A Novel Biologically Inspired Developmental Indirect Encoding for the Evolution of Neural Network Controllers for Autonomous Agents

Stefan Tsokov, Milena Lazarova, Adelina Aleksieva-Petrova

***Abstract* — Evolutionary algorithms provide the ability to automatically design robot controllers, but their wider use is hampered by a number of problems, including the difficulty of obtaining complex behaviors. This paper proposes a biologically inspired indirect encoding method for developing neural networks that control autonomous agents. The model is divided into three stages, the first two stages determine the structure of the network – the positions of the neurons and the network connectivity, and the third stage, occurring during the lifetime of the agent, determines the strength of connections based on the network activity. The model was tested experimentally by simulating an agent in an artificial environment, and the results of these simulations show that the method successfully evolved agents, capable of distinguishing between several types of objects, collecting some while avoiding others, without the use of a complex fitness function.**

***Index Terms* — artificial nervous system, autonomous agent, developmental representation, indirect encoding, neuroevolution**

I. Introduction

Designing a robot's control system is not an easy task because of the difficulty in predicting how the robot will interact with the environment, especially when the task and environment are complex. The use of automated methods for designing robot controllers not only facilitates the work of researchers and engineers, but also provides an opportunity to find "non-standard" ways of solving the problem that designers have not thought of. Evolutionary algorithms are one of the most popular methods used to automatically design robot controllers.

The field of evolutionary robotics faces a number of problems [1], one of which is the difficulty of obtaining complex behaviors, which in turn limits its application in real situations. There are a number of attempts to solve this problem such as the use of modules [2], [3], incremental evolution [4], [2], [5], the use of neuronal models with more complex dynamics, such as Continuous-time recurrent neural networks (CTRNN) [6] and spiking neural networks [7], open-ended evolution [5], as well as the use of indirect encoding [8], which allows the efficient evolution of more complex networks.

This paper proposes and experimentally tests a biologically inspired developmental model for the evolution of neural networks for the control of autonomous agents. The article is structured as follows – section II presents related works, section III describes the model, section IV describes the performed experiment, section V presents the experimental results, and section VI presents the conclusion.

II. Related Work

Cellular encoding [9] is an indirect method for encoding neural networks based on cellular duplication, that starts with a single cell and through a series of cell divisions and transformations results in a complete network. The genome is a list of trees describing developmental rules, such as cell division, adding and removing connections and changing the weights of the connections. A tree can be called by other trees, which makes it easier to evolve modular networks. The method successfully evolved a modular neural network that controlled the gait of a six-legged artificial insect.

In [10] a geometrically-oriented modification of [9] is proposed, in which by specifying the sensory neurons, the motor neurons, the precursor cells and their positions, a desired geometry can be introduced into the network. The method was used to evolve a simulated six-legged insect that can reach a source of odor by avoiding obstacles in its path.

The authors of [11] used a neural network developmental model that includes a cell division and migration phase and an axonal growth and branching phase, to evolve a simulated creature that must learn to eat food when hungry and drink water when thirsty. The evolved networks show a modular structure with two modules – one for food and one for water.

The authors of [12] evolved stimulus-avoiding agents as well as agents that follow a curve, using a model based on genetic regulatory networks represented by a random Boolean network, which determines the cellular division and axonal growth.

The authors of [13] propose a model in which the development of the network occurs during the life of the organism, by controlling the growth of the axon using the variability in the activation of the neuron. The authors showed experimentally that organisms that had evolved in one environment were capable of adapting to a different environment.

The authors of [14] propose an evolutionary model in which a growth phase of the connections between neurons alternates with a phase of generating spontaneous activity, which in turn regulates subsequent growth steps. Excess connections are removed after all growth is completed.

III. Proposed Model

In nature, the development of the nervous system is influenced by the environment [15], thus facilitating

https://doi.org/10.47978/TUS.2021.71.01.005

evolution and allowing for the ability to adapt to a particular environment, as it is not necessary to genetically encode the entire nervous system in all its detail, but only a rough template that is further refined through interactions with the environment. The neural development begins with the birth of neurons, their migration and the formation of a large number of synapses, followed by an activity-dependent synaptic pruning [16].

Based on this, a model for the development of an artificial nervous system for autonomous agents is proposed, in which the development can be divided into three phases. In the first, neurons are generated and migrate, in the second, connections are formed, and in the third phase, which takes place during the lifetime of the agent, the connections are modified depending on the activity of the network. The neuronal migration stage is inspired by the development of the cerebral cortex, which is formed from the inside out in layers by successive waves of migrating neurons [17].

