

Dynamics From Patterns: Creating Neural Controllers with SENMP

Janne Haverinen

Department of Electrical and Information Engineering
University of Oulu
Oulu, Finland
e-mail: janne.haverinen@oulu.fi

Juha Rönig

Department of Electrical and Information Engineering
University of Oulu
Oulu, Finland
e-mail: juha.roning@oulu.fi

Abstract—In this paper we show how simple laterally interacting computational entities, *i.e.* neurons, can be guided by a selectionist process into spatial patterns that show interesting and purposeful dynamics with regard to a particular utility measure. In other words, if a suitable population of laterally interacting mobile entities exist, it is possible to gradually arrange the entities into a spatial pattern that exhibits the desired dynamics. In this paper, the selectionist process is implemented with the Stochastic Evolutionary Neuron Migration Process (SENMP) and approach is used to evolve Dynamic Recurrent Neural Networks (DRNNs) for controlling complex dynamic systems such as autonomous mobile robots, for example. The feasibility and advantages of the approach are demonstrated by evolving neural controllers for solving a non-Markovian double pole balancing problem. In addition, we have earlier used SENMP to evolve navigation behaviors for mobile robots in complex simulated and real environments.

I. INTRODUCTION

Our original idea was to design, using some mechanism, a spatial pattern of computational entities in n -space with simple lateral interaction rules, which could act as a simple control system for a mobile robot [10], and which could adapt to a new environment by undergoing gradual morphological changes. The idea is based on the observation that nature has a remarkable ability to create complex patterns on various scales from physical matter. If these patterns are composed of entities that are able to interact with the environment, *i.e.* they are not closed systems, a dynamic system emerges in which interactions of entities are inevitably regulated by the physical reality, including space. Assuming that space has an important role in defining the properties of a dynamic system composed of laterally interacting entities, it is possible to modify the dynamics of the system by altering the spatial pattern formed by the entities.

It is clear that the patterns that exist in living organisms are to a large extent genetically determined. However, an infinite number of non-genetic spatial patterns exist on various scales in the physical world that form dynamic systems which are composed of laterally interacting entities – both living and non-living. These patterns can be a result of physical forces acting in the physical reality, *e.g.* the gravity in start systems, or they can be a result of a complex stochastic selection process, *i.e.* adaptation, as in population dynamics and spatial ecology [9], [23].

In the Stochastic Evolutionary Neuron Migration Process (SENMP) [10], the efficacy of the connection between individual neurons is a function of the Euclidean distance between the neurons in 2-space. At present, the model is exclusively based on lateral interaction between neurons, and no distinct connections between neurons are used in order to emphasize the role of spatial distribution of neurons in defining the behavior of the neural network. This practice is based on the assumption that during the evolution of the nervous system neural conduction was preceded by more primitive forms of communication in which signals were propagated directly between neighboring cells. Indeed this form of non-neural communication exists alongside neural conduction in some cnidarians¹. The basic cnidarian nerve net is a two-dimensional network of neurons which has both a sensory and motor capacity, and in which there is no distinction between axons and dendrites – nervous impulses, therefore, propagate in both directions between cells [15].

II. SENMP

A. The Encoding Scheme

Following the original idea, the purpose was to create, through some mechanism, a spatial pattern of computational entities with simple interaction rules. The spatial interaction scheme adopted for this work should have at least the following three properties: lateral interaction, selectively sensitive neuron groups, and feedback. The encoding scheme used in SENMP is illustrated in Fig. 1.

The neighborhood function used to implement the lateral interaction scheme in SENMP is a bell-shaped Gaussian function. The definition of the neighborhood function is given in Eq. 1 which describes the efficacy of the connection between neurons j and k . In Eq. 1 θ_i and θ_j are the phase terms of the connected neurons, $d_{k,j}$ is the normalized Euclidean distance of neurons in 2-space, σ_j^2 represents the region of influence of the neuron j , ϕ_k is the feedback factor of neuron k , and $\Theta(\phi_k)$ is the feedback function of a neuron. The Gaussian function was selected because it is a continuous, monotonically decreasing function which asymptotically approaches zero

¹cnidarians include the corals, hydras, jellyfish, Portugese men-of-war, sea anemones, sea pens, sea whips, and sea fans.

