Evolving Networks Processing Signals with a Mixed Paradigm, Inspired by Gene Regulatory Networks and Spiking Neurons

Borys Wróbel[1,2,3], Ahmed Abdelmotaleb[2], and Michał Joachimczak[3]

[1] Systems Modelling Laboratory, IOPAS in Sopot (Poland)
[2] Evolutionary Systems Laboratory, Adam Mickiewicz University in Poznan (Poland)
[3] Institute for Neuroinformatics, University of Zurich / ETHZ (Switzerland),
wrobel@evosys.org

Abstract. In this paper we extend our artificial life platorm, called GReaNs (for Genetic Regulatory evolving artificial Networks) to allow evolution of spiking neural networks performign simple computational tasks. GReaNs has been previously used to model evolution of gene regulatory networks for processing signals, and also for controling the behaviour of unicallular animats and the devlopment of multicellular structures in two and three dimensions. The connectivity of the regulatory network in GReaNs is encoded in a linear genome. No explicit restrictions are set for the size of the genome or the size of the network. In our previous work, the way the nodes in the regulatory network worked was inspired by biological transcriptional units. In the extension to genes here we modify the equations governing the behaviour of the units so that they describe spiking neurons: either leaky integrate and fire neurons with a fixed threshold or adaptive-exponential integrate and fire neurons. As a proof-of-principle, we report the evolution of spiking networks that match desired spiking patterns.

Key words: evolutionary algorighms, gene regulatory networks, spiking neural networks, signal processing, leaky integrate and fire neurons, adaptive-exponential neurons

1 Introduction

Drawing from biology when building artificial systems aims to distil the crucial features of natural processes which allow for the existence and constant generation of extremely complex, efficient entities – biological organisms. The field of Artificial Life (AL), closely related to Artificial Intelligence, aims to understand-by-construction these processes, in order to understand the life – including intelligent life – on our planet and possibly also elsewhere in the universe, and in order to allow for novel technological approaches.

Research in AL often centres around software (much less often hardware) platforms which are based on a particular biologically-inspired paradigm. We are building one such platform, based on the paradigm of the encoding of regulatory network in a linear genome. This platform, GReaNs (which stands for

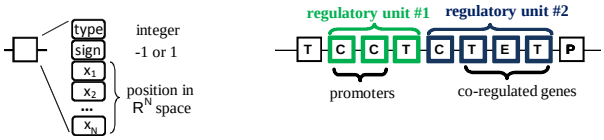

Fig. 1. Schematic structure of the genome in GReaNs. The left part shows the internal structure of a genetic element which consists of an integer specifying the type (*cis-regulator*, C; *trans-regulator*, T; or *external element*, E), a sign field (1 or -1, which determines if a trans-cis interaction is inhibitory or activatory), and real numbers that specify a point in space (which determines trans-cis affinity). The genome (left) is a series of genetic elements that build regulatory units, which correspond to nodes in the regulatory network. A regulatory unit is at least one cis-regulator followed by at least one trans-regulator. External elements correspond to the inputs and outputs of the network.

Gene Regulatory evolving artificial Networks) has been previously used to evolve regulatory networks to control development of multicellular structures in two dimensions [1, 2, which also introduced a method to transform the structures into soft-bodied animats swimming in a fluid-like environment] and three dimensions [3, 4, 5], and to investigate the ability of single cells to forage for resources in an artificial environment [6, 7] and to process signals [8].

In our previous work we were inspired by the way transcriptional units work in biology to formulate the rules governing the nodes in the regulatory network. The structure of the network in GReaNs is encoded in a linear genome (Fig. 1). The genome consists of "regulatory units": series of "genetic elements" of type C, followed by a series of elements of type T. C stands for "cis-regulators" (it is common to use the term promoters in the field, although this does not follow strictly the biological nomenclature). T elements code for "products", some of which act as "trans-regulators". Trans-regulators have "affinity" to cis-regulators. This affinity determines the computational properties of the network. All products can change concentration in each simulation step. All products coded in the same regulatory unit have the same concentration. In addition to the elements of type C and T, there are elements of type E. This letter stands for "external": the elements of type E allow for external inputs and outputs of the network. Each genetic element is by itself a series of numbers: an integer specifying the type, a bit specifying the sign (signs determine if a particular cis-trans interaction is inhibitory or excitatory), and real numbers (coordinates) that specify a point in space (affinity is a function of the Euclidean distance between the corresponding points). Because the connection is always from a product-coding element (a "gene") to the cis-regulator, the connections between regulatory units are asymmetric, and the net regulatory effect of one unit on the other results from the combined effect of all trans-regulators coded in the former on all cis-elements of the latter. An important feature of our encoding is

that the topology of the network is not restricted, although in most experiments (also here) we do not allow for direct connections between the input and the output. However, we do not limit in any way the size of the genome, and thus of the network. There is, of course, a limit imposed by the computer memory, but it is never reached in practice.

