit of microcircuit C can implement any of the 2^m possible binary classifications of the m inputs, that is, any given target output y_i at time t_0 resulting from the input u_i can be mapped by a linear readout unit (Maass et al., 2005).

Another important measure regarding a neuronal microcircuit is its ability to generalize a learned computational function to new inputs. Consider a finite set S of s inputs consisting of many noisy variations of the same input signal. One can build an $n \times s$ matrix M whose columns are the state vectors $\theta_{u_s}(t_0)$ for all inputs u in S. An estimate of the generalization capability of this circuit is then given by the rank r of matrix M. (See

Vapnik, 1998, and Maass et al., 2005, for a more complete description of the method.)

In the experiment, we evaluate oscillatory neuronal networks comprising of 80 ± 4 neurons organized in different assembly configurations. Ideally, for consistency in the comparisons among the measurements, the system should always have the same number of states across different trials; however, as we are interested on the gradient of performance when comparing different assemblies' setup, we have used architectures with a few more or somewhat fewer states for a broader set of configurations.

In this way, for a variety of possible architectures of microcircuits C, the task consists of classifying noisy variations u of 20 fixed spike patterns arbitrarily divided into two classes (0 or 1). For one randomly chosen classification task (there are 2^{20} possible classifications of the spike patterns), the objective is to train a linear readout unit to output at time $t = 4$ s the class of the spike pattern from which the noisy variation input had been generated. Each spike pattern u consisted of a Poisson spike train with a rate of 1 Hz and a duration of 4 s. Inputs are always applied to node 2 of the network, according to equation 2.2. A Euler integration time step of 0.02 s is used.

At the beginning of a simulation, 20 fixed spike patterns are generated. For each pattern, we produced 30 jittered spike trains by jittering each spike in each spike train by an amount drawn from a gaussian distribution with zero mean and standard deviation of 0.1 s. If after jittering, a spike was outside the time interval of $[0, 4]$ seconds, it was discarded. Twenty of the jittered sequences are used for training, and 10 are used for testing the performance. Figure 2a shows some examples of input spike trains and the respective jittered versions. For each simulation, we randomly classified 10 spike patterns as belonging to class 1 and 10 to class 0 (recall that there are 2^{20} possible forms of classifying the patterns).

To calculate the computational power, we generated 76 different spike patterns in the same way as for the classification task. The state vectors of the neuronal circuit at time $t = 4$ s ($\theta(t_0 = 4)$) with one of the 76 spike patterns as input were stored in the matrix M, and its rank r was estimated by singular value decomposition. To calculate the generalization performance, the procedure was similar to the one just described, but instead of using 76 spike patterns as inputs to the network, we used 38 jittered versions of two different spike patterns, following the recommendation that the number of network states should be superior to the size of S (Legenstein & Maass, 2007).

3.1.2 Redundancy and Synergy. Another insight into the activity of neuronal assemblies can be given by measurements of redundancy and synergy (Reich, Mechler, & Victor, 2001; Schneidman, Bialek, & Berry, 2003; Narayanan, Kimchi, & Laubach, 2005). In a given network composed of many interacting neurons arranged in assemblies, if the information encoded by a given pair of neurons is greater than the sum of the information

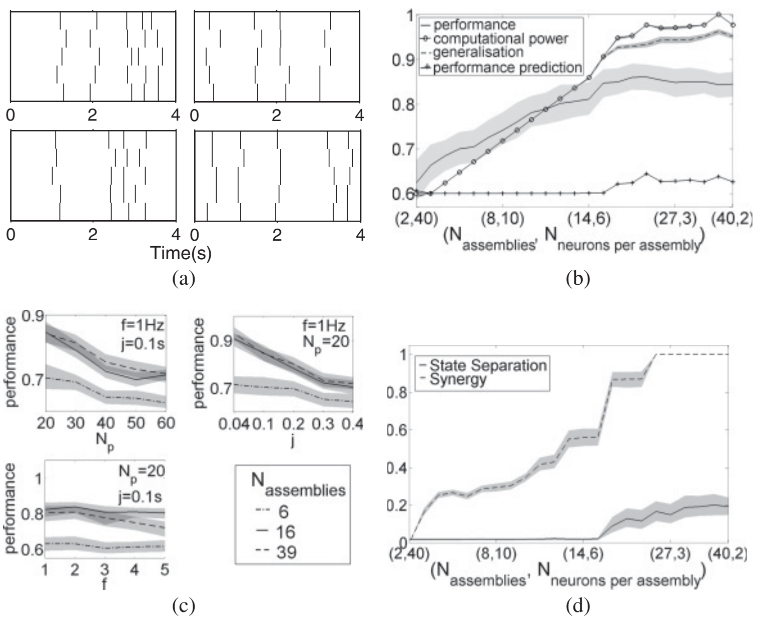


Figure 2: Simulation results for experiment 1. (a) Examples of spike trains used as inputs. In each of the four panels, five spike trains are presented: the original spike pattern (top train of each panel) and four respective jittered versions (subsequent trains in each panel). (b) Classification performance (fraction of correct classifications) obtained by architectures consisting of approximately 80 neurons arranged in a diverse number of assemblies; computational power, calculated as the value of $rank(M_{n,m})$, where each column of M is the state $\theta_{u_m}(t_4)$ of the network at time $t = 4$ s when submitted to an input u_m—the higher this value, the better a linear readout unit can discriminate among different input spike patterns (values are normalized between 0.6 and 1 to improve visualisation); generalization capability, similar to the computational power, but the inputs are now jittered versions of the same spike train—the smaller this values, the more likely the variations in a spike train will be interpreted as noise instead of consisting of a different spike train; performance prediction, calculated as the difference between the computational power and generalization capability. (c) Impact on the classification performance of three different architectures (6, 16, and 39 assemblies composed of 13, 5, and 2 neurons, respectively) caused by variations in three parameters of the input spike train (each parameter is varied while keeping the other two constant): the standard deviation of the gaussian jitter in the spike trains j (in s), the spike firing rate f (in Hz), and the number of patterns to classify N_p. (d) State separation and synergy (rescaled to vary between 0 and 2 to improve visualization). Higher values of the first indicate that the network state $\theta(t)$ reflects more details of the input stream that occurred some time in the past; higher values of the latter indicate a more synergistic (less redundant) system. All the previous results are mean values over 20 different simulations, and shaded areas are the 95% confidence interval.

encoded by the individual neurons, we say that there is a synergistic interaction; if it is less, we say that the interaction is redundant.

Consider a neuronal network with an activity set A_n of each individual neuron n composed of a states and a finite set S of s inputs. The mutual information (in bits) between the stimuli and the responses, $I(S; A)$, that is, the reduction of uncertainty about the stimuli given that the neuronal activity A is known is given by

$$I(S; A) = H(S) - H(S|A) = \sum_{s \in S} \sum_{a \in A} p(s, a) \log_2 \left[\frac{p(s, a)}{p(s)p(a)} \right]. \tag{3.1}$$

The equation for a pair of neurons is thus

$$I(S; A_1, A_2) = \sum_{s} \sum_{a_1, a_2} p(s, a_1, a_2) \log_2 \left[\frac{p(s, a_1, a_2)}{p(s)p(a_1, a_2)} \right]. \tag{3.2}$$

Given equations 3.1 and 3.2, the synergy between a pair of neurons is then defined as (Schneidman et al., 2003)

$$Syn(A_1, A_2) = \frac{I(S; A_1, A_2) - I(S; A_1) - I(S; A_2)}{I(S; A_1, A_2)}. \tag{3.3}$$

Notice that if the mutual information between the two neurons is 0, that is, if they have unrelated activity, equation 3.2 reduces to $I(S; A_1, A_2) = I(S; A_1) + I(S; A_2)$, and the synergy value given by equation 3.3 is 0. Synergy varies from -1, if the interaction between the neuronal pair is completely redundant, to 1, when the information conveyed by the pair activity is greater than the information conveyed individually by the neurons.

To estimate the synergy value, stimuli consisted of eight noisy variations of eight different spike patterns, lasting for 200 iterations and with the same characteristics as detailed before, and the neuronal activity A_n is the phase value of neuron n at the end of simulation. We performed 20 experiments for each pair of neurons, and 10 different randomly chosen pairs were used. The results were then averaged. Sets S and A were discretized into eight equiprobable states, which improves the robustness of the statistics (Marschinski & Kantz, 2002), and finally the joint probabilities associated with the information-related measures were estimated using histograms (Lungarella, Ishiguro, Kuniyoshi, & Otsu, 2007). In this way, at the end of each experiment, a table whose columns are all the possible combinations of $[a_1, a_2]$ and whose lines are all the possible stimuli $s_1, \ldots, s_8$ is formed, and each field of this table contains the probability $p(a_1, a_2|s_n)$, from which the synergy calculations were performed (see equations 3.1 to 3.3). An important point to stress is that ideally, we should have tested all possible neuron pairs and assembly combinations, but that would have been computationally prohibitive. Nevertheless, considering the standard deviations observed in the experiments that follow, we believe the results are informative.

3.2 Results. Figure 2b shows the results for the classification performance, the computational power, and the generalization capability of the system. Notice the increase in performance as one moves from networks with fewer assemblies (and more neurons per assembly) to architectures constituted by many assemblies (and few neurons per assembly). The computational power and the generalization capability have the same values until a critical architecture is reached, after which they start to behave differently. Recall that both measures are based on calculations of matrices' ranks, which indicate the maximum number of linearly independent rows or columns (whichever is smaller). With just a few assemblies composed of several neurons, each assembly works as a single large oscillator, and interassembly modulations due to external perturbations are minimum. The rank value thus is directly connected with the number of assemblies in the system. As the assemblies increase in number and decrease in size, interassembly modulations become more prominent, and this is captured by the rank of the state matrix. The results therefore indicate that networks with more assemblies have the potential to classify a greater number of input patterns. In contrast, the greater the value of the rank of the state matrix M, the worse the generalization of the circuit is likely to be, which means that small perturbations in spike times for a given spike pattern tend to be classified as belonging to a different spike pattern.

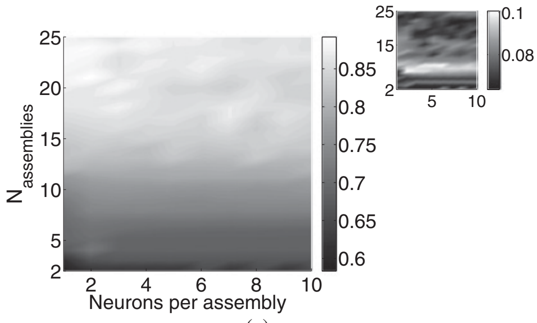
Maass et al. (2005) and Legenstein and Maass (2007) showed that combined, the above two measures may provide a good estimate of the computational capabilities of a given neuronal microcircuit and may also be used to predict its performance in a classification task. There is no ultimate method for combining them both, but simply using the difference between the computational power and the generalization performance can be a good indicator. Figure 2b shows the result. Due to the properties of the model, the matrix ranks calculated for each measure differ only for architectures with a higher number of assemblies; the prediction of computational performance therefore is applicable only for a subset of all possible configurations of our model. Nevertheless, the prediction points at the correct region of possible architectures where performance is maximum.

