On the automatic calibration of fully analogical spiking neuromorphic chips

Daniele M. Papetti
Department of Informatics, Systems and Communication
University of Milano-Bicocca
Milan, Italy
d.papetti1@campus.unimib.it

Simone Spolaor
Department of Informatics, Systems and Communication
University of Milano-Bicocca
Milan, Italy
simone.spolaor@disco.unimib.it

Daniela Besozzi
Department of Informatics, Systems and Communication
University of Milano-Bicocca
SYSBIO/ISBE.IT Centre of Systems Biology
Milan, Italy
daniela.besozzi@unimib.it

Paolo Cazzaniga
Department of Human and Social Sciences
University of Bergamo
Bergamo, Italy
SYSBIO/ISBE.IT Centre of Systems Biology
Milan, Italy
paolo.cazzaniga@unibg.it

Marco Antoniotti
Department of Informatics, Systems and Communication
Milan Center for Neuroscience
University of Milano-Bicocca
Milan, Italy
marco.antoniotti@unimib.it

Marco S. Nobile
Department of Industrial Engineering and Innovation Sciences
Eindhoven University of Technology
Eindhoven, The Netherlands
SYSBIO/ISBE.IT Centre of Systems Biology
Milan, Italy
m.s.nobile@tue.nl

***Abstract*—Nowadays, understanding the topology of biological neural networks and sampling their activity is possible thanks to various laboratory protocols that provide a large amount of experimental data, thus paving the way to accurate modeling and simulation. Neuromorphic systems were developed to simulate the dynamics of biological neural networks by means of electronic circuits, offering an efficient alternative to classic simulations based on systems of differential equations, from both the points of view of the energy consumed and the overall computational effort. Spikey is a configurable neuromorphic chip based on the Leaky Integrate-And-Fire model, which gives the user the possibility to model an arbitrary neural topology and simulate the temporal evolution of membrane potentials. To accurately reproduce the behavior of a specific biological network, a detailed parameterization of all neurons in the neuromorphic chip is necessary. Determining such parameters is a hard, error-prone, and generally time consuming task. In this work, we propose a novel methodology for the automatic calibration of neuromorphic chips that exploits a given neural activity as target. Our results show that, in the case of small networks with a low complexity, the method can estimate a vector of parameters capable of reproducing the target activity. Conversely, in the case of more complex networks, the simulations with Spikey can be highly affected by noise, which causes small variations in the simulations outcome even when identical networks are simulated, hindering the convergence to optimal parameterizations.**

***Index Terms*—Spikey, neuromorphic chip, neural networks, PyNN, Fuzzy Self-Tuning PSO, swarm intelligence, global optimization, modeling and simulation**

I. Introduction

Neuromorphic Chips (NC) are devices designed to mimic neuro-biological architectures by means of electric analog circuits. The idea behind these systems is to represent efficient alternatives to the computational simulation of neural systems from both the points of view of speed and power consumption. NCs have been applied in different contexts [1], ranging from integrate and fire spiking circuits used in neuromorphic vision sensors, to large-scale neural systems realizing fast and parallel computation, or from real-time large-scale neural emulation, to bidirectional brain–machine interfaces. Other implementations of neuromorphic hardware focused on the realization of machine learning techniques, as in the work presented in [2], where a device with artificial synapses was introduced to implement unsupervised learning.

Among the existing implementations [3], the Spikey chip [4] is a peculiar reconfigurable neuromorphic system, able to simulate the temporal evolution of the membrane potentials of a network consisting of up to 384 neurons, governed by a conductance-based Leaky Integrate-And-Fire (LIF) model [5]. All measurable quantities in Spikey's circuitry have corresponding biological equivalents. For example, a neuron's membrane potential is modeled by the voltage over a capacitor that, in turn, represents a model of the capacitance of the cell membrane. Thanks to this approach, the dynamics of potentials evolve continuously in time. Each neuron can receive input signals from up to 256 synapses, whose connection topology and weights can be arbitrarily controlled by the user.

Although Spikey can only provide support for small neural networks, it could (in principle) be used to replicate the emergent phenomena observed in some specific neural region of model animals, like insects. Once the topology is defined—for instance, by replicating the connections observed by means of confocal imaging [6] or immunohistochemistry [7], or by fetching the topological information from dedicate databases [8]—the ultimate spiking behavior of the NC is determined

TABLE I
PARAMETERS OF SPIKEY'S NEURONS.

Parameter name	Parameter code	Parameter interval	Notes
Leak conductance	`g_leak`	$[20.0, 60.0]$	Individually configurable for each neuron
Absolute refractory period	`tau_refrac`	$[0.0, \infty]$	Individually configurable for each neuron
Resting potential	`v_rest`	$[-80.0, -55.0]$	Shared among neurons in the network
Reset potential	`v_reset`	$[-80.0, -55.0]$	Shared among neurons in the network
Threshold potential	`v_thresh`	$[-80.0, -55.0]$	Shared among neurons in the network
Inhibitory reversal potential	`e_rev_I`	$[-80.0, -55.0]$	Shared among neurons in the network

by its parameterization, that is, the vector of values controlling conductances, refractory periods, weights, and so forth. In this work, we investigate the possibility of automatically calibrating such parameters with respect to an expected behavior, e.g., the electrophysiological data recorded *in vivo*.

