

Neurocontroller Analysis via Evolutionary Network Minimization

Zohar Ganon¹, Alon Keinan¹, Eytan Ruppin^{1,2,†}

¹School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel

{*ganonzoh,keinanak,ruppin*}@post.tau.ac.il (tel: +972-3-6406528)

²School of Medicine, Tel-Aviv University, Tel-Aviv, Israel

[†]To whom correspondence should be addressed.

Abstract

This study presents a new evolutionary network minimization (ENM) algorithm. Neurocontroller minimization is beneficial for finding small parsimonious networks that permit a better understanding of their workings. The ENM algorithm is specifically geared to an evolutionary agents setup, as it does not require any explicit supervised training error, and is very easily incorporated in current evolutionary algorithms. ENM is based on a standard genetic algorithm with an additional step during reproduction in which synaptic connections are irreversibly eliminated. It receives as input a successfully evolved neurocontroller and aims to output a pruned neurocontroller, while maintaining the original fitness level. The small neurocontrollers produced by ENM provide upper bounds on the neurocontroller size needed to solve a given task successfully, and can provide for more efficient hardware implementations.

Keywords: Neurocontroller analysis, Network minimization, Synaptic pruning, Evolutionary algorithms, Evolutionary agents.

1 Introduction

Pruning a neural network is a standard approach in the connectionist literature by which unimportant synaptic connections are removed, usually in order to enhance the network’s generalization performance [12]. This study focuses on pruning and its applications in the context of neurally driven *Evolved Autonomous Agents (EAA)*. We propose a new, *Evolutionary Network Minimization (ENM)* algorithm, which is essentially a standard genetic algorithm with an additional step during reproduction in which synaptic connections are irreversibly eliminated. Starting from successfully evolved agents, ENM is utilized to prune their neurocontrollers while maintaining their fitness.

Minimization of successful agents is motivated by several goals. First and foremost, finding a small underlying skeleton of an agent permits a direct inspection of its functioning. By analyzing and comparing the several minimized successful neurocontrollers one can gain an understanding of the typical mechanisms of successful agents, one of the major open problems in the field of EAA. Second, the size and complexity of the minimized agent may stand as an estimate for the complexity of the task it solves (while the size of the original agent is arbitrary). Third, the minimized agent is more efficient computationally, hence, if the agent is implemented in hardware, fewer components will be needed.

The quest for network minimization has received considerable attention in the past. Several pruning algorithms in the neural networks literature, such as Optimal Brain Damage (OBD) [9] and Optimal Brain Surgeon (OBS) [6] are non-evolutionary methods which rely on the existence of a supervised error function. OBD and OBS analytically predict the effect of a synaptic connection removal on the error in a feed-forward neural network. In the case of EAAs no such explicit error function is available and the neurocontrollers may be recurrent. Another approach, contribution based pruning [1, 7],

deletes the synapses of a neurocontroller according to their contribution (importance) in the original network, without taking into account how the network operates after some of the synaptic connections have been eliminated. This is in contrast to ENM where the remaining synaptic connections can be mutated to compensate for the pruned ones. An iterative pruning algorithm [3] which does not require a retraining phase was suggested but it relies on a training set for its operation. A pruning algorithm for analyzing asynchronous random boolean networks was also presented [13] but is restricted to the context of random boolean networks. Some evolutionary minimization algorithms have been suggested [15, 16], but they too rely on supervised learning and retraining. Another approach to evolutionary minimization has been to incorporate an explicit network complexity term in the fitness function [17]. Such algorithms are problematic since balancing the coefficient of the original fitness and that of the network complexity term is a delicate art. ENM does not rely on such an arbitrarily defined explicit complexity term in the fitness function. Furthermore, the tree encoding used in [17] is inappropriate for recurrent neurocontrollers. Last, additions as well as deletions have been introduced to an evolutionary process from its beginning, starting with a random population [11]. ENM introduces a more strict pressure, starting with an already evolved network, to minimize it, without allowing it to increase in size.

ENM has a unique combination of features which distinguish it from previous evolutionary minimization algorithms: 1. Minimization begins after functional neurocontrollers have been successfully evolved. 2. An explicit penalty term is not added to the fitness function. 3. Successful minimization relies solely on the workings of a pressure which removes unimportant synaptic connections and, importantly, on continuing adaptive modifications of the magnitudes of the remaining weights. 4. ENM does not utilize a supervised error function and therefore is applicable for the minimization of autonomous agents.

The rest of this paper is organized as follows: Section 2 presents the ENM algo-

rithm. In sections 3 and 4 we describe the results of applying ENM in two different environments. These results demonstrate the potential utility of ENM for minimizing neurocontrollers, facilitating the understanding and the identification of general characteristics of successful agents. The results also testify to the effectiveness and the consistency of ENM. Finally, section 5 discusses the results.

2 The ENM Algorithm

The ENM algorithm receives as input a successfully evolved neurocontroller and aims to output a pruned neurocontroller, while maintaining the fitness levels of the original agent. ENM is a genetic algorithm which evolves a population of genomes, directly encoding the networks' weights (and possibly also other parameters such as the neuronal biases). Receiving a neurocontroller to be minimized as its input, the algorithm starts by duplicating the input network's genome to form the first generation. Then, a standard genetic algorithm is used, with an additional elimination step which takes place after applying the standard selection, crossover and mutation operators, in each generation. In the elimination step, each synaptic connection is eliminated (zeroed) with a certain fixed *deletion probability*. Other elimination criterions can also be used during the elimination step, such as the one we have previously introduced [5], according to which a synaptic connection is eliminated if the magnitude of its weight is smaller than a predefined threshold. Eliminated synapses remain zeroed and cannot be revived by mutation, thus creating pressure for network minimization. As demonstrated in section 3.2, a crossover operator does not prevent successful minimization. The fitness function used for evaluating the new generation agents is simply the original one. Evolution continues until both the average fitness and the average number of remaining synaptic connections in the population converge. The result of the evolutionary process is then a population of networks from which ENM selects and outputs

the network with the best fitness.