Consider Figure 2c, which shows the response of three different network configurations to variations in some simulation parameters: the number of different spike patterns presented to the network for classification, the frequency of the input spike trains, and the noise rate used to generate the jittered spike trains. Notice that increasing the value of the first or the latter results in a fall in performance, while the performance peaks at an intermediate value of the input frequency. This shows that classifying 60 different patterns (2^{60} possible classifications) is harder than classifying 20 using the same framework. Also, the relatively small variation in performance due to the input frequency indicates that the model has a good spike pattern discrimination time resolution.

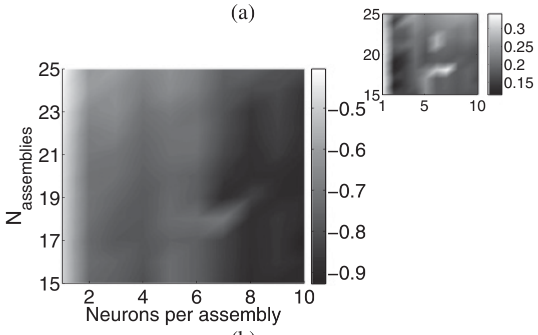
Not surprisingly, noisier spike trains result in more classification mistakes, for an otherwise noisy train is now viewed as a different spike pattern, but notice that the drop in performance is sharper for networks composed of more assemblies (22.6% for a network with 40 assemblies in contrast with a 10% fall for a network with 10 assemblies), in agreement with what the generalization analysis predicted. One of the reasons this might occur is illustrated in Figure 2d. It shows the state separation of the system, a measure that captures how much the state $\theta(t)$ of one network reflects details of the input stream that occurred in the past. Consider two input patterns u and v over 3000 iterations that differ only during the first 1000, with the same properties as described before. The state separation is given by $\|\theta_u(t) - \theta_v(t)\|$ for $t = 3000$. Notice that the architectures with fewer assemblies have a lower value of state separation than the ones with more assemblies, which means that perturbations caused by earlier input differences persist more in the latter configurations. For noisy spike trains, such amplified differences may have an impact on the overall pattern classification performance. The results also highlight that networks with more assemblies are affected more by inputs, which can be explained considering that in the latter case, the network state $\theta(t)$ at a given time t is less influenced by the activity of a single neuron and more a product of the whole network interaction.

The synergy analysis confirms this last point (see Figure 2d). Notice that for architectures with fewer assemblies, the level of redundancy is high ($Syn = -1$, equation 3.3), but it reduces as the number of assemblies grows. The vast majority of networks with higher numbers of assemblies present information independence ($Syn = 0$), that is, the information conveyed by the pair of neurons is the sum of the information they convey separately. Importantly, Schneidman et al. (2003) make the point that information independence may relate to neurons being responsive to different features of the stimulus, but the synergy measurement reflects an average over the whole set of stimuli S; for that reason, the neuronal pair may be redundant, synergistic, or independent for different subsets of S. Also, Reich et al. (2001) found neuronal pairs in nearby cortical neurons presenting varied forms of interactions—more specifically, independent and redundant interactions. Thus, the results portrayed in the figure may vary depending on which pair is recorded and the measurements may be a result of averaging, not from independence, across the whole trial.

In the results above, we investigated networks with roughly the same number of states ($N_{assemblies} \times N_{neurons/assembly} \approx constant$). This constraint had to be imposed in order for the calculations of computational power and generalization capability to hold. However, another interesting aspect of assembly computations is how performance and synergy change as one



(a)



(b)

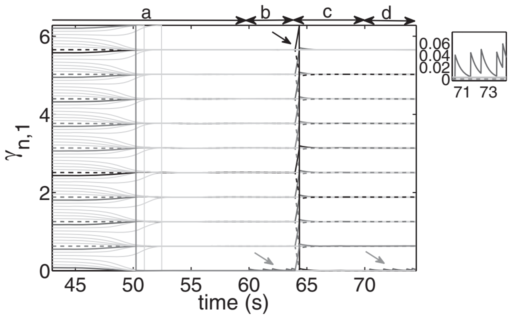
Figure 3: Simulation results for experiment 1. (a) Classification performance and (b) synergy values (not normalized) for different network configurations. In contrast to figure 2, in which all architectures had approximately 80 neurons, here the numbers of assemblies and neurons are varied independently. Results are mean values over 20 different simulations, and the small plot within each figure is the standard deviation.

varies the number of neurons within each assembly for a given number of assemblies in the network. Figure 3a shows the results. Performance is predominantly higher in networks with more assemblies, regardless of the number of neurons within each. In other words, performance increases as the number of assemblies increases, but given a certain network with a fixed number of assemblies, adding neurons to each assembly does not cause a salient increase in performance (e.g., networks with 2 assemblies with one or five neurons within each assembly have a classification performance of approximately 0.6, whereas networks with 20 assemblies with one or five neurons in each cluster have a classification performance of approximately 0.85). In contrast, the level of redundancy or independence is related mainly to the number of neurons within each assembly, regardless of the total number of assemblies (see Figure 3b). For example, neurons in a network with 20 assemblies with 1 neuron in each assembly present a much more independent activity than neurons in a network with 20 assemblies composed of 10 neurons each. This is in accordance with results obtained in motor cortex studies, which show that the synergistic or redundant interactions depend on the size of each neuronal assembly, and redundancy increases with the size of assemblies (Narayanan et al., 2005).

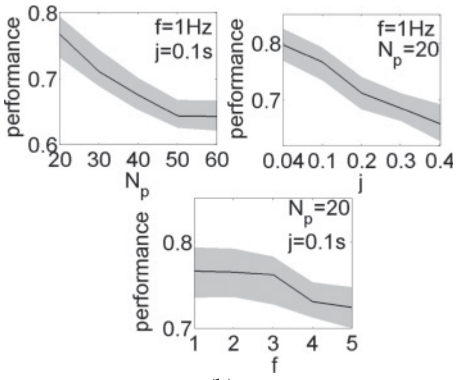
Recall that assemblies are formed by their phase relationship, that is, two neurons belong to the same assembly only if they are synchronized with near-zero phase lag. In this sense, the synchronization properties of the network (dictated by the phase interaction function defined in equation 2.3) make the dynamics of neurons constituting the same assembly similar and the dynamics of neurons constituting different assemblies dissimilar. Thus, increasing the number of assemblies, not the number of neurons, has an impact more on performance, and that is possibly due to an increase in entropy. However, the rationale is not simple because of nonlinear effects and the intrinsic dynamics of the network responding to inputs. Notably, the Kuramoto model presents second-order phase transitions, and a given node can influence in different ways other nodes in the network, depending on the relationship between natural frequencies and on whether nodes are directly connected. Some of these effects may be in place, given the saturation in performance and the nonlinear impact on classification to adding assemblies or neurons to the network. In this sense, it is not trivial that adding neurons maximizes the classification performance because this is determined by the way these neurons are organized (assemblies) and limited by nonlinear effects (highlighted by the saturations depicted in Figures 2b and 3a).

To conclude experiment 1, we explore the experimental evidence (Steinmetz, Roy, Fitzgerald, Hsiao, & Johnson, 2000; Lakatos, Karmos, Mehta, Ulbert, & Schroeder, 2008), which suggests that attentional mechanisms can promote phase resetting and modulate the ongoing neuronal oscillations to respond differently to stimuli to investigate whether the system can cope with multiple tasks by just relying on the phase dynamics, without any changes in the readout unit after training. Therefore, as described in section 2, we manipulate the phase relationship between nodes (emulating attention mechanisms) and investigate the performance in opposite versions of a classification task. The network architecture has been arbitrarily chosen to have 10 clusters of 8 neurons each, with similar results obtained for other configurations.

To begin, we present to the network spike patterns that have to be classified. At the end of each pattern presentation, the network state $\theta(t)$ is stored, representing the system's response for this given input. After all the patterns are shown, the phase relationships in the network are reorganized (see Figure 4a). Then we present the same spike patterns once more and the network state is stored, but the corresponding classification label (1 or 0) for each pattern is made exactly the opposite from the ones previously used. Finally, an output readout unit is trained by linear regression using the network state and the desired classification label for each pattern. Essentially



(a)



(b)

Figure 4: Multiple classification task with dynamic network reconfiguration in an architecture of 10 assemblies with 8 neurons each. (a) Phase dynamics portrayed as the phase difference $\gamma_{n,1}$ of each node n to node 1. Dark gray solid and dashed lines indicate nodes that change assembly membership (the black arrow shows the moment of change). The solid black line depicts the phase behavior of node 2, which has its ongoing phase dynamics modulated by the spike train input that has to be classified. Light solid gray lines relate to the remaining nodes. Phase a shows the washout phase, when the phase relationships stabilize in 10 clusters of 8 neurons each, phase b shows the dynamics during the classification task, phase c comprises the reorganization of the assemblies, and phase d shows the classification task (same input train, opposite classification labels). Assemblies are rearranged by perturbing the phase of a given node with an input of magnitude ±1.7 for 20 iterations. Light gray arrows point at the perturbation caused by the input in node 2 (detailed in the small plot). (b) Impact on the classification performance caused by variations in three parameters of the input spike train (notation similar to Figure 2c): the standard deviation of the gaussian jitter in the spike trains j, the spike firing rate f, and the number of patterns to classify N_p.

the procedure replicates the previous experiment, but with the classification task changing with a reorganization of the nodes' phase relationship.

Figure 4a presents the resulting network dynamics. First, the system goes through a washout phase (3000 iterations) and has its phase activity stabilized in 10 clusters of 8 neurons each. Then a spike train input to node 2, lasting 200 iterations, modulates the phase dynamics; at the end, the final network state is stored. In sequence, the phase relationships are rearranged by inputs to certain nodes, and the same classification procedure is executed, with the network state stored at the end. This process is performed for every spike pattern used for training, and finally the readout unit weights are calculated. Figure 4b shows the network performance obtained for different parameter configurations. Notice that the performance is comparable to the one obtained in the previous task (see Figure 2b), which suggests that the phase reorganization dynamics can be exploited to solve different tasks without the need for adaption or plasticity mechanisms at the readout unit level.

The results for experiment 1 suggest that neuronal assemblies and phase reorganization dynamics can play a significant part in supervised classification tasks and, perhaps most relevant to cognition, can cope with multiple classification tasks without the need for additional adaptive mechanisms. However, the major part of (natural) neural and cognitive dynamics is bound up in the generation of unsupervised embodied behavior. Hence, in order to explore the possible roles of neuronal assembly dynamics further, the properties of the model were investigated in a second experiment in which it was used in an unsupervised embodied learning scenario, as described in the next section.