Due to the inherent complexity of neural networks, such parameterization cannot be manually fine-tuned. In order to create a completely automatic methodology, Spikey needs to be coupled to an optimization algorithm able to realize the calibration of the NC parameters, such that the difference between the simulated and the target spiking activities is reduced to zero. This non-convex and non-linear minimization problem can be tackled by means of population-based bio-inspired global optimization meta-heuristics. Among the existing algorithms, we selected Fuzzy Self-Tuning Particle Swarm Optimization (FST-PSO) [9]. FST-PSO extends the classic PSO algorithm [10], which is based on a population of individuals (the particles) moving inside a bounded high-dimensional search space. Particles move as the result of a social attraction (towards the best particle in the swarm), a cognitive attraction (each particle tends to stay close to the best position it found so far), and a small amount of inertial weight. FST-PSO leverages fuzzy logic reasoning to autonomously and dynamically adjust PSO's hyper-parameters (e.g., the strength of the social and cognitive attractions), and adapt the swarm behavior to the problem under investigation.

Our results on the automatic calibration of Spikey by FST-PSO show that our method is effective on relatively small test cases, allowing a perfect fit with respect to the target spiking data. However, the automatic calibration becomes unfeasible for complex networks, due to the intrinsic stochasticity of the simulated spiking activity of Spikey. This randomness in neurons' potentials—mainly due to hardware production variability and electronic noise (including thermal noise)—implies that two fitness evaluations using the same parameterization might yield different values, misleading the trajectory of particles. A similar issue was previously highlighted in [11]. We argue that the inconsistent results we obtained are due to a fitness function that was not designed to handle the unpredictable stochastic fluctuations that lead to Spikey's variable responses.

The paper is structured as follows. We introduce Spikey and the FST-PSO, together with the fitness function used for the optimization, in Section II. In Section III we report some results of automatic calibration on three networks of increasing complexity. We conclude the paper in Section IV with some discussions about the limitations of Spikey and our approach, and provide future directions of this work.

II. MATERIALS AND METHODS

In this section we present an overview of Spikey's characteristics, briefly describing how the device can be programmed, configured and controlled. Then, we introduce the FST-PSO optimization algorithm, used to perform the calibration of Spikey, and the novel fitness function introduced in this work.

A. The neuromorphic chip Spikey

Spikey [4] is a re-configurable NC developed by the Electronic Vision(s) group at Universität Heidelberg, designed to model and simulate an arbitrary topology of 384 interconnected neurons. At the core of Spikey there is a 180 nm CMOS with die size equal to 5 mm $\times$ 5 mm, which contains the analog circuitry actually performing the simulation. Although the communication with the host computer is established by digital circuits, the simulation is completely analog without any additional level of abstraction or discretization of the signal.

Spikey is based on the LIF model with conductance-based synapses [12] and can follow the temporal evolution of all neurons, keeping track of their spiking activity. Specifically, as outcome of a simulation Spikey returns to the host a vector containing the spiking time instants of each neuron involved in the neural network. The detailed temporal trace of the membrane potential can also be recorded, but only for a single selected neuron. This limitation is due to the small amount of RAM available on the system (512 MB).

The 384 neurons are organized into two arrays of 192 neurons each; each neuron can be connected up to 256 synapses, whose weights are defined by the user using 4 bits registers [13]. A synapse with a weight equal to 0 is assumed to be turned off by the system. The main advantage provided by Spikey is its high speed when simulating the neural network; as a matter of fact, thanks to its analog circuitry, Spikey is 10^4 times faster than the biological neural networks [4].

Spikey uses an extension of simple LIF neurons, by introducing the concept of refractory period: as soon as the neuron fires, it cannot produce other spikes during the refractory interval. This LIF neuron model consists in 6 parameters (listed in Table I) that can be set in the Spikey network [4]. To be more precise, each neuron of the network has 2 parameters (leak conductance and refractory period), while the other 4 parameters (resting, reset, threshold and inhibitory reversal potentials) pertain to the entire network. In addition, as mentioned earlier, each synapse stores a configurable 4-bit weight. Spikey supports in hardware a long-term learning mechanism known as Spike-Timing-Dependent Plasticity (STDP) [14], a biologically inspired process in which the connection strengths (in this case, the weights) are adjusted according to the relative timing between a neuron's input and the output spikings. Specifically, all inputs that cause a post-synaptic neuron reaction get strengthened, whereas inputs that do not cause post-synaptic activity are weakened and will contribute, with a lower strength, in future simulations.

The neural connection topology in the Spikey chip can be arbitrarily reconfigured by the user. Technically, this process is performed by the Spikey chip by means of a FPGA. In order to simplify the control of the FPGA, and the re-configuration of the neural networks, Spikey can be programmed using PyNN [15], a simulator-independent Python library for building neural network models. PyNN is an agnostic language implemented to program the neuromorphic systems and software simulators with a high level of abstraction. PyNN provides both an object-oriented interface and a procedural interface. In what follows, we briefly describe only the object-oriented API, which was used in all tests presented in this work.

The most important PyNN classes are `Population` and `Projection`, which are fundamental to define the network topology:

- `Population` represents a (uniform) group of cells of the same type;
- `Projection` is a set of *connections* between pre- and post-synaptic populations.

In principle, the PyNN library provides several models of neurons, but only the LIF model can be used when working with Spikey, relying on synapses with exponentially decaying or alpha-shaped conductances (`IF_facets_hardware1`). Two different stimuli sources can be exploited to provide an input to the network, and thus activating the simulation: `SpikeSourcePoisson` and `SpikeSourceArray`. These sources generate a sequence of spikes according to a Poisson process or at user-defined times, respectively. From a technical standpoint, Spikey considers both sources as actual neurons, so that a population of sources can be created and propagated through the neuron populations downstream.