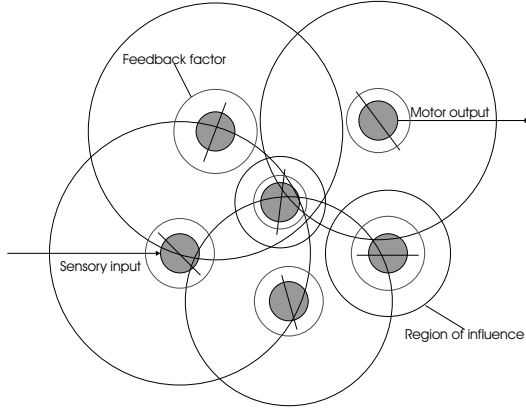


Fig. 1. Lateral interaction in SENMP. The line drawn through a neuron represents the phase term θ of the neuron. The feedback factors ϕ are drawn as circles around the neurons as they can be thought to define a region inside which the feedback is received. Each network consist of input, hidden, and output neurons. Some of the input neurons can be considered as bias neurons if their output is constant (=1).

and has the maximum, *i.e.* one, at zero (distance). These properties makes it an ideal candidate for the lateral interaction scheme proposed here. The Gaussian neighborhood function is responsible for defining the absolute physical limits for the efficacy of any connection departing from a neuron.

$$w_{kj} = \Theta(\phi_k) \sin(\theta_k - \theta_j) e^{-d_{kj}^2/2\sigma_j^2} \quad (1)$$

According Eq. 1, connection weights are implicitly determined by the spatial distribution of neurons and the neurons' phases. The definition of normalized distance d_{kj} between neurons j and k is given in Eq. 2 where $p_i = [x_i y_i]^T$ represents a position of the neuron i in 2-space.

$$d_{kj} = \frac{\|p_k - p_j\|}{\max_{l \neq m} (\|p_l - p_m\|)} \quad (2)$$

In order to break the symmetry of the neighborhood function and to establish both excitatory and inhibitory connections to the network, the variable called phase is introduced for each neuron state. The rationale behind the neuron phase is to provide a way of making neurons selectively more sensitive to some subset of neurons than to others and to create orthogonal groups of neurons, *i.e.* sets of neurons that do not interact with each other.

The term $\sin(\theta_k - \theta_j)$ in Eq. 1 is a simple way of breaking the symmetry of the neighborhood function and to provide different connection types using only one variable for each neuron. Furthermore, the phase term has an important role in defining the network topology as it can make a neuron selectively more sensitive to some subset of neurons in the network that to others. Indeed, the phase term is a simpler and more elegant way of tuning the network connectivity than the one used by [11] and [12], who used excitatory and inhibitory 'connectivity segments' in each neuron defining the connectivity of the evolved neural network. For the segments, the neuron

state contained six parameters defining the radius, angular extent and orientation for both excitatory and inhibitory connections.

The rationale behind the feedback factor ϕ is that by applying it, each neuron can tune the global feedback effect of the other neurons. More precisely, the outputs of the neurons propagating through the recurrent connections are multiplied by an asymmetric and bounded function $\Theta(\phi_k)$ in the target neuron k . Prior to SENMP all feedback factors are reset to zero and correspondingly $\Theta(\phi_k) = 0$. This means that initial neural networks are feedforward networks and recurrent connections evolve under the selection pressure during the migration process as the feedback factors of the neurons get non-zero values when subjected to the random noise. The definition of $\Theta(\phi_k)$ in Eq. 1 depends on whether the connection from neuron j to neuron k is a forward connection, *i.e.* a connection from the lower layer, or a recurrent connection, *i.e.* a connection from the upper or within the same layer: $\Theta(\phi_k) = \tanh(\phi_k)$ is used for recurrent connections enabling both positive and negative feedback, and $\Theta(\phi_k) = 1$ is used for forward connections.

