Neuro-Evolved Cellular Automata

Emil Fazzi, emil.fazzi@studenti.unitn.it

***Abstract*—This Project aims to explore the field of Cellular Automata starting from the well-known Convey's model "Game of Life". The goal of this research project is to understand and study the mechanics of such systems controlled by a set of simple rules composing a very chaotic and complex behaviour. The main idea is to insert the neuro-evolution into a Cellular Automata system in order to make the shallow neural network "learn" basic rules to drive the Cellular Automata system towards new mechanics and patterns.**

***Index Terms*—Cellular Automata, Neuro-Evolution, Conway's Game of Life**

I. Introduction

CELLULAR automation is a discrete model of computation wich have found application in several areas such as physics, mathematics, theoretical biology and graphics. In this paper we will focus the attention on the application of the cellular automata and neuro-evolution algorithms for study purposes and for graphical applications, exploring different scenarios and behaviours of the system.

The following introductory section first describes the Cellular Automata model in detail describing the rules at the basis of the system; then, we will define and describe the Neuro-Evolution algorithm and its role inside the cellular automata system. Finally, we will explain the motivations behind the project as well as some possible applications.

The next section deals with the actual experimentation, hence formally defining the metrics for fitness evaluation and Neuro-Evolution parameters as well as the motivations behind the adopted solutions, concluding with the respective results. In the last section we draw some conclusions about the results and the study path that guided us to the final version of the system, additionally proposing some guidelines for future works.

A. Cellular Automata

The Cellular Automation model is composed by a finite grid of cells in an N-dimensional space in wich each cell can be found in a finite number of states. The general rule that describes cellular automata systems is that the behaviour of each cell will be determined by simple rules that consider the states of the neighborhood of that cell. The system will evolve over time by applying for each generation the same simple rules to every cell in the grid and updating the state of every block, resulting in a complex overall behaviour created by simple interactions and rules among a considerable number of cells.

The specific case of Cellular Automata System in we will focus our research will be a 2-dimensional case with cells that can be found in only two different states: Dead or Alive. Given the definition of Cellular Automation system, we need a set of rules that will define whenever a cell in the grid will be dead or alive in the next iteration of the process, based on the neighbor cells.

Manuscript received November 09, 2022; revised November 09, 2022.

Conway's Game of Life

An example of 2-State, 2-dimensional orthogonal grid Cellular Automata algorithm is Conway's Game of Life: a simple and amazing algorithm created by the British mathematician Jorn Horton Conway in 1970. The neighborhood is described as a 3x3 matrix around the selected cell in the grid (8 neighbor cells) and the rules that determine if the selected cell will be dead or alive in the next iteration are the following:

- Any live cell with fewer than two live neighbours dies, as if by underpopulation.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any live cell with more than three live neighbours dies, as if by overpopulation.
- Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Given the outcome of those simple rules, each cell in the grid will survive, born or die in the next iteration of the process. The amazing mechanic of this quite simple algorithm is the complexity that emerge by the fact that the state of cells at each teration will strongly depend by the previous state. This make the system very difficult to predict even if the initial state and the evolution rules are known.

The proposed project will be based on the Conway's Game of Life as starting point to develop different mechanics and evolutionary rules such as the insertion of the Neuro-Evolution algorithm.

B. Neuro-Evolution

Neuroevolution, or neuro-evolution, is a form of artificial intelligence that uses evolutionary algorithms to generate artificial neural networks (ANN), parameters, and rules [1]. In contrast to other learning algorithms such as supervised learning algorithms that need a ground-truth for every training input, neuro-evolution uses evolutionary algorithms, that only need a fitting function, to set weights, structure and rules of a given artificial neural network. This way the learning process utilizes the fitness score to measure "how good" is the ANN and to evolve all the parameters in the next generations. In the proposed method we will use a specific neuro-evolution algorithm that exploits a simple evolutionary computation which genotype aims to improve the weights of a fixed-structure Feedforward Neural Network (FFNN).

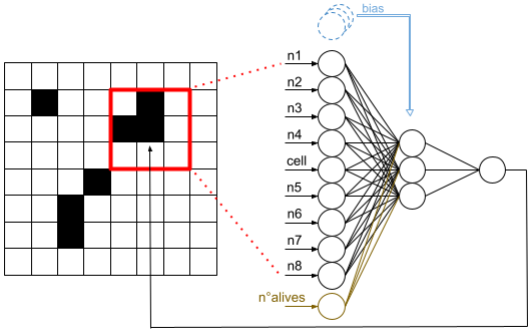

Fig. 1. Scheme of the Neuro-evolved Cellular automata

C. Neuro-Evolved Cellular Automata

The main characteristic of this project is the combination of the Cellular Automata model with the Neuro-Evolution algorithm. This union is ment to create new mechanics and behaviours to the analyzed Cellular Automata system, exploring the results given by the interaction between the two models.

As is shown in Fig. 1, the FFNN is fed with the status of the neighbor cells of every alive cell in the grid of the Cellular automata system. The Network has 10 inputs in total and only one hidden layer, making it a shallow network, composed by three hidden neurons. The output of the network, having a binary output given by a threshold value, will decide the status of the selected cell in the next iteration.