In our previous papers [1-8] each regulatory unit in GReaNs was considered to be an analog of a node in a biological gene regulatory network. But they could equally well be seen as computationally equivalent to neurons (or groups of neurons) in a neural network. Indeed, the recurrent networks explored previously in GReaNs can be seen as networks of discretely or continuously variable artificial neurons (cf. [9, 10]). In this paper we explore the analogy further by analysing the evolvability of a model in which linear genomes encode the structure of a spiking neural network (SNN), in other words, when the equations governing each computational unit produce a spiking behaviour. We have recently introduced two models in GReaNS: the leaky integrate and fire model with a fixed threshold (LIF) [11, 12, 13] and the adaptive exponential model (AdEx) [12, 13, 14, 15]. In biology the encoding of the neural structures in the genome is much more indirect, so our model has at this point has little to do with biological reality. However, the introduction of spiking neuron models in GReaNs is the first step towards a more biologically plausible model. Furthermore, this introduction allows us to investigate the evolvability of this encoding, and its potential for possible practical applications.

2 Evolving Spiking Neural Networks in GReaNs

To provide more detail to the description above, the connectivity of the network is specified in GReaNs by trans-cis affinity, using a function that relates the weights of the connections in an inverse exponential way to the Euclidean distance between points in the abstract space of interactions, with a cut-off value to prevent full connectivity. Once the topology of the network and the weights are determined, every regulatory unit is transformed into a single neuron, the value of the concentration of the products in the unit to the neuron membrane potential, and the connections between the regulatory units, to synapses.

In the experiments described here, units are either of type LIF or AdEx. In the former case, they are described by the equation:

$$\frac{dV}{dt} = \frac{g_L(V_R - V) + g_E(E_{rev,E} - V) + g_I(E_{rev,I} - V)}{C} \tag{1}$$

where V is the membrane potential, $V_R = -65.0$ mV is the resting potential, $g_L = 0.05$ μS is the leak conductance, g_E is the conductance of the excitatory synapses, g_I is the conductance of the inhibitory synapses, $E_{rev,E} = 0$ mV is the reversal potential of the excitatory input, $E_{rev,I} = -70.0$ mV is the reversal potential of inhibitory input, and $C = 1$ nF is the capacitance of the membrane. When the membrane potential of a LIF neuron crosses the threshold

($V_{threshold} = -50$ mV), a stereotypical spike is created: the membrane potential is first forced to reach a maximum value, and then it decreased to the reset voltage value ($V \leftarrow V_{reset}$, $V_{reset} = -70.0$ mV).

The AdEx neurons are described by two equations:

$$\frac{dV}{dt} = \frac{g_L(E_L - V) + \delta e^{\frac{V - V_T}{\delta}} + g_E(E_E - V) + g_I(E_I - V) - W}{C} \tag{2}$$

$$\frac{dW}{dt} = \frac{a(V - E_L) - W}{\tau_W} \tag{3}$$

where V is the membrane potential, $E_L = -65.0$ mV is the leak reversal potential, $\delta = 2.0$ mV is the slope factor, $V_T = 50.0$ mV is the threshold potential, $g_L = 0.05$ S is the leak conductance, g_E (or g_I) is the conductance of the excitatory (inhibitory) synapses, $E_E = 0$ mV ($E_I = -70.0$ mV) is the reversal potential of the excitatory (inhibitory) input, W is the adaptation variable, $C = 1$ nF is the capacitance of the membrane, $a = 4.0$ nS is the adaptation coupling parameter, and $\tau_w = 40.0$ ms is the adaptation time constant.

In contrast to the LIF neurons, AdEx neurons do not have a threshold for spike generation, but when the neuron is activated enough, its membrane potential will diverge to infinity. It is necessary, therefore, to specify the maximum value of the potential. When this value is reached ($V_{spike} = -40.0$ mV), both the potential and the adaptation variable are decreased or increased, respectively, in a discontinuous manner ($V \leftarrow V_{reset}, w \leftarrow w + b$, where $V_{reset} = -65.0$ mV, and $b = 0.0805$ nA). It has been shown that the behaviour of the AdEx model does not depend on the value of the maximum potential. In other words, the behaviour would be essentially the same if a more biologically realistic value (e.g., +40 mv) was used for V_{spike} [15].