The proposed model does not aim to accurately model biological processes, but is an abstraction of some of their features for computational applications.

The neural networks are located in a 2D plane and are encoded indirectly as a sequence of neuronal subpopulations that are placed sequentially in the plane one after the other, i.e. the neurons from the first subpopulation are placed first at the beginning of the horizontal axis, after that the neurons from the next subpopulation are placed to the right of them, and so on.

A. Genome

The genome is of variable length and represents a sequence of neuronal subpopulations, with a minimum length of 2, as the first subpopulation is always interpreted as the input neurons and the last as the output neurons.

Each neuronal subpopulation (except the first and last) is defined by the following parameters – number of neurons, neuronal personal space, four vertices of a quadrilateral, maximum length of the axon, activation threshold and learning rate. The first subpopulation is encoded by neuronal personal space, maximum axonal length, activation threshold, and learning rate, and the last one by neuronal personal space and activation threshold. The neuronal personal space determines the minimum distance between two neurons from the same subpopulation, at which they will not interfere with each other, i.e. the two neurons are thought to overlap if they enter each other's personal spaces. The coordinates of the four vertices determine the area of a quadrilateral in which the neurons can be located, and the maximum length of the axon determines the maximum distance at which neurons from this subpopulation will form connections with other neurons. The activation threshold and learning rate are used to modify the weights depending on the neuronal activity (for more information see section III *D*).

All values in the genome (except for the number of neurons) are normalized from 0 to 1, and when the network is constructed the actual values for the respective subpopulation are obtained by multiplying these values by the number of neurons in the subpopulation, with the exception of the values for the activation threshold and the learning rate which are left within the range from 0 to 1.

B. Placing and moving neurons in a subpopulation

The neurons of the first subpopulation are considered to be the input neurons and are placed at the beginning of the horizontal axis, one on top of the other at a vertical distance from each other determined by the real value of their personal space (normalized value multiplied by the number of neurons in the subpopulation) so that they do not enter each other's personal spaces. The neurons from the next subpopulation are placed to the right of the input neurons at a minimum possible horizontal distance – 0.1*number of neurons of the input subpopulation.

For intermediate subpopulations, the first four neurons are placed in the four vertices of the quadrilateral describing the shape of the subpopulation. Each subsequent neuron is placed at the center of the subpopulation at this time: $(x_c, y_c) = (\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i)$, where n are the neurons placed so far.

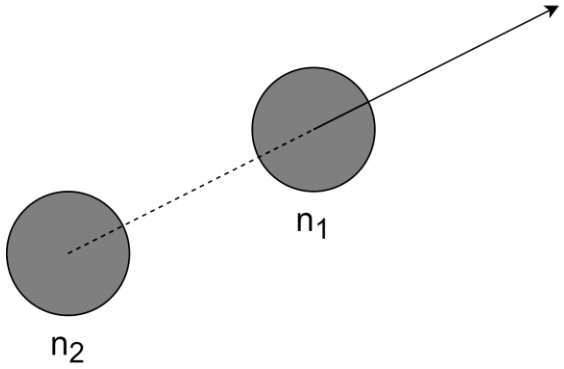

Fig. 1. Neuron n_2 acts on neuron n_1 with a pushing force because n_1 has entered its personal space.

Each time a new neuron is placed, a check is performed to determine whether there are any overlapping neuron pairs (neurons that enter each other's personal space) and an attempt is made to resolve the overlap by moving the neurons. Attempts to resolve overlaps are made until a maximum number of steps is reached (in the experiment, the maximum number of steps is set to 10) or when there are no more overlapping pairs. When two neurons get close enough to each other (enter each other's personal space), they begin to push each other, i.e. when a neuron enters the personal space of another neuron, the second neuron acts with a repulsive force on the first one trying to push it out of its personal space in a direction determined by the line connecting the two neurons (Fig. 1) and magnitude of the force: $|F| = 2 * personalspace - d$,where d is the distance between the two neurons. When a direction cannot be determined, i.e. the coordinates of the two neurons are the same, the force is directed parallel to the horizontal axis, and if the neuron on which the force acts is older than the other one, the direction is to the left, otherwise – to the right.

The displacement of a neuron is determined by the net force acting on it as a result of all the other neurons with which it overlaps, i.e. has entered their personal spaces (Fig. 2). The neurons placed at the vertices of the quadrilateral cannot be moved. When a neuron moves outside of the quadrilateral, it is returned inside by generating a mirror image of its coordinates relative to the wall of the quadrilateral which it had crossed to go outside.