The deletion operator is the source of the pressure towards a minimized genome. The smaller the deletion probability is, the longer it will take the algorithm to converge, while a too large deletion probability might result in poor fitness scores. The deletion probability parameter is hence set by a search process that finds its highest value that still leads to a high average population fitness score. An example for a search technique would be to start from a low deletion probability and verify that the average population fitness remains high throughout evolution. Then, the deletion probability is gradually increased, while verifying each time that the average fitness remains high, resulting in a reduction in the convergence time. The increase of the deletion probability is ceased when the average population fitness drops.

A standard genetic algorithm for evolving neurocontrollers can be easily transformed into an ENM in the following simple way: All the genetic operators and their parameters from the original algorithm (which were possibly optimized for a specific environment), as well as the fitness evaluation are retained as is and the following three enhancements are introduced:

1. Start from a first generation of duplicates of one successful agent.
2. Modify the mutation operator to perform the following operation on each parameter: if the parameter value is zero retain this value. When it is not zero then with a certain given deletion probability zero the value of the parameter, otherwise change the parameter value according to the original mutation operator.
3. Perform a preliminary coarse search to identify a good deletion probability.

Following this procedure, in the two environments described in this paper the genetic operators are the same as those in the original algorithms that created the successful agents. In both environments the weights are directly encoded in the genome,

as in the original algorithms. In section 3, roulette wheel selection is used with uniform point crossover (probability of 0.35). Mutation is applied with probability of 0.02, adding a uniformly distributed random value between -0.2 and 0.2 . Deletion probability is 0.002. In section 4 fitness based selection is used. Mutation is applied such that with probability of 0.02 each bit of the genotype is replaced with a new randomly selected binary value. Deletion probability is 0.001. In both environments the ENM is not sensitive to the deletion probability parameter and converges to the same results with a deletion probability that is smaller by an order of magnitude.

3 ENM in a Memory-dependent Foraging and Navigation Task

3.1 The Model

The EAA model used in this section is described in detail in [2]. A population of agents performing a navigation and foraging task is evolved in a discrete grid arena surrounded by walls. Poison items are scattered in the whole arena, 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 poison items. Its fitness is the number of food items minus the number of poison items it has consumed, normalized by the total number of food items that were available giving a maximal value of 1. Due to its short lifetime, a fitness score of 1 is practically unattainable.

The sensory system of the agent is made of 5 sensors: *front*, *left*, *right* and *here* which provide an indication whether there is a resource (food or poison, without a distinction between the two), a wall or an empty cell in front, left-front, right-front and underneath the agent, respectively. The *smell* sensor gives an indication of food or poison if they are directly underneath the agent and a random reading otherwise. In

several simulations the agent was also equipped with position sensors indicating its x and y grid coordinates, otherwise it had only local information from its 5 sensors. Four motor neurons dictate movement forward, a turn left or right, and control the state of the mouth (open or closed). Eating takes place if the agent opens its mouth and is neither moving nor turning. The agent is controlled by a fully recurrent neurocontroller of binary McCulloch-Pitts neurons, with the exception of the non-binary sensor neurons which have no input from other neurons.

Previous analysis [2] revealed that successful agents possess one or more *command neurons* that determine the agent’s behavioural strategy. Artificially clamping these command neurons to either constant firing activity or to complete quiescence causes the agent to constantly maintain one of the two behavioural modes it exhibits, regardless of its sensory input. These two behavioural modes are *exploration* and *grazing*. Exploration, which takes place when the agent is outside of the food zone, consists of moving in straight lines, ignoring resources in the sensory field that are not directly under the agent, and turning at walls. Grazing, which takes place when the agent is in the food zone, consists of turning towards resources to examine them, turning at walls and maintaining the agent’s location on the grid in a relatively small region.

Throughout this section we apply the ENM to three successful evolved agents: A1, A2 and AP with 22, 10 and 10 neurons (including the motors), respectively. A1 and A2 are equipped with only the 5 local sensors while AP is also equipped with the position sensors.

3.2 Minimization of Successful Neurocontrollers

We applied ENM for minimizing agent A1 having 22 neurons and 594 synapses. Figure 1 shows the average fitness and average fraction of remaining synaptic connections out of the total initial fully connected set of synaptic connections, throughout the

minimization process. Evidently, ENM markedly reduces the number of network connections while preserving high fitness values. The minimized neurocontroller consists of only 26 synaptic connections out of the original 594. The average fitness declines in the beginning of the evolutionary process, and is then gradually restored. This initial decline is due to the random elimination of important synaptic connections in some agents, which are later removed from the population by the selection pressure.

To assess the effectiveness of ENM, we compared the fitness of ENM pruned agents with agents pruned by other methods, including completely random pruning, magnitude based pruning and contribution based pruning of the synaptic connections (Figure 2). Random pruning removes the synaptic connections in random order. Magnitude based pruning removes in each iteration the synapse with the smallest magnitude. Contribution based pruning removes in each iteration the synaptic connection with the smallest contribution according to a Multi-perturbation Shapley value Analysis (MSA) of the intact neurocontroller [8, 7]. ENM prunes 96% of the synaptic connections while maintaining the original fitness almost intact, whereas the other methods result in a significant decrease in the agent’s fitness already at much smaller levels of pruning. The difference in the final pruning percentages between ENM (96%) and contribution based pruning (90%) is not only quantitative but a qualitative one. Understanding the fully recurrent network with 26 synaptic connections simply by observing it is manageable while doing it with 60 synaptic connections is an intractable task. It should be noted here that there is an inherent advantage in this comparison towards ENM that stems from the nature of these three methods. In all three methods, synaptic connections are irreversibly deleted without any change to the remaining ones. ENM, on the other hand, allows the remaining synapses to compensate for the deleted ones.