D. Motivations

The motivations behind the choice of this study are the curiosity and the idea of the interaction between two quite simple models that can potentially create brand new complex behaviours to study, rising a novel research branch that could potentially give inspirations in other studies that analyse cellular automata systems for applications in fields like biology, finance and graphics video effects. The proposed project takes as starting model Conway's Game of Life: a famous and fascinating algorithm because of its very simple rules and the extremely complex behaviour that creates. The project aims to create a similar result using a different and dynamic criteria for the rule management of the system, aiming to create a continuous graphical illusion that can be compared with the optical effect of the evolving cells in the Game of Life model.

The main idea of making a continuous system, a cellular automata model driven by rules that make it never ends, comes from a personal project of a low visual quality (12x12 pixels) DIY LED panel designed for simple graphical animations. The implementation of Conway's Game of Life on that DIY project was motivated by the fact that is very hard to mathematically code good animations into simple programming language for microcontrollers with limited hardware resources: given the nature of simple rules and implementation of Life algorithm, is easy to transpose all the rules into effortless commands, making a simple and light implementation on a little microcontroller that will be able to create complex and fascinating graphical animations. The algorithm implemented by Jorn Horton Conway itself was not good-suited for that application though: given the little space grid of the panel, the visual effect stops after few iterations because of repetitive patterns or the death of all the cells in the grid. The development of the proposed project will also try to solve this problem, creating a continuous visual effect on a small grid that will be utilized on the previously built DIY project.

II. Experimentation

The next section will describe in details the project and the process that led to the final version of it, taking Conway's Game of Life as the State of the Art and evolving and implementing the algorithm adding the neuro-evolution system. After a brief description of the architecture of the final version of the system, the attention of this report will move to the testing phase of the project describing the implementation of the algorithm and its validation algorithm and parameters and fitness criteria used in the evolutionary computation.

A. Starting point: Game of Life

As mentioned before, the Life algorithm has been a starting point for the development of this project because of its success and particular behaviour. The first logical step is to recreate the Game of Life behaviour on the grid by using the FFNN, setting the structure of the network and the weights in order to artificially recreate the same result.

Being the basic rules of the Life algorithm linear, this will be a possible task for the network by implementing only one hidden layer: First of all, the Life algorithm does not include the information about the relative position of the neighbor cell with respect to the selected central cell, that makes all the weights set to either 1 or 0.

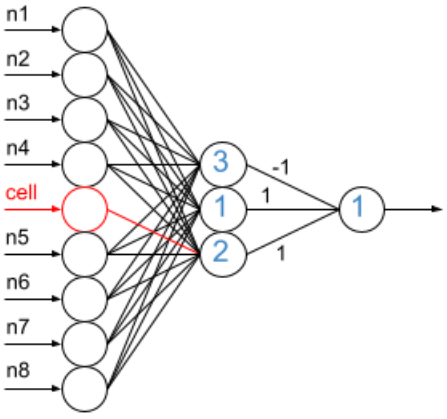

Fig. 2. Scheme of the FFNN settled to recreate the behaviour of the Life algorithm

To find a method to insert the Life rules into the network, we created one hidden neuron for every Life rule, as shown in Fig 2: Starting from the hidden neuron on the top

- The first neuron ensures that the cell will die by overpopulation, if the number of neighbors will be more than 3.

- The last neuron activates if the analyzed cell is alive and if has more than one neighbor, preventing a cell to born if has less than 2 neighbors and allowing cells with more than 1 neighbor to live.
- The neuron in the middle is used in the case when the selected cell is dead but has exactly 3 neighbor cells, making it live in the next generation

Once that we stated that the Life algorithm is possible with a Feedforward Neural Network, we added another extra input to the network to make the current setting work endlessly. A way to implement this behaviour is to add the information about the number of the current number of alive cells in the grid. This way the network can be tuned during the iteration process and ”kill” less cells if there are few alive cells.

B. Architecture

The final version of the system is composed by a grid of 12x12 cells, motivated by the personal application of the project and to prevent memory and resources overloading in the training phase of the network. The FFNN, described in its full version in the Fig. 1, has 10 inputs and 3 hidden neurons with a binary output, making it have a total of 36 parameters including weights and biases to be trained by the evolutionary algorithm. The learning algorithm uses an evolutionary computation with the parameters displayed in the table I, selected in order to maximize the efficiency and accomplish the best result in the lower number of evaluations.

TABLE I
PARAMETERS OF THE EVOLUTIONARY ALGORITHM

N population	50
N generations	20
Tournament Size	4
Mutation rate	0.3
Crossover rate	0.8
Selection size	50
N Elites	2

While the Evolutionary algorithm is developed in python using the inspyred library, the evaluation program designed to try the network configuration during the learning phase is coded in C++ for a better resource management and to maintain a likelihood between the evaluation algorithm and its application on the LED panel. The evaluation environment is composed by a 12x12 grid surrounded by dead cells: the trial is executed two times with random-initialized cells in order to avoid rare cases for specific particular starting configuration and only the output statistics of the case with the lowest number of total iterations is selected. In order to avoid infinite-iteration trials, the maximum numbers is limited to 5000 iterations for the cellular automata execution, making a value of 5000 the best in terms of fitness evaluation. Finally, the criteria that make the execution stop (before the limit of 5000 iterations) is the death of every cell in the grid or the repetition of a maximum of 5 iterations: preventing infinite looping through the same states.