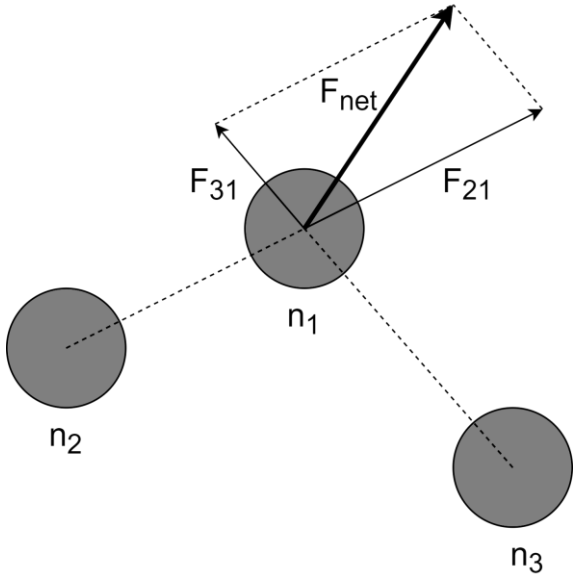

Fig. 2. Neuron n_1 has entered the personal spaces of neurons n_2 and n_3, which begin to push n_1 with forces F_{21} and F_{31}, respectively. The final displacement is determined by the sum of these forces – F_{net}.

Each subsequent neuronal subpopulation is placed after the previous one relative to the horizontal axis, i.e. to the right, with a minimum possible horizontal distance from the rightmost neuron from the previous subpopulation of 0.01*number of neurons of the previous subpopulation.

The last subpopulation represents the output neurons, which are placed similarly to the input ones – one on top of the other at a vertical distance so that they do not enter each other's personal space.

C. Connecting the neurons

After the completion of the neuronal placement phase, the network connectivity is determined. Each neuron connects to all neurons to its right that are within its range, i.e. are at a distance less than the maximum length of its axon. The initial values of the weights are determined by the distance between the two neurons: *1/d*.

After the connection phase is complete, those parts of the network that do not contribute to the output, as well as those that do not receive input information, are removed.

D. Using network activity to train the connection weights

The weights of the connections of the neurons are modified based on the activity of the network during the lifetime of the agent. The presynaptic rule is used to modify them – the weight changes only when the presynaptic neuron is active (has an output above the activation threshold), if the postsynaptic neuron is also active, the weight increases, otherwise it decreases:
$w_{ij}^{t} = w_{ij}^{t-1} + \eta \Delta w_{ij}, 0 < \eta \leq 1$, where $\Delta w_{ij}$is +1 or -1 depending on whether the postsynaptic neuron is active or not, η is the learning rate. Weights can have values between -1 and 1.

IV. EXPERIMENT

The model was tested experimentally by simulating an agent in an artificial world, the experiment was performed on a system with Intel Core i7-9750H CPU @ 2.60GHz and 16 GB RAM.

A. Simulated World

The artificial world is a square 2D arena composed of cells (squares) (Fig. 3). The arena is completely closed on all sides with walls. In addition to the walls that occupy the outermost squares, there are two types of objects – food and poison. The agent has orientation and energy, and can sense what object is directly in front of it, as well as how "tired" it is, i.e. its available energy.

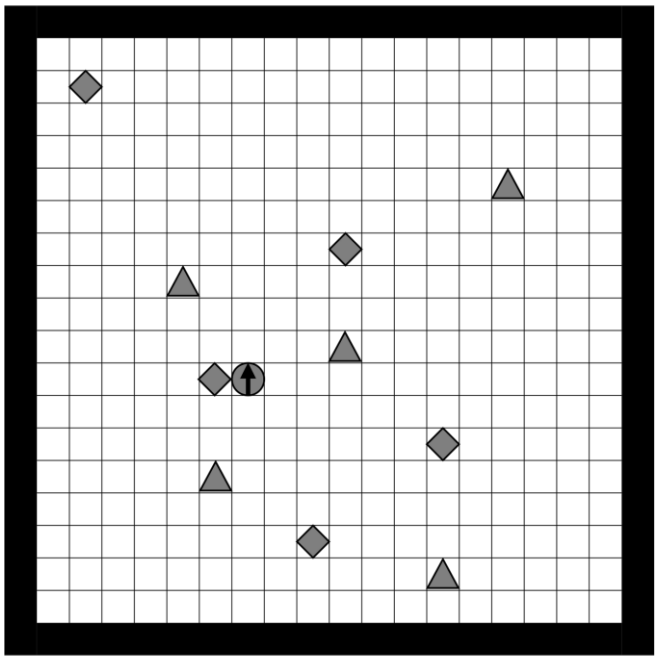

Fig. 3. A sample world of size 20x20 with 5 food items and 5 poison items. The agent is represented by a circle (the arrow indicating its orientation), the food is represented by a triangle, and the poison by a rhombus. The filled squares represent the walls.