In order to test the consistency of ENM, we repeated 10 times the minimization of each of the evolved agents A1, A2 and AP. Figure 3a presents the fitness levels of the resulting minimized agents. In all three agents the fitness is very close to the fitness

of their original intact agent, with small standard deviations, the latter testifying to the consistency of ENM. Figure 3b presents the number of synaptic connections of the minimized agents. Evidently, the number of synaptic connections is very small compared with the original intact agents and is consistent across different ENM runs.

3.3 Preservation of Functional Characteristics

Since the ENM minimized agent is an evolutionary descendant of the original successful agent, one may hope that it preserves the functional characteristics of the latter. Nevertheless, such preservation is not guaranteed. In this section we show that minimized agents in this environment do preserve the essential functional characteristics of their ancestors, based on their behavior (3.3.1), their neural activity patterns (3.3.2) and the set of synaptic connections retained after the minimization process (3.3.3). Preservation of functional characteristics is a much desired property if one aims to use ENM to obtain insights into the workings of the original, non-pruned agent.

3.3.1 Apparent Behavior

The minimized agents exhibit behavioral patterns similar to those exhibited by the original ones. For example, A2 has a unique characteristic, not found in other successful agents evolved in the same environment: When walking along a wall, the agent keeps a one cell distance from it. All the ENM minimized agents of A2 exhibit this exact same unique characteristic.

3.3.2 Neural Activity Patterns

In order to compare the neural activity patterns of the original agents with those of the minimized agents, we evolved agents to solve a more complex task, based on the work presented in [14]. In this variant of the original task, in order for the agent to

successfully eat, it must wait on the food item for 5 steps without moving or turning. Due to the deterministic update rule of the network, a specific agent has a characteristic neural activity pattern during counting when starting from the same initial activity pattern. To examine how close are the dynamics of the minimized agent to those of its original ancestor, we compared the activities of the 10 non-sensory neurons during the different time steps of the counting period, measuring their similarity as the percentage of neurons that fire identically. We found that the similarity of the dynamics averaged over a population of 5 minimized agents and their ancestors is 91%. The similarity of the firing between minimized agents and their non-minimized ancestors is much higher than the similarity between the dynamics over population of agents evolved separately to solve this task (the latter similarity averaged over a population of 5 pairs is 49%).

3.3.3 The Retained Synaptic Connections

Repeating ENM many times and considering the surviving synaptic connections in each minimized agent, we plot the distribution of the number of times that the different synaptic connections have been retained in the final minimized agents (Figure 4). Note that a large portion of the original synaptic connections of a typical minimized network (14 out of an average of 27) have survived in all the minimized agents. This observation indicates that the ENM implicitly strives to preserve a core of important synaptic connections while explicitly deleting randomly. If the synaptic connections that survive the ENM were to be selected randomly and not as a function of their merit then the number of surviving synapses would have been strictly monotonously decreasing with the frequency of survival. The characteristic distribution manifested in Figure 4 shows that the latter is clearly not the case.

Focusing on a specific minimized neurocontroller (analyzed in section 3.4), it has the 14 synaptic connections that survived all the ENM runs and 8 other synaptic connections that survived only some of the runs. A further investigation shows that

these 8 synaptic connections can be deleted without any fitness deterioration simply by adding a corresponding bias to the post-synaptic neuron, which simulates the condition of the pre-synaptic neuron always being active. The genetic algorithm could not find such solution as the biases were not evolved but rather set to a fixed value of zero. Regarding the 14 synaptic connections that always survive, 13 out of them cannot be replaced by biases without a drastic fitness deterioration. This is consistent with the fact that there are 3 different possible sources of bias in the neurocontroller (interneurons 1, 2 and 4). The different minimized agents of A2 obtained with ENM have hence successfully kept the synaptic connections which convey important information while each has used different alternatives as sources of bias.

We turn to further investigate the hypothesis that the more important synaptic connections of the input network stand a better chance of being retained during minimization. To quantify the importance of each synaptic connection in the original neurocontroller, we use a perturbation paradigm [1], afflicting numerous multiple perturbations upon the synaptic connections, while measuring the corresponding performance score of the agent during each such perturbation. Based on these data, the importance of each synaptic connection to the agent’s overall fitness can be rigorously assessed via the Multi-perturbation Shapley value Analysis (MSA) [8, 7]. We find that the 14 synaptic connections that survived all the 20 different ENM runs are all among the first 20 most important synaptic connections according to this analysis. In contrast, the 14 synaptic connections are only among the first 70 synaptic connections (out of 150) when considering them by order of their synaptic magnitude. This testifies that the probability of the synaptic connections to be retained following an ENM minimization is indeed related to their functional importance in the original, intact agent. This ENM tendency to remove the less important synaptic connections is paramount for preserving the agent’s functional characteristics.

3.4 Analyzing a Minimized Neurocontroller

A prime advantage of pruning which preserves the agent's functional characteristics is that it can help in understanding the kind of neural processing involved in solving the task in hand by direct inspection of the minimized neurocontroller that it yields. The conclusions derived from such an analysis of the minimized neurocontroller may then be further tested on the functionally similar original agent. To demonstrate this, we applied ENM to the fully recurrent network of agent A2, consisting of 150 synaptic connections which obviously is non-amenable for any visual inspection of its workings. The resulting pruned network consists of only 22 synaptic connections (Figure 5) and has a fitness almost as good as the original network of A2 (0.37 compared with 0.39). The minimized network is simple enough to be fully understood by observing its structure. In the following we discuss the network's structural characteristics that underlie three main behavioral features of the agent. The rest of the behavioral features may be explained in a similar, meticulous manner.

1. **When reaching a cell with food, the agent stops and eats it.** Standing on a food item, both the *here* and the *smell* sensors are active. Together, they inhibit the *move* motor, causing the agent to stop. Furthermore, the *smell* neuron activates the *eat* motor. The inhibition on the *move* motor is ceased after eating, since the *here* sensor is inactivated.
2. **When not in a food cell, the agent always moves.** Interneuron 2 is always active due to a loop of mutual excitatory connections with Interneuron 1 which is always active too. Interneuron 2 in turn constantly activates the *move* motor and causes it to fire. Its activation is only antagonized when the agent enters a food cell and the *smell* and the *here* neurons fire and inhibit the *move* motor.