B. The Evolutionary Algorithm

The stochastic motion of neurons establishing the spatial organization of the neural network ensemble was realized by using an Evolutionary Algorithm (EA). The fundamental reason for using the stochastic motion of neurons over developmental models [3], [5], [6], [14], [17], [18], [24] was to avoid heuristic constraints in pattern formation and to overcome the difficulty of designing an artificial genome which develops into a particular pattern. In addition, the stochastic process was used to learn and visualize how the neural network ensemble behaves while adapting to the task at hand.

In SENMP, the genotype G_i of the neural network i is a list

$$G_i = (\vec{g}_k : k = 0, 1, \dots, n-1),$$

where n is the number of neurons and the gene \vec{g}_k of the neuron k is

$$\vec{g}_k = (x_k, y_k, \theta_k, \phi_k, \tau_k, 2\sigma_k^2)^T,$$

where $x_k, y_k, \theta_k, \phi_k, \tau_k$, and σ_k^2 are the coordinates in 2-space, phase, feedback factor, time constant, and the region of influence of the neuron k , respectively. Index k specifies whether \vec{g}_k represents an input neuron, a hidden neuron, or an output neuron: $k = [0, l-1]$ for input, $k = [l, l+m-1]$ for hidden, and $k = [l+m, n-1]$ for output neurons, where l and m are the numbers of output, and hidden neurons, respectively. Correspondingly, the number of output neurons is $n - (m+l) > 0$.

SENMP is started by creating a list P of random genotypes

$$P = (G_n : n = 0, 1, \dots, N-1),$$

where N is the population size, *i.e.* the number of neural networks in the neural network ensemble. For each neuron, *i.e.* gene k , the phase θ_k gets a random value between

$[-\pi, \pi]$ and the coordinates x_k and y_k , and τ_k get random values between $[-\lambda, \lambda]$, where λ is the maximum amplitude of the random noise ν introduced to the parameters \vec{g}_k of the neuron (gene) k by the mutation operator. The region of influence σ^2 is initialized with some suitable² real value and the feedback factor ϕ is initialized, in general, to zero for all neurons, meaning no recurrent connections prior to the adaptation process. After the initial neural network ensemble is created, a fitness function $f(G)$ is used to assign a fitness for each genotype (neural network) G .

In SENMP, the recombination of genes is implemented using a factored sampling [2], [20] based method. This means that the neural network ensemble is represented with a single network composed of a fixed number of neurons, and each neuron having N (*i.e.* population size) possible states, *i.e.* genes. The factored sampling based EA was used to simplify and generalize the implementation of the SENMP and to emphasize the locality principle in the definition of neuron state transitions, *i.e.* decision about the next state of a neuron is made within the neuron itself based on the global reward received by each state during the previous generation.

Suppose that the neuron's position, phase, feedback factor, time constant and region of influence are encoded in a state vector $X \in R^{N_x}$, and the fitness values, measuring the performance of the neural network that the current state of the neuron is part of, observed at time t are denoted Z_t , with a measurement history $Z_t = (Z_1, \dots, Z_t)$. The Bayesian technique of factored sampling is a random-sampling method of approximating a distribution $p(X|Z)$ when it is too complicated to sample directly, but when prior $p(X)$ can be sampled, and the measurement density $p(Z|X)$ can be evaluated. Factored sampling proceeds by generating a set of N samples $s^{(n)}$ from *a priori* $p(X)$ and then assigning to each sample a weight $\pi^{(n)} = p(Z|X = s^{(n)})$ corresponding to the measurement density. The $\pi^{(n)}$ are normalized to sum 1 and then the weighted set $\{(s^{(n)}, \pi^{(n)})\}$ represents an approximation $\tilde{p}(X|Z)$ to the desired *posterior* $p(X|Z)$, where a sample is drawn from $\tilde{p}(X|Z)$ by choosing one of the $s^{(n)}$ with probability $\pi^{(n)}$. As $N \rightarrow \infty$ samples from $\tilde{p}(X|Z)$ arbitrarily close approximate fair samples from $p(X|Z)$.