In GReaNs, as in other implementations of SNNs, a spike is marked at time t when the threshold is crossed ($V = V_{threshold}$), for the LIF neuron, or $V = V_{spike}$, for AdEx. This spike will arrive at all the connected neurons at the next simulation time step ($t + 1$), and the membrane potential of these neurons will be affected by this spike starting from the next step still ($t + 2$). The way it is affected depends on the weight of the synaptic connection (determined by the Euclidean distance between the points specified by the genetic elements, in this paper we use two coordinates for both LIF and AdEx) and its sign (positive for excitatory, negative for inhibitory; a product of the sign of the genetic elements).

The conductance of the synapses was modeled using decaying exponential functions:

$$\frac{dg_E}{dt} = \frac{-g_E}{\tau_E} \quad \text{and} \quad \frac{dg_E}{dt} = \frac{-g_I}{\tau_I} \tag{4}$$

where $\tau_E = 5.0$ ms and $\tau_I = 5.0$ ms are the decay time constants of the excitatory and inhibitory synaptic conductance, respectively. Each arriving spike produces an increase of conductance which is directly proportional to the weight of the synapse.

In this paper we use a genetic algorithm to evolve SNNs which are able to generate a specific spike pattern if presented with a specific input. The population size was kept constant (300 individuals). The genetic algorithm used the tournament selection (pick two, select the best). The genomes in the initial population were generated randomly (with random signs and coordinates of the elements), with 5 regulatory units, but a random number of C and T elements in each. At each generation, a new population was formed by taking the best 5 individuals from the previous one without mutation (elitism), 100 with crossover and mutation, and 195 with mutation only. Mutations could change the element type, sign, coordinates, or consist of a deletion of a random number of elements or a duplication of a random number at a random position.

While we use GReaNs for the time-integration of the network, we have allowed for the export of the network description using PyNN [12] which allows testing the network with other simulation backends, e.g. Brian simulator [13] (We have used IF_cond_exp class for LIF and EIF_cond_exp_isfa_ista class for AdEx in pyNN, and Euler integration with 1 ms integration time step in both Brian and GReaNs).

The genetic algorithm used a fitness function which rewards the match between the output of the recurrent network in GReaNs with the desired output:

$$f_{fit} = \frac{\alpha|S_{desired} - S_{GReaNs}| + \beta(S_{desired} - m_{GReaNs})}{S_{desired}} \tag{5}$$

where $S_{desired}$ is the desired number of spikes, S_{GReaNs} is the number of spikes generated by a network under evaluation, m_{GReaNs} is the number of matching spikes in these two spike trains, while α and β are constant fractions ($\alpha + \beta = 1$, we used $\alpha = 0.3$ and $\beta = 0.7$ in this work). We have searched for matching spikes over the window $[t_{desired} - 9, t_{desired} + 9]$ and weighted them using a Gaussian function, $f_{Gaussian} = e^{(t_{desired} - t_{GReaNs})^2/15}$, where $t_{desired}$ is the spike time in the desired output, and $t_{desired}$ is the spike time in GreaNs. Only one specific spike train was used for evaluation during evolution (Fig. 2, 3, 4, 5).

3 Results

In this paper we used GReaNs to evolve recurrent SNNs able to perform two types of tasks. In one task the network generates a specific spike train in response to a specific input, generated by a Poisson spike source with 100 Hz spike frequency. The target spike train was generated by one LIF or AdEx neuron (modeled, for convenience, using Brian, with the same parameters as the LIF/AdEx neurons in the evolving network, with the exception of using $V_{reset} = -65.0$ mV for the LIF neuron in Brian).

Because direct connection of the input to the output was not allowed in the evolving networks, and because synaptic connection introduce delays, it is not possible to match the desired output exactly unless the spikes are "shifted". Therefore, in this type of task we evolved networks which produced an output shifted (in comparison to one LIF or AdEx neuron) by 20 ms.