The agent is capable of three actions – turning clockwise, turning counterclockwise and moving one position forward (in the direction of its orientation). Any object, including the walls, can decrease or increase the energy of the agent when it is encountered (moving on top of the object or in the case of a wall – an attempt to move). When the agent eats food, its energy increases, when it eats poison – it decreases, and the energy also decreases when the agent collides with a wall. For each unit of time elapsed, the energy of the agent decreases, and in addition to that, each action leads to a further reduction in energy. The agent must learn to explore its environment without bumping into walls and to learn which objects are good and which are bad, i.e. to eat food but avoid poison.

B. Genetic Algorithm

A genetic algorithm is used to evolve the neural network that controls the agent in the environment, or more precisely, the number of neuronal subpopulations and their parameters are evolved.

In the experiment, a population of 100 individuals was evolved over 200 generations. A tournament selection was used for the selection of parents, with a tournament size of 2. Elitism was used, and the 2 best solutions from the previous generation were added directly to the next population. The number of neurons in the input subpopulation and the number of neurons in the output

subpopulation were fixed. For the input one they were 5, one for each type of sensory information – no object (empty square), wall, food, poison, internal energy. For the output subpopulation, there were 3 neurons, one for each type of action – clockwise rotation, counterclockwise rotation and forward movement. The output neuron with the highest activation determines which action will be taken. Noise with normal distribution with $\mu = 0$ and $\sigma = 0.05$ was added to the activations of the input and output neurons. The sigmoid function was used to obtain the output signal of the neurons.

C. Initial population

The initial population consists of random individuals with an input and an output subpopulation and any number of intermediate (between the input and the output one) subpopulations in the range between 0 and 10, each with a random number of neurons in the range between 1 and 10.

D. Crossover

As in [18], the probability of crossover is related to how similar the two individuals are in terms of the number of neuronal subpopulations: $p = 1 - \frac{|l_1 - l_2|}{\max(l_1, l_2) - 1}$, where l_1 and l_2 are the lengths (number of neuronal subpopulations) of the two parent solutions. A minimum crossover probability of 0.05 was set. A modified single-point crossover for solutions of different lengths [18] was used, in which each parent has a seperate crosover point, the one for the shorter solution is determined by scaling the selected crossover point for the longer one so that the two points have the same relative positions in the two solutions. When no crossover takes place, the children are just copies of the two parents.

E. Mutation

After crossover, each of the two newly obtained solutions (children) can be mutated with a certain probability. The value of this probability starts from 0.9 in the first generation and decreases with each subsequent one by 0.01 until reaching a minimum probability of 0.2. There are three possible mutations:

- *Modifying a subpopulation* – the value of one of the parameters of a randomly selected neuronal subpopulation is modified. When modifying the vertices, one of the coordinates of a randomly selected vertex is modified. The number of neurons in the intermediate subpopulations can be increased or decreased by one (with a minimum allowed number of 1 neuron in the subpopulation), and the remaining parameters can be mutated by adding noise with a normal distribution with $\mu = 0$ and $\sigma = 0.01$.
- *Adding an intermediate subpopulation* – a new (randomly generated) intermediate subpopulation is added at a specific location in the solution between the input and the output subpopulations.
- *Removing an intermediate subpopulation* – one of the existing intermediate subpopulations is selected and removed.

The probability of modifying a subpopulation is 50%, the removal and the addition of an intermediate subpopulation are equally likely.

F. Fitness function

The performance of the network is assessed by placing the agent (with random orientation) in a random position in the world and leaving it to perform actions for a certain period of time or until the agent has run out energy. In order to avoid the introduction of additional bias, the simplest possible fitness function is used, which is determined by how long the agent has managed to keep its internal energy above a certain energy threshold (i.e. how long the agent has been in good "health"). The size of the world is set to be 20x20, with 100 food items and 100 poison items randomly placed at each initialization. The agent always starts with 1000 units of energy, the energy threshold is set to 500. For each time step of the simulation the energy of the agent is reduced by 1, and when an action is performed the energy is further reduced by 10. When ingesting food, the energy of the agent increases by 400, in case of ingestion of poison the energy decreases by 400, and in case of a collision with a wall the energy decreases by 200. The maximum number of time steps for quality assessment is 1500.