In SENMP, the measurement density $p(Z|X)$ represents the normalized fitness of the state X of a neuron, *i.e.* a high fitness corresponds to a high probability. Consequently, the *posterior* $p(X|Z)$ tends to evolve to a distribution $p(X|Z_{max})$ that tries to maximize the fitness of the set of N samples $s^{(n)}$ that represent the possible states of a neuron. When the adaptation process has converged, *i.e.* the diameter of the neuron distribution is large, the clusters of neurons are fairly localized and correspondingly, the *posterior* $p(X|Z)$ for each neuron is approximately unimodal. However, at the beginning of the adaptation process, while the diameter of the neuron distribution is still small

² $\sigma^2 = 0.025$ was used in the experiments presented in this work. This value was chosen in order to constrain the region of influence of any neuron approximately to the normalized distance 0.5, in which the Gaussian neighborhood function is close to zero.

and the variability of the distribution is correspondingly high, the *posterior* $p(X|Z)$ can be multimodal supporting multiple hypotheses. In other words, at the beginning of the adaptation process, SENMP permits the set of N samples $s^{(n)}$ of a neuron to populate a large area of the state space and to effectively explore the fitness landscape. However, as the process matures and the system converges the clusters of neurons get localized and the behavior of the system stabilizes.

After recombination, *i.e.* resampling of possible neuron states, the mutation operator is applied for all n neurons and their N states in such a way that for all k and i the new mutated state (gene) \vec{g}'_{k_i} is

$$\vec{g}'_{k_i} = \vec{g}_{k_i} + \vec{v},$$

where $k = 0, 1, \dots, n-1$ and $i = 0, 1, \dots, N-1$, and where

$$\vec{v} = (\nu_x, \nu_y, n_\theta(\nu_\theta), n(\nu_\phi), n(\nu_\tau), \nu_{\sigma^2})^T,$$

where $\nu_x, \nu_y, \nu_\theta, \nu_\phi$ and ν_τ get random values from the interval $[-\lambda, \lambda]$, ν_{σ^2} gets a random value from the interval $[-0.1\lambda, 0.1\lambda]$, $n_\theta(\nu) = 2\pi\nu/d$, and $n(\nu) = \nu/d$. After the mutation, τ_k and σ_k^2 of the new gene g'_k are constrained to be positive real numbers by replacing them with their corresponding absolute values. The reason why ν_{σ^2} is constrained to the interval $[-0.1\lambda, 0.1\lambda]$ is to emphasize the role of local interaction in the model, but at the same time to allow the region of influence of a neuron to vary in some limited degree in the long run.

The purpose of the normalization functions $n_\theta(\nu)$ and $n(\nu)$ is to control the short term variances of the parameters for which the functions are applied³. Normalization decreases the effect of mutation into these variables as the diameter of the neuron distribution increases. In other words, the diameter of the neuron distribution can be seen as the temperature of the system. Indeed, the evolution of the diameter of the distribution is a direct analogy to the cooling schedule of the simulated annealing method [13]. However, in SENMP, the 'cooling schedule' is controlled by the evolution process, not by the experimenter. If normalization is not used for the parameters θ, ϕ , and τ then $n_\theta(\nu) = n(\nu) = \nu$.

III. NON-MARKOVIAN DOUBLE POLE BALANCING PROBLEM

The pole balancing problem was selected for easier comparison of SENMP with existing neuroevolution (NE) methods as it has been used as a reinforcement learning benchmark for the last three decades [1], [8], [16], [21], [22], [25], [26].

The pole balancing experiments were set up as described in [21] and [22]. In double pole balancing experiments two rigid poles are connected to a moving cart by an hinge. The cart is constrained to move along a linear horizontal track. The neural network is used to apply force to the cart to keep the poles balanced for as long as possible without

³for convenience this is called normalization of that particular variable in the remaining text.

going beyond the boundaries of the track. The two poles must be of different length to be balanced, *i.e.* they respond differently to different control inputs [27]. In this problem, the ratio of the natural frequencies of the poles determines the size of the region of controllability [27].