4 Experiment 2

Plasticity mechanisms are a common feature in the brain and mediate many (if not all) cognitive processes during learning and development (Turrigiano & Nelson, 2004; Masquelier et al., 2009). There is a rich literature exploring models of artificial neuronal networks with some kind of synaptic plasticity in the context of real or simulated agents engaged in a behavioral task (Urzelai & Floreano, 2001; Sporns & Alexander, 2002; Di Paolo, 2003; Edelman, 2007; Shim & Husbands, 2012), but normally the techniques involve the modulation of the electric connections between nodes of the network as a response to the agent's actions and the environment. Here, we explore the way in which neurons and assemblies relate to each other and how a modulation of this relationship alone, without other plasticity mechanisms, can be exploited to generate adaptive behavior.

We conduct the analysis following an evolutionary robotics (ER) approach, where an evolved simulated robotic agent controlled by a variation of the system investigated in experiment 1 has to solve multiple tasks. In the following sections, we first present the concepts of transfer entropy, an information-theoretic measure used to analyze the results; then we explain the ER approach, the robotic model used, and the control system framework, and the unsupervised learning task, and conclude with the outcomes of the experiment.

4.1 Methods

4.1.1 Transfer Entropy. Agent-environment systems pose extra challenges in devising and interpreting a sensible measurement of information flow, for they normally have noisy and limited data samples, asymmetrical relationships among elements of the system, and temporal variance (i.e., sensory and motor patterns may vary over time). Transfer entropy (TE) (Schreiber, 2000), in this context, is suggested as a suitable and robust information-theoretic tool (Lungarella, Ishiguro et al., 2007; Lungarella, Pitti, & Kuniyoshi, 2007), and has also been applied to investigate real neuronal assemblies and other neuroscience problems (Borst & Theunissen, 1999; Gourévitch & Eggermont, 2007; Buehlmann & Deco, 2010; Vicente, Wibral, Lindner, & Pipa, 2011); it will thus be used in our analysis.

TE is based on classical information theory and allows one to estimate the directional exchange of information between two given systems. The choice of TE in this work is based on a study conducted by Lungarella, Ishiguro et al. (2007), who compared the performance of different IT tools in bivariate time series analysis, which will be the case here, and concluded that TE is in general more stable and robust than the other tools explored. The next paragraphs describe the technique.

Consider two time series, $X = x_t$ and $Y = y_t$, and assume they can be represented as a stationary higher-order Markov process. Transfer entropy calculates the deviation from the generalized Markov property $p(y_{t+1}|y_t^n, x_t^m) = p(y_{t+1}|y_t^n)$ where $x_t^m \equiv (x_t, x_{t-1}, \ldots, x_{t-m+1})^T$, $y_t^n \equiv (y_t, y_{t-1}, \ldots, y_{t-n+1})^T$, and m and n are the orders of the higher-order Markov process (note that the above property holds only if there is no causal link between the time series). Schreiber (2000) defines transfer entropy as

$$TE(X \to Y) = \sum_{y_{t+1}} \sum_{x_t} \sum_{y_t} p(y_{t+1}, x_t^m, y_t^n) \log \frac{p(y_{t+1}|y_t^n, x_t^m)}{p(y_{t+1}|y_t^n)}. \qquad (4.1)$$

Therefore, from equation 4.1, one can estimate the information about a future observation y_{t+1} given the available observations x_t^m and y_t^n that goes beyond the information of the future state y_{t+1} provided by y_t^n alone. It is thus a directional, nonsymmetrical estimate of the influence of one time series on another.

The original formulation of transfer entropy suffers from finite sample effects when the available data are limited, and the results obtained may not be correctly estimated. To attenuate these limitations, Marschinski and

Kantz (2002) introduced an improved estimator, effective transfer entropy (ETE), which is calculated as the difference between the usual transfer entropy (see equation 4.1) and the transfer entropy calculated after shuffling the elements of the time series X, resulting in the following equation:

$$ETE(X \rightarrow Y) \equiv TE(X \rightarrow Y) - TE(X_{shuffled} \rightarrow Y). \tag{4.2}$$

The ETE formulation is the one used in this letter. We adopt the orders of the Markov processes as $m = n = 1$ (see equation 4.1), and the conditional probabilities are calculated by rewriting them as joint probabilities, which are then estimated using histograms.

4.1.2 Evolutionary Robotics. Evolutionary robotics (ER) is a relatively new field of interdisciplinary research grounded in concepts from computer science and evolutionary biology (Harvey, Di Paolo, Wood, Quinn, & Tuci, 2005; Floreano, Husbands, & Nolfi, 2008; Floreano & Keller, 2010). Originally devised as an engineering approach to automatically generate efficient robot controllers in challenging scenarios, where traditional control techniques have limited performance, ER is now well regarded among biologists, cognitive scientists, and neuroscientists, as it provides a means to simulate and investigate brain-body-environment interactions that underlie the generation of behavior in a relatively unconstrained way, thus penetrating areas that disembodied studies cannot reach.

Consider a real or simulated robot, with sensors and actuators, situated in an environment with a certain task to accomplish. Each solution candidate (individual) is represented by a genotype, which contains the basic information of the agent's body or its controller's parameters (e.g., the number of wheels the robot has or the values of the weights of an artificial neuronal network acting as its controller). According to some criteria, normally the previous performance of that individual in solving the task (fitness), parents are selected and undergo a process of mutation and recombination, generating new individuals, which are then evaluated in the task. This process is repeated through the generations, eventually obtaining individuals with a higher performance in the given task.

In this sense, ER is a reasonable approach to studying embodied and situated behavior generation because it can be used as a powerful model synthesis technique (Beer, 2003; Husbands, 2009). Relatively simple, tractable models can be produced and studied in the context of what have been called minimally cognitive tasks (Beer, 2003), which are tasks that are simple enough to allow detailed analysis and yet are complex enough to motivate some kind of cognitive interest.

4.1.3 Robotic Model. The robot is based on the Khepera II model (K-Team Corporation). It has two wheels with independent electric motors,

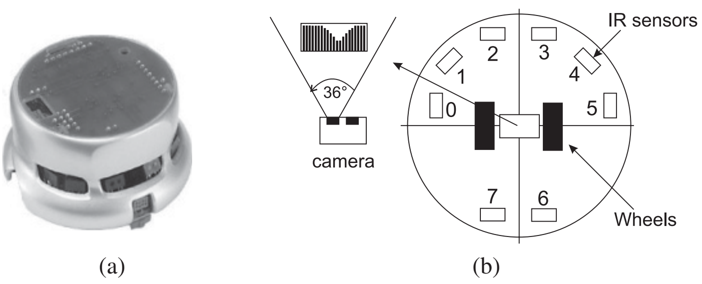


Figure 5: (a) Real Khepera II robot and (b) its schematic representation, including the IR sensors and the camera.

eight infrared sensors, and a camera (see Figure 5). The sensors measure the environmental luminosity (ranging from 65 to 450–65 being the highest luminosity that can be sensed) and the distance to nearby objects (ranging from 0 to 1023—the latter value represents the closest distance to an object). The camera provides a 36 degrees 64 pixel gray-scale horizontal image from its field of view. These 64 pixels are grouped into three mean inputs for the system: the mean value of pixels 0 to 13 representing the left reading, the mean value of pixels 24 to 39 representing the central reading, and the mean value of pixels 48 to 63 representing the right reading. The readings range from 50 to 175—the first value representing the maximum perception of a black stripe. In all experiments, a sensorimotor cycle (time between a sensory reading and a motor command) lasts 400 ms. The KiKS Khepera robot simulator was used (Storm, 2004); it simulates with great fidelity motor commands and noisy sensory readings that are observed in the real robot.

4.1.4 Framework. The model studied in experiment 1 was adapted so that it could be applied to control a simulated robotic agent. The framework, illustrated in Figure 6, is composed of 12 fully connected oscillators, with some nodes connected to the robot's noisy sensors (1 sensor per node). The rationale for a network with 12 nodes relates to richer dynamical behaviour in the Kuramoto model with this number of nodes (Popovych, Maistrenko, & Tass, 2005). The frequency of each node is the sum of its natural frequency of oscillation, w_n, and the value of the sensory input related to that node (0 if there is no input), scaled by a factor ϵ_n. The natural frequency w_n can be associated with the natural firing rate of a neuron or a group of neurons, and the sensory inputs mediated by ϵ_n alter its oscillatory behavior according to environmental interactions, thus improving the flexibility of the model to study neuronal synchronization (Cumin & Unsworth, 2007) within a behavioral context.

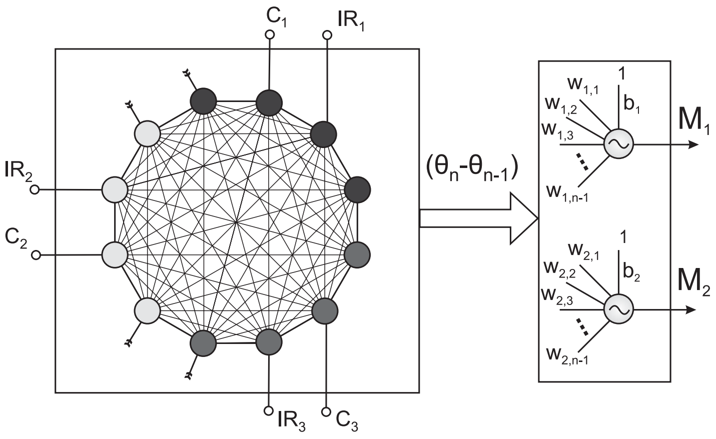


Figure 6: Framework for application in evolutionary robotics. The oscillatory network is composed of 12 fully connected neuronal oscillators, with nodes 2, 6, and 10 connected to the robot's infrared sensors and nodes 3, 7, and 11 connected to the visual sensors. Nodes 4, 5, 8, and 9 receive internal inputs only. The phase differences $\theta_n - \theta_{n-1}$, $n = 2, \ldots, 12$, plus a bias term, are linearly combined by a weight matrix W and fed into two nonlinear output units that have as an activation function the sin function, which can be interpreted as two output neurons that capture the ongoing network activity. The activation of each output neuron is used to command the motors M_1 and M_2.

At each iteration, the phase differences γ from a node n to nodes $n-1$, $n = 2, \ldots, 12$, are calculated following equation 2.2. Then the phase differences plus a bias term are linearly combined by a weight matrix W and fed into two nonlinear output units that have as an activation function the *sin* function, which can be interpreted as two output neurons that capture the ongoing network activity. According to Pouget, Dayan, and Zemel (2008), nonlinear mappings (such as the one developed here) can be used as a comprehensive method to characterize a broad range of neuronal operations in sensorimotor contexts. The calculation results in two signals that will command the left and right motors of the agent,

$$\boldsymbol{M} = \sin(\boldsymbol{W}'\boldsymbol{\gamma}), \tag{4.3}$$

where $\boldsymbol{M} = [M_1, M_2]^T$ is the motor state space, with M_1 corresponding to the left motor command and M_2 to the right motor command.

