Neural Processing of Counting in Evolved Spiking and McCulloch-Pitts Agents

Keren Saggie-Wexler
Alon Keinan
School of Computer Science
Tel-Aviv University
Ramat-Aviv, Tel-Aviv, 69978
Israel
kerensa@post.tau.ac.il
keinanak@post.tau.ac.il

Eytan Ruppin
School of Computer Science
Tel-Aviv University
Ramat-Aviv, Tel-Aviv, 69978
Israel
and
School of Medicine
Tel-Aviv University
Tel-Aviv, Israel
ruppin@post.tau.ac.il

Abstract This article investigates the evolution of autonomous agents that perform a memory-dependent counting task. Two types of neurocontrollers are evolved: networks of McCulloch-Pitts neurons, and spiking integrate-and-fire networks. The results demonstrate the superiority of the spiky model in evolutionary success and network simplicity. The combination of spiking dynamics with incremental evolution leads to the successful evolution of agents counting over very long periods. Analysis of the evolved networks unravels the counting mechanism and demonstrates how the spiking dynamics are utilized. Using new measures of spikiness we find that even in agents with spiking dynamics, these are usually truly utilized only when they are really needed, that is, in the evolved subnetwork responsible for counting.

Keywords
Evolutionary computation, counting, spiking dynamics, neurocontroller analysis

1 Introduction

The evolution of artificial neural networks of embedded autonomous agents constitutes a powerful tool for creating agents with complex behaviors [9–11, 17, 21], and for studying properties of the emergent network dynamics [20]. This study extends an evolutionary study of foraging [2], involving an evolved autonomous agent (EAA) living in a 2D grid arena containing food and poison items. In that study, the agent, possessing limited sensory inputs, had to consume as many food items as possible, while avoiding poison items. The agent's eating action consisted of standing still on food for one time step with its mouth open. Aharonov et al. [2] concentrated on the short-term memory mechanism underlying the foraging and navigation behaviors that evolved in this nonstationary environment. Our study aims at finding out whether we can replace the rather simple stimulus-response eating action with a memory-dependent delayed action, in which the agent has to remain still on a food item for an exact constant number of steps before it can consume it. This kind of delayed response is not trivial; the agent must develop a counting mechanism that will allow it to "remember" how many steps it has already waited. The goals of this study are to explore the limits of evolving such a counting mechanism, and study the evolved network dynamics solving this type of task.

There is a large pool of evidence concerning the ability of animals to time their responses and count. In the traditional Sidman avoidance operant conditioning procedure [6] a dog positioned in a two-compartment shuttlebox learns to move back and forth from one compartment to the other according to a predefined time schedule, to avoid receiving an aversive electrical current. There is no sensory signal that marks the onset of the shock; hence the dog has to learn the fixed time interval between shocks. Platt and Johnson [19] have shown that rats can be trained to press a lever a specific number of times and then activate a feeder to receive food. Fetterman [7] has demonstrated that pigeons learn to choose between two different responses based on the number of times they peck on a lighted key before it is darkened, simultaneously counting the number of pecks and timing the pecking duration. There is some similarity between the counting task and traditional delayed-response match-to-sample behavioral tasks [6]. In a match-to-sample task, the animal receives a cue indicating the appropriate action, which should be performed only after the presentation of a second sensory *trigger* cue. In both types of tasks there is a significant delay between a stimulus and the corresponding appropriate response that makes them impossible to solve by a simple sensory-motor mapping. However, having the trigger cue, the challenge of a match-to-sample task is to remember, throughout the delay period, which action should be performed, rather than *when* it should be preformed, as in the counting task.

We evolve agents with two types of neurocontrollers: networks of *McCulloch-Pitts neurons*, and spiky networks of discrete-time *integrate-and-fire* neurons. Models of spiking neurons have been extensively studied in the neuroscience literature. Spiky networks have a greater computational power than networks of sigmoidal and McCulloch-Pitts neurons [15], and are able to model the ability of biological neurons to convey information by the exact timing of an individual pulse, and not only by the frequency of the pulses [5, 16]. It is appealing to use spiky neural networks in EAA studies, since they are biologically more plausible: Biological neurons perform integration over their presynaptic inputs such that a neuron accumulates its membrane potential over time, and fires if it exceeds a threshold. After firing, the neuron's membrane potential returns to a resting value, remaining there for a given refractory period. Recent studies that combine evolutionary computation with spiky neural networks have analyzed properties of the spiking dynamics in the evolved networks, for example, whether the spiking dynamics result in a time-dependent or a rate-dependent computation, and the effect of noise on the emerging network [8, 18].

The goal of this study is to single out the effects of *integration* and *memory* in the integrate-and-fire model. Thus we chose to employ a minimal integrate-and-fire model, which is similar in every other respect to McCulloch-Pitts. However, while the McCulloch-Pitts model only permits a limited type of dynamic behavior, by its self-connections and recurrent synapses, the integrate-and-fire model allows for subthreshold dynamics. This type of dynamics may be useful in performing tasks that require memory, such as the counting task. We compare the evolution of McCulloch-Pitts and integrate-and-fire neurocontrollers performing the counting task, focusing on the evolutionary success, the overall complexity of the evolved neurocontrollers, and their counting dynamics. To perform this analysis we use a newly developed method, *multi-perturbation Shapley value analysis* [14], with which we quantify various properties of the evolved networks.

The analysis of spiking neurocontrollers in the framework of EAAs brings up substantial issues regarding spiking dynamics: whether an evolved network with spiking neurons is truly "spiky," and how we can define and measure the spikiness level of each neuron. This study addresses these questions by presenting two new fundamental ways by which we define and quantify different aspects of spikiness: the spikiness functional contribution, and the level of integration of inputs over time of a spiky neuron.

The rest of this article is organized as follows: Section 2 describes network architecture and the evolutionary process. Section 3 describes the multi-perturbation Shapley value analysis framework, which we use to analyze the evolved agents. Section 4 presents the results of the evolutionary experiments, and analyzes the evolved neurocontrollers and their dynamics. In Section 5 we present and quantify two basic properties of spiking neurocontrollers, with which we analyze the evolved spiky agents. These results and their implications are discussed in Section 6.

2 The EAA Environment

2.1 The Task

The EAA environment is similar to that of [2]. The agents live in a discrete 2D grid world surrounded by walls (Figure 1). Poison items are scattered all around the world, while food items are scattered only in a *food zone* in one corner. The agent's goal is to find and eat as many food items as possible during its life, while avoiding the poison items. The fitness of the agent is proportional to the number of food items minus the number of poison items it consumes. The agent is equipped with a set of sensors, motors, and a fully recurrent neurocontroller.