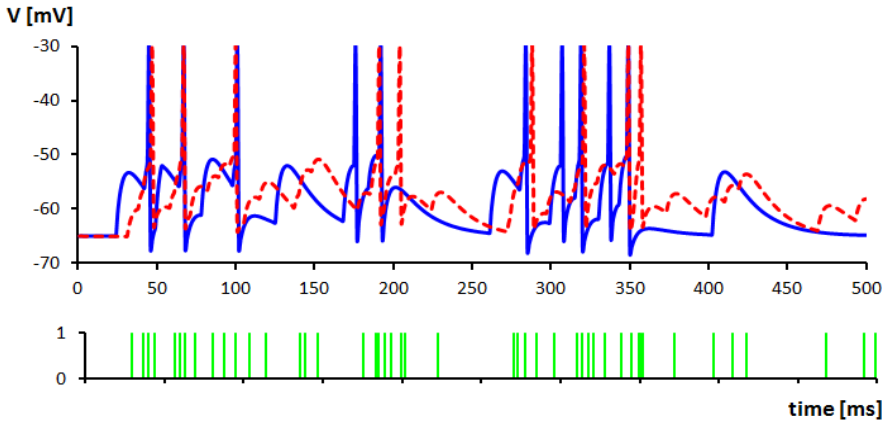

Fig. 2. The membrane potential of the output neuron of the best network (in terms of generalization, in 10 independent evolutionary runs) of LIF neurons evolved with GReaNs (blue line) to match the spikes of one AdEx neuron (red line), shifted by 20 ms in response to the Poisson spike train (green) used during evolution.

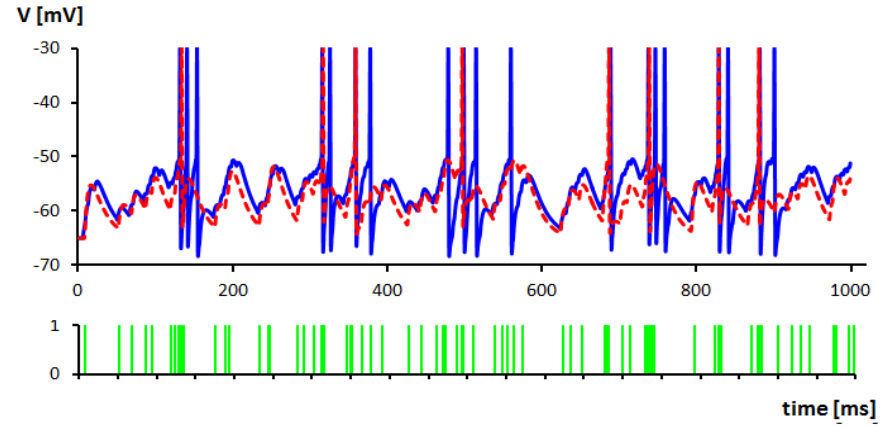

Fig. 3. The membrane potential of the output neuron of the best network (in terms of generalization, in 10 independent evolutionary runs) of LIF neurons evolved with GReaNs (blue line) to double the spikes of one AdEx neuron (red line), shifted by 5 ms in response to the Poisson spike train (green) used during evolution.

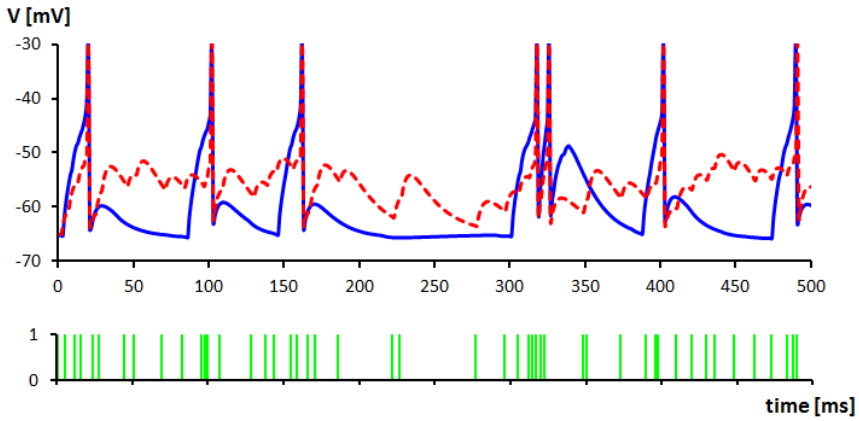

Fig. 4. The membrane potential of the output neuron of the best network (in terms of generalization, in 10 independent evolutionary runs) of AdEx neurons evolved with GReaNs (blue line) to match the spikes of one LIF neuron (red line), shifted by 20 ms in response to the Poisson spike train (green) used during evolution.