Differently from classic *in silico* simulations of neural networks—typically based on numerical integrators developed to solve the corresponding systems of coupled differential equations—when specific hardware is employed for the simulation of such networks, as in the case of analog circuits, noise can affect the outcome. There are two main types of noise that can affect Spikey: fixed-pattern noise, and temporal noise. The former is due to physical imperfections on the chip that were generated during the fabrication process; this noise can be recognized and mitigated as it remains identical over time. Temporal noise, on the contrary, is a form of stochasticity introduced by many factors, including electronic and thermal disturbances, which cause Spikey to generate quantitatively different results at each simulation. We will show in Section III how temporal noise can represent a limiting factor for the effective calibration of NCs.

As described above, Spikey can record the spiking times of all neurons in parallel, producing collective traces of neurons' spiking activity. A spike is detected and annotated when the neuron's membrane potential crosses the user-defined firing threshold. The spiking traces are the only information that can be leveraged to perform the calibration on this type of neuromorphic systems. In order to automatically identify the optimal parameters that precisely reproduce any desired spiking behavior, we exploited here FST-PSO, as described in the next section.

B. Fuzzy Self-Tuning Particle Swarm Optimization

The calibration problem of NCs can be restated as an optimization problem. Specifically, the goal is to minimize a fitness function defined as the distance between a target neural spiking activity—e.g., collected by means of microelectrodes during *in vivo* intracellular recordings [16]—and the activity of the network simulated by using a putative parameterization.

More precisely, given a candidate solution $\mathbf{x}$, the fitness function is defined as follows. We denote by T the total time of the simulation, J the number of neurons in the network, and $\boldsymbol{\omega}_\sigma^j = (t_1, \ldots, t_M)$, $\boldsymbol{\omega}_\tau^j = (t_1, \ldots, t_N)$ the vectors of spiking times of the j-th neuron sampled during the simulation and in laboratory experiments, respectively (please note that M can be different from N). Then, the fitness is defined as:

$$f(\mathbf{x}) = \sum_{j=0}^{J-1} \left| \phi(\boldsymbol{\omega}_\sigma^j) - \phi(\boldsymbol{\omega}_\tau^j) \right|, \tag{1}$$

where ϕ is the area of a curve that increases by 1 whenever the j-th neuron spikes:

$$\phi(\boldsymbol{\omega}^j) = \sum_{k=1}^{|\boldsymbol{\omega}^j|-1} k \cdot (t_{k+1} - t_k). \tag{2}$$

The fitness function defined in Equation 1 represents a cumulative difference between the area of the two curves $\phi(\boldsymbol{\omega}_\sigma^j)$ and $\phi(\boldsymbol{\omega}_\tau^j)$, calculated on both the target and the simulation, for all neurons in the network. Please note that if $\boldsymbol{\omega}_\tau^j = \boldsymbol{\omega}_\sigma^j$, then, for all $j = 0, \ldots, J-1$, the two areas are identical and the fitness value of $\mathbf{x}$ is equal to zero. Every difference in the spiking time, or in the number of spikes, increments the final fitness value.

In this work we aimed at realizing the calibration of NCs, with respect to a desired behavior, without the need for any user intervention. Specifically, the algorithm used to minimize Equation 1 is FST-PSO [9], a self-tuning variant of the global optimization meta-heuristic PSO [17]. FST-PSO is a settings-free method in which particles automatically determine the optimal functioning settings (i.e., inertia weight, cognitive and social factors, minimum and maximum velocities) by leveraging a fuzzy rule-based system that dynamically adjusts such hyper-parameters for each particle during the optimization process. In addition, FST-PSO automatically calculates the size P of the swarm according to the number of dimensions D of the search space, using the formula $P = \lfloor 10 + 2\sqrt{D} \rfloor$. The search space, in which the swarm can move, is bounded to avoid the exploration of unfeasible parameterizations. In this work, the limits are determined by the physical limitations of Spikey (see Table I), but they can be adapted to the characteristics of arbitrary NCs. When a particle goes outside the boundaries of the search space along one of its components (i.e., a putative parameterization not supported by the NC chip), FST-PSO adopts a damping strategy: the particle is relocated inside the search space with a stochastic displacement, and the velocity component is changed in the opposite direction for that component. During each iteration, the position of the best particle $\mathbf{g}$ of the swarm is updated. At the end of the process—in this work, after 100 iterations—the last value of $\mathbf{g}$ is returned as the optimal solution of the problem under investigation.

The optimization method was implemented using Python 2.7 and the following libraries: PyNN 0.6, FST-PSO 1.6, numpy 1.14.3, pexpect 4.5, scipy 1.1, and miniful 0.4. In all the experiments shown in Section III, the optimization was repeated 30 times to collect statistical information and analyze the average behavior. Since we aim at determining all missing parameters of a model, including the synaptic weights, the STDP functionality of Spikey was disabled.

III. Results

We performed the calibration of the parameters of neural networks characterized by an increasing complexity. Besides the connection topology, in the figures of the networks we show the parameters to be estimated and their nominal values. For each network, we ran a simulation of the spiking activity that was used as *in silico* target for the calibration. The input stimuli, fed to each network, was an irregularly spaced sequence of spiking impulses.