Four agent sensors (input neurons) encode the presence of a wall, a resource (food or poison, without distinction between the two), or a vacancy in the cell the agent occupies and in the three cells directly in front of it. A fifth sensor is a *smell* sensor, which can differentiate between food and poison underneath the agent, but gives a random reading if the agent is in an empty cell (Figure 1). The four output neurons dictate a forward movement (neuron 1), a left turn (neuron 2), or a right one (neuron 3), and control the state of the mouth (open or closed, neuron 4). Network updating is synchronous: in every time step a sensory reading occurs, network activity is updated, and a motor action is taken according to the resulting activity in the designated output neurons.

In [2], the agent consumed a resource by standing still on a grid cell containing it for one time step. Here, the agent has to remain still on the resource for a *waiting period* of precisely K steps, without moving or turning, in order to eat. Food is consumed by the agent closing its mouth on the last waiting step and then moving forward at the next step. That is, the mouth motor neuron should be open (a value of 0) in the $(K-1)$th waiting step, and closed (a value of 1) in the Kth step, the state of this motor having no effect for the first $K-2$ steps. Hence, in essence, the agent has to learn to count precisely to K. The agent has a limited life span of 150 sensorimotor steps. In order to facilitate the evolution of the agents performing the counting task, waiting steps (steps in which the agent does not move or turn) are not counted as part of the life span.

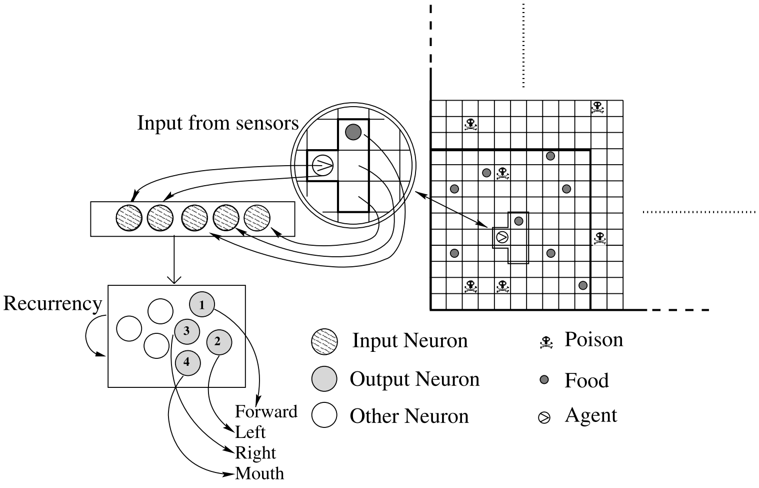


Figure 1. The EAA environment. An outline of the grid world and the agent's neurocontroller. The agent is marked by a small arrow on the grid, whose direction indicates its orientation. The curved lines indicate where in the arena each of the sensory inputs comes from. A value of 0, 1, or 3 in four of the input neurons encodes respectively the existence of a vacancy, a resource, or a wall in the corresponding grid-cell (an input value of 2 was used in [2] but is no longer needed). The fifth input neuron receives a value of 1 if the agent stands on food, a value of 0 if the agent stands on poison, and a random value of either 0 or 1 otherwise.

2.2 The Neurocontrollers

All neurocontrollers are fully recurrent with self-connections, containing ten neurons (of which four are the output, motor neurons), such that the five sensory input neurons are connected to all network neurons. We compare between neurocontrollers with McCulloch-Pitts (MP) neurons, employed conventionally in most EAA studies, and ones with spiky discrete-time integrate-and-fire neurons. In both types of networks, a neuron fires if its voltage exceeds a threshold (set to 0.05 in all evolutions). The spiking dynamics of an integrate-and-fire neuron *i* in our model are defined by

$$V_i(t) = \lambda_i[V_i(t-1) - V_{\text{rest}}] + V_{\text{rest}} + \frac{1}{N}\sum_{j=1}^{N} A_j(t) W_{i,j}, \tag{1}$$

where $V_i(t)$ is the voltage (membrane potential) of neuron *i* at time *t*, λ_i is the *memory factor* of neuron *i* (which corresponds to its membrane time constant), $A_j(t)$ is the activation (firing) of neuron *j* at time *t*, $W_{i,j}$ is the synaptic weight from neuron *j* to neuron *i*, *N* is the number of neurons, including the input sensory neurons, and V_{rest} is the resting voltage (set to zero in all simulations). After firing, the voltage of a spiky neuron is reset to the resting voltage, with zero refractory period.

This simple and minimal discrete-time integrate-and-fire model mimics the ability of biological spiking neurons to integrate sensory information over time. In each time step, the voltage of a spiky neuron results from the interplay between the history of its inputs and its current input field (the last summand in Equation 1). The memory factor, which ranges between 0 and 1, determines the amount of integration over time that the neuron performs: The higher the memory factor, the more important is the neuron's history $V_i(t–1)$, compared with the current input field. Different memory factors may be required for neuronal computations demanding a different amount of integration over time. A spiky neuron with a zero memory factor reduces to an MP neuron, in which only the current input field determines the voltage. That is, the voltage of an MP neuron is given by

$$V_i(t) = \frac{1}{N}\sum_{j=1}^{N} A_j(t) W_{i,j}, \tag{2}$$

and the neuron fires if its voltage exceeds the neuron's firing threshold. Hence both the MP and the integrate-and-fire network models used in this article communicate through spikes, so that the only difference between them is the additional memory ability of the integrate-and-fire model, allowing it to employ subthreshold dynamics.

2.3 The Evolution

A genetic algorithm is used to evolve the synaptic weights $W_{i,j}$ (in the range $[-1, 1]$) of both types of networks. For the spiky neurocontrollers, the memory factors λ_i are evolved as well in the range [0,1], allowing different neurons to perform a different amount of integration over time. The synaptic weights and the memory factors are directly encoded as real values in the genome. Evolution is conducted over a population of 100 agents for 30,000 generations, starting from random neurocontrollers. In each generation every agent is evaluated, after which the parents forming the next generation are chosen with probability proportional to their fitness. The next generation is created using a mutation rate of 0.02, a mutation range of $[-0.2, 0.2]$, and uniform point crossover with a rate of 0.35. In each evolutionary run, the number of waiting steps is constant. Thirty evolutionary runs were performed for each number of waiting steps K, ranging between 1 and 6.