The state of the cart-pole system, shown in Fig. 2, is defined by the cart position x and velocity \dot{x} , the first pole's position θ_1 and angular velocity $\dot{\theta}_1$, and second pole's position θ_2 and angular velocity $\dot{\theta}_2$. The motion equations were simulated by using the 4th order Runge-Kutta method with a step size of 0.02 seconds. All state variables were scaled to $[-1.0, 1.0]$ before being fed into the network. The network's output response, *i.e.* action, serves as a force on the simulated system. The force can get only two distinct values; -10N or 10N following the common "bang-bang" approach for controlling the cart-pole system. The poles were 0.1m and 1.0m long. The initial position of the long pole was randomly selected from the interval $[-1, 1]$ degrees and the short pole was set vertical. The length of the track was 2.4m. An acceptable solution for this problem must maintain both poles balanced for 30 minutes of simulated time, *i.e.* 100,000 time steps. The fitness of the neural network was measured as the number of time steps that both poles remained balanced. A pole was considered balanced between -36 and 36 degrees from vertical.

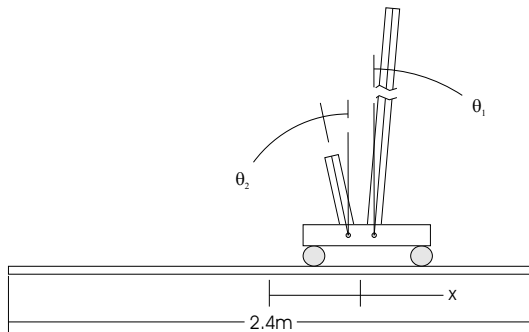


Fig. 2. The cart-pole system used in the double pole balancing experiments. An acceptable solution, *i.e.* neural controller must maintain the balance of the system for 30 minutes of simulated time without going beyond the boundaries of the track.

The "bang-bang" approach enforces the neural controller to apply non-zero force to the cart, making the cart-pole system unstable. Controlling an unstable and non-linear system with a neural network can be a difficult problem. Even though the proper behavior of the system might be describable, it is difficult to provide training examples for a supervised learning method for a system with complex dynamics. Under such conditions, a gradient-based search of the weight space for minimum error is unsuitable, making an Evolutionary Computation (EC) based procedure more attractive for addressing these kind of control problems.

In the non-Markovian version of the double pole balancing problem, the velocity information for both the cart (\dot{x}) and the poles ($\dot{\theta}_1$ and $\dot{\theta}_2$) are omitted which makes the task more difficult because the network has to estimate the internal state in order to compensate for the absence of the

velocity information, which requires recurrent connections. The schema of the non-Markovian double pole balancing experiment is illustrated in Fig. 3.

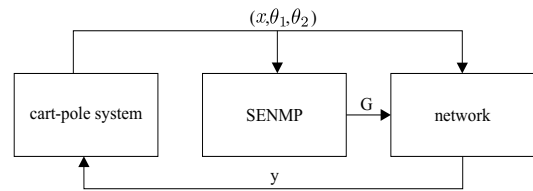


Fig. 3. Description of the non-Markovian double pole balancing experiment. y is the network's response, *i.e.* force applied to the cart, to the current state of the cart-pole system. G represents the current state, *i.e.* genotype of the neural network. SENMP receives the cart position and the angles of the poles in order to evaluate the performance of the network.

The network architecture used in the double pole balancing experiments was 3-10-2 with 1 bias node. The activation function used at the hidden and output layers was the hyperbolic tangent function. The most active output neuron represented the force, either 10N or -10N, applied to the cart. The network architecture is illustrated in Fig. 4.

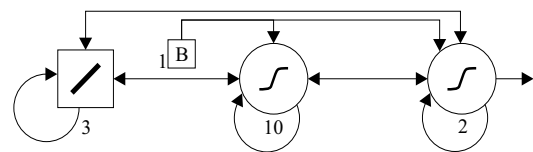


Fig. 4. The 3-10-2 network architecture used in the non-Markovian double pole experiments. The arrows are used to indicate fully connected layers. B represents bias nodes.