A. Network A - Two sequential neurons

The first analyzed network is a simple system with two sequential neurons and $D = 6$ parameters (Network A, Figure 1). Figure 4 shows the result of the first test on Network A, by comparing the best solution found (bottom panel) and the target spiking activity (top panel). Each dot in the plots represent a spike produced by one of the two neurons in the network (denoted by green and blue colors, respectively). In the case of Network A we correctly estimated the 6 parameters, being able to converge to an optimal solution with respect to the target spiking activity. Thus, from a strictly phenomenological point of view, the Spikey chip is correctly reproducing the expected behavior of the target neural network.

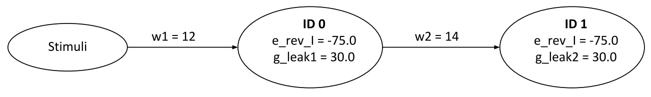

Fig. 1. Network A: a simple system with 2 sequential neurons and $D = 6$ parameters.

Figure 5 (left) shows the convergence plot of FST-PSO: the x-axis represents the number of iterations of the algorithm, while the y-axis represents the average fitness of the best individuals calculated over the 30 repetitions of the optimization process. The plot highlights the rapid convergence to optimal solutions with respect to the fitness function, and with a very limited standard deviation in the results.

The six box-plots reported in Figure 5 (right) show the distributions of the optimal parameters identified over 30 runs, and provide a different perspective on the previous result. The red solid lines denote the nominal values used to generate the target data, while the black dashed line represent the best fitting solution (that was used to generate Figure 4). These boxplots highlight that different executions of FST-PSO converged to different parameterizations that are all equivalent from the point of view of the spiking activity, and are hence characterized by a similar fitness value. As a matter of fact, the only information stored by Spikey is the time when a neuron spikes: the NC discards all the data concerning the exact dynamics of membrane potentials. Two different solutions can yield spikes characterized by very different nature (e.g., having different amplitudes, decays, slopes, width of half-spike amplitude) but happening in the same moment. This circumstance leads to different optimal solutions characterized by the same fitness value. The only information available are these vectors of spiking times, which are used in Equation 1 to assess the fitness, meaning that our approach cannot discriminate between phenomenologically equivalent behaviors.

B. Network B - Ring network

We then performed the calibration of a feedback ring network with five neurons and $D = 12$ parameters (Network B, Figure 2). Figure 6 shows that also in this case FST-PSO identified a parameterization allowing to reproduce the expected behavior (as confirmed by the convergence plot in Figure 7, left), by fitting all targets with the exception of some exceeding spiking activity mainly at the end of the simulation. By inspecting the distribution of the 12 parameters (Figure 7, right), we observe that some values are very different with respect to the nominal values, while some others show a high sensitivity. One example is the parameter corresponding to the synaptic weight of neuron 1. This neuron is upstream of the whole network: it receives the initial stimulus and ultimately governs the dynamics of the whole network. Thus, this parameter has the highest impact on the fitness value, and must be calibrated with a high precision to enable a correct spiking activity.

C. Network C - Starfish network

Finally, we calibrated a topology proposed in [18], corresponding to a simplified model of the starfish nervous system, consisting in five neurons and $D = 17$ parameters (Network C). The central nervous system of the starfish is organized as a nerve ring around its mouth, with radial nerves running along the arms. Although lacking a centralized brain, this system is able to control and coordinate the movements of each arm, and

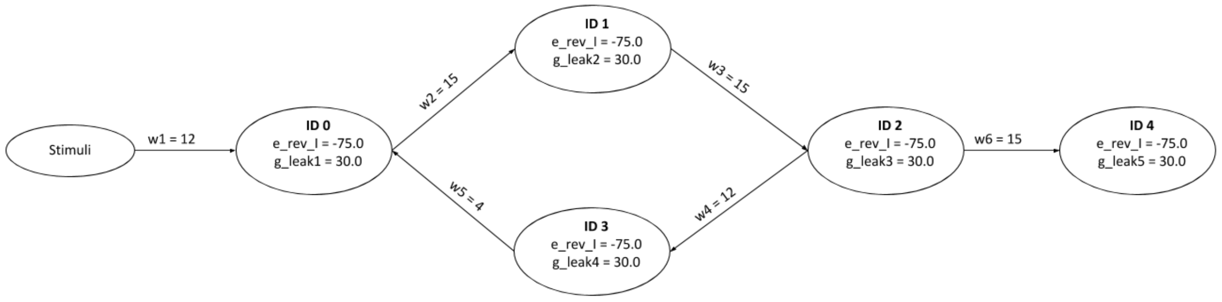

Fig. 2. Network B: a feedback network with 5 neurons and $D = 12$ parameters.

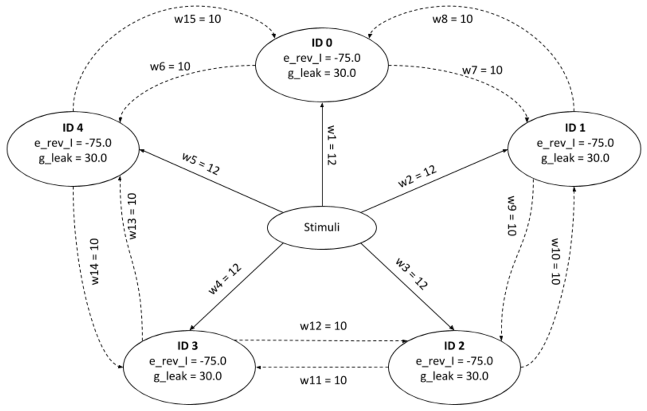

Fig. 3. Network C: the starfish network with 5 neurons and $D = 17$ parameters. Solid and dashed arcs represent excitatory and inhibitory connections, respectively.

it is considered as one of the earliest and most simple example of nervous system in animals. The purpose of this test was to investigate the performance of FST-PSO in inferring the weights of inhibitory synapses. Figure 3 shows the topology of this neural network, where the excitatory and inhibitory connections between the neurons are denoted by solid and dashed lines, respectively.