3 Multi-Perturbation Shapley Value Analysis

In order to decipher the mechanisms underlying the agents' behavior and understand the inner workings of the evolved neurocontrollers, we utilize a newly developed method called multi-perturbation Shapley value analysis (MSA) [14]. This method is designed to meet one of the fundamental challenges of neuroscience, the identification of the individual roles of the network elements. The basic idea behind the MSA and other related methods is that mere correlation between the activations of the neural elements and the different behaviors is not sufficient to identify causality. To allow for the correct identification of the elements that are causally important for a given function, the deficit in performance should be measured after perturbing specific elements. Additionally, single-lesion studies, in which only one element is disabled at a time, are limited in their ability to reveal the significance of interacting elements. The MSA analyzes a data set composed of numerous multiple perturbations that are inflicted upon a neurocontroller, along with the performance score in a given set of functions. In each multi-perturbation experiment, several elements are perturbed by concurrently disrupting their operation, and the agent's performance in each function is measured. The MSA views a set of multi-perturbation experiments as a *coalitional game*, borrowing relevant concepts from the field of game theory. Specifically, the desired set of contributions is captured by the *Shapley value* [23], which corresponds to the *unique* fair division of the game's worth (the system's performance score when all elements are intact) among the different players (the system's elements). Hence, in this framework, a *contribution* of an element to a function measures its importance, that is, the part it causally plays in the successful performance of the function.

The basic MSA requires for its calculation the performance scores under all possible multi-perturbation experiments (the full knowledge of the behavior of the game for all possible coalitions). Thus, the computations needed to calculate the element's contributions grow exponentially with the number of elements in the analyzed system. For larger systems containing many elements, these computations become infeasible. Hence, the estimated MSA variant [13, 14] computes an approximation of the elements' contributions with high accuracy and efficiency from a relatively small set of multi-perturbation experiments.

In the results to follow in Section 4, the contributions of a network's neurons are computed using the basic MSA. The perturbation method used is *stochastic lesioning* [1], performed by randomizing the firing pattern of a perturbed neuron while keeping its firing probability equal to its intact overall mean firing rate. In Section 5 we define a different perturbation method, which allows us to segregate the contribution of a neuron's spiking dynamics. To compute the contributions of a network's synapses, we use the estimated MSA, allowing us to use only a small sample of all 2^{100} possible synaptic perturbation configurations (resulting from 100 internal synapses). Each perturbation configuration denotes a subgroup of synapses, when a synapse is perturbed by stochastically clamping its presynaptic neuron, while keeping the mean firing rate it exhibits when it is intact.

The MSA was utilized in the past in several studies to uncover the mechanisms underlying EAAs' neurocontroller operations. In [13] four neurocontrollers were analyzed using various MSA variants, with which their contributing elements were accurately identified. In [12] fault-tolerant neurocontrollers were examined with the MSA, uncovering the important neurons, the interactions between the neurons, and the fault-tolerance mechanism. In all these studies, MP dynamics were explored, and the stochastic lesioning perturbation method was utilized, finding the contributions of the network's elements, whether neurons or synapses. In this article a new perturbation method is employed, which singles out the contribution of the spiking dynamics of each neuron to the evolutionary task.

4 Results

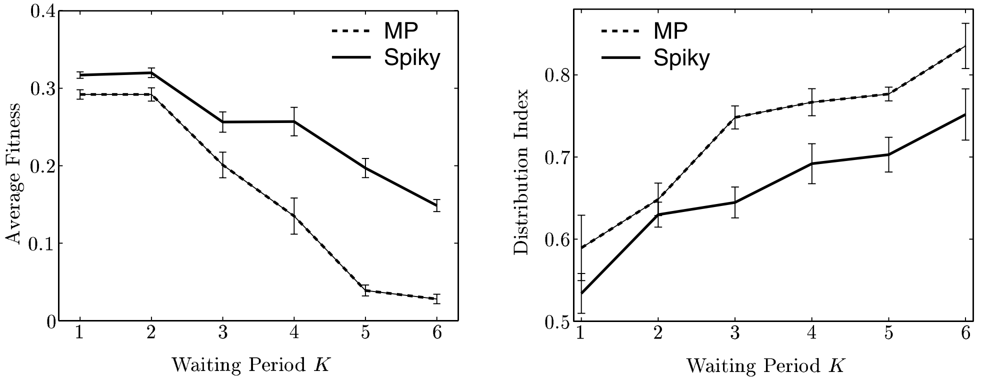
This section describes and analyzes the results of the evolutionary process. After presenting the basic evolutionary results for both types of agents (Section 4.1), the successfully evolved spiky and MP networks are compared in terms of the network's distribution of processing (Section 4.2). In Sections 4.3 and 4.4 the networks' counting mechanism is analyzed. Section 4.5 presents the results of incremental evolutions of the task, where the best agent evolved to perform a task with a waiting period K is taken as a seed for the evolution of agents performing the task with a waiting period of $K + 1$.

4.1 Performance Evaluation

The evolutionary task is fairly difficult, as many evolutionary runs converged without yielding successful agents. Figure 2a compares the difficulty of evolving agents with different waiting periods. For a specific waiting period K, we average the fitness score of the best agent from the last generation of each evolutionary run over many evolutionary runs. Evidently, the task is harder as the agent has to wait on food for a longer period, manifested by a decreasing average fitness score. Evolved spiky neurocontrollers are more successful than MP ones in performing this task.

4.2 Distribution of Processing

An important step in understanding an evolved network is to measure the amount in which the network processing is spread across the different neurons. For this, we first compute the functional contribution of each neuron to the counting task, using the MSA (Section 3). We measure the performance score of the agent under the entire set of neuronal perturbation configurations, using the stochastic lesioning method, and compute by the MSA the *causal importance* of each neuron to the behavioral task, namely, the contribution c_i of neuron i.



(a) (b)

Figure 2. (a) Average fitness versus waiting period K. The fitness score of the best evolved agent from the last generation of each evolutionary run, with the specified waiting period (x axis). Results are mean and standard error across 30 evolutionary runs for each waiting period. (b) The distribution index—mean and standard error—across all successful agents from the initial 30 evolutionary runs, with the specified waiting period (x axis). The number of successful agents is 11, 9, 8, 6, 5, 4 for $K = 1, \ldots, 6$, respectively, for the MP agents, and 30, 24, 16, 12, 6, 6 for the spiky agents. For $K = 5, 6$ we performed more than 30 evolutionary runs to produce the specified number of successful agents, since in the original 30 runs we received only one successful MP agent for $K = 5$, and no successful MP agents and only four successful spiky agents for $K = 6$.

Based on these neuronal contributions, we calculate the *distribution of processing* index D [1], which measures how distributed is the function in the network, according to

$$D = 1 - \frac{\sigma(\mathbf{c})}{\sqrt{(N-1)/N^2}}, \tag{3}$$

where $\mathbf{c}$ is the vector of all neuronal contributions, $\sigma(\mathbf{c})$ is the standard deviation of $\mathbf{c}$, and N is the number of neurons. Distribution values are in the range [0, 1], where a zero distribution score indicates localization of the task to one neuron alone. The higher the distribution index, the more evenly spread is the network processing across many neurons, with a distribution value of 1 corresponding to equal participation of the neurons in the network's processing.