3. **Switching between exploration and grazing behavioral strategies** (Section 3.1). When interneuron 3 is silent, the *move* motor activates the *right* motor which then activates the *left* motor. Both stay active, dictating a direct forward move, which constitutes the exploration. When interneuron 3 is active, triggered by food consumption causing the silencing of the *move* motor, it inhibits the *right* motor. Whenever the *left* sensor is now active, there is high probability that the agent will turn left, forming the grazing behavior (turning towards resources to examine them). Thus, interneuron 3 is the command neuron controlling the behavioral strategy.

Indeed, it should be noted that this kind of detailed analysis may not be possible in general. Yet it should be emphasized that the smaller, minimized neurocontrollers derived via ENM make further analysis of their workings, with any other method, easier and simpler.

3.5 General Characteristics of Successful Agents

An important application of ENM (independent of the preservation of functional characteristics) is the possibility to understand characteristics of successful agents solving a given behavioral task. Using an analysis of the type described in section 3.4 over minimized neurocontrollers of several successful agents solving the same task, one may differentiate between *general principles* and *specific principles*, general principles being the ones common to many successfully minimized agents. In this task, the existence of a command neuron (or several command neurons) is a general principle, as well as the loop causing the constant activation of the *move* motor. Another general principle revealed by this population analysis is the mechanism which causes the agent to stop on food, based on the inhibition of the *move* motor by the *here* and the *smell* sensors. In contrast, in this task, the exact mechanism for toggling the state of the command

neuron is a specific principle, as well as the way the command neuron controls the behavior of the agent.

4 ENM in a Homing Navigation Task

4.1 The Model

The EAA model used in this section is described in detail in [4]. We perform all experiments using Nolfi's simulator [10], replicating the experiments that were originally performed using real Khepera robots [4].

A simulated Khepera robot moves in the environment while avoiding walls. It is equipped with a restricted energy supply which is recharged when the agent returns to a recharging area. The environment consists of a 40x45 cm arena with an 8 cm radius sector in the corner that represents the recharging area. The recharging area is illuminated by a light source and has a different brightness than the rest of the environment. When the robot arrives at the recharging area its battery is instantaneously fully recharged. The fitness function has two components: The first one is maximized by the agent's speed and the second by its success in obstacle avoidance. A robot's fitness is accumulated and therefore influenced by the duration of its life.

The sensory system of the robot consists of eight proximity sensors (six in front and two in the back), two light sensors (one in front and one in the back), one sensor that senses the floor of the recharging area (by its different brightness) and one sensor that provides information regarding battery status.

The motor system consists of two motor neurons. The activation of each motor neuron is used to set the velocity of the corresponding wheel in a range where 0.0 is maximum speed in one direction, 0.5 corresponds to absence of motion, and 1.0 is the maximum speed in the other direction.

The neurocontroller is a multi layer perceptron of continuous sigmoid neurons, with one hidden layer consisting of five neurons with recurrent connections between themselves. The number of network parameters consisting of both synaptic connections and neuronal biases is 102.

The evolved successful agents have a common behavioral pattern (exhibited by all the ten best agents evolved): They move in an organized manner in almost straight lines as shown in Figure 6, until they reach the corner with the light, recharge and continue with a similar path as before.

F1, a typical agent, has a behavior (Figure 6a) which is consistent over different initial conditions. It moves in straight lines and turns right when in front of a wall. When it reaches the recharging corner (top left in Figure 6a) and the light source is located behind it, it moves backwards until it enters into the recharging area itself, recharges and then moves forward away from the recharging area. As we shall demonstrate, the neural mechanisms of these behavioral characteristics can be traced by examining the agent's minimized network (Section 4.3).

4.2 Performance of Minimized Neurocontrollers

We applied ENM 10 times for minimizing each of two evolved agents, F1 and F2. Figure 7a presents the fitness levels of the resulting minimized agents. In both cases the fitness is very close to the fitness of the original intact agent, with small standard deviations, testifying to the effectiveness and consistency of ENM in this environment. Figure 7b presents the number of parameters surviving the minimization. Evidently, the number of remaining parameters is much smaller compared with the original agents, and is consistent across many ENM runs.

4.3 Analyzing a Minimized Neurocontroller

In this section we focus on MF1, one of the ENM minimized descendants of F1. While F1's original network is complex and non-amenable for visual inspection of its workings, the minimized network consists of only 26 parameters (24 synaptic connections and 2 biases; Figure 8) while still having a high fitness of 0.62 (compared with 0.63 of the original F1). In the following we describe the main behavioral features of agent MF1 and their origins in the network structure. Figure 6c presents the behavior of the ENM minimized agent recorded in the environment for a limited time of 100 cycles. MF1 has the same apparent behavior as F1 as can be seen in comparison with Figure 6a.

The neurocontroller of MF1 is shown in Figure 8. The proximity sensors are located on the circumference of the agent. The relative positions of the proximity sensors are: P0 is the left front sensor, P1 is located to its right, P2 and P3 are in the front and P4 and P5 further to their right. P6 and P7 are located at the back of the agent.

We begin the analysis of the underlying neuronal mechanisms by identifying the role played by each neuron in the hidden layer. Three hidden neurons, H2-H4, play a specific and crucial role in the agent's behavior. H2 is responsible for right turns. It activates the left motor (M0), while inhibiting the right one (M1). H3 is responsible for moving forward. It strongly activates both motors and has a self excitation. H4 is responsible for moving backwards as it inhibits both motors. H0 functions as a 0.5 constant bias to H3. H1 and five sensors were removed from the network during minimization, as all of their incoming and outgoing synaptic connections have been deleted. The removed sensors are the proximity sensors on the right side of the agent (P4 and P5), the proximity sensors on the back of the agent (P6 and P7) and the sensor which senses the floor in the recharging area.