In this way, the phase dynamics and the environmental input to the robotic agent will determine its behavior. It is important to stress that nodes that receive no input participate in the overall dynamics of the network; hence, their natural activity can modulate its global activity.

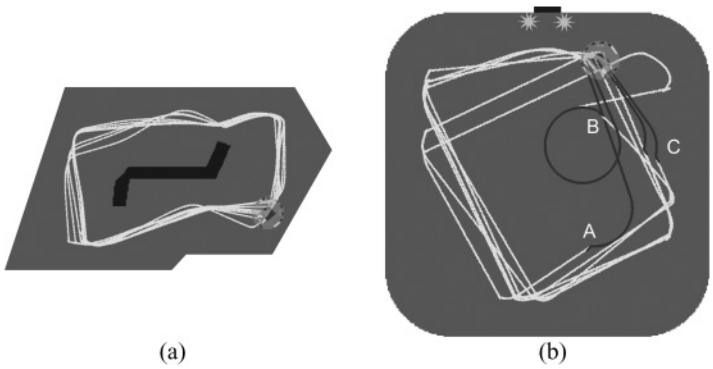


Figure 7: Scenarios used in experiment 2. (a) Obstacle avoidance training scenario and the behavior displayed by a successfully evolved individual. (b) Task scenario. The arena has a recharging area represented by two light sources located next to a black stripe landmark (central top part of the figure). Light gray and black trajectories show the robot's behavior when controlled by different assembly configurations.

4.1.5 Task. The robot described in section 4.1.3 has two main objectives: it has to explore the environment while avoiding collisions (O_1), and it has to ensure that its battery level remains above a threshold (O_2), actively searching for the recharging area otherwise. The environment is a square arena with a recharging area represented by two light sources located next to a black stripe tag (see Figure 7(b)). Whenever the robot's light sensory readings are below 100, it is considered to be inside the recharging area. The battery level *BL* dynamics is given by

$$BL(t+1) = \begin{cases} BL(t) - \alpha(BL(t) - \mathrm{Min}(BL)), & \text{if ROCA} \\ BL(t) + \beta(\mathrm{Max}(BL) - BL(t)), & \text{if RICA} \end{cases}, \tag{4.4}$$

where α and β control the battery consumption and recharge rate, respectively, $\mathrm{Min}(BL)$ and $\mathrm{Max}(BL)$ are the lower and upper limit of *BL* (set here to 0 and 100, respectively), and *ROCA* and *RICA* stand for *Robot Outside Charging Area* and *Robot Inside Charging Area*.

The network consists of $N = 12$ neurons with initial phases uniformly distributed in $[0, 2\pi)$. Nodes 2, 6, and 10 are connected to the robot's infrared distance sensors, and nodes 3, 7, and 11 are connected to the camera sensors. We set $\epsilon_n = 4$ for $n = 4, 8$ and $\epsilon_n = -4$ for $n = 5, 9$ (for $n = 2, 3, 6, 7, 10, 11, \epsilon_n$ is evolved, and 0 otherwise (see the next section for details). We also adopted $M = 3$ (see equation 2.3), which leads to the formation of three assemblies with four neurons in each, denoted (1, 2, 3, 4)-(5, 6, 7, 8)-(9, 10, 11, 12) (see Figure 8a).

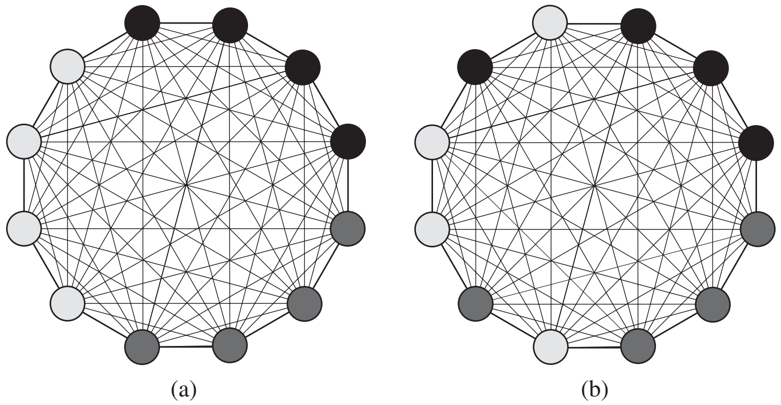


Figure 8: Network assembly structure used in experiment 2. Nodes with same shadings are synchronized. (a) Configuration at the beginning of the experiment. (b) Configuration after an internal signal (caused by the drop of the battery level below 15) changes the assemblies' setup.

Whenever the battery level drops below 15 ($t = t_{low}$), an internal signal is generated that reorganizes the network in terms of neuronal synchronization (see Figure 8b). This signal consists of an input lasting 400 ms (the same duration of a sensorimotor cycle) applied to nodes 4, 5, 8, and 9, that is, considering equation 2.2, $I_{4,5,8,9}(t) = 1$ for $t_{low} \leq t \leq t_{low} + 0.4$, 0 otherwise. Given the setup of the network described in the previous paragraph, this input shifts the phase of oscillators 4 and 8 and lags the phase of oscillators 5 and 9 enough to move them from their original basin of attraction to a neighboring assembly. The network final configuration is thus (1, 2, 3, 5)-(4, 6, 7, 9)-(8, 10, 11, 12). Whenever the battery level increases above 95 ($t = t_{high}$), an internal signal $I_{4,5,8,9}(t) = -1$ for $t_{high} \leq t \leq t_{high} + 0.4$, 0 otherwise, brings the network back to its original configuration (see Figure 8a). Recall, from the task description in the previous paragraph that the goal is to investigate if dynamic assembly formation can underpin the coordination of different, possibly conflicting behaviors in an autonomous agent.

Tasks are conflicting in the sense that the first (O_1) requires the agent to move and explore while minimizing sensory readings (and hence avoiding collisions), whereas in the latter (O_2), it has to approach a certain area of the environment and maximize the inputs from the light sensors (to recharge) while suppressing its movement to increase the time spent in the charging area (until the battery is recharged above the threshold).

4.1.6 Genetic Algorithm. We used a geographically distributed genetic algorithm with local selection and replacement (Husbands, Smith, Jakobi, &

O'Shea, 1998) to determine the parameters of the system: the input weights $\epsilon_n \in [-0.5, 0.5]$, $n = 2, 3, 6, 7, 10, 11$, and the two output neurons' weights, $W_{N,o}$, $o = 1, 2$, with elements in the interval $[-5, 5]$, resulting in a genotype of length 30.

The network's genotype consists of an array of integer variables lying in the range [0, 999] (each variable occupies a gene locus), which are mapped to values determined by the range of their respective parameters. For all the experiments in this letter, the population size was 49, arranged in a 7×7 toroidal grid. There are two mutation operators: the first operator is applied to 20% of the genotype and produces a change at each locus by an amount within the $[-10, +10]$ range according to a normal distribution. The second mutation operator has a probability of 10% and is applied to 40% of the genotype, replacing a randomly chosen gene locus with a new value within the [0, 999] range in an uniform distribution. There is no crossover.

In a breeding event, a mating pool is formed by choosing a random point in the grid together with its eight neighbors. A single parent is then chosen through rank-based roulette selection, and the mutation operators are applied, producing a new individual, which is evaluated and placed back in the mating pool in a position determined by inverse rank-based roulette selection. (For further details about the genetic algorithm, see Husbands et al., 1998.)

During evolution, we adopted a shaping technique (Dorigo & Colombetti, 1998; Bongard, 2011), in which the robot is required to execute and succeed in one task environment before proceeding to more complex scenarios. This technique has been shown to improve the evolvability of controllers in tasks that involve the accomplishment of many different objectives.

Therefore, considering the task previously described, the first phase of evolution, phase 1, consists of 800 iterations of the algorithm where the fitness f is defined as the robot's ability to explore the environment while avoiding collisions with the environment walls and obstacles (see equation 4.5, based on Floreano & Mondada, 1994). Note that there is no influence of the battery level in this first stage of evolution. Figure 7a depicts the training scenario.

$$f = V(1 - \sqrt{\Delta v})(1 - i), \tag{4.5}$$

where V is the sum of the instantaneous rotation speed of the wheels (stimulating high speeds), Δv the absolute value of the algebraic difference between the speeds of the wheels (stimulating forward movement), and i is the normalized value of the distance sensor of highest activation (stimulating obstacle avoidance).

A generation is defined as 10 breeding events, and the evolutionary algorithm runs for a maximum of 300 generations. If, at the end of this first evolutionary process, the agent attains a fitness above 0.4, it can proceed to the next phase.

During phase 2 (the scenario depicted in Figure 7b), robots are evaluated according to their ability to avoid collisions and the time they spend with the battery level below the threshold, that is, if the battery level is above 15, fitness is scored following equation 4.5; otherwise, fitness is given by the fraction of time it took the robot to recharge its battery above 95:

$$f = \begin{cases} V(1-\sqrt{\Delta v})(1-i), & \text{if } BL(t) \geq 15 \\ 1 - t_b/T, & \text{if } BL(t) < 15 \text{ and } BL(t+\tau) < 95 \end{cases}, \tag{4.6}$$

where V, Δv and i, and BL are as described in equations 4.5 and 4.4, respectively; t_b is the number of iterations the robot spent with its battery level below 15; and T is the number of iterations counting from the moment the battery dropped below 15 until it reached a level above 95.

At each iteration of the trial, the corresponding fitness value is calculated, and the final fitness is given by the mean fitness obtained across the whole trial. Notice that there is a selective pressure toward agents that reach the recharging area as fast as possible and remain in the area until the battery is recharged. Also, the learned behavior in phase 1 cannot be completely overwritten in phase 2, as part of the evaluation function still accounts for the robot's ability to avoid collisions and explore the environment. Importantly, the second part of the fitness function described in equation 4.6 does not reward a specific sequence of actions, only the final behavior of the robot (reach the recharging area as fast as possible and remain there until recharged). There is no influence of the light or camera sensors in the calculations; thus, the robot has to associate the distance sensors and vision information to find the area where recharge occurs.

4.2 Results. Robots successfully evolved to execute phases 1 and 2. Figure 7a portrays one of the evolved agents that navigates throughout the environment while avoiding collisions (phase 1). Notice that because the agent is surrounded by walls and obstacles, sensory readings are nearly always present; thus, the maximum fitness obtained is less than the maximum 1. Figure 7b shows the same agent after phase 2 of evolution, and Figure 9 shows the spiking activity (based on the phase dynamics) for every node of the network, the battery level, the distance and camera sensors, and the motor commands. The next paragraphs explore in detail the results of this latter phase.