The difficulty of evolving an agent that solves the counting task increases with the waiting period (Section 4.1). Is there a correlation between the difficulty of evolving a network that solves the task and the distribution level of the network? From all agents evolved in the 30 evolutionary runs performed for each waiting period, we denote the successful agents as those whose fitness score is higher than 0.3.[1] Figure 2b presents the distribution score, averaged over all the successful agents, for each waiting period. In both types of networks, the distribution index increases for longer waiting periods. Since the tasks differ only by the duration of the waiting period K, the difference in the distribution scores is due to the counting process. Comparing the dynamics of the two networks, spiky networks have a more localized (less distributed) processing than MP networks. Evidently, there is a high correlation between the distribution index, averaged over the successful agents for each waiting period (Figure 2b), and the average fitness score of each waiting period computed over all evolutionary runs, successful and unsuccessful (Figure 2a) (correlation coefficient of -0.8 for the spiky networks, and -0.9 for the MP ones). These results indicate that counting tasks with longer waiting periods require more complex network solutions in terms of distribution of processing, and that the network's complexity in turn is negatively correlated with evolutionary success in performing the task. This is true comparing agents employing a similar type of dynamics. Comparing between different types of dynamics, we find that spiky agents have simpler network solutions (lower distribution levels) and are more successful in performing the counting task than MP agents. We shall return to explain this finding in Section 4.4.

4.3 Identifying the Counting Subnetwork

Given a successfully evolved network, consisting of 10 neurons and 100 internal synapses, it is almost impossible to identify the synaptic functional backbone of the network that is in charge of the counting process. However, this can be quite easily done by using the estimated MSA (Section 3), this time examining the network's synapses rather than the neurons. First, we refer to the counting process as an independent network function, among various different subgoals that the agent solves in order to successfully forage for food. We define the *counting fitness score* as the number of food items that the agent consumed divided by the number of times that the agent arrived at a grid cell containing food. We perform multi-perturbation experiments on the neurocontrollers (each perturbation configuration corresponds to a subgroup of synapses) and measure the counting fitness score. The synaptic perturbations are performed only while the agent is standing on food, preserving all other aspects of its foraging abilities. The perturbations are stochastic lesioning, using as the intact mean firing rate of each neuron the firing rate it exhibits when the agent stands on food. As a result, an estimated contribution of each synapse to the counting function is obtained. A statistical t test (with $\alpha = 0.01$) is then carried out for each of the synapses to determine whether its contribution is significant.

---

[1] This fitness value has been observed to ensure satisfactory foraging and food consumption abilities. See [2] for a comparison of the fitness values of the evolved agents with several benchmarks.

Figure 3 presents the synapses that significantly participate in the counting of agent S7, a spiky agent that waits for $K = 7$ steps on food, and their corresponding neurons. This minimized backbone *counting subnetwork* identifies the elements in the network that are in charge of the counting process: Only neurons 1, 4, 7, and 3 participate in the counting process, so that all synapses between neurons 1, 4, and 7 are significant (their magnitudes and signs are given in Figure 3). The food inputs excite neuron 4, while inhibiting neuron 7. In the next section we explore the interactions between the counting elements, explaining the role of each of S7's synapses in the counting process.

4.4 Activation-Pattern Analysis of the Counting Process

To fully understand the counting process of successful neurocontrollers, we turn to study the activation patterns of the agent's neurons during counting. To demonstrate this we focus on two agents, one with an MP network and one with a spiky network, and compare their activation patterns. For a specific agent, the firing sequence during waiting periods usually repeats itself precisely in all counting incidents (this is not guaranteed even though the dynamics are deterministic, since the initial state at the entry point to counting may vary across the agent's lifetime). Figure 4a presents the activation pattern sequence during the waiting period of agent MP5, an MP agent with a waiting period of 5 time steps. The sensory inputs, which are constant during the waiting period, inhibit the first three motor neurons, responsible for moving and turning. On each time step, a different subgroup of the remaining seven neurons is active, so that at the last waiting step the agent closes its mouth and eats, and at the next step the forward motor (neuron 1) is reactivated and the agent moves. In general, since the sensory inputs stay exactly the same throughout the waiting period, for an MP agent to count to K, the network has to pass through K different activation states.

Figure 4b shows the firing pattern of the counting neurons of S7, the spiky agent whose backbone was presented in Section 4.3. As was also observed from the synaptic backbone, neurons 1, 4, 7 participate in the counting process. The food inputs inhibit neuron 7 but activate neuron 4, which has a excitatory synapse to neuron 7. Neuron 7, a spiky neuron with a memory factor of 0.43, gradually accumulates voltage during steps 3 to 5, till it passes the threshold and fires at the sixth time step. This firing inhibits the fourth neuron on the seventh step, closing the agent's mouth to consume the food. Closing the mouth removes the inhibition from neuron 4 to the forward motor, which causes the agent to move, due to a parallel excitation from neuron 7. The excitatory synapse from neuron 7 to neuron 3 (the turn-right motor) makes it more likely for the agent to turn right after eating, thus remaining in the food zone. The three excitatory synapses (from the food inputs to neuron 4, from neuron 4 to neuron 7, and from neuron 7 to the forward neuron) create the dynamic

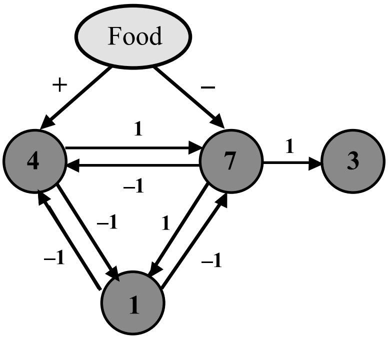


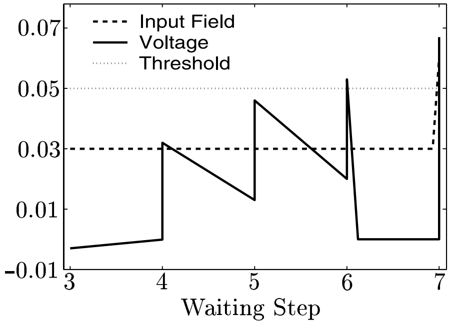
Figure 3. The synaptic backbone forming the subnetwork responsible for the counting of agent S7. Numbered circles represent neurons; arrow lines represent synapses, with the synaptic weight marked next to them. "Food" represents the joint influence of the two food sensory inputs (the first sensory input receives a value of 1 when the agent stands on a resource; the fifth sensory input identifies the exact resource type underneath the agent, and receives a value of 1 when the agent stands on food). These two sensory inputs both excite neuron 4 and inhibit neuron 7 when the agent stands on a food item. The presented synapses are the ones that came out significant from the synaptic MSA.