Having a "skeletal" minimized network in hand as the one portrayed in Figure 8, we can now turn to describe the specific network structures underlying MF1's main

behavioral characteristics:

1. **Moving forward, turning at walls.** When the agent approaches a wall, the proximity sensor P2 is activated, which activates hidden neurons H2 and H4 and inhibits H3. This causes the agent to turn right and backwards away from the wall. When the agent turns right, proximity sensor P1 becomes activated which causes the agent to further turn right by activating H2 and inhibiting H3. Then, P0 becomes close to the wall and causes a further slower movement to the right and backwards, after which the agent no longer faces the wall so the inhibition from P1 and P2 on H3 is removed and the agent starts moving forward.
2. **Moving backwards towards the light and the recharging area.** When the agent arrives to the corner where the light is placed and starts its slow backwards movement from the wall, the *light back sensor* (LB) is activated, inhibiting H3 and activating H4, causing the agent to move backwards towards the light and the recharging area which is adjacent to the light.
3. **Moving away from the recharging area after a recharge.** After recharging, the *battery sensor* (B) is activated and it activates H3, while inhibiting H4. As a consequence, the agent moves forward away from the recharging area.

Finally, it is interesting to note that since the evolutionary pressure within ENM strives at removing as many synaptic connections as possible while maintaining the agent's fitness, behavioral characteristics and neural processing features that are not necessary for achieving high fitness may be discarded in the process. This is manifested when testing F1 and its minimized version MF1 in a different environment from the one they were evolved in, in which the recharging area is abolished and no recharging occurs. In this environment, Floreano et al. [4] describe an agent which exhibits a behavior of moving in the vicinity of the light as if waiting for recharging. A similar

behavior is also exhibited by our evolved agent F1; when there is no recharge it goes back and forth near the light. However, this behavior is not preserved in MF1 which bumps into the wall and ceases to move when no recharging occurs.

5 Discussion

This study presents a new evolutionary network minimization algorithm, ENM. Given a successful neurocontroller, ENM consistently finds significantly smaller neurocontrollers, while preserving their fitness. We demonstrated the application of ENM in two different evolutionary autonomous agents environments which span two different tasks and different types of neurocontrollers. ENM succeeds in minimizing agents in both environments to the level that the minimized neurocontrollers, in contradistinction to their ancestors, are amenable to human inspection and understanding.

ENM allows for the analysis of minimized agent neurocontrollers, as well as for identifying common features underlying the neurocontrollers of different agents solving the same task. It can markedly enhance the utility of other methods which are conventionally used to understand neurocontrollers such as receptive field analysis, lesioning, or clamping experiments, since these methods can now be used on the minimized and much simpler neurocontrollers generated by ENM. In addition, ENM potentially enables the generation of more efficient hardware neurocontroller implementations. Finally, ENM allows for estimating a task's complexity by providing an upper bound on the number of components needed to solve a given task successfully. An example is provided in section 3, where A1 and A2, which solve the same task, have a similar number of synaptic connections after minimization (Figure 3b) even though their original sizes vary considerably (594 and 150 synaptic connections respectively). Compared to A1 and A2, AP which has additional sensors and thus performs an easier task, has a smaller number of synaptic connections after minimization.

ENM has a unique combination of features which distinguish it from previous evolutionary minimization algorithms: 1. Minimization begins after functional neurocontrollers have been successfully evolved. 2. No explicit penalty term is added to the fitness function. 3. Successful minimization relies solely on the workings of a pressure and on continuing adaptive modifications of the magnitudes of the remaining weights. 4. ENM is applicable to the analysis of autonomous agents' neurocontrollers (with any given architecture) since it does not require a supervised error function. These four features were found to be necessary for successful minimization of agents. The first feature, minimizing only successful neurocontrollers, is important because if a deletion is introduced in the earlier stages of the evolution when the synaptic connections have not yet captured their eventual functional roles, then synaptic connections that might later become important are irreversibly deleted. Using a very small deletion probability may overcome this problem but results in extremely slow convergence. Introduction of an addition operator can partially circumvent this problem. Our results indicate however that this solution yields fewer successful agents than achieved by the original evolution which ENM is based on (out of 25 runs, in the first environment of section 3 we obtained 10 successful agents with ENM versus 5 with addition and deletion, and in the second environment we obtained 5 versus 3 successful agents, correspondingly). Adding an explicit fitness term also did not improve the effectiveness of ENM and in certain cases even decreased it by "masking" the original fitness function.

ENM is a very simple evolutionary process and hence can be easily incorporated into existing evolutionary systems. The procedure described in section 2 for transforming a genetic algorithm which evolves neurocontrollers into a minimization algorithm has been successfully followed in two distinct environments described in this paper. In both environments it consisted of adding only two statements to the original source code; indeed a minimal intervention in its own right.

Acknowledgments

We acknowledge the valuable help provided by Ranit Aharonov and Isaac Meilijson and the technical help provided by Oran Singer. This research has been supported by the Adams Super Center for Brain Studies in 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] Castellano, G., Fanelli, A., & Pelillo, M. (1997). An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks*, 8, 519–531.
- [4] Floreano, D. & Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 26, 396–407.
- [5] Ganon, Z., Keinan, A., & Ruppin, E. (2003). Evolutionary network minimization: Adaptive implicit pruning of successful agents. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J. T., & Ziegler, J. (Eds.), *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, volume 2801 of *Lecture Notes in Artificial Intelligence*, (pp. 319–327). Springer Verlag Berlin, Heidelberg.
- [6] Hassibi, B. & Stork, D. (1993). Second order derivatives for network pruning: Optimal Brain Surgeon. In Hanson, S. J., Cowan, J. D., & Giles, C. L. (Eds.),