In this tests, FST-PSO was not able to identify any solution that fits with the target data, as shown in Figure 8. As a matter of fact, the target and the simulated spiking sequences differ, since the best solution found is characterized by a slightly different activity (e.g., neuron 4 spikes two times in the simulation during the interval 0–50 ms, while only a single spike was present in the target).

We investigated this circumstance in order to determine what was preventing the convergence to an optimal solution. By checking the convergence plots of all runs (data not shown), we observed that the best solutions found throughout the 30 runs were actually excellent from the point of view of the fitness value. In particular, we noticed that these solutions were characterized by fitness values much lower than those obtained for Networks A and B, and that many of them were not improved by FST-PSO throughout the optimization.

To better understand the results obtained for the calibration of Network C, we performed a simulation using each one of the best solutions found during the 30 repetitions of the experiment. We then re-calculated the fitness values of these solutions using the newly simulated traces, obtaining very different values with respect to the fitness values returned at the end of the optimization process with FST-PSO. Specifically, the new fitness values were much higher, thus denoting a very poor fitting with respect to the target traces. To confirm this finding, we repeated 1000 times the same simulation using one of the best solutions found, to analyze the distribution of the spiking events. In Figure 9, the traces of the spiking events show that the spikes are scattered in time, and the presence of many lighter areas confirm that different simulations can yield radically different outcomes, corresponding to different fitness values (points are alpha blended, so that darker areas correspond to a higher probability of a spiking event at that specific time point). This problem is caused by the fact that the same parameterization can lead to simulations with different spiking activities, due to the presence of temporal noise in analog circuits, as also discussed in [11]. One striking example is the trace for neuron 3: every simulation basically produces a different outcome, affecting the fitness values at each repetition.

The stochasticity of Spikey's simulation introduces sudden, unpredictable and misleading modifications of the spiking activity which, in turn, affects the fitness value calculated for the same solution. As a consequence, the best particle found by FST-PSO—characterized by an excellent fitness value, yet not necessarily reflecting the real quality of the corresponding parameterizations—was actually driving the optimization process of the swarm in a region of the fitness landscape characterized by low quality, ultimately yielding a poor final parameterizations of the NC.

IV. Conclusion

In this paper we investigated the automatic calibration of spiking neuromorphic chips, by means of a settings-free variant of PSO—called FST-PSO—exploiting a fitness function specifically defined for this problem. The automatic calibration method presented in this work has been tested on the Spikey NC, taking into account three neural networks

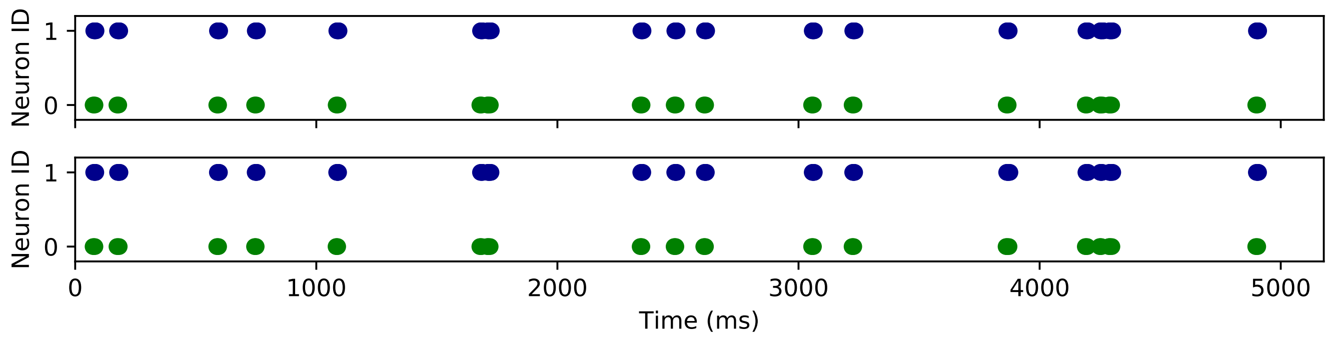

Fig. 4. Comparison of the target trace (top panel) and the trace simulated using the best fitting individual (bottom panel), in the case of Network A.