	1	2	3	4	5	6	7	8	9	10
	1	0	0	0	0	1	1	0	1	0
1	0	0	0	0	0	1	1	0	1	0
2	0	0	0	0	0	1	0	0	0	1
3	0	0	0	0	0	0	0	0	1	0
4	0	0	0	1	0	1	1	0	0	0
5	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**
6	1	0	0	0	0	0	0	0	1	0

(a)

	1	4	7
	1	0	0
1	0	0	0
2	0	1	0
3	0	1	0
4	0	1	0
5	0	1	0
6	0	1	1
7	**0**	**0**	**1**
8	1	0	0

(b)



(c)

Figure 4. Information processing during counting in an MP versus a spiky agent: (a) The activation pattern of the neurons of agent MP5 throughout the waiting period. An activation value of 1 states that the neuron fires. Each column represents a neuron; the four leftmost neurons (1–4) are the forward, left, right, and open-mouth motor neurons, respectively. Each row represents the firing at a different time step, in a consecutive order. Eating occurs at the fifth time step (marked in bold) when the agent closes its mouth, which is open in the previous step. After eating, the forward motor is activated and the agent moves. (b) The activation pattern of neurons 1, 4, and 7 of agent S7 during the waiting period (no other neurons fire throughout this period). Columns and rows as in (a); eating occurs at the seventh time step. (c) The input field and the voltage of neuron 7 of agent S7 in time steps 3 to 7 of the waiting period, along with the threshold above which the neuron fires (0.05). The memory factor of neuron 7 is 0.43. After firing at the sixth time step the voltage is reset to V_{rest}, and the neuron fires again at the seventh step, activated by neuron 4 and self-excitation, this time without any integration period.

pathway of the counting flow. The remaining inhibitory synapses ensure that this flow will take place with the correct timing. For instance, the inhibitory synapse from neuron 1 to neuron 4 ensures that the fourth neuron will start firing only a step after the agent stands still on a food item, thus initiating the described sequence of activations responsible for counting, such that the agent precisely counts to 7. As shown in Figure 4c, during steps 3 to 6 of the waiting period the input field of neuron 7 is constant, below the threshold; hence the neuron's accumulated input field history is the one causing the increase in its membrane potential. In a spiky network, the same network activation pattern can be repeated several times during the counting process, since the state of a neuron includes its voltage. Thus, theoretically, using spiking dynamics, a network can count with a single neuron that accumulates voltage over $K - 1$ steps and fires on the Kth step.

We now examine if we can generally show a difference in the efficiency of the counting process between the evolved MP and spiky agents. We perform activation-pattern analysis on all the successful agents from the original evolutionary runs, and count the number of neurons that are active during the waiting period for each agent. As observed from Figure 5, more neurons are active during counting as the waiting period increases in both spiky and MP agents. However, for the same waiting period, more neurons are counting in the MP networks than in the spiky network. Thus, the evolved spiky networks exhibit more efficient counting than the MP networks, but not the most efficient kind (counting with only one neuron). The reason for this is that usually in the spiky networks several neurons that do not perform integration participate in the counting process as well. However, one or more neurons do use their spikiness to accumulate voltage and count for a few steps. As a result, less neurons participate in the counting process than in MP networks, explaining the lower distribution levels of spiky networks shown in Section 4.2.

4.5 Incremental Evolution

Since evolving agents with long waiting periods is a difficult evolutionary task (Section 4.1), we use incremental evolution techniques [10] to further evolve agents counting for even longer periods: Each evolutionary process starts with a population of 100 mutated copies of an already evolved agent counting to K, and develops an agent with a waiting period of $K + 1$. We present an example of two sets of incremental evolutions, one performed with MP agents and one with spiky agents. For

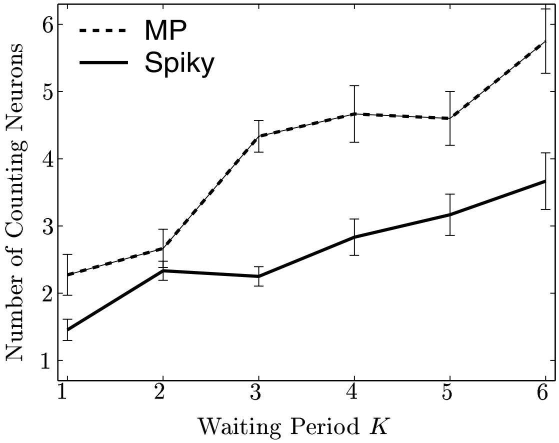


Figure 5. The number of active neurons during counting. Mean and standard error across all successful agents from the original 30 evolutionary runs for each waiting period (the same agents were averaged as in Figure 2b). For each agent the number of neurons active during the waiting period was counted (not including the forward motor, which becomes active after eating).

the MP networks, we have succeeded in evolving agents that count up to 15 (compared with a waiting period of 7 time steps, the longest we reached in regular evolutions). Examining the activation patterns of these agents during the waiting period revealed that agents counting to 3 use two neurons in the counting process, agents that count to 4 and 5 achieve it with four neurons, agents that count to 6 and 7 use five neurons, and an additional neuron is used for counting to 8 and more. This exemplifies, as discussed in Section 4.4, that under MP dynamics more neurons are needed to count for a longer period.

Using spiky networks, we have incrementally evolved agents that count up to 35, and arbitrarily stopped at this number. This is compared with a waiting period of 8 time steps, which is the longest we reached with regular evolutions using spiking dynamics. Exploring the counting dynamics of the incrementally evolved spiky agents revealed two main counting patterns: The spiky networks with waiting periods of 3 to 9 time steps all employ one pattern, in which two neurons are used for counting: one is spiky, and the other has a vanishing memory factor, and de facto behaves like an MP neuron that does not perform integration over its inputs. In the networks with a waiting period of 10 and longer, the counting MP-like neuron was transformed into a neuron with a significant memory factor that accumulates voltage over time as well, changing the counting pattern to use two spiky neurons, counting together. This result exemplifies that a spiky network can indeed employ a very efficient localized counting method (as described in Section 4.4). Spiky agents with even larger waiting periods can be constructed by continuing with the incremental evolutionary runs, further modifying the memory factors and the synaptic weights.