Due to the stochastic nature of the world generation (food items and poison items are initialized at random positions) and the initial placement and orientation of the agent, the evaluation is performed 10 independent times with different randomly initialized worlds, these 10 worlds (including the starting position of the agent and its orientation) are the same for all individuals. The fitness function is:

$$fitness = \frac{1}{n}\left(\sum_{i=1}^{n} fit_i\right) - \alpha \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} \left(fit_i - \overline{fit}\right)^2}$$

where n is the number of evaluations. Successful individuals are not only those that show good average performance, but also ones that show consistency between the evaluations. The parameter α was set to 0.1. If one of the intermediate subpopulations is shaped by a crossed quadrilateral, the individual's performance is not evaluated and it is given a fitness of 0. When the center of the intermediate neuronal subpopulation is outside the quadrilateral, the fitness is also 0.

V. RESULTS

Due to the stochastic nature of the algorithm, the experiment was repeated 3 times, one run taking on average around 18.5 hours. Table I lists the genomes of the best individual of the last generation for each experimental run.

Figures 4-6 show the constructed neural networks of the best individuals after removing the excess parts. Input neurons are shown in blue and the output neurons – in orange. From the bottom up, each input neuron represents information about an empty square, a wall, a food item, a poison item and internal energy, respectively. Again, from the bottom up, each output neuron is responsible for the following actions – turning clockwise, turning counterclock-wise and moving forward, respectively.

TABLE I
THE BEST GENOME OF EACH RUN

Run	Subpopulation	Genome
1	1	[0.3537433688394607, 0.3475420898389171, 0.4151090370591758, 0.8419326101666478]
	2	[1, 0.1582100867781533, [[0.7006565490373394, 0.6103082106656933], [0.581796878365856, 0.37042622915757495], [0.5414104758227963, 0.9092641150092111], [0.5633864207692753, 0.8427584874220297]], 0.7928915002854481, 0.783577013841125, 0.6358753118520005]
	3	[0.4867450286691157, 0.42427574085087977]
2	1	[0.6075084990678143, 0.8128391036963282, 0.28948784454879833, 0.2882099985055757]
	2	[3, 0.4866316784971686, [[0.8606437748466655, 0.13740843357560717], [0.4675881117879297, 0.6203542772742386], [0.02702814608933235, 0.06441176767996604], [0.3313173733960509, 0.03238670717154912]], 0.33978593718623096, 0.6771205787451327, 0.4363457951423555]
	3	[1.0, 0.0504312514750648]
3	1	[0.15814631702005488, 0.277922483013492, 0.8543802425120453, 0.477519802746862]
	2	[6, 0.5438169103208217, [[0.03862065744047294, 0.8069786783406282], [0.3740547063910148, 0.5139355081411615], [0.27081966217606757, 0.5674932988411782], [0.6564083293575926, 0.36205149468600506]], 0.8800661379498583, 0.7871085047283626, 0.0]
	3	[0.017519626872972642, 0.2858690602994516]

Genomes are a list of neuronal subpopulations. For the first subpopulation, the parameters are - neuronal personal space, maximum axon length, activation threshold and learning rate. For the last subpopulation the parameters are neuronal personal space and activation threshold. For the intermediate subpopulations the parameters are number of neurons, neuronal personal space, vertices of a quadrilateral, maximum axon length, activation threshold and learning rate.

From these figures it can be seen that in two of the runs the algorithm evolves similar simple structures. In both cases, neural networks ignore input information about poison objects and internal energy. The input neuron, activated in the presence of food in front of the agent, is directly connected to the motor neuron responsible for moving forward. The input neuron, activated in the presence of a wall, is directly connected to the motor neuron responsible for counterclockwise rotation and the input neuron, activated when there are no objects in front of the agent is connected to the motor neuron responsible for clockwise rotation. In the third run, the best neural network only takes into account the input information about poison objects and ignores everything else. None of the best neural networks use information about the internal energy of the agent.