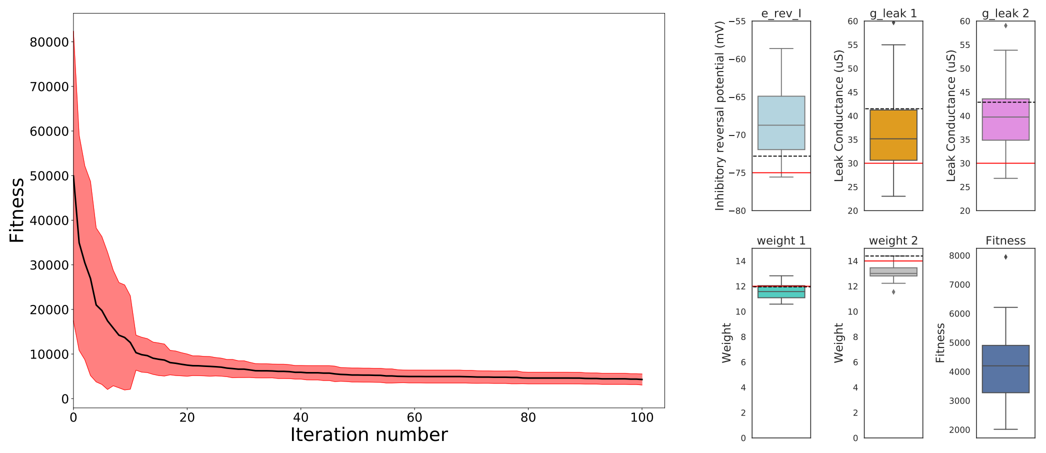

Fig. 5. Calibration of Network A. Left: convergence plot of FST-PSO. The mean and the standard deviation of the 30 runs are denoted by the black solid line and the red filled area, respectively. Right: distribution of the optimal parameters. The red solid lines denote the nominal values used to generate the target. The black dashed line denote the best parameterization found.

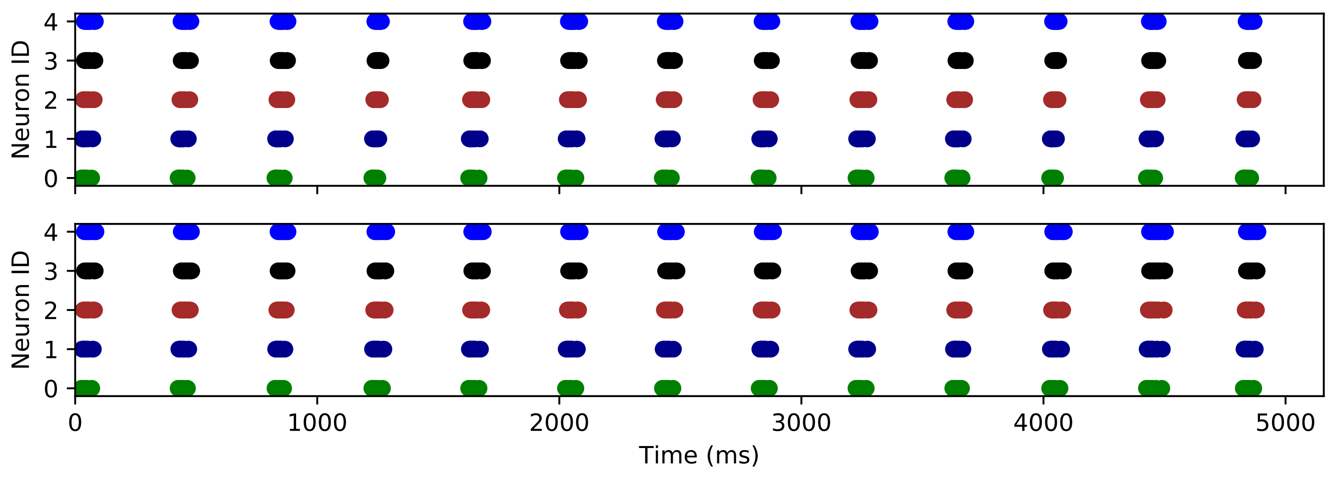

Fig. 6. Comparison of the target trace (top panel) and the trace simulated using the best fitting individual, in the case of Network B.

of increasing complexity, having up to 17 parameters to be estimated. Our results show that in the case of simple neural networks, FST-PSO can identify a parameterization capable of reproducing the expected behavior of the network. However, when the complexity of the neural network increases, and both excitatory and inhibitory connections are considered, our approach failed in identifying an appropriate parameterization.

We speculate that this is due to some specific features of

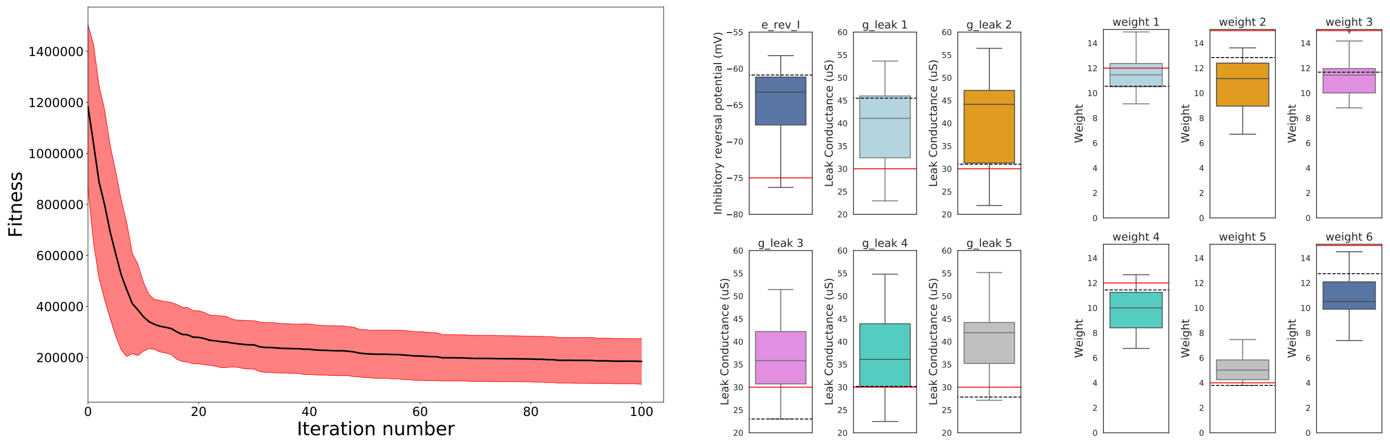

Fig. 7. Calibration of a Network B. Left: convergence plot of FST-PSO. The mean and the standard deviation of the 30 runs are denoted by the black solid line and the red filled area, respectively. Right: distribution of the optimal parameters. The red solid lines denote the nominal values used to generate the target. The black dashed line denote the best parameterization found.