5 Quantifying Spikiness

5.1 Spikiness Measurements

Given a network evolved with spiking dynamics, we would like to answer the questions: which neurons really utilize their spiking dynamics, and are these the neurons involved in the counting subtask? That is, we would like to know if indeed spiking dynamics are evolved and utilized in the place where we intuitively would assume they are needed—where memory resides. To answer these questions, one has to carefully examine the "spikiness" of neurons. First, having encoded the neuronal memory factors in the genome gives rise to the possibility that evolution will come out with non-spiky solutions. Second, even if the memory factor is high, it does not ensure that the neuron indeed utilizes its "integration potential" in its firing.[2] For example, a neuron may receive a large excitatory input field in every time step and fire at a very high frequency, without performing any integration over its past input fields. That is, its pattern of firing would be indistinguishable from that of an MP neuron with the same input field. Third, even if the spikiness is utilized for firing, it does not necessarily contribute to the agent's performance. Essentially, we aim to distinguish between the observation that a given neuron has been assigned spiking dynamics by evolution, that is, obtained a nonvanishing memory factor, and the true level of its *spikiness*, that is, the amount by which it really utilizes its spiking dynamics. As it turns out, no such measures exist in the literature, and we hence decided to devise such measures and study them in our agents. In this section we present two methods for measuring the "true spikiness level of a neuron," which are based on two fundamentally different perspectives.

5.1.1 Spiking Dynamic Factor

The first index of spikiness—the *spiking dynamic factor* (SDF)—measures *how much the spiking dynamics of a neuron influences its firing.* If the firing pattern of a neuron is the same whether it possesses spiking dynamics or not, then it will be considered non-spiky. The SDF is calculated by comparing the firing of a spiky neuron with that of an MP neuron receiving an identical input field at each time step (last summand in Equation 1). The fraction of time steps in which there is a difference between the binary activations of the spiky neuron and the corresponding benchmark MP neuron quantifies the average percentage of lifetime steps in which the spiking dynamics made a difference in the firing of the neuron.[3] It is also possible to measure the *counting SDF*, reflecting the influence of the spiking dynamics of a neuron on its firing pattern specifically during the counting process. This is performed by simply calculating the SDF score only during steps in which the agent stands on food.

5.1.2 Spikiness Relevance

The second measurement of spikiness—the *spiking relevance* (SR)—answers the following question: *how essential are the spiking dynamics of a neuron for the good performance of the agent?* If on abolishing the spiking dynamics of a neuron the agent's performance deteriorates considerably, then its spiking dynamics contribute to the agent's behavior. If, in contrast, the fitness of the agent is maintained during this procedure, this neuron's spiking dynamics are functionally insignificant. To quantify this type of spikiness we use the MSA (Section 3) on data consisting of perturbations to the memory factors of the neurons only, leaving the rest of their dynamics unaltered. A neuron is perturbed by clamping its memory factor to zero (turning it into an MP neuron) and each perturbation configuration corresponds to a different subgroup of perturbed neurons. The contributions yielded by the MSA quantify the causal importance of the spiking dynamics of each neuron to successful behavior.

It is possible to further identify the neurons whose spikiness is specifically utilized for the counting mechanism, by measuring the agent's counting fitness score (Section 4.3) under each perturbation configuration, so that the network is perturbed only when the agent stands on food. The contribution yielded by the MSA on this data, called as the *counting SR* index, quantifies how important the spikiness of each neuron is to the counting process.

---

[2] This phenomenon frequently manifests itself in evolutionary optimization processes where some of the parameters may attain almost arbitrary values in "flat" regions of the fitness landscape.

[3] For each neuron i the SDF index is calculated as

$$\mathrm{SDF}(i) = \frac{1}{L}\sum_{t=1}^{L} |S_i(t) - O_i(t)|, \tag{4}$$

where L is the number of life steps of the agent, $S_i(t)$ is the activation of the spiky neuron at time t, and $O_i(t)$ is the activation of an MP neuron observing an identical input field at time t.

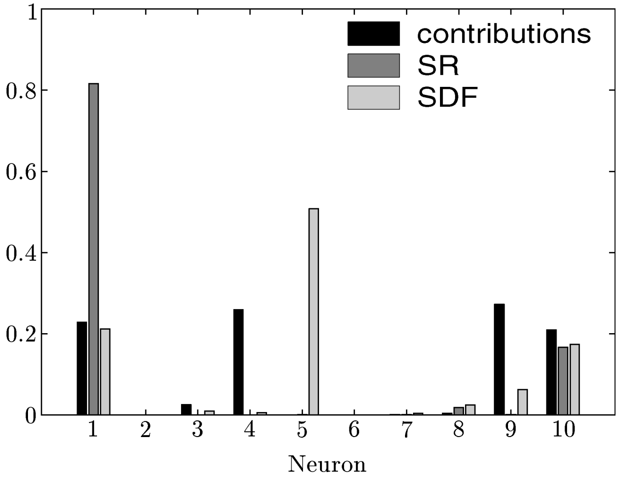


Figure 6. Spikiness measurements of S5. SR and SDF scores of the neurons, along with their basic contributions. To be comparable, all three measures are normalized so that the sum over all neurons equals one.

5.2 Analysis of Spikiness

5.2.1 Comparing the SDF and SR Indexes

We examine the spikiness level of the neurocontrollers evolved with spiking neurons, focusing on two spiking agents, S5 and S7, with a waiting period of 5 and 7 time steps, respectively. For S5, Figure 6 compares the SR values of the different neurons with the contributions of the neurons (Section 3). These contributions quantify how much each neuron is generally responsible for successful behavior. Notably, neurons 1, 4, 9, and 10 contribute significantly to the agent's behavior, as shown by their contributions, while the spikiness of only neurons 1 and 10 has a significant contribution, according to their SR values. Figure 6 also presents the SDF values, which differ greatly from the SR ones: Neuron 5 receives a very high SDF score, but a vanishing SR score. A more pronounced difference between the two spikiness measurements is apparent in Figure 7a, which shows both measures for agent S7 (previously analyzed in Sections 4.3 and 4.4). Here, neuron 7 has

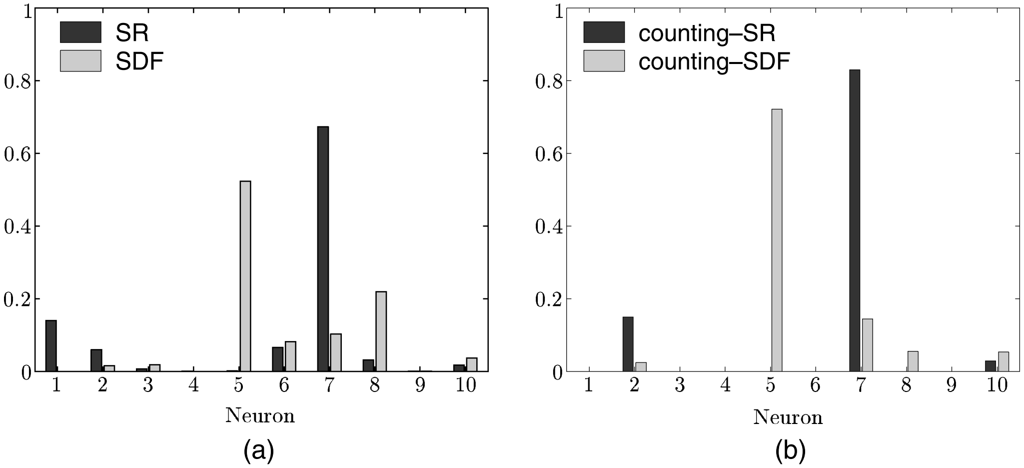


Figure 7. Spikiness measurements of S7. (a) SR and SDF scores. (b) Counting SR and counting SDF scores. All measures are normalized so that the sum over all neurons equals one.