At the beginning of the task, the agent wanders around the environment in straight lines, adjusting its trajectory only when faced by a wall. Notice from Figure 9 that both motor outputs are close to the maximum value of 10 and change only in response to the distance sensors' stimuli; the motors remain unresponsive to changes in the camera input (recall that the network

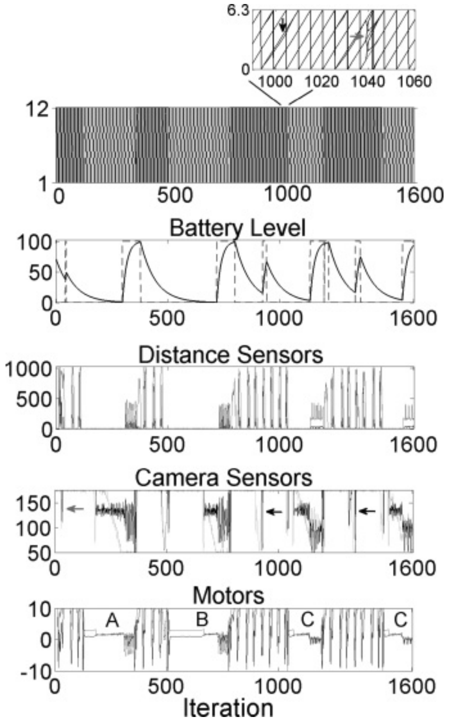


Figure 9: Experiment 2 variables dynamics. From the top down: network raster plot (the dark and light gray shaded areas relate to different network configurations, the small plot shows details of the phase dynamics, the black arrow shows how sensory stimulus modulates the ongoing dynamics, and the gray arrow points at a moment of assembly reorganization); the battery level, with dashed lines indicating when the agent is within the recharging area (0 is outside, 100 is inside); the distance sensors (0 when there is no obstacle, 1023 if very close to one); the camera sensors (175 if no black stripe is seen, 50 if all the camera pixels detect black); and the motor commands (positive values indicate forward movement—backward movement otherwise). Letters A, B, and C refer to the trajectories displayed in Figure 7b.

receives input from all sensors at all times, and there are no ontogenetic plasticity mechanisms). Incidentally, the robot passes near the recharging area (the gray arrow near iteration 50), but because its battery level is still above the threshold, the predominant behavior remains "explore and avoid collisions."

However, near iteration 300, the battery level drops below 15 (point A in Figures 7b and 9), which triggers an internal signal that reorganizes the network configuration. The agent now should stop exploring the environment and drive toward the recharging area as fast as it can to maximize its fitness. Notice that at the moment this occurs, the robot is far from the recharging area, so it has to use its visual information to orient and move toward the correct direction. The adopted strategy is to move in circles until the visual stimulus (the black landmark) is perceived and then progress in a straight line toward it. This can be seen in Figure 9, with the consistent camera readings. As the robot approaches the landmark, the distance sensory readings increase but do not cause the same response as before the network reorganization: the agent remains relatively still within the recharging area until the battery is recharged and does not display the characteristic turn around movement of obstacle avoidance. After the battery is above 95, another internal signal is triggered, and assemblies are rearranged to their previous state. The robot returns to explore and avoid obstacles.

This same sequence of behavior can be observed near iteration 500 (point B in Figures 7b and 9). The turning behavior, brought about by differential wheel speeds, is much more noticeable here, and although the robot is closer to the recharging area, it does not have the visual stimulus at the time the battery drops below 15. When the first visual stimulus is perceived, the agent fixates on the landmark, and the sensory readings increase as it slowly moves toward the black stripe. A similar sequence of behavior is displayed as the task continues, but the following moments when the battery drops below 15 occur when the agent has a visual stimulus; therefore, there is no need to move in circles before heading toward the recharging area (points *C*).

Figure 10 depicts the activity of the assemblies during the task, represented by the phase difference $\gamma_{n,1}$ of each node n to node 1. The oscillators rapidly synchronize and form three neuronal assemblies equally spaced according to their phase differences (see Figure 8). Each assembly has inputs from one distance sensor and one camera sensor. This sensory stimuli modulate the ongoing network activity, causing small phase deviations from the respective assembly's mean phase (examples are indicated by light gray arrows in Figure 10), yet all nodes remain within the basin of attraction of their respective cluster; they do not change their assembly membership. The small phase modulations of each cluster are captured by both output neurons and are responsible for adjusting the agent's motor commands and, consequently, its trajectory. Internal signals triggered by the battery-level dynamics change the assemblies' original arrangement (compare with Figure 9, top), and this new phase relationship, together with the sensory modulation, accounts for the change in the robot's behavior.

The relationship between the rearrangement of the assemblies and the different behaviors displayed by the agent can be seen by plotting the network's phase differences together with the corresponding motor outputs at every sensorimotor time step. Because there are $\gamma_{i,i-1} = 11$ phase differences, we have to perform a dimension reduction to visualize the system's dynamics. This is done by projecting the original phase differences into the first two principal components calculated using a principal component

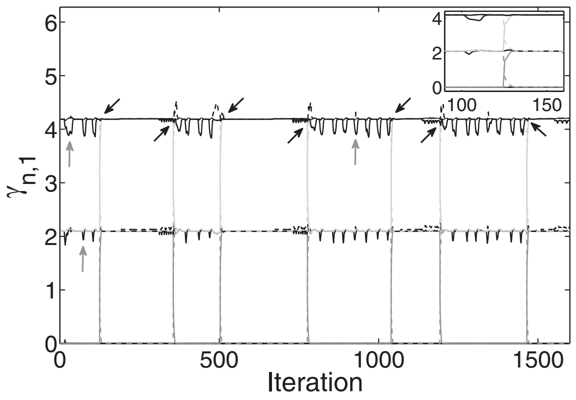
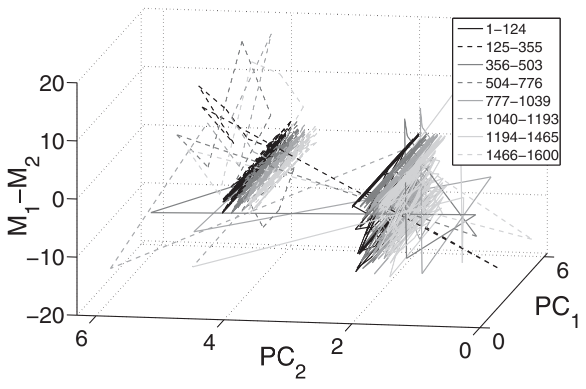


Figure 10: Phase dynamics portrayed as the phase difference $\gamma_{n,1}$ of each node n to node 1. The inset shows the moment of an assembly reorganization (solid and dashed light gray and dark gray lines show the change in assembly membership). Gray arrows point at examples of phase modulations due to sensory stimuli; black arrows indicate moments of assembly reorganization. Notice how sensory stimuli modulate the ongoing dynamics in the whole network but ultimately do not cause an assembly change.

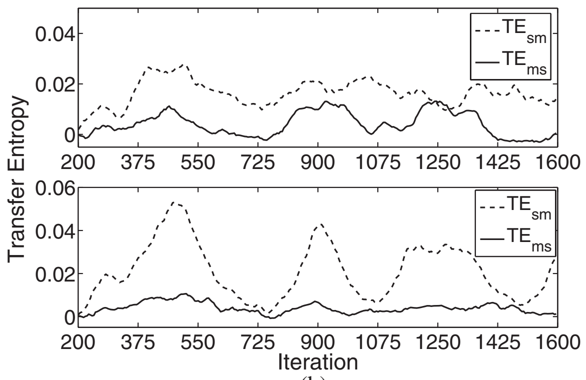
analysis (PCA) (Jolliffe, 2002). A single time series is obtained from the two motor commands by subtracting the left from the right wheel commands. Figure 11a shows the results. Note that there are two clearly discernible regions in the state-space—one comprising trial iterations 1–124, 356–503, 777–1039, and 1194–1465, and the other iterations 125–355, 504–776, 1040–1193, and 1466–1600. These regions relate to different assembly configurations (see Figure 9, top, and Figure 10); therefore, rearranging the assemblies causes movement in the state-space of the network motor system, which has a direct correspondence with the behavior of the robot.

The assembly reconfiguration is the main mechanism responsible for changing the way the robot behaves; there are no other plasticity mechanisms, and both distance and visual sensors are always fed into the network. The effects of the inputs (or their relevance to the behavior observed) vary depending on the assembly configuration. Observe the black arrows near iterations 900 and 1300 in the camera sensors panel in Figure 9. The battery level is above 15 and the robot is exploring the environment and avoiding collisions (notice the distance sensors dynamics), but it also receives visual input. This input, however, does not affect the ongoing behavior (see the motors dynamics). To highlight this effect, we conducted an information dynamics analysis, exploring how information flows from sensors to motors and from motors to sensors as the task progresses.

Figure 11b shows the transfer entropy between the robot's distance and camera sensors and its motors for the duration of the trial, calculated



(a)



(b)

Figure 11: (a) System dynamics depicted by the projection of the 11 phase differences $\gamma_{i,i-1}$, $i = 2, \ldots, 12$, of the 12-node network into their first two principal components (PC_1 and PC_2), and the motor output represented by the difference between the values of M_1 and M_2 (see equation 4.3). Solid and dashed lines relate to different iteration intervals. Notice that whenever there is an assembly rearrangement (see Figures 9 and 10), there is a corresponding shift in the network motor state-space region, which relates to different behaviors displayed by the robot. (b) Transfer entropy between the distance sensors' time series and the motors' time series (top panel) and between the camera sensors' time series and the motors' time series (bottom panel). A sliding window of length 200 iterations is used to obtain each time series at every iteration of the transfer entropy analysis. Results are smoothed using a gaussian filter with time constant 0.08.

according to section 4.1.1. To obtain the time series, we used a sliding window containing data from the past 200 iterations; therefore, note that the results reflect a history of interactions and are not an instantaneous measurement of information flow. More specifically, the sensors' time series (three infrared sensors and three camera sensors) are submitted to PCA to perform a dimension reduction. The calculated principal component and the original time series of each sensor modality are used to create a single time series that captures the most significant features of the multidimensional input space. The motor commands are also combined to generate a single time series by subtracting the value of the left wheel command from the right wheel command. This data were then discretized into six equiprobable states, and finally the transfer entropy is calculated. We performed a series of analyses with different parameter choices, and although there were differences in the values obtained, the overall qualitative aspect of the curves was maintained.

In the top panel, the information flow from distance sensors to the motors (in other words, the causal influence from distance sensors to motor commands) oscillates throughout the task and has peaks between iterations 375 and 550, 900 and 1075, and 1300 and 1500. In the bottom panel, the information flow from visual sensors to the motors also oscillates throughout the task and has peaks between iterations 375 and 550, 700 and 900, and 1100 and 1400. Comparing these with Figure 9, one can see a relationship between the distance and camera sensors and the respective information flows. This relationship is to complex brain-body-environment interactions and not only to the presence of sensory stimulus, as, for example, one can verify observing iterations 500, 900, and 1350 in the camera sensors plot (see Figure 9) and the respective information flow plot: although there are variations in the visual input, there is no corresponding increase in the transfer entropy values. Therefore, sensory inputs may or may not affect the motor commands, and the current assembly configuration will modulate this interaction.