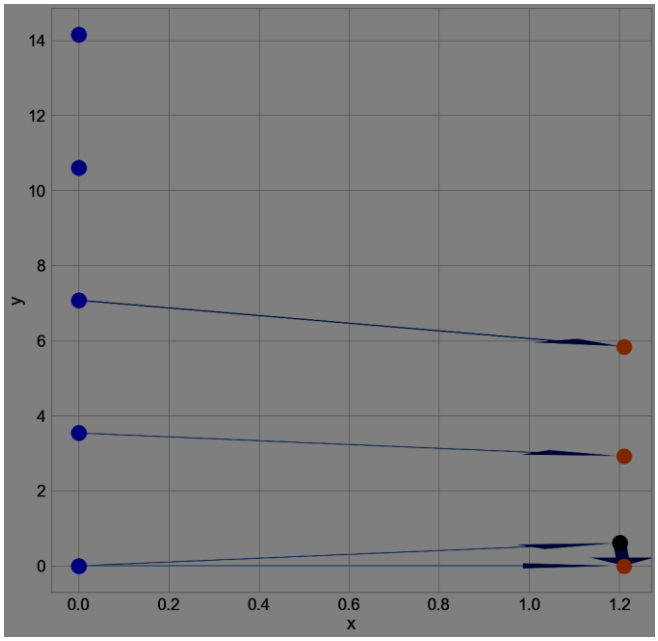

Fig. 4. The resulting neural network from the best individual of the last generation of the first run of the experiment. The input neurons are shown in blue and the output neurons in orange.

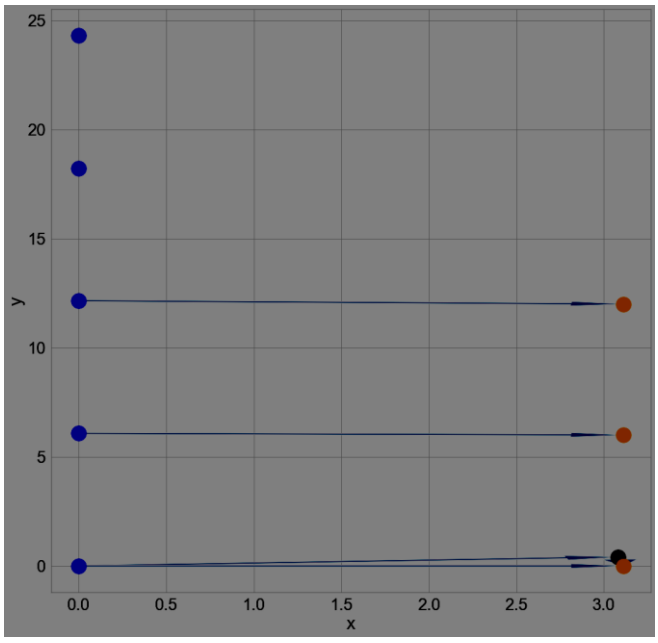

Fig. 5. The resulting neural network from the best individual of the last generation of the second run of the experiment. The input neurons are shown in blue and the output neurons in orange.

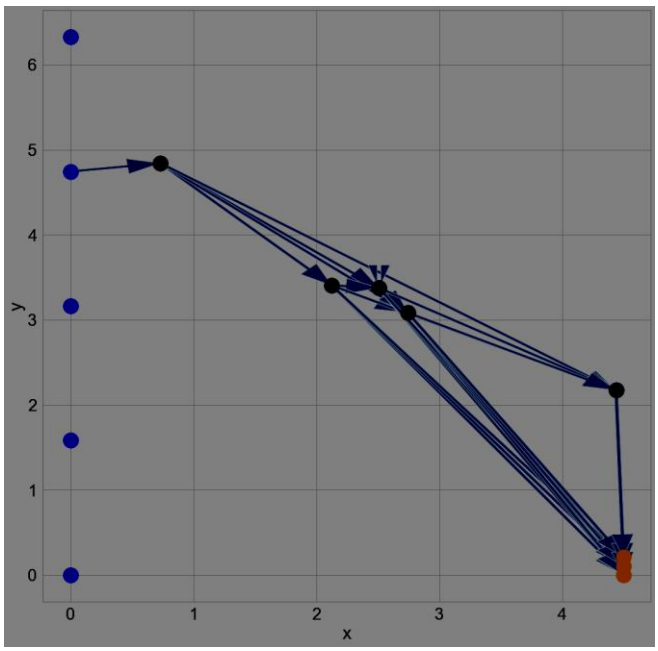

Fig. 6. The resulting neural network from the best individual of the last generation of the third run of the experiment. The input neurons are shown in blue and the output neurons in orange.