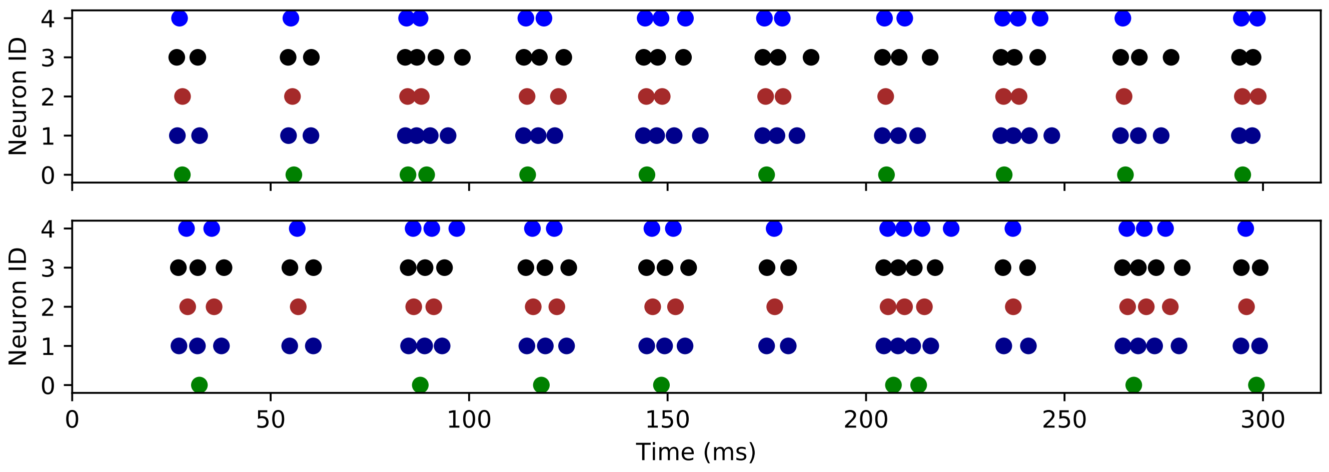

Fig. 8. Comparison of the target trace (top panel) and the trace simulated using the best fitting individual (bottom panel), in the case of Network C.

the NC. In particular, Spikey's stochasticity implies that two independent simulations of the same model, performed using the same parameterization, yield different spiking traces. As a consequence, if a candidate parameterization of the network under investigation generates, just by accident, even one simulation corresponding to a spiking activity that perfectly fits with the target behaviour, then, thanks to its very good fitness value, that parameterization will be considered as an optimal solution, possibly driving the exploration carried out by the swarm towards unsuitable areas of the search space.

To overcome this issue, we will test the feasibility of the automatic calibration of Spikey by using a different fitness function, able to deal with the stochasticity of the system. However, it should be kept in mind that Spikey's simulation produces a peculiar trace composed of a list of spiking events that are temporally misaligned, and whose number can differ across multiple simulations. Due to the nature of this data, it would be unrealistic, for instance, to perform a large number of simulations and calculate an "average" dynamics able to mitigate the stochastic fluctuations and outliers. As an alternative, we could partition the traces in small time intervals to bin the spikes, in order to collect statistical information about the neuron's activities. By using this approach, we could determine whether the putative parameterization actually yields a spiking event in that time interval with a high probability, and use this information to implement an expectation maximization algorithm. So doing, FST-PSO could be used to maximize the likelihood of experimental data, given the putative parameterization. We expect this approach to be robust with respect to Spikey's stochastic behavior.

The proposed improvements in the optimization process would allow the investigation of larger networks representing biologically meaningful topologies, as the sensory networks of model insects [4], [19].