- Advances in Neural Information Processing Systems*, volume 5, (pp. 164–171), San Mateo, CA. Morgan Kaufmann.
- [7] Keinan, A., Sandbank, B., Hilgetag, C. C., Meilijson, I., & Ruppin, E. Axiomatic scalable neurocontroller analysis via the shapley value. *Artificial Life*, to appear.
- [8] 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.
- [9] LeCun, Y., Denker, J., Solla, S., Howard, R. E., & Jackel, L. D. (1990). Optimal Brain Damage. In Touretzky, D. S. (Ed.), *Advances in Neural Information Processing Systems*, volume 2, (pp. 598–605), San Mateo, CA. Morgan Kauffman.
- [10] Nolfi, S. Evorobot simulator 1.1 (page visited on september 2003). <http://gral.ip.rm.cnr.it/evorobot/simulator.html>.
- [11] Quinn, M., Smith, L., Mayley, G., & Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, 361, 2321–2344.
- [12] Reed, R. (1993). Pruning algorithms - a survey. *IEEE Transactions on Neural Networks*, 4, 740–747.
- [13] Rohlfshagen, P. & Di-Paolo, E. A. (2004). The circular topology of rhythm in asynchronous random boolean networks. *Biosystems*, 73, 141–152.
- [14] Saggie-Wexler, K., Keinan, A., & Ruppin, E. Neural processing of counting in evolved spiking and McCulloch-Pitts agents. *Artificial Life*, to appear.

- [15] Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing*, 14, 347–361.

- [16] Yao, X. & Liu, Y. (1996). Evolutionary artificial neural networks that learn and generalise well. In *Proc. of IEEE Int. Conf. on Neural Networks*, (pp. 159–164).

- [17] Zhang, B. T. & Mühlenbein, H. (1993). Genetic programming of minimal neural nets using Occam’s razor. In Forrest, S. (Ed.), *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, (pp. 342–349), University of Illinois at Urbana-Champaign. Morgan Kaufmann.

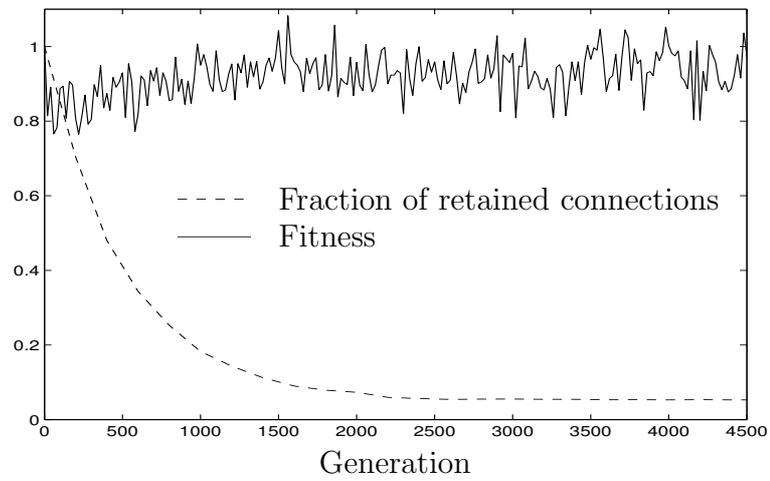


Figure 1: *Average fitness and average fraction of retained synaptic connections across generations during ENM of agent A1. The fitness is normalized such that the fitness of the original ancestor agent equals 1. The averaging is over a population of 100 agents.*

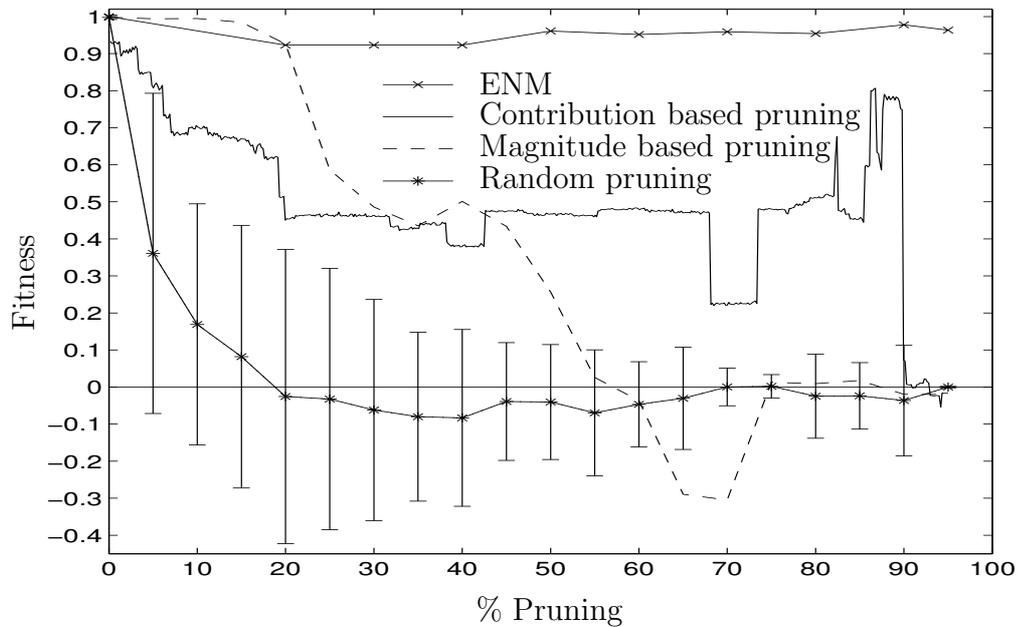


Figure 2: Comparison of different pruning methods for minimizing agent A1. The agent’s normalized fitness is plotted against the percentage of pruned synaptic connections, using four different pruning methods: ENM, contribution based pruning, magnitude based pruning and pruning the synaptic connections in random order (mean and standard deviation across 25 runs). ENM converged to a final network architecture after 96% of the synaptic connections were pruned.