Agents controlled by the best networks of the first two runs show similar behaviors by eating the food in front of them and avoiding the walls – whenever there is food in front of them, they move towards it, when they are facing a wall they always turn counterclockwise and when there is no object in front of them, they most often move forward, turning from time to time. These agents show indifference to poisonous objects. The agent controlled by the best network of the third run successfully avoids all poisonous objects by turning clockwise, but as expected shows indifference to all other objects.

Figures 7-9 show the best fitness and the average fitness of the population for each generation for the three experimental runs. They show that fitness increases in the process of the evolutionary search, both for the best individuals and for the entire population.

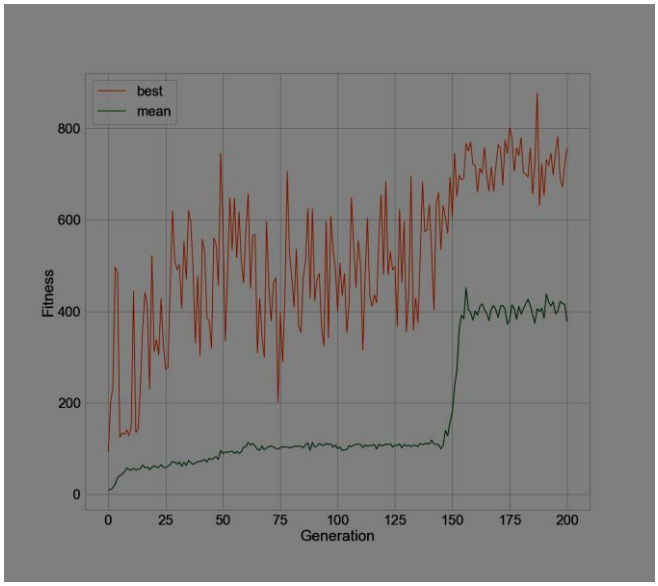

Fig. 7. The best fitness and the average fitness per generation of run 1.

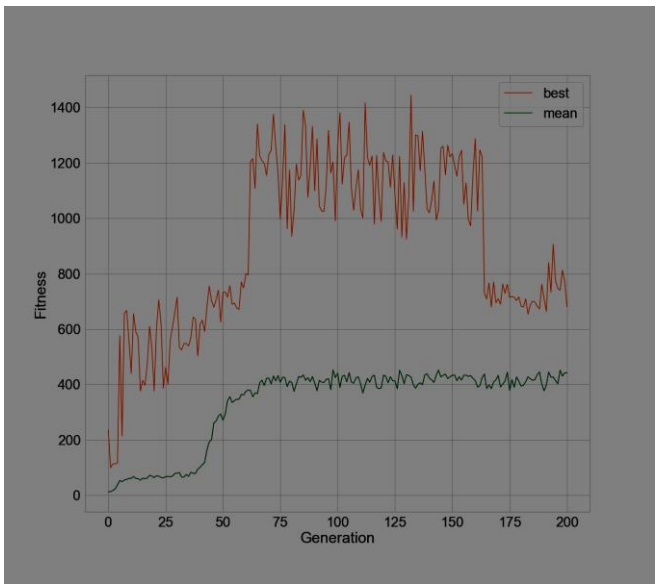

Fig. 8. The best fitness and the average fitness per generation of run 2.

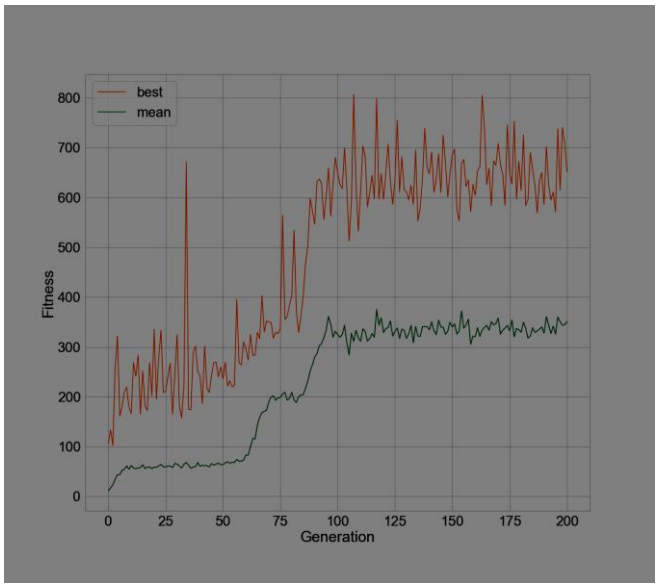

Fig. 9. The best fitness and the average fitness per generation of run 3.