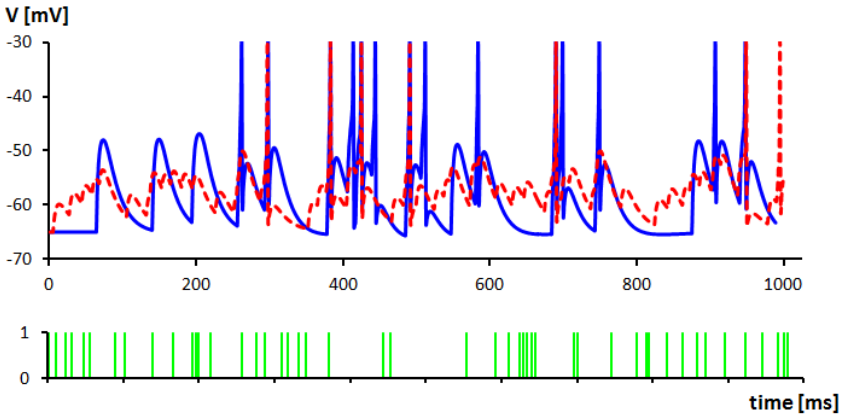

Fig. 5. The membrane potential of the output neuron of the best network (in terms of generalization, in 10 independent evolutionary runs) of AdEx neurons evolved with GReaNs (blue line) to double the spikes of one LIF neuron (red line), shifted by 5 ms in response to the Poisson spike train (green) used during evolution.

The second task consists also of producing a specific output, but here the target is created by first presenting the input (also generated by a Poisson source, with 75 Hz frequency) to, again, one neuron (same as above, modeled using Brian), then shifting the spikes by 5 ms, and then adding one spike 20 ms after each spike generated by the Brian neuron. In other words, the tasks consists of producing a "shifted-doubled" output.

For each task, we have evolved a network or LIF neurons to match (shifted or shifted-doubled) output of one AdEx neuron or vice versa. We have run 10 independent evolutionary runs in each case, with either 500 generations (for shifting) or 750 generations (for shifting-doubling). This proved sufficient to obtain networks able to perform these tasks (Fig. 2, 3, 4, 5); the average value for 10 runs of $f_{fit} = 0.24$ for the shifting networks of LIF neurons, $f_{fit} = 0.23$, for shifting-doubling; for AdEx, the average was, respectively: $f_{fit} = 0.087$, $f_{fit} = 0.258$), although some runs resulted in slightly worse results (the range of values and the standard deviation was, respectively: $range = [0.146, 0.309]$, $\sigma = 0.046$, $range = [0.112, 0.3]$, $\sigma = 0.06$, $range = [0.006, 0.158]$, $\sigma = 0.065$, $range = [0.17, 0.301]$, $\sigma = 0.044$).

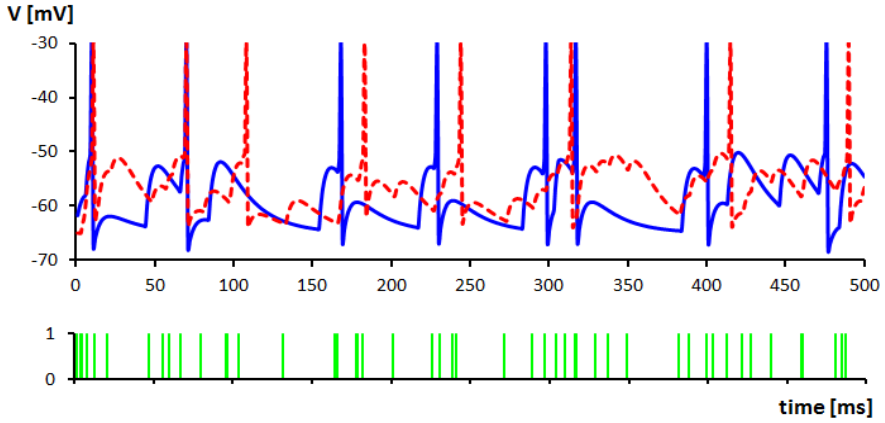

Fig. 6. The membrane potential of the output neuron of the best network (in terms of generalization, in 10 independent evolutionary runs) ofof LIF neurons evolved with GReaNs (blue line) to match the spikes of one AdEx neuron (red line), shifted by 20 ms in response to a different spike train, not used during evolution.