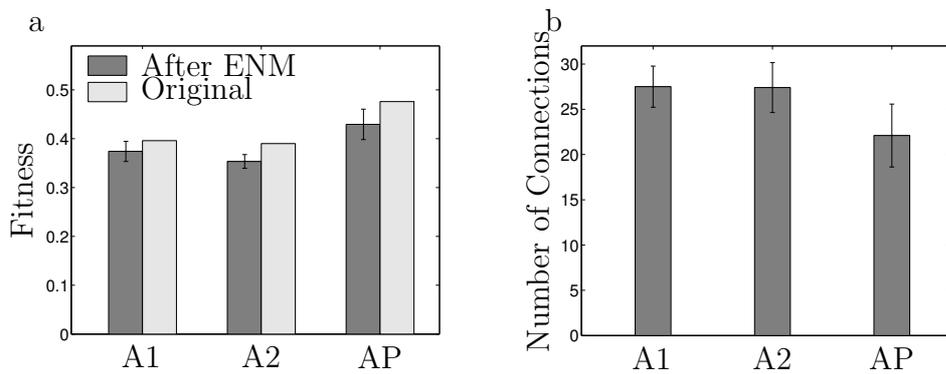


Figure 3: *Fitness and number of synaptic connections of ENM minimized agents.* Mean and standard deviation across 10 ENM runs. (a) Fitness after minimization portrayed along with the fitness of the original agents. (b) Number of synaptic connections after minimization. The original A1, A2 and AP agents have 594, 150 and 170 synaptic connections, respectively.

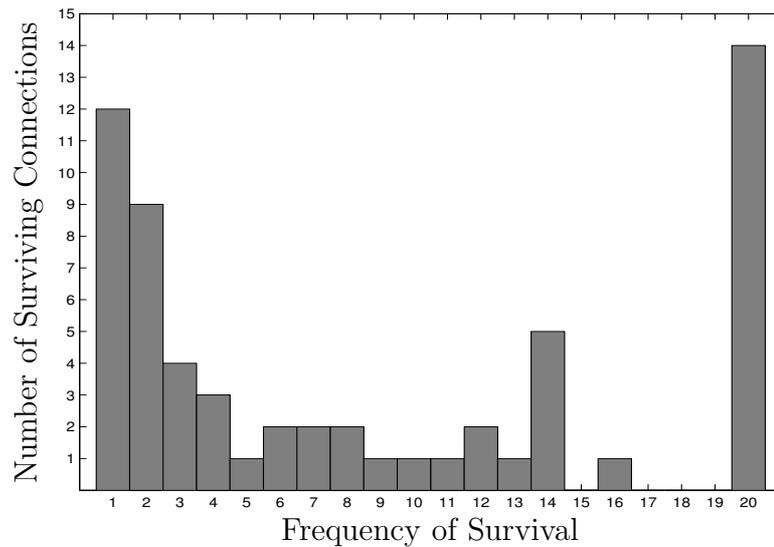


Figure 4: *Distribution of synaptic connections across different ENM runs of agent A2.* The histogram plots the number of different synaptic connections (y-axis) surviving the pruning a certain number of times (x-axis) out of 20 ENM runs. For instance, 12 different synaptic connections survived in only 1 run and 14 synaptic connections survived all 20 runs.

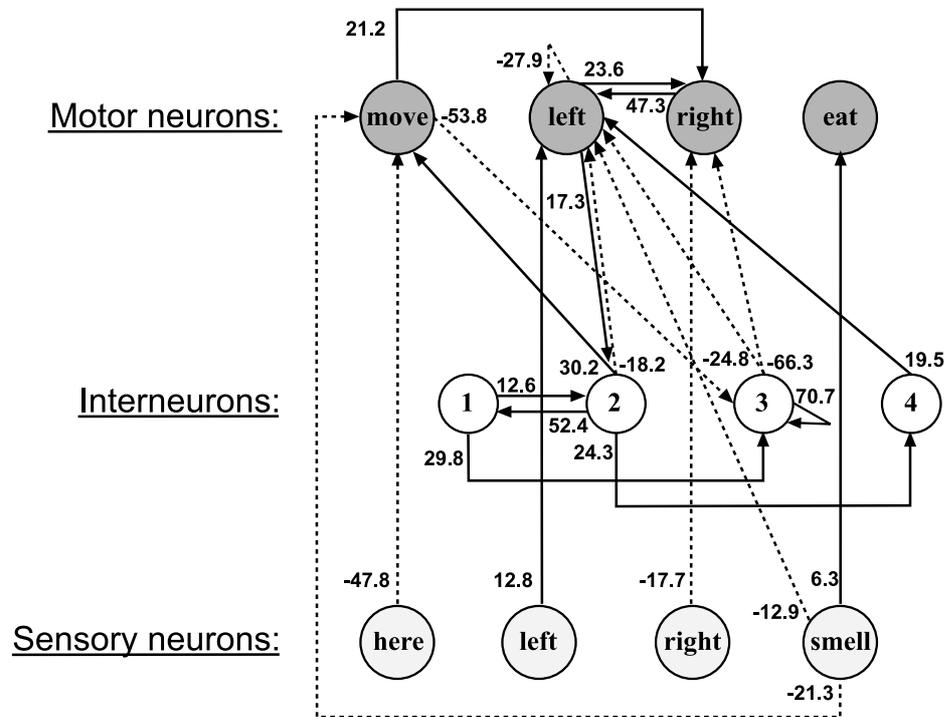


Figure 5: *The minimized neurocontroller of agent A2.* The neurocontroller consists of 22 synaptic connections. Dashed arrows represent inhibitory synaptic connections while solid arrows represent excitatory synaptic connections, with the synaptic weights indicated next to each arrow. Three neurons are not shown in the figure since all their synaptic connections were eliminated.