All experiments were conducted using Dynamic Recurrent Neural Networks (DRNNs) [4], [19]. The corresponding neuron model is shown in Eq. 3, where v_k is the action potential, of neuron k , w_{kj} is the weight of the connection from neuron j to neuron k , I_k is the external (sensory) input to the neuron k , τ_k and $\varphi(v_k)$ are the time constant, and the activation function of the neuron k , respectively.

$$\tau_k \dot{v}_k = -v_k + \sum w_{kj} \varphi(v_j) + I_k \quad (3)$$

SENMP was compared to the published results of three other NE systems: CE is Cellular Encoding [8], ESP is Enforced Subpopulations [7], and NEAT is NeuroEvolution of Augmenting Topologies [22], that are able to solve the non-Markovian double pole balancing problem. Table I shows that SENMP requires the fewest evaluations for finding a solution. In addition, SENMP finds solutions by using a population size, which is over an order of magnitude smaller than the next smallest population size suggesting that SENMP can effectively search the fitness landscape and maintain the diversity of the population despite the small population size used. The results presented in Table I were obtained without normalization of the feedback factors. For SENMP the results are averages over 120 experiments. Other results are averaged over 20 experiments. Standard deviations for the SENMP and

NEAT are 13,070, and 21,790 evaluations, respectively. The standard deviation for the other methods were not reported. The difference between the average number of evaluations of SENMP and NEAT is statistically significant ($p < 0.001$) favoring SENMP.

TABLE I

Method	Evaluations	Generations	No. Networks
CE	840,000	51	16,384
ESP	169,466	169	1,000
NEAT	33,184	33	1,000
SENMP	21,552	450	48

The recurrent connections are required in order to solve the non-Markovian double pole balancing problem, which was the reason why the experiments were repeated using the normalization of the feedback factors. The results are summarized in Table II. In this case, SENMP clearly shows a better performance in comparison with the other methods. Table II shows that with normalization, the solution is found using a smaller number of evaluations than without normalization. In fact, this time SENMP requires about half a magnitude less evaluations than the next best method, which is NEAT. The standard deviation for SENMP in this experiment was 2,102. SENMP results are averages over 85 experiments.

TABLE II

Method	Evaluations	Generations	No. Networks
CE	840,000	51	16,384
ESP	169,466	169	1,000
NEAT	33,184	33	1,000
SENMP	5,946	124	48

Fig. 5 shows the evolution of the neuron distribution of the neural network ensemble for four runs of the double pole balancing experiment. Fig. 5 illustrates how neuron clusters emerge and the neuron parameters θ , ϕ , and σ evolve.

The pole angles and the corresponding cart position for the neural network ensemble, corresponding to the neuron distribution in Fig. 5, is shown in Fig. 6 for 500 time steps, *i.e.* 10 seconds. The plots illustrate how the two poles are kept balanced and the cart remains within the required interval. The cart position plot in Fig. 6 is quasiperiodic with a period that is approximately 450 time steps, *i.e.* 9 seconds. The plot also shows that the cart is moving within the interval $[-0.5m, 0.5m]$, which is well within the required interval $[-1.2m, 1.2m]$. The data in Fig. 6 demonstrates clearly that SENMP is also able to solve the non-Markovian double pole balancing task.

IV. CONCLUSION

The motivation behind SENMP was to study if, and how, spatially distributed and laterally interacting computational entities could be arranged and tuned by a stochastic process, driven by selectionist pressure, to create dynamic systems that exhibit purposeful behavior. The stochastic