The information flow dynamics offers another perspective in the robot's behavior analysis. Notice that the information flow magnitude is nearly twice as much in the bottom panel as in the top panel. We saw in the previous dynamics and behavior analyses that when the battery drops below 15, the robot moves toward the recharging area, but it does not display the otherwise natural obstacle avoidance behavior when it eventually finds a wall. The transfer entropy analysis highlights that the visual information is the main source of behavior modulation even though the distance sensors still have some influence. This is clear between iterations 1075 and 1250: the visual information flow (bottom panel) increases as the robot approaches the recharging area (higher visual input), and on finding the wall, there is an increase in the flow from motors to sensors in the top panel. This means that the robot's trajectory, mainly influenced by the visual inputs, determines the incoming sensory readings—the robot actively "produces" its inputs—while the information flow from distance sensors to motors decreases, meaning that there is little influence of the distance sensors in the robot's behavior.

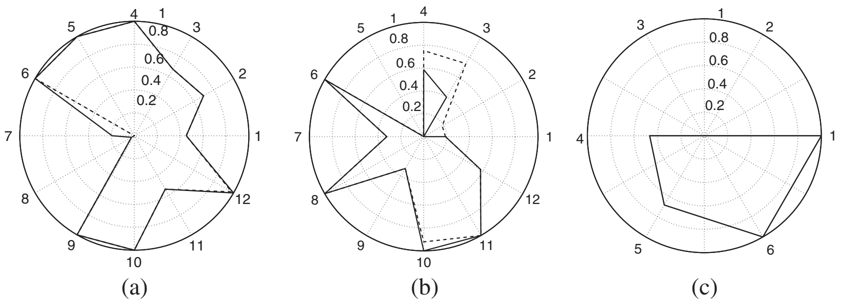


Figure 12: Values of the weights of the output unit neurons (a, b) and of the input weights ϵ_n of nodes that have sensory input (c) at the end of phase 1 (solid black) and phase 2 (dashed black) of evolution. Values are normalized between 0 and 1 to improve visualization. See Figure 6 for details.

Finally, to stress the relevance of assembly reorganization in the evolutionary process, observe Figure 12. It shows the values at the end of phases 1 and 2 of evolution of the weights of the output unit neurons and of the input weights of nodes that have sensory input (see Figure 6). Notice that though there are minor adjustments in the value of a few parameters, most of them remain unchanged as evolution progress from one phase to another. This further supports the relevance and flexibility of dynamic assembly reorganization in multi-objective tasks.

5 Discussion

It is now established that synchronization mechanisms and dynamic assembly formation in neuronal networks have a relationship with cognitive processes and behavior; however, the underlying computational functions and interplay with behavior are still to be uncovered. In this work, we conducted experiments in both supervised and unsupervised learning scenarios exploring concepts drawn from the binding-by-synchrony hypothesis, which considers neuronal assembly computations from a spike time perspective. In fact, a growing body of literature attests that neuronal codes based solely on spike rates underperform or do not contribute in a variety of cognitive tasks (Borst & Theunissen, 1999; Carmena et al., 2003; Jacobs et al., 2009; Rabinovich et al., 2012).

The neuronal network model used is inspired by the Kuramoto model of coupled phase oscillators and allows one to fine-tune the network synchronization dynamics and assembly configuration. The model has an intrinsic, ongoing oscillatory activity that can only be modulated—not determined—by external stimuli, in contrast with models that consider a static system with responses elicited only by stimulus onset. As reiterated throughout this work, cognitive processes unfold over time and therefore cannot rely only on external events. Also, one can precisely determine the number and constitution of assemblies in the model. Although evidence points at assembly formation as a result of emergent processes, and several models capture this property (Izhikevich, 2006; Burwick, 2008; Ranhel, 2012), it is hard to foresee or design how the network will self-organize; hence, the model presented here contributes to studies that require a consistent emergent configuration and studies focused on a systematic exploration of different synchronization regimes.

In experiment 1, a supervised learning task, we studied the influence of the number and size of neuronal assemblies in a spike pattern classification task. The input spike patterns and the network phase dynamics both had roughly the same spike count across the task, similar to what has been observed in real cortical neuronal ensembles (Carmena et al., 2003). A linear readout unit generated the circuit outputs based on the state of the network nodes at a particular time, which resembles the approach adopted in brain-machine interface studies (Lebedev et al., 2005; Hatsopoulos & Donoghue, 2009) but is also suggested as a more appropriate form to understand assembly activity (Buzsaki, 2010). Although it is still not clear how the temporal relations in the brain are organized, and thus how reading the network state at a predetermined time could be justified, there are some possible solutions that may have evolved in natural brains: a redundancy in the circuitry may exist so that at any time an event occurs or a classification task is required, an output is produced (Ranhel, 2012); there can be an interaction of the external rhythms with internally generated ones, forcing synchronized firing events to occur in strict time windows (Masquelier et al., 2009; Kopell et al., 2010); attention mechanisms may also interfere and promote phase resetting (Steinmetz et al., 2000; Lakatos et al., 2008).

Considering a network with a total number of neurons equal to 80 ± 4, our results show that performance boosts as we increase the number of assemblies, and that can be predicted up to a certain extent by the computational power and generalization capability of the system following the same procedure described in Legenstein and Maass (2007), although in our case, it is possible that simply using the difference between these two measurements may not be the most appropriate form of combining them; this is an open problem that goes beyond the scope of this work. A further analysis, which varied the number and size of assemblies, revealed that the first has an impact more on performance than the latter does, a fact that can be attributed to the increased variability of possible network states due to a larger number of emergent clusters rather than fewer but larger assemblies. Also, the system presents a saturation in performance with respect to the number of clusters. Therefore, the results indicate that simply increasing the number of neurons or assemblies in the system does not necessarily originate a corresponding increase in performance. A similar phenomenon is described in neuronal assembly physiology as "the neuronal mass principle"

(Nicolelis & Lebedev, 2009), which states that a minimal number of neurons is needed in a neuronal population to stabilize its information capacity (captured by a readout unit) at a satisfactory level. Reducing the number of neurons causes an increasingly sharp drop in the information capacity of this population, whereas increasing the number of sampled neurons above a certain level does not increase the accuracy of predictions (Carmena et al., 2003; Lebedev et al., 2008).

Also, the results showed that in our model, most of the neuronal architectures were highly redundant, most of the neurons in the higher-performance configurations presented independent activity, and increasing the number of neurons in a network with a fixed number of assemblies increased the redundancy. All of these findings resemble the results obtained in real cortical experiments; therefore, a few remarks should be made: first, real neuronal ensembles are highly redundant, and that can be associated with resistance to error and natural mechanisms of probability distribution estimation (Barlow, 2001; Szczepanski, Arnold, Wajnryb, Amigó, & Sanchez-Vives, 2011); second, neuronal independence (as observed in our results) can be linked to code efficiency because the information capacity of individual neurons is not compromised by redundant scenarios (Schneidman et al., 2003); and third, there is still a lack of studies comparing the information flow dynamics due to neuronal interactions and due to single neurons alone—attesting that the timescales of the interactions as well as spurious effects such as averaging are still works in progress (Reich et al., 2001; Narayanan et al., 2005).

The computational power analysis emphasized that multiple readout units could be trained to perform different classification tasks based on the same network state. In contrast, to conclude experiment 1, we investigated whether the system could cope with multiple classification tasks relying only on a manipulation of the phase dynamics by means of an internally generated signal, employing the same readout unit without any plasticity mechanisms. To support the approach, there are clinical studies suggesting that intracortical electrical stimulation can induce cortical plasticity (Jackson, Mavoori, & Fetz, 2006), but functional plasticity can also be obtained faster as a result of attentional processes (Steinmetz et al., 2000; Lakatos et al., 2008; Schroeder & Lakatos, 2009). Our results show that the system can be trained in multiple classification tasks on rearrangement of assembly configuration.

Although the results of the first experiment show that such a temporal code carries information and suggest that it can be exploited in a variety of tasks, it is challenging to determine to what extent the brain uses a temporal code. Moreover, there is evidence that the neuronal activity evoked by the body’s sensorimotor interactions with the environment differs from the activity evoked by passive stimulus (Lungarella & Sporns, 2006; Eliades & Wang, 2008). Hence, research on temporal neuronal codes and assembly formation benefits if linked with behavioral studies (Engel, 2010; Panzeri, Brunel, Logothetis, & Kayser, 2010); in this sense, evolutionary robotics emerges as a suitable technique to combine both approaches (Floreano et al., 2008; Floreano & Keller, 2010).

In experiment 2, an ER unsupervised learning task, we evolved a simulated robotic agent, controlled by a variation of the system investigated in experiment 1, to solve multiple tasks depending on its battery state. The results showed that the evolved framework, together with the dynamic assembly formation, can generate minimally cognitive behaviors. In increasingly complex tasks, the changes in the parameters of the system are relatively small, which indicate that the different assemblies formed dynamically also facilitate the evolutionary process. Finally, we highlighted the context-based neuronal dynamics showing that the phase space formed by the motor readings and the nodes' phases have different orbits due to changes in assembly organization, and an analysis of the information flow in the network reveals that such changes modulate the influence of the inputs in the robot's behavior (determined by the motor commands).

Taken together, experiments 1 and 2 employed information theory and decoding methods to provide further evidence that the dynamic formation of assemblies and the relative neuronal firing times can mediate processes involving the classification of spike patterns and can selectively modulate the influence of external signals in the current network activity. Ultimately there is no guarantee that the brain makes use of a time-based decoding procedure, neither that it is able to exploit the information content revealed by the synergy analysis and the transfer entropy approach; nevertheless, it may shed light on aspects of brain-body-environment interactions and provide upper bounds on code efficiency when testing hypotheses (Quiroga & Panzeri, 2009; Jacobs et al., 2009).

There are several directions for future research. First, it is common to construct the Kuramoto model (and its variations) having additive noise at the input level equivalent to noise applied at the network level (Acebrón, Bonilla, Pérez Vicente, Ritort, & Spigler, 2005). Based on equation 2.2, the following equation shows the usual form of the Kuramoto model with inputs $I_n(t)$ and noise $\chi_n(t)$:

$$\dot{\theta}_n = \omega_n + \frac{K}{N}\sum_{m=1}^{N} g(\theta_n - \theta_m) + I_n(t) + \chi_n(t), \quad n = 1, \ldots, N. \tag{5.1}$$

In this sense, experiments 1 and 2 had a subset of noisy neurons (only neurons that had inputs). Given the relevance of widespread noise to many neuronal and cognitive phenomena (Rolls & Deco, 2010), future investigations should explore in depth its impact on the framework. As a preliminary study, we have run two further simulations of experiment 1 adding gaussian noise of zero mean and standard deviation σ to all nodes of the network. The results are presented in Figure 13. Notice that classification

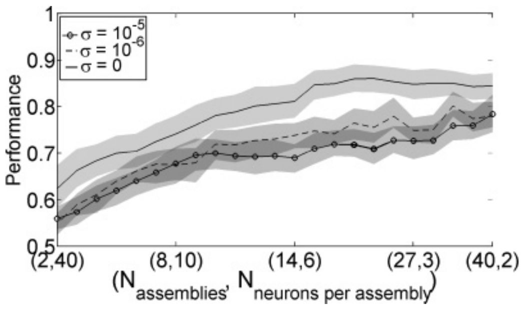


Figure 13: Effect of gaussian noise of zero mean and standard deviation σ applied to all nodes of the network in the performance of the system. Results are mean values over 20 different simulations; shaded areas are the 95% confidence interval.

performance falls with increasing noise magnitude (an effect also present in Figures 2c and 4b), but the trend observed in our original results is kept, and higher performance levels are obtained in architectures with more assemblies. Thus, at least for this experiment, noise applied to all neurons alters the classification performance in a quantitative rather than a qualitative way.