We have then tested if the networks could generalize the tasks: we tested what was the fitness value when the desired output was generated using a different specific Poisson spike train as an input (Fig. 6, 7, 8, 9). The values of the fitness function were higher than for the spike train used during evolution (the genetic algorithm aims to minimize the f_{fit}), but still quite low for shifting (the average

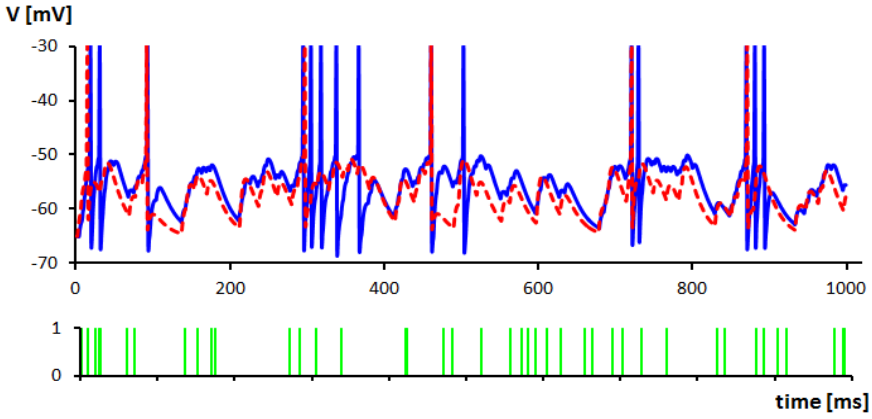

Fig. 7. The membrane potential of the output neuron of the best network (in terms of generalization, in 10 independent evolutionary runs) of LIF neurons evolved with GReaNs (blue line) to double the spikes of one AdEx neuron (red line), shifted by 5 ms in response to a different spike train, not used during evolution.

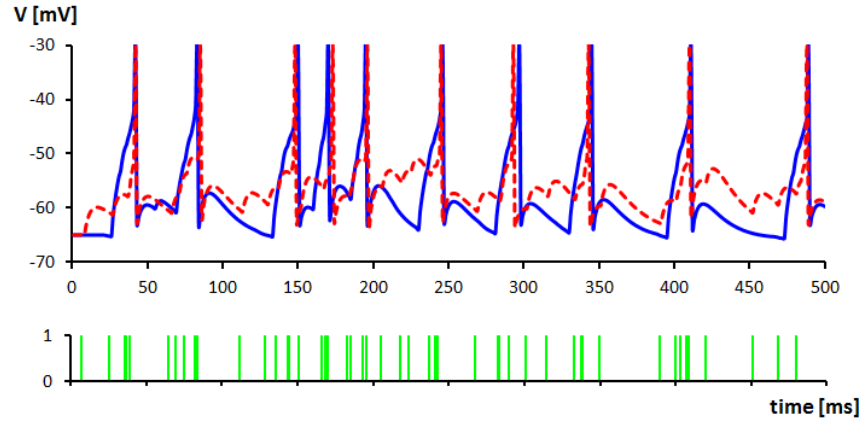

Fig. 8. The membrane potential of the output neuron of the best network (in terms of generalization, in 10 independent evolutionary runs) of AdEx neurons evolved with GReaNs (blue line) to match the spikes of one LIF neuron (red line), shifted by 20 ms in response to a different spike train, not used during evolution.

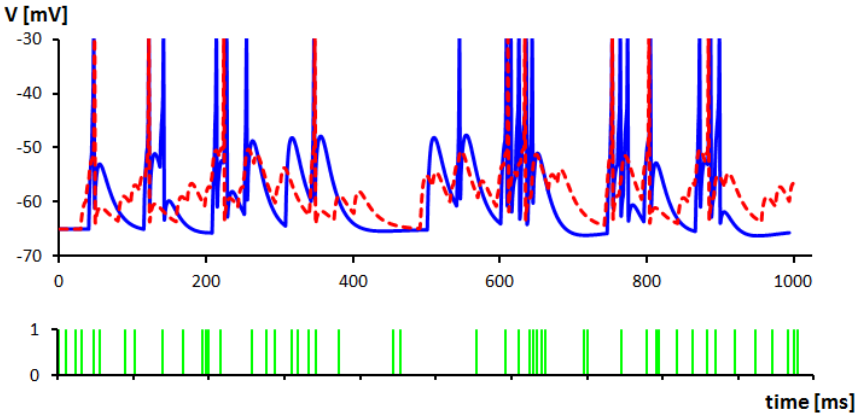

Fig. 9. The membrane potential of the output neuron of the best network (in terms of generalization, in 10 independent evolutionary runs) of AdEx neurons evolved with GReaNs (blue line) to double the spikes of one LIF neuron (red line), shifted by 5 ms in response to a different spike train, not used during evolution.