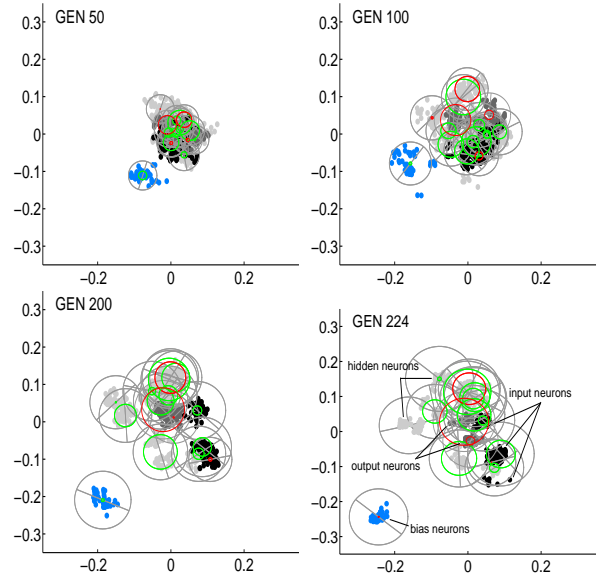


Fig. 5. Snapshots from the evolution of a neuron distribution during a non-Markovian double pole balancing experiment: generations 50, 100, 150, 200, 224. The average $\sigma (= \sqrt{\sigma^2})$ for each neuron cluster is shown with a circle at the center of gravity of the corresponding cluster of neurons. The mean feedback factor (scaled with σ) is shown with a smaller circle at the center of a neuron cluster. The line drawn through the center of a neuron cluster represents the average θ , *i.e.* phase of the neurons within the cluster.

process was selected rather than a development program, because the development program is difficult to formulate and is unsuitable for gaining new insights into the adaptation process of an artificial life-form.

SENMP is started from a dense cloud of neurons. The radius of this cloud can be interpreted as the temperature of the system. The gross features of the final state are seen at higher temperature (small radius), while fine details of the state appear at lower temperatures (larger radius), as the neurons migrate under the selective pressure implementing the necessary neural plasticity for the system. In this way SENMP can effectively maintain the diversity of the neural network population until a solution for the problem is found.

The feasibility of SENMP was demonstrated by evolving neural controllers that solve the non-Markovian double pole balancing problem. In the double pole balancing experiments, SENMP emerges as the most efficient approach, requiring the fewest evaluations and the smallest population size. In addition to the non-Markovian double pole balancing experiments, SENMP has earlier been used to evolve neural controllers for mobile robot navigation [10].

SENMP provides a novel and promising alternative for evolving neural controllers for dynamic systems such as mobile robots interacting with their environment [10]. The analysis⁴ of SENMP shows that the spatial distribution of neurons has a crucial role in the operation of SENMP, which gives further motivation to study systems composed

⁴excluded from this paper due to limited space.

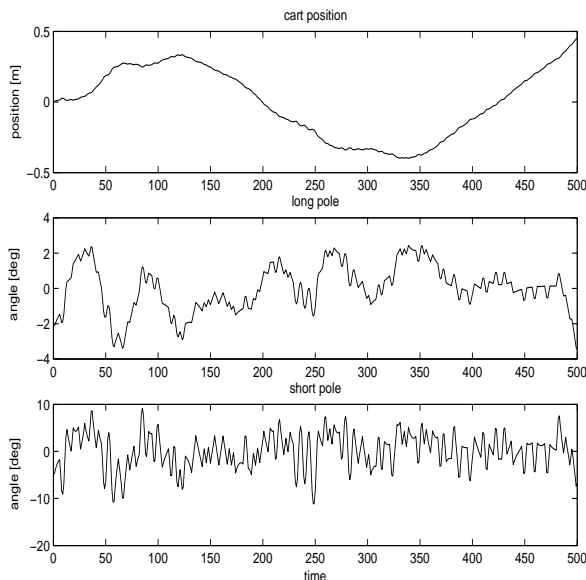


Fig. 6. The dynamics of the double pole system. These plots show the position of the cart and the angles of the two poles for 500 time steps, *i.e.* 10 seconds. The system is controlled by a neural network that represents an acceptable solution for the double pole balancing problem. The plots illustrate how the cart remains within the 2.4m long track, *i.e.* at the interval $[-1.2m, 1.2m]$ while the poles are kept balanced. This data is produced by the neural network ensemble shown in Fig. 5.

of laterally interacting computational entities and their dynamics.

Interestingly, the approach taken in SENMP is to some extent related to the spatial ecology [9], [23], which addresses the fundamental effects of space on the dynamics of individual species and on the structure, dynamics, diversity, and stability of multispecies communities. Indeed, it has been shown [23] that the spatial structure of a habitat can fundamentally alter the dynamics and outcomes of ecological processes. It is also clear that the spatial structure of a habitat and the resulting dynamics is an adaptation – a result of a complex stochastic selection process.