How would additive noise affect measures of redundancy and synergy? The intuition that noisy scenarios are better tackled with redundant architectures is justified: there is work showing that cortical circuits, which operate in an intrinsically noisy environment, are highly redundant (Narayanan et al., 2005; Szczepanski et al., 2011). However, there is criticism regarding the interpretation of information-theoretical measurements such as redundancy (Schneidman et al., 2003; Latham & Nirenberg, 2005), as well as findings showing predominantly synergistic or independent activity in neuronal circuits, instead of redundancy, depending on factors such as which area and which neurons are recorded or which kind of task is performed (Reich et al., 2001). Additionally, as Szczepanski et al. (2011) showed, neurons can dynamically switch their interactions during the execution of the task; thus, synergetic, independent, or redundant activity may be masked by averaging processes. Finally, redundancy and synergy are found to be largely influenced by the network architecture and the decoding unit used (Schneidman et al., 2003). The conclusion is that redundancy is not necessary for good performance in noisy scenarios, but, most important, it depends largely on the experimental paradigm used.

One limitation encountered in the methods used in experiment 1, chosen for their ability to assess computational performance in generic neuronal microcircuits independent of task paradigm, is that in some circumstances, noisy neurons may make the computational power and generalization capability analyses inconclusive due to state matrices having complete rank most of the time (Legenstein & Maass, 2007). This was not an issue in previous work that used these methods (e.g., Maass et al., 2005; Legenstein & Maass, 2007) because emergent properties of the neural architecture resulted in highly silent networks with dynamics that were marginally affected by noise; conversely, the model in this letter is composed of self-sustained oscillators, which are always active. Considering that silence in the brain is still a point of much controversy (Shoham, O'Connor, & Segev, 2006), the applicability of the methods used in experiment 1 to a variety of problems and neural architectures remains an open question.

Another possible future extension to the model would be to substitute the continuously coupled oscillators used in this work with pulse-coupled oscillators, which are not only a more biologically plausible abstraction of neuronal synaptic activity but also present rich metastable dynamics that can be exploited to compute arbitrary logic operations (Neves & Timme, 2009, 2012; Wildie & Shanahan, 2012). However, the network cluster states in the works just cited are emergent processes found numerically (despite the switching dynamics being controllable), while the model studied in this work can be systematically tuned into predefined assembly configurations.

Finally, it would be interesting to extend the model to include multiple assembly membership (i.e., entitle a given neuron to participate simultaneously in two or more assemblies), as it has been shown to enhance the computational power of a neuronal circuit (Izhikevich, 2006).

To conclude, implementing the methods or the experiments described in this work in a biological network is impractical at the moment for limitations in both recording and stimulation technologies: the best technologies are able to record and stimulate a limited number of neurons. However, more important than trying to implement the methods or experiments in a biological network are the insights and future work opportunities we gain. There are many open questions in neuroscience regarding neural assemblies, their properties, and their relationship to behavior. This very simple model, based on a model that is being increasingly applied to study neuroscience problems (the Kuramoto model), has shown promising results in supervised and unsupervised learning tasks. The point to stress is not solely performance levels—support vector machines, for instance, would surely excel in experiment 1, attaining far better results than our approach—but the ability to solve relatively complex tasks mimicking mechanisms that current research suggests is exploited by the brain: neuronal assembly dynamics. Therefore, a better understanding of the framework, its limitations and possible extensions, and ultimately understanding of the computational properties of neuronal assembly dynamics, whether at solving data mining tasks or as part of novel behavior generation mechanisms, should precede biological implementations.

Acknowledgments

R.M. was partly funded by a Sussex TA scholarship and a de Bourcier scholarship. Special thanks to Bruno Santos, James Thorniley, and other members of the CCNR for useful discussions relating to this work. We thank two anonymous reviewers for their valuable comments.

References

Acebrón, J. A., Bonilla, L. L., Pérez Vicente, C. J., Ritort, F., & Spigler, R. (2005). The Kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of Modern Physics*, *77*(1), 137–185.

Ashwin, P., & Timme, M. (2005). When instability makes sense. *Nature*, *404*, 36–37.

Ashwin, P., Wordsworth, J., & Townley, S. (2007). Dynamics on networks of cluster states for globally coupled phase oscillators. *SIAM J. Applied Dynamical Systems*, *6*, 728–758.

Averbeck, B. B., & Lee, D. (2004). Coding and transmission of information by neural ensembles. *Trends in Neurosciences*, *27*(4), 225–230.

Barlow, H. (2001). Redundancy reduction revisited. *Network: Comput. Neural Syst.*, *12*(3), 241–253.

Beer, R. (2003). The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, *11*(4), 209–243.

Bhowmik, D., & Shanahan, M. (2012). How well do oscillator models capture the behaviour of biological neurons? In *Proc. of the IEEE 2012 Int. Joint Conf. on Neural Networks* (pp. 1–8). Piscataway, NJ: IEEE.

Bongard, J. C. (2011). Innocent until proven guilty: Reducing robot shaping from polynomial to linear time. *IEEE Trans. Evolutionary Computation*, *15*(4), 571–585.

Borst, A., & Theunissen, F. E. (1999). Information theory and neural coding. *Nature Neurosci.*, *2*(11), 947–957.

Breakspear, M., Heitmann, S., & Daffertshofer, A. (2010). Generative models of cortical oscillations: Neurobiological implications of the Kuramoto model. *Front. Hum. Neurosci.*, *4*.

Buehlmann, A., & Deco, G. (2010). Optimal information transfer in the cortex through synchronization. *PLoS Comput. Biol.*, *6*(9).

Burwick, T. (2008). Temporal coding: Assembly formation through constructive interference. *Neural Computation*, *20*(7), 1796–1820.

Buzsaki, G. (2010). Neural syntax: Cell assemblies, synapsembles, and readers. *Neuron*, *68*, 362–385.

Canolty, R. T., Cadieu, C. F., Koepsell, K., Ganguly, K., Knight, R. T., & Carmena, J. M. (2012). Detecting event-related changes of multivariate phase coupling in dynamic brain networks. *Journal of Neurophysiology*, *107*(7), 2020–2031.

Carmena, J. M., Lebedev, M., Crist, R. E., O'Doherty, J. E., Santucci, D. M., Dimitrov, D. F., et al. (2003). Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS Biology*, *1*(2), 193–208.

Cover, T. M., & Thomas, J. (1991). *Elements of information theory.* New York: Wiley.

Cumin, D., & Unsworth, C. (2007). Generalising the Kuramoto model for the study of neuronal synchronisation in the brain. *Physica D*, *226*(2), 181–196.

Dayan, P. (1999). Unsupervised learning. In R. A. Wilson & F. Keil (Eds.), *The MIT encyclopedia of the cognitive sciences*. Cambridge, MA: MIT Press.

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: Computational and mathematical modelling of neural systems*. Cambridge, MA: MIT Press.

Di Paolo, E. (2003). Evolving spike-timing-dependent plasticity for single-trial learning in robots. *Philosophical Transactions of the Royal Society A*, *361*(1811), 2299–2319.

Dorigo, M., & Colombetti, M. (1998). *Robot shaping: An experiment in behavior engineering*. Cambridge, MA: MIT Press.

Edelman, G. M. (2007). Learning in and from brain-based devices. *Science*, *318*(5853), 1103–1105.

Eliades, S. J., & Wang, X. (2008). Neural substrates of vocalization feedback monitoring in primate auditory cortex. *Nature*, *453*(7198), 1102–1106.

Engel, A. (2010). Directive minds: How dynamics shapes cognition. In J. Stewart, O. Gapenne, & E. A. Di Paolo (Eds.), *Enaction: Toward a new paradigm for cognitive science*. Cambridge, MA: MIT Press.

Engel, A., Fries, P., & Singer, W. (2001). Dynamic predictions: Oscillations and synchrony in top-down processing. *Nat. Rev. Neurosci.*, *2*(10), 704–716.

Ermentrout, G., & Kleinfeld, D. (2001). Traveling electrical waves in cortex: Insights from phase dynamics and speculation on a computational role. *Neuron*, *29*, 33–44.

Ermentrout, G., & Kopell, N. (1986). Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM J. Appl. Math.*, *46*, 233–253.

Floreano, D., Husbands, P., & Nolfi, S. (2008). Evolutionary robotics. In B. Siciliano & O. Khatib (Eds.), *Springer handbook of robotics* (pp. 1423–1451). New York: Springer.

Floreano, D., & Keller, L. (2010). Evolution of adaptive behaviour in robots by means of Darwinian selection. *PLoS Biology*, *8*(1).

Floreano, D., & Mondada, F. (1994). Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In D. Cliff, P. Husbands, J. Meyer, & S. W. Wilson (Eds.), *From animals to animats III: Proc. of the Third Int. Conf. on Simulation of Adaptive Behavior* (pp. 402–410). Cambridge, MA: MIT Press.

Gourévitch, B., & Eggermont, J. (2007). Evaluating information transfer between auditory cortical neurons. *J. Neurophysiol.*, *97*, 2533–2543.

Hansel, D., Mato, G., & Meunier, C. (1995). Synchrony in excitatory neural networks. *Neural Computation*, *7*, 307–337.

Harvey, I., Di Paolo, E., Wood, R., Quinn, M., & Tuci, E. (2005). Evolutionary robotics: A new scientific tool for studying cognition. *Artificial Life*, *11*(1–2), 79–98.

Hatsopoulos, N. G., & Donoghue, J. P. (2009). The science of neural interface systems. *Annual Review of Neuroscience*, *32*, 249–266.

Hebb, D. (1949). *The organization of behavior*. New York: Wiley.

Husbands, P. (2009). Never mind the iguana, what about the tortoise? Models in adaptive behaviour. *Adaptive Behavior*, *17*(4), 320–324.

Husbands, P., Smith, T., Jakobi, N., & O'Shea, M. (1998). Better living through chemistry: Evolving GasNets for robot control. *Connection Science*, *10*, 185–210.