References

[1] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger,

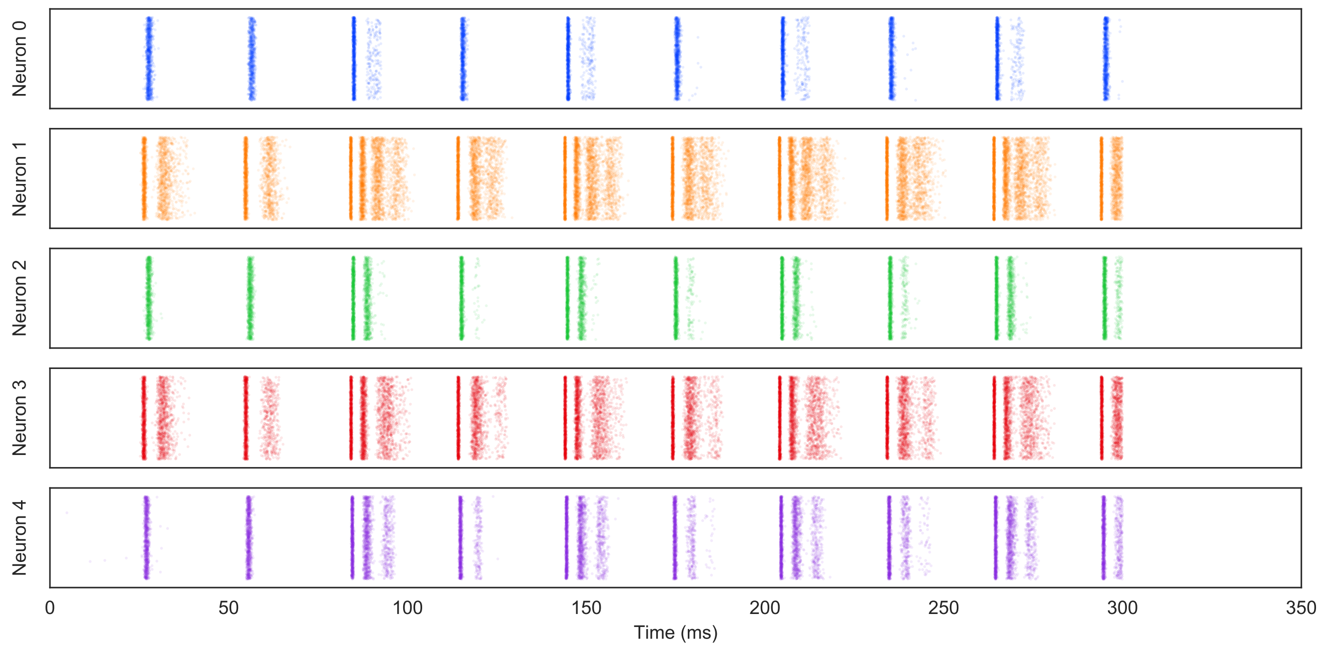

Fig. 9. 1000 simulated traces of the spiking activity of the five neurons in the starfish network. All simulations were performed using the same parameterization. A darker color corresponds to a higher probability of a spiking event. The scattering and misalignment of points highlight the stochasticity of Spikey's simulation.

S. Renaud *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, p. 73, 2011.

[2] E. Covi, S. Brivio, A. Serb, T. Prodromakis, M. Fanciulli, and S. Spiga, "Analog memristive synapse in spiking networks implementing unsupervised learning," *Frontiers in Neuroscience*, vol. 10, p. 482, 2016.

[3] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[4] T. Pfeil, A. Grübl, S. Jeltsch, E. Müller, P. Müller, M. A. Petrovici, M. Schmuker, D. Brüderle, J. Schemmel, K. Meier, "Six networks on a universal neuromorphic computing substrate," *Frontiers in Neuroscience*, vol. 7, no. 11, pp. 1–17, 2013.

[5] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[6] T. Mishima and R. Kanzaki, "Physiological and morphological characterization of olfactory descending interneurons of the male silkworm moth, Bombyx mori," *Journal of Comparative Physiology A*, vol. 184, no. 2, pp. 143–160, 1999.

[7] I. Nishikawa, M. Nakaumura, Y. Igarashi, T. Kazawa, H. Ikeno, and R. Kanzaki, "Neural network model of the lateral accessory lobe and ventral protocerebrum of Bombyx mori to generate the flip-flop activity," *BMC Neuroscience*, vol. 9, no. 1, p. P23, 2008.

[8] T. Kazawa, H. Ikeno, and R. Kanzaki, "Development and application of a neuroinformatics environment for neuroscience and neuroethology," *Neural Networks*, vol. 21, no. 8, pp. 1047–1055, 2008.

[9] M. S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, G. Mauri, G. Pasi, "Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization," *Swarm and Evolutionary Computation*, vol. 39, pp. 70–85, 2018.

[10] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[11] T. Wunderlich, A. F. Kungl, E. Müller, A. Hartel, Y. Stradmann, S. A. Aamir, A. Grübl, A. Heimbrecht, K. Schreiber, D. Stöckel *et al.*, "Demonstrating advantages of neuromorphic computation: a pilot study," *Frontiers in Neuroscience*, vol. 13, p. 260, 2019.

[12] J. Schemmel, D. Brüderle, K. Meier, B. Ostendorf, "Modeling Synaptic Plasticity within Networks of Highly Accelerated I&F Neurons," in *Proceedings of the 2007 International Symposium on Circuits and Systems (ISCAS)*. New Orleans, LA, USA: IEEE Press., 2007, pp. 3367–3370.

[13] T. Pfeil, T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, K. Meier, "Is a 4-bit synaptic weight resolution enough?–constraints on enabling spike-timing dependent plasticity in neuromorphic hardware," *Frontiers in Neuroscience*, vol. 6, no. 90, pp. 1–19, 2012.

[14] J. Schemmel, A. Grubl, K. Meier, and E. Mueller, "Implementing synaptic plasticity in a VLSI spiking neural network model," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006, pp. 1–6.

[15] A. P. Davison, D. Brüderle, J.M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, P. Yger, "PyNN: a common interface for neuronal network simulators," *Frontiers in Neuroinformatics*, vol. 2, no. 11, pp. 1–10, 2009.

[16] S. Namiki and R. Kanzaki, "Reconstructing the population activity of olfactory output neurons that innervate identifiable processing units," *Frontiers in Neural Circuits*, vol. 2, p. 1, 2008.

[17] J. Kennedy, R. Eberhart, "Particle swarm optimization," in *Proceedings IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[18] R. Suzuki, I. Katsuno, and K. Matano, "Dynamics of "neuron ring"," *Kybernetik*, vol. 8, no. 1, pp. 39–45, 1971.

[19] S. Namiki, S. S. Haupt, T. Kazawa, A. Takashima, H. Ikeno, and R. Kanzaki, "Reconstruction of virtual neural circuits in an insect brain," *Frontiers in Neuroscience*, vol. 3, p. 28, 2009.