the highest SR value, but a significantly lower SDF score. Neurons 5 and 8 are the most spiky ones according to the SDF measure, while both have low SR values. Finally, examining the size of the memory factors of the two agents, illustrating the extent to which the neuron's history influences its current membrane potential (Equation 1), reveals that they are not highly correlated with either of the measurements. For instance, for agent S7, the memory factors of neurons 5, 6, and 9 equal 1, although neuron 9 has vanishing values of both SR and SDF. In fact, it is common for a neuron with a high SR score to have a relatively low memory factor, if the neuron counts over a long waiting period.

The difference between the results of the two spikiness measurements originates in their different nature, as each measurement is designed to capture inherently different aspects of spiking dynamics. The SR measure targets the *semantic* role of spikiness—that is, its importance to the actual performance of the agent (either relating to the agent's overall survival, or computed for various network functions). The SDF measure is confined to the *syntactic* perspective of spikiness—that is, how the spiking dynamics alter neuronal firing compared with the null alternative of MP dynamics. We term the latter effect "syntactic" because large effects on the firing (as measured by the SDF) may turn out to have no effect on the actual behavior of the agent (as measured by the SR). However, neurons with large SR must have a nonzero SDF, since influencing the firing pattern of a neuron is a prerequisite of having a behavioral contribution. Such neurons also tend to be functionally important in general, with high neuronal contributions. There is another fundamental difference between the two measurements: The SDF index measures the influence of the neuron's spikiness on that neuron itself, performing the analysis on a "recorded" agent's life, without changing the actual neuronal activations. The SR index, by clamping the memory factor of the neuron during an active simulation, propagates the influence of the clamping through time, and thus takes into account the influence of a neuron on other neurons in consecutive time steps. In the case of agent S7, as described in Section 4.4, the spikiness of neuron 7 plays a pivotal role in the agent's counting ability by accumulating voltage for 3 time steps and then firing. Clamping this neuron's memory factor disables its firing on the sixth counting step, and further modifies the firing of neurons 4 and 1 in the following time steps, impairing the network's counting, and thus explaining its very high SR value. However, since the fraction of steps in which the spiking dynamics influence the activation of neuron 7 is only about 3% of its total lifetime steps (the counting steps), this neuron receives a low SDF score.

5.2.2 Counting SDF and Counting SR Indexes

Figure 7b plots the counting SR and counting SDF scores of agent S7. Comparing first between the SR and counting SR, and between the SDF and counting SDF scores, reveals that they are not identical for either of the spikiness measurement. For example, when comparing the SDF indexes, the relative influence of neuron 8's spikiness on its overall firing pattern (its SDF value) is much larger then its relative influence on the firing pattern during counting (its counting-SDF value). For the SR indexes, the spikiness of neuron 1 is generally important for successful behavior, but has no functional contribution to the counting mechanism (manifested by a zero counting-SR score). Neuron 7, on the other hand, has both high SR and high counting-SR scores. Hence, by considering solely the counting SR, we can detect that neuron 7's spikiness is crucial for the counting process. Still, it receives a relatively small counting-SDF score, since during the counting process the spikiness of neuron 7 influences its firing pattern in a single time step (the sixth one). In the next section we will further test whether it is generally the case in the evolved agents that the spikiness critically contributes to the counting process.

Generally, the counting-SDF and counting-SR scores help us better understand the counting dynamics of the spiky networks. As in the case of the SDF and SR indexes, the same fundamental differences between the two measurements exist. While the counting SDF only reflects the syntactic influence of spikiness on counting—namely, how the spiking dynamics alter neuronal firing during counting—the counting SR identifies which of the neurons actually uses its spikiness for the counting, in such a manner that impairing its spikiness will have an effect on the agent's counting performance. In this respect the SR and counting-SR indexes are the true measures of the causality and functionality of spikiness.

5.2.3 Overall Analysis

We turn to examine the extent to which the spiking dynamics is functionally utilized in the evolved spiky agents. We compute the SR scores for all successfully evolved spiky agents counting to 4 and higher (see Figure 2b for the number of agents for each *K*). Each neuron whose SR score is at least 10% of the sum of the agent's SR scores is considered to have a high SR contribution. Under this definition, we receive a mean value of 2.16 neurons with a large spikiness functional contribution (standard error of 0.168).[4] This indicates that in the evolved spiky agents, on average, only a small fraction of the network's neurons utilize their spiking dynamics for successful behavior (have high SR scores). But when considering, for the same agents, the number of neurons with large contributions (Section 3), reflecting the size of the functional neuronal backbone, we obtain a mean value of 5.35 important neurons (standard error of 0.23).[5] We can see that less than half of the neurons that crucially contribute to the network's successful behavior utilize their spiking dynamics for this purpose. To further identify the relation between the SR index and the counting SR in the evolved agents, we calculate for each tested agent the correlation between its SR and counting SR over the neurons. Out of the 23 agents analyzed, in 14 agents (more than 60%) we find a significant positive correlation (p value < 0.05) between the SR and the counting SR over the neurons. Thus, the neurons that have high SR scores tend to also have high counting-SR scores. The spiking dynamics are hence usually utilized by the agent for performing the counting subtask that really requires them.

6 Discussion

In this work we have succeeded in evolving agents solving a nontrivial, memory-dependent counting task, without using any special structure for counting, or giving an external reinforcement to the agent while waiting. Through incremental evolution we succeeded in evolving agents that count over very long waiting periods. But even by a regular evolutionary process we evolved an agent that stands still on a food item for precisely 8 time steps and then consumes it. We have shown that as the agents have to count over longer time periods, the resulting evolved networks use more distributed processing, and more neurons are active during the counting process. We found that the distribution level of a network, and the difficulty of evolving such a network in evolution, are negatively correlated. That is, in our settings, as the network solutions of a problem become more complex in terms of the distribution of processing, it becomes harder to evolve such solutions. This might be generally the case in other settings, making the chosen network architecture and dynamics an important component in determining the success or failure of the evolutionary process. Comparing two network dynamics, we have further shown that MP networks are more distributed and reach poorer evolutionary results than spiky networks. Hence in a counting task, as may be the case for other memory-dependent tasks, networks of spiking neurons can be less complex and easier to evolve than MP networks. This accounts for the importance of choosing a network architecture with characteristics and abilities that match the requirements of the given task. Additionally, we have tackled the question of how the networks perform the counting task, using several functional analysis methods. Specifically, we have illustrated how the ability of a spiky neuron to accumulate its membrane potential is utilized for counting, in a way that enables the spiky networks to use less neurons for the same counting process than do the MP ones.