Izhikevich, E. (1999). Weakly pulse-coupled oscillators, FM interactions, synchronization, and oscillatory associative memory. *IEEE Trans. Neural Networks*, *10*(3), 508–526.

Izhikevich, E. (2006). Polychronization: Computation with spikes. *Neural Computation*, *282*, 245–282.

Izhikevich, E. M. (2007). *Dynamical systems in neuroscience: The geometry of excitability and bursting*. Cambridge, MA: MIT Press.

Jackson, A., Mavoori, J., & Fetz, E. E. (2006). Long-term motor cortex plasticity induced by an electronic neural implant. *Nature*, *444*(7115), 56–60.

Jacobs, A. L., Fridman, G., Douglas, R. M., Alam, N. M., Latham, P. E., Prusky, G. T., & Nirenberg, S. (2009). Ruling out and ruling in neural codes. *PNAS*, *106*(14), 5936–5941.

Jolliffe, I. T. (2002). *Principal component analysis*. New York: Springer-Verlag.

Kayser, C., Montemurro, M. A., Logothetis, N. K., & Panzeri, S. (2009). Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns. *Neuron*, *61*(4), 597–608.

Kitzbichler, M., Smith, M., Christensen, S., & Bullmore, E. (2009). Broadband criticality of human brain network synchronization. *PLoS Comput. Biol.*, *5*(3).

Knudsen, E. I. (1994). Supervised learning in the brain. *Journal of Neuroscience*, *14*(7), 3985–3997.

Kopell, N., Kramer, M., Malerba, P., & Whittington, M. (2010). Are different rhythms good for different functions? *Frontiers in Human Neuroscience*, *4*(187).

Kopell, N., Whittington, M. A., & Kramer, M. A. (2011). Neuronal assembly dynamics in the beta1 frequency range permits short-term memory. *PNAS*, *108*(9), 3779–3784.

Kuramoto, Y. (1984). *Chemical oscillation, waves, and turbulence*. New York: Springer.

Lakatos, P., Karmos, G., Mehta, A. D., Ulbert, I., & Schroeder, C. E. (2008). Entrainment of neuronal oscillations as a mechanism of attentional selection. *Science*, *320*(5872), 110–113.

Latham, P. E., & Nirenberg, S. (2005). Synergy, redundancy, and independence in population codes, revisited. *Journal of Neuroscience*, *25*(21), 5195–5206.

Lebedev, M., Carmena, J. M., O'Doherty, J. E., Zacksenhouse, M., Henriquez, C. S., Principe, J. C., & Nicolelis, M. A. L. (2005). Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface. *Journal of Neuroscience*, *25*(19), 4681–4693.

Lebedev, M. A., O'Doherty, J. E., & Nicolelis, M. A. L. (2008). Decoding of temporal intervals from cortical ensemble activity. *Journal of Neurophysiology*, *99*(1), 166–186.

Legenstein, R., & Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, *20*(3), 323–334.

Lopes dos Santos, V., Conde-Ocazionez, S., Nicolelis, M., Ribeiro, S. T., & Tort, A. B. L. (2011). Neuronal assembly detection and cell membership specification by principal component analysis. *PloS One*, *6*(6), 20996.

Lungarella, M., Ishiguro, K., Kuniyoshi, Y., & Otsu, N. (2007). Methods for quantifying the causal structure of bivariate time series. *Int. J. Bifurcation and Chaos*, *17*(3), 903–921.

Lungarella, M., Pitti, A., & Kuniyoshi, Y. (2007). Information transfer at multiple scales. *Physical Review E*, 056117.

Lungarella, M., & Sporns, O. (2006). Mapping information flow in sensorimotor networks. *PLoS Comput. Biol.*, *2*(10), e144.

Maass, W., Legenstein, R., & Bertschinger, N. (2005). Methods for estimating the computational power and generalization capability of neural microcircuits. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems, 17* (pp. 865–872). Cambridge, MA: MIT Press.

Marschinski, R., & Kantz, H. (2002). Analysing the information flow between financial time series: An improved estimator for transfer entropy. *European Physical Journal B*, *30*(2), 275–281.

Masquelier, T., Hugues, E., Deco, G., & Thorpe, S. (2009). Oscillations, phase-of-firing coding, and spike timing-dependent plasticity: An efficient learning scheme. *Journal of Neuroscience*, *29*(43), 13484–13493.

Moioli, R. C., Vargas, P. A., & Husbands, P. (2012). Synchronisation effects on the behavioural performance and information dynamics of a simulated minimally cognitive robotic agent. *Biological Cybernetics*, *106*, 407–427.

Narayanan, N. S., Kimchi, E. Y., & Laubach, M. (2005). Redundancy and synergy of neuronal ensembles in motor cortex. *Journal of Neuroscience*, *25*(17), 4207–4216.

Neves, F. S., & Timme, M. (2009). Controlled perturbation-induced switching in pulse-coupled oscillator networks. *J. Phys. A: Math. Theor.*, *42*, 345103.

Neves, F. S., & Timme, M. (2012). Computation by switching in complex networks of states. *Phys. Rev. Lett.*, *109*, 018701.

Nicolelis, M. A. L., & Lebedev, M. A. (2009). Principles of neural ensemble physiology underlying the operation of brain-machine interfaces. *Nature Reviews Neuroscience*, *10*(7), 530–540.

Orosz, G., Moehlis, J., & Ashwin, P. (2009). Designing the dynamics of globally coupled oscillators. *Progress of Theoretical Physics*, *122*, 611–630.

Panzeri, S., Brunel, N., Logothetis, N. K., & Kayser, C. (2010). Sensory neural codes using multiplexed temporal scales. *Trends in Neurosciences*, *33*(3), 111–120.

Pérez, T., Garcia, G. C., Eguíluz, V. M., Vicente, R., Pipa, G., & Mirasso, C. (2011). Effect of the topology and delayed interactions in neuronal networks synchronization. *PloS One*, *6*(5), e19900.

Pikovsky, A., Rosenblum, M., & Kurths, J. (2001). *Synchronization: A universal concept in nonlinear sciences*. Cambridge: Cambridge University Press.

Popovych, O., Maistrenko, Y., & Tass, P. (2005). Phase chaos in coupled oscillators. *Physical Review E*, *71*(6), 3–6.

Pouget, A., Dayan, P., & Zemel, R. (2008). Information processing with population codes. *Nature Reviews Neuroscience*, *1*(2), 125–132.

Quiroga, R. Q., & Panzeri, S. (2009). Extracting information from neuronal populations: Information theory and decoding approaches. *Nature Reviews Neuroscience*, *10*(3), 173–185.

Rabinovich, M., Afraimovich, V., Bick, C., & Varona, P. (2012). Information flow dynamics in the brain. *Physics of Life Reviews*, *9*(1), 51–73.

Ranhel, J. (2012). Neural assembly computing. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(6), 916–927.

Reich, D. S., Mechler, F., & Victor, J. D. (2001). Independent and redundant information in nearby cortical neurons. *Science*, *294*(5551), 2566–2568.

Rieke, F., Warland, D., van Steveninck, R. R., & Bialek, W. (1997). *Spikes: Exploring the neural code*. Cambridge, MA: MIT Press.

Rolls, E. T., & Deco, G. (2010). *The noisy brain: Stochastic dynamics as a principle of brain function.* New York: Oxford University Press.

Schneidman, E., Bialek, W., & Berry II, M. J. (2003). Synergy, redundancy, and independence in population codes. *Journal of Neuroscience*, *25*(21), 5195–5206.

Schreiber, T. (2000). Measuring information transfer. *Phys. Rev. Lett.*, *85*, 461–464.

Schroeder, C. E., & Lakatos, P. (2009). Low-frequency neuronal oscillations as instruments of sensory selection. *Trends in Neurosciences*, *32*(1), 9–18.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, 379–423.

Shim, Y., & Husbands, P. (2012). Chaotic exploration and learning of locomotion behaviours. *Neural Computation*, *24*(8), 2185–2222.

Shoham, S., & O'Connor, D. H., & Segev, R. (2006). How silent is the brain: Is there a "dark matter" problem in neuroscience? *J. Comp. Physiol. A*, *192*, 777–784.

Singer, W. (1999). Neuronal synchrony: A versatile code for the definition of relations? *Neuron*, *24*(1), 49–65.

Sporns, O., & Alexander, W. (2002). Neuromodulation and plasticity in an autonomous robot. *Neural Networks*, *15*, 761–774.

Steinmetz, P. N., Roy, A., Fitzgerald, P. J., Hsiao, S. S., & Johnson, K. O. (2000). Attention modulates synchronized neuronal firing in primate somatosensory cortex. *Nature*, *436*, 187–190.

Storm, T. (2004). *KiKS, a Khepera simulator for Matlab 5.3 and 6.0*. http://theodor.zoomin.se/index/2866.html.

Szczepanski, J., Arnold, M., Wajnryb, E., Amigó, J. M., & Sanchez-Vives, M. V. (2011). Mutual information and redundancy in spontaneous communication between cortical neurons. *Biol. Cybern.*, *104*, 161–174.

Turrigiano, G., & Nelson, S. (2004). Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience*, *5*(2), 97–107.

Uhlhaas, P. J., Pipa, G., Lima, B., Melloni, L., Neuenschwander, S., Nikolic, D., & Lin, S. (2009). Neural synchrony in cortical networks: History, concept and current status. *Frontiers in Integrative Neuroscience*, *3*(17), 1–19.

Urzelai, J., & Floreano, D. (2001). Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evol. Comput.*, *9*(4), 495–524.

Vapnik, V. N. (1998). *Statistical learning theory.* New York: Wiley.

Varela, F., Lachaux, J., Rodriguez, E., & Martinerie, J. (2001). The brainweb: Phase synchronization and large-scale integration. *Nat. Rev. Neurosci.*, *2*(4), 229–239.

Vicente, R., Gollo, L. L., Mirasso, C. R., Fischer, I., & Pipa, G. (2008). Dynamical relaying can yield zero time lag neuronal synchrony despite long conduction delays. *Proc. Nat. Acad. Sci. USA*, *105*(44), 17157–17162.

Vicente, R., Wibral, M., Lindner, M., & Pipa, G. (2011). Transfer entropy: A model-free measure of effective connectivity for the neurosciences. *J. Comput. Neurosci.*, *30*, 45–67.

Wildie, M., & Shanahan, M. (2012). Metastability and chimera states in modular delay and pulse-coupled oscillator networks. *Chaos*, *22*(4), 043131.

Womelsdorf, T., Schoffelen, J., Oostenveld, R., Singer, W., Desimone, R., Engel, A., & Fries, P. (2007). Modulation of neuronal interactions through neuronal synchronization. *Science*, *316*(5831), 1609–1612.

Wordsworth, J., & Ashwin, P. (2008). Spatiotemporal coding of inputs for a system of globally coupled phase oscillators. *Physical Review E*, *78*(6), 1–10.

Received October 19, 2012; accepted May 6, 2013.