Based on the experiments conducted with SENMP it is argued that space have a fundamental role in the dynamics of a system consisting of laterally interacting entities. Furthermore, it is argued that rather simple mobile entities can be gradually guided by a selectionist process into a spatial pattern that exhibits purposeful dynamics with regard to a particular utility measure. In other words, if a suitable population of laterally interacting mobile entities exists, it is possible to gradually arrange the entities into a spatial pattern that exhibits the desired dynamics. Based on this observation, it can be hypothesized that biological evolution, for example, could utilize spatial patterns when creating purposeful complex dynamic systems – like our brain.

REFERENCES

[1] C. W. Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9:31–37, 1989.
 [2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking.

IEEE Transactions on Signal Processing, 50(2):174–188, February 2002.
 [3] J. Astor and C. Adami. A developmental model for the evolution of artificial neural networks. *Artificial Life*, 6(3):189–218, 2000.
 [4] Randall D. Beer and John C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, 1992.
 [5] A. Cangelosi, S. Nolfi, and D. Parisi. Cell division and migration in a ‘genotype’ for neural networks. *Network: Computation in Neural Systems*, 5:497–515, 1994.
 [6] J. Chavas, C. Corne, P. Horvai, and J. Kodjabachian. Incremental evolution of neural controllers for robust obstacle-avoidance in Khepera. *Lecture Notes in Computer Science*, 1468, 1998.
 [7] F. Gomez and R. Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1356–1361, Denver, 1999. Morgan Kaufmann.
 [8] F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 81–89, 1996.
 [9] Ilkka Hanski. Metapopulation dynamics. *Nature*, 396(5):41–49, 1998.
 [10] Janne Haverinen and Juha Röning. Adaptation through a stochastic evolutionary neuron migration process (SENMP). In *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1008–1013, EPFL, Lausanne, Switzerland, Sep 30 - Oct 4 2002.
 [11] Phil Husbands. Evolving robot behaviours with diffusing gas networks. In P Husbands and J Meyer, editors, *Proceedings of the First European Workshop on Evolutionary Robotics*, pages 123–136. Springer Verlag, 1998.
 [12] N. Jakobi and M. Quinn. Some problems (and a few solutions) for open-ended evolutionary robotics. In P Husbands and J-A Meyer, editors, *Proceedings of the First European Workshop on Evolutionary Robotics: EvoRobot98*, pages 108–122. Springer Verlag, 1998.
 [13] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
 [14] J. Kodjabachian and J.-A. Meyer. Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects. *IEEE Transactions on Neural Networks*, 9(5):796, September 1998.
 [15] G. O. Mackie. The elementary nervous system revisited. *American Zoologist*, 30:907–920, 1990.
 [16] D. Michie and R. A. Chambers. Boxes: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence*, pages 125–133. Oliver and Boyd, Edinburgh, UK, 1968.
 [17] S. Nolfi and D. Parisi. Growing neural networks. Technical Report PCIA-91-15, Institute of Psychology, CNR, Rome, 1991.
 [18] Stefano Nolfi and Domenico Parisi. Evolving artificial neural networks that develop in time. In *European Conference on Artificial Life*, pages 353–367, 1995.
 [19] B. A. Pearlmutter. Dynamic recurrent neural networks. Technical Report MCU-CS-90-196, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1990.
 [20] B. D. Ripley. *Stochastic Simulation*. New York: Wiley, 1987.
 [21] N. Saravanan and D. B. Fogel. Evolving neural control systems. *IEEE Expert*, 10(3):23–27, 1995.
 [22] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.
 [23] David Tilman. *Spatial Ecology: The Role of Space in Population Dynamics and Interspecific Interactions*. Princeton University Press, 1997.
 [24] Jari Vaario, Akira Onitsuka, and Katsunori Shimohara. Formation of neural structures. In P. Husbands and I. Harvey, editors, *The Proceedings of the Fourth European Conference on Artificial Life*, pages 214–233. The MIT Press, 1997.
 [25] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.
 [26] D. Whitley, S. Dominic, R. Das, and C. W. Anderson. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284, 1993.
 [27] A. Wieland. Evolving controls for unstable systems. In *Proceedings of the 1990 Connectionist Models Summer School*, pages 91–102, San Francisco, 1990. Morgan Kaufman.