---

[4] If we change the threshold over which a neuron is considered to have a high SR value to 5% of the sum of the agent's SR scores, we receive a similar mean value of 2.33 (0.81) neurons with a large spikiness contribution. Changing to 15% results in 1.95 (0.13).

[5] This was, analogously, calculated while considering a neuron with a large contribution as one whose contribution is at least 10% of the sum of the agent's contributions.

The study of spiky neural networks in the context of EAAs brings forward basic questions regarding spiking dynamics that have not yet been raised. Apparently, the potential spiking dynamics does not necessarily transcribe to actual spikiness in the network. We have presented two different ways by which the spikiness level of each neuron can be defined and quantified. Specifically, we have shown that the evolved spiky networks contain a truly spiky subnetwork responsible for the counting.

As a benchmark we compared the evolutionary results with another network model, of continuous-time recurrent neural networks (CTRNNs) [3]. CTRNNs, like the spiky networks, are capable of integrating inputs over time, and have been recently used in neuroscience experiments [4, 22]. We used CTRNN neurocontrollers composed of 10 continuous neurons, possessing a sigmoidal activation function. A motor action is performed if the activation level of the corresponding motor neuron exceeds a threshold. Each neuron has a memory factor (in the range [0, 1]) that evolves through evolution, and the neuron's dynamics are similar to the dynamics of the spiking neurons (Equation 1), except that CTRNNs do not quench information by spiking. The CTRNNs were successful in performing the counting task, also showing a decrease in the average fitness score as the agent has to wait for a longer period. Their average fitness score is higher than that of the MP networks, but lower than that of the spiky networks. The distribution index of the successfully evolved CTRNNs is similar to that of the MP neurocontrollers (higher than that of the spiky ones). Further work remains in comparing and understanding the differences between CTRNNs and spiky network models in this counting task.

Future studies may further compare spiking neural networks with other network models, and explore the spiking model in different environments and behavioral tasks. Specifically, it will be interesting to continue exploring this in the framework of EAAs, focusing on the interplay between the neural substrate, the environment, and the actual behavior. It is also possible to combine counting with traditional delayed-response match-to-sample tasks, by having different sensory inputs signal a different delay period for the requested behavior. The methods and indices presented in this work provide the basic tools for the analysis of such more complex agents, performing more complex types of tasks.

Acknowledgments

We acknowledge the valuable contributions and suggestions made by Hezi Avraham, Shay Cohen, and Ben Sandbank. This research has been supported by the Adams Super Center for Brain Studies at Tel-Aviv University and by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities.

References

1. Aharonov, R., Segev, L., Meilijson, I., & Ruppin, E. (2003). Localization of function via lesion analysis. *Neural Computation*, *15*, 885–913.
2. Aharonov-Barki, R., Beker, T., & Ruppin, E. (2001). Emergence of memory-driven command neurons in evolved artificial agents. *Neural Computation*, *13*, 691–716.
3. Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, *3*, 469–509.
4. Beer, R. D. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. In P. Maes, M. Mataric, J. Meyer, J. Pollack, & S. Wilson (Eds.), *Proceedings of the 4th International Conference on Simulation of Adaptive Behavior (SAB'96)* (pp. 421–429). Cambridge, MA: MIT Press.
5. Bugmann, G. (1997). Biologically plausible neural computation. *Biosystems*, *40*, 11–19.
6. Deese, J., & Hulse, S. (1980). *The psychology of learning* (5th ed.). New York: McGraw-Hill.
7. Fetterman, J. G. (1993). Numerosity discrimination: Both time and number matter. *Journal of Experimental Psychology: Animal Behavior Processes*, *19*, 149–164.

8. Floreano, D., & Mattiussi, C. (2001). Evolution of spiking neural controllers for autonomous vision-based-robots. In T. Gomi (Ed.), *Evolutionary Robotics IV* (pp. 38–61). Berlin: Springer-Verlag.

9. Floreano, D., & Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, *26*, 396–407.

10. Gomez, F., & Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, *5*, 317–342.

11. Jakobi, N. (1997). Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior*, *6*, 325–368.

12. Keinan, A. (2004). Analyzing evolved fault-tolerant neurocontrollers. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife9)* (pp. 557–562). Cambridge, MA: MIT Press.

13. Keinan, A., Sandbank, B., Hilgetag, C. C., Meilijson, I., & Ruppin, E. Axiomatic scalable neurocontoller analysis via the Shapley value. *Artificial Life* (in press).

14. Keinan, A., Sandbank, B., Hilgetag, C. C, Meilijson, I., & Ruppin, E. (2004). Fair attribution of functional contribution in artificial and biological networks. *Neural Computation*, *16*, 1887–1915.

15. Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*, 1656–1671.

16. Maass, W., & Ruf, B. (1999). On computation with pulses. *Information and Computation*, *148*, 202–218.

17. Marocco, D., & Floreano, D. (2002). Active vision and feature selection in evolutionary behavioral systems. In B. Hallam, D. Floreano, G. Hayes, & J. Meyer (Eds.), *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior (SAB'02)* (pp. 247–255). Cambridge, MA: MIT Press.

18. Paolo, E. A. D. (2002). Spike timing dependent plasticity for evolved robots. *Adaptive Behavior*, *10*, 243–263.

19. Platt, J. R., & Johnson, D. M. (1971). Localization of position within a homogeneous behavior chain: Effects of error contingencies. *Learning and Motivation*, *2*, 386–414.

20. Ruppin, E. (2002). Evolutionary autonomous agents: A neuroscience perspective. *Nature Reviews Neuroscience*, *3*, 132–141.

21. Scheier, C., Pfeifer, R., & Kunyioshi, Y. (1998). Embedded neural networks: Exploiting constraints. *Neural Networks*, *11*, 1551–1569.

22. Tuci, E., Harvey, I., & Todd, P. M. (2002). Using a net to catch a mate: Evolving CTRNNs for the dowry problem. In B. Hallam, D. Floreano, G. Hayes, & J. Meyer (Eds.), *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior (SAB'02)* (pp. 292–302). Cambridge, MA: MIT Press.

23. Shapley, L. S. (1953). A value for *n*-person games. In H. W. Kuhn & A. W. Tucker (Eds.), *Contributions to the theory of games, Vol. II* (pp. 307–317). Princeton, NJ: Princeton University Press.