for networks of LIF neurons: $f_{fit} = 0.729$, $range = [0.517, 0.963]$, $\sigma = 0.137$, for AdEx: $f_{fit} = 0.39$, $range = [0.142, 0.625]$, $\sigma = 0.185$), and even for shifting-doubling (the average for LIF networks: $f_{fit} = 0.379$, $range = [0.29, 0.434]$, $\sigma = 0.058$, AdEx: $f_{fit} = 0.478$, $range = [0.373, 0.503]$, $\sigma = 0.044$). Not surprisingly, there was some trade-off between the value of f_{fit} that a particular network has reached during evolution (this can be seen as the performance for the training set) and its generalization. For example, the best shifting network of LIF neurons in 10 runs had $f_{fit} = 0.146$ but only $f_{fit} = 0.65$ when tested on a different input, while the best generalizing network had the values of $f_{fit} = 0.285$, and $f_{fit} = 0.517$, respectively. The situation was similar for shifting-doubling LIF networks (the best network had $f_{fit} = 0.112$ for the "training set", and for a different input: $f_{fit} = 0.327$, while the best generalizing network had, respectively, $f_{fit} = 0.239$ and $f_{fit} = 0.29$). For the AdEx networks, the best shifting network had also the best generalization (for the "training set" $f_{fit} = 0.006$, for a different input: $f_{fit} = 0.142$), but the trade-off was observed for shifting-doubling (the best network for the "training set" had $f_{fit} = 0.171$ in training and $f_{fit} = 0.5$ for generalization, the best generalizing network, respectively, $f_{fit} = 0.301$ and $f_{fit} = 0.373$).

While shifting-doubling solutions generalized less efficiently, they did so still quite well. Interestingly, although shifting could in principle be accomplished by a feed-forward network in which the first layer, connected directly to the input, has one neuron (or several) which would generate the desired output without any shift (after all, it was generated by one neuron) and the neurons in subsequent

layers would generate a spike for every spike received (to provide the shift). Shifting-doubling could be achieved in a similar fashion, using two feed-forward networks (of different sizes) working in parallel. However, we did not observe any such trivial topology. Instead, all the best networks were heavily interconnected, with recurrent connections, so (as is typical in the field of evolving networks) it is difficult to find out how they actually work. The analysis of the spiking pattern of the interneurons did, however, reveal that there is sometimes a match between their spiking pattern and the target pattern. Perhaps even more interestingly, we have observed the presence of neurons spiking with very high frequency in the shifting-doubling networks. We plan to investigate this latter issue in more detail in further work.

4 Discussion and Future Work

We have described here a biologically-inspired (even if not biologically plausible) system for evolution of SNNs. In our system the networks are encoded in a linear genome without imposing any limitations on the size of the network or on the size of the genome. We use this encoding and the genetic operators inspired by the biological mutations (point mutations, deletions, duplications; the last two may also result from cross-over) to evolve recurrent SNNs able to perform simple single processing tasks.

Our platform, GReaNs, is only one of several existing approaches to the evolution of genetic and neural networks (see e.g., [16] and references therein for examples, and [17] and references therein for approaches involving SNNs). Obviously, other software platforms and paradigms differ in design details. For example, when compared to Analog Genetic Encoding [16], GReaNs uses an entirely different way of specifying the connection weights, and the number of connections per unit is not pre-specified.

The networks evolved in this paper are able to perform quite simple tasks, but these tasks are not entirely trivial. We use these tasks as a proof-of-principle of the evolvability of our system after its extension with SNN models. Of course, we would like to use GReaNs to evolve networks able to perform even more difficult tasks, for possible applications in signal processing. We believe that such an evolutionary approach would be interesting for the field of neuromorphic engineering, because the evolved networks could be implemented in neuromorphic hardware [18], or even evolved using such hardware.

Indeed, both models used here, LIF and AdEx, have been implemented in neuromorphic silicon neuron circuits (both analog circuits and digital very large scale integration circuits, and implementations are available also for multi-core based architectures, based on multiple ARM cores and GPUs) [17, and references therein]. Such circuits might allow for a speed-up of the evolutionary process, especially for larger networks. This is the direction of research we are already pursuing, allowing for the possibility of interfacing through different software/hardware platforms using Python scripts (such as PyNN [12]).

In simulations reported here we have investigated the evolution of SNNs with homogenous nodes. Not only were all the nodes of the same type, but they all had the same, specific values of the parameters in the equations describing their behaviour. We are currently introducing the possibility of encoding neurons with different parameters, specifying various behaviours (e.g., bursting or chaotic response [14, 15]). The approach we would like to explore first is not to evolve the parameters in a continuous fashion, but rather to specify several stereotypical neurons (with particular behaviour). In other words, one mutation in the system would change the type of the neuron (setting in one step another set of values specific for another stereotype). We believe that this approach can keep the size of the search space small enough for the genetic algorithm to explore efficiently the search space of solutions.