VI. Conclusion

A biologically inspired indirect encoding scheme for the evolution of neural networks for autonomous agents was proposed, in which the neural networks develop and subsequently, during the agent's lifetime, are tuned by its interactions with the environment. The model was tested experimentally on a simulation of an artificial world, and the results show that by using a simple implicit fitness function, agents capable of distinguishing between different objects can be successfully evolved. Future work will be to test the model on a simulation of a real environment and robot, as well as a subsequent test on a real physical robot.

Acknowledgment

The work presented in the paper is supported by research grant 202ПД0002-09 financed by the Research and Development Sector of TU-Sofia.

REFERENCES

[1] Eaton, M. (2015). Evolutionary Humanoid Robotics. Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-662-44599-0

[2] de Garis, H. (1991). Genetic Programming: GenNets, Artificial Nervous Systems, Artificial Embryos. PhD thesis, Université Libre de Bruxelles, Belgium, https://doi.org/10.1016/B978-0-444-89178-5.50068-3

[3] Mouret, J. B., Doncieux, S., & Meyer, J. A. (2006, September). Incremental evolution of target-following neuro-controllers for flapping-wing animats. In International Conference on Simulation of Adaptive Behavior (pp. 606-618). Springer, Berlin, Heidelberg, https://doi.org/10.1007/11840541_50

[4] Auerbach, J. E., & Bongard, J. C. (2011). Evolving monolithic robot controllers through incremental shaping. In New Horizons in Evolutionary Robotics (pp. 55-65). Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-18272-3_5

[5] Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. Evolutionary computation, 19(2), 189-223, https://doi.org/10.1162/EVCO_a_00025

[6] Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. Adaptive Behavior, 3(4), 469-509, https://doi.org/10.1177/105971239500300405

[7] Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. Neural networks, 10(9), 1659-1671, https://doi.org/10.1016/S0893-6080(97)00011-7

[8] Silva, F., Correia, L., & Christensen, A. L. (2016). Evolutionary robotics. In Evolutionary Robotics (No. 7). Scholarpedia, https://doi.org/10.4249/scholarpedia.33333

[9] Gruau, F. (1994). Automatic definition of modular neural networks. Adaptive behavior, 3(2), 151-183, https://doi.org/10.1177/105971239400300202

[10] Kodjabachian, J., & Meyer, J. A. (1998). Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects. IEEE transactions on neural networks, 9(5), 796-812, https://doi.org/10.1109/72.712153

[11] Cangelosi, A., Parisi, D., & Nolfi, S. (1994). Cell division and migration in a 'genotype'for neural networks. Network: computation in neural systems, 5(4), 497-515, https://doi.org/10.1088/0954-898X_5_4_005

[12] Dellaert, F., & Beer, R. D. (1996). A developmental model for the evolution of complete autonomous agents. In Proceedings of the fourth international conference on simulation of adaptive behavior (pp. 393-401). Cambridge, MA: MIT Press.

[13] Nolfi, S., Miglino, O., & Parisi, D. (1994, September). Phenotypic plasticity in evolving neural networks. In Proceedings of PerAc'94. From Perception to Action (pp. 146-157). IEEE.

[14] Rust, A. G., Adams, R., George, S., & Bolouri, H. (1997). Activity-based pruning in developmental artificial neural networks. In Proc. of the European Conf. on Artificial Life (ECAL'97) (pp. 224-233).

[15] Greenough, W. T., Black, J. E., & Wallace, C. S. (1987). Experience and brain development. Child development, 539-559, https://doi.org/10.2307/1130197

[16] Collin, G., & Van Den Heuvel, M. P. (2013). The ontogeny of the human connectome: development and dynamic changes of brain connectivity across the life span. The Neuroscientist, 19(6), 616-628, https://doi.org/10.1177/1073858413503712

[17] Ayala, R., Shu, T., & Tsai, L. H. (2007). Trekking across the brain: the journey of neuronal migration. Cell, 128(1), 29-43, https://doi.org/10.1016/j.cell.2006.12.021

[18] Tsokov, S., Lazarova, M., & Aleksieva-Petrova, A. (2021). An evolutionary approach to the design of convolutional neural networks for human activity recognition. Indian Journal of Computer Science and Engineering, 12(2), 499-517, https://doi.org/10.21817/indjcse/2021/v12i2/211202145