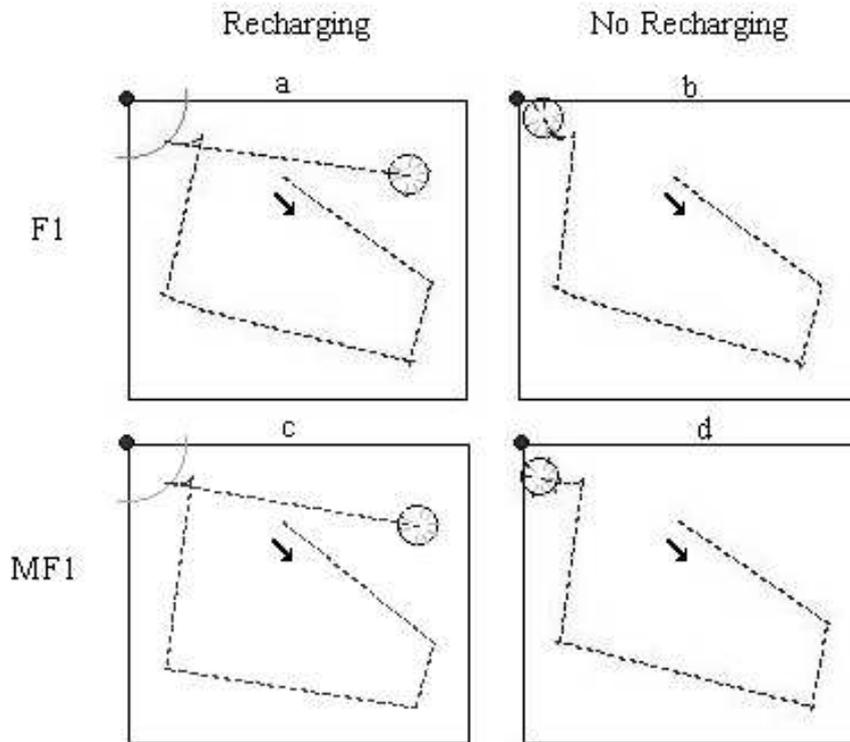


Figure 6: *Agents' behavior*. The figures are output of the simulator [10] showing the agent in the arena from a top view. The circle represents the agent. The small gray lines inside the agent's circle show the proximity sensors, six in front and two in the back. The black dot in the top left corner represents the light source and the arc adjacent to it represents the border of the recharging area. The dashed line shows the path that the agent took in the arena during its movement, with the arrow indicating the direction of movement. (a) and (c) show the behavior when recharging takes place as soon as the agent reaches the recharging area. (b) and (d) show the behavior when light in the top left corner is still on but recharging does not take place (see main text). (a-b) Original F1 (c-d) Minimized F1 (MF1). All are placed in the same environment and start from the same initial conditions.

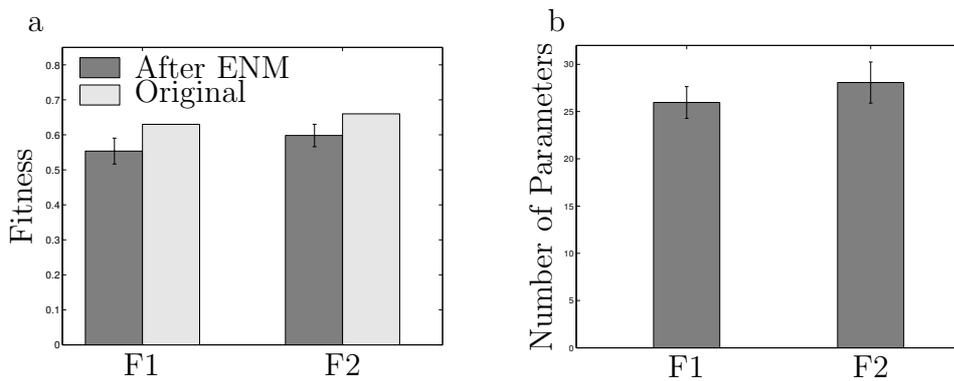


Figure 7: *Fitness and number of parameters of ENM minimized agents.* Mean and standard deviation across 10 ENM runs. (a) Fitness after minimization portrayed along with the fitness of the original agents. (b) Number of parameters retained after minimization. The original F1 and F2 agents both have 102 parameters (95 synaptic connections and 7 neuronal biases).

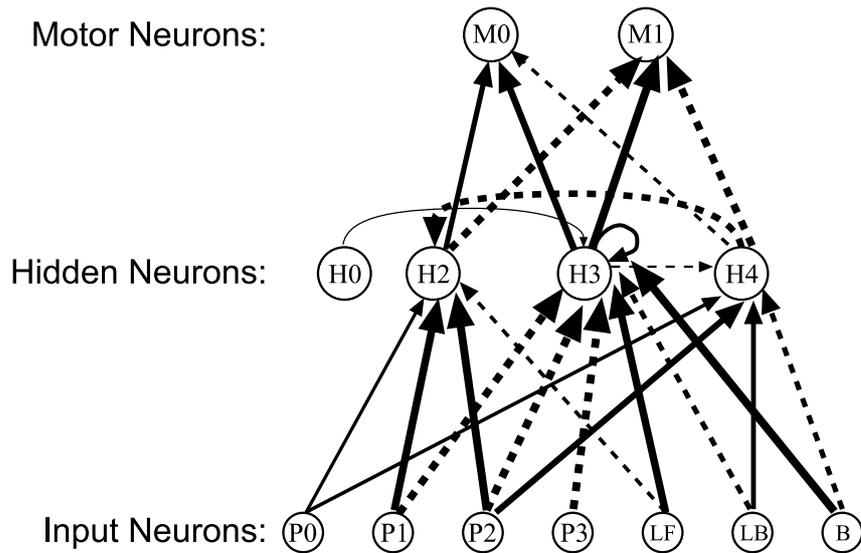


Figure 8: *The minimized neurocontroller of agent F1 (MF1), consisting of 24 synaptic connections. $M0$ and $M1$ are the motor neurons; $H0 - H4$ are the hidden neurons ($H1$ was removed during the minimization); $P0 - P3$ are the proximity sensors; LF is the Light Front sensor; LB is the Light Back sensor; B is the battery sensor. Dashed arrows represent inhibitory synaptic connections while solid arrows represent excitatory ones. The width of an arrow is proportional to the magnitude of its corresponding synaptic weight. Not shown in the figure are two biases, inhibitory to $H3$ and excitatory to $M1$. Also not shown are the six neurons removed from the network during minimization.*