Acknowledgement. This work was supported by the Polish Ministry of Science and Higher Education (project 2011/03/B/ST6/00399 to BW); BW acknowledges the support of the Swiss-Polish Research Fund, AA the support of the Foundation for Polish Science, co-financed by EU Regional Development Fund (Innovative Economy Operational Programme 2007-2013). We are also grateful to Volker Steuber for discussions.

References

[1] Joachimczak, M., Kowaliw, T., Doursat, R., Wróbel, B.: Brainless bodies: Controlling the development and behavior of multicellular animats by gene regulation and diffusive signals. In: Artificial Life XIII: Proceedings of the 13th International Conference on the Simulation and Synthesis of Living Systems, MIT Press (2012) 349–356

[2] Joachimczak, M., Wróbel, B.: Co-evolution of morphology and control of soft-bodied multicellular animats. In: Proceedings of the 14th International Conference on Genetic and Evolutionary Computation. GECCO '12, ACM (2012) 561–568

[3] Joachimczak, M., Wróbel, B.: Evo-devo *in silico*: a model of a gene network regulating multicellular development in 3D space with artificial physics. In: Artificial Life XI: Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems, MIT Press (2008) 297–304

[4] Joachimczak, M., Wróbel, B.: Evolution of the morphology and patterning of artificial embryos: Scaling the tricolour problem to the third dimension. In: Advances in Artificial Life. Darwin Meets von Neumann: Proceedings of the 10th European Conference on Artificial Life (ECAL 2009). Volume 5777 of LNCS., Springer (2011) 35–43

[5] Joachimczak, M., Wróbel, B.: Open ended evolution of 3d multicellular development controlled by gene regulatory networks. In: Artificial Life XIII: Proceedings of the 13th International Conference on the Simulation and Synthesis of Living Systems, MIT Press (2012) 67–74

[6] Joachimczak, M., Wróbel, B.: Evolving gene regulatory networks for real time control of foraging behaviours. In: Artificial Life XII: Proceedings of

the 12th International Conference on the Simulation and Synthesis of Living Systems, MIT Press (2010) 348–355
[7] Wróbel, B., Joachimczak, M., Montebelli, A., Lowe, R.: The search for beauty: Evolution of minimal cognition in an animat controlled by a gene regulatory network and powered by a metabolic system. Volume 7426., Springer Berlin Heidelberg (2012) 198–208
[8] Joachimczak, M., Wróbel, B.: Processing signals with evolving artificial gene regulatory networks. In: Artificial Life XII: Proceedings of the 12th International Conference on the Simulation and Synthesis of Living Systems, MIT Press (2010) 203–210
[9] Flood, I., Kartam, N.: Artificial Neural Networks for Civil Engineers: Advanced Features and Applications. American Society of Civil Engineers (1998)
[10] Beer, R.D.: On the dynamics of small continuous-time recurrent neural networks. Adaptive Behavior **3**(4) (1995) 469–509
[11] Dayan, P., Abbott, L.F.: Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. First edn. The MIT Press (2001)
[12] Davison, A.P., Brüderle, D., Eppler, J., Kremkow, J., Muller, E., Pecevski, D., Perrinet, L., Yger, P.: Pynn: A common interface for neuronal network simulators. Front. Neuroinform. **2** (2008)
[13] Goodman, D., Brette, R.: Brian: a simulator for spiking neural networks in python. Front. Neuroinform. **2** (2008)
[14] Brette, R., Gerstner, W.: Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. J. Neurophysiol. **94**(5) (2005) 3637–3642
[15] Touboul, J.: Bifurcation analysis of a general class of nonlinear integrate-and-fire neurons. SIAM J. Appl. Math **68**(4) (2008) 1045–1079
[16] Mattiussi, C., Floreano, D.: Analog genetic encoding for the evolution of circuits and networks. Trans. Evol. Comp **11**(5) (2007) 596–607
[17] Veredas, F.J., Vico, F.J., Alonso, J.M.: Evolving networks of integrate-and-fire neurons. Neurocomput. **69**(13-15) (2006) 1561–1569
[18] Indiveri, G., Linares-Barranco, B., Julia, T., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.C.C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., Boahen, K.: Neuromorphic silicon neuron circuits. Front. Neurosci. **5** (2011)