C. fitness function

For the evaluation of the fitness of the FFNN on the trial algorithm of the cellular automata, a multi-objective fitness function is selected. The fitness function is combined with the a priori scalarization method, making it comprehend the following parameters:

- **N. Iterations:** Number of the total iterations of the trial. (0 - 5000) As mentioned before, only the statistics of the execution with the lower value is selected for the fitness evaluation
- **N. Particles:** This value calculates the average number of alive cells during the execution. (0 - 144)
- **N. changed:** Average number of cells that changed between one iteration and the following. (0 - 144)
- **Aggregation:** This value weights the concentration of cells in groups inside the grid. (0 - 12)

Every parameter is normalized in a range between 0 and 100 with 0 indicating the desired value for that parameter (e.g. 5000 for the N. Iterations). All the four parameters are summed together with the others without any weighting, making the values of the total fitness space between 0 and 400. While the first three parameters are straightforward in the mathematical formulation and meaning, the **Aggregation** parameter was added only in a second phase of the development of the algorithm: The experimentation on the algorithm using only the first three parameters was succesful, but there was something missing. The result was respecting the requirements in terms of fitness regarding the endelss behaviour, average number of cells and average number of changing cells but was still very different from the Life algorithm behaviour. In fact, the final optical behaviour result in few particles moving (mainly in a specific direction) uniformly scattered on the grid, developing a simple behaviour compared with the Game of Life. The main difference between the two results was the fact that, in Life, all the cells are grouped in small clusters moving independently from the others: the missing part on the project was a parameter in the fitting function that measures this particular behaviour of the cells in the grid. To overcome this insufficiency, the **Aggregation** parameter calculates the way cells are scattered in the grid with a sliding window over the 12x12 space, collecting the total number of cells in the window for every point and calculating the standard deviation of the final vector. This way a value of 0 represent a uniform distribution of the cells in the grid while a value close to 12 indicates a high concentration of cells in groups

D. Results

The final version of the system led to satisfying results, showing a good complexity in terms of behaviour and graphical effect. Given the nature of the project, there was no unique solution to the analysed problem: the result was different at every trial of the learning process, showing in any case the required characteristics expressed in the fitness function. During the training phase, the convergence to a good solution was found in a good number of evaluations, demonstrating the efficiency of the selected evolutionary algorithm and the tuned parameters. Being the values of the fitness function selected

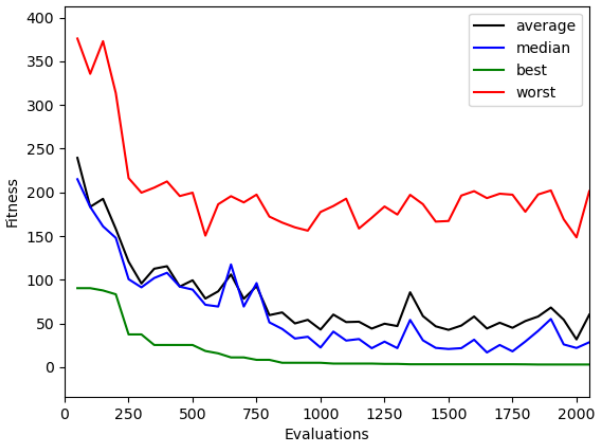

Fig. 3. Learning trend up to 2000 evaluations by one of the training trials

a priori by the user and the highly ill-posed fitness landscape showing a lot of possible optimal solutions, the priority of the algorithm was not the absolute precision of the optimal solution. Instead, the goal of the evolutionary algorithm is the observance of the required parameters that need to match as good as possible with the behaviour of the trained system without any ideal match.

During the testing phase, a clear result was the fact that the structure and the parameters used to build the fitness function highly affect the outcome of the training phase in terms of behaviour of the cellular automata. In fact the result hugely depends on the user choices in terms of average number of cells, average number of changing cells and desired aggregation value. This due to the fact that we are not searching for a specific optimal solution but looking for a class of desired behaviours of the algorithm that depends by our choice.

The algorithm was succesfully applied also in the personal DIY LED panel, showing a satisfying visual effect. Im fact, the system after the training phase require minimal hardware resources and is able to work without any problem also on economic embedded devices such Arduino. The Fig 4 shows the personal DIY project displaying the particles on the 12x12 grid of the elaborated system.

III. Conclusions

This project has mainly proven the successful binding of the two algorithms: the evolutionary computation and the cellular automata, which has demonstrated good results and reliability on the expected and requested outcomes. Some principles and theoretical aspects analyzed during the lectures emerged from the experimentation of the investigated system such as the ”you get what you evaluate” rule and the dynamics beyond the a priori scalarization method for the multi-objective fitness equation. The idea to merge two important technologies studied in class allowed the practical study of both algorithms in detail, enlarging and integrating the knowledge acquired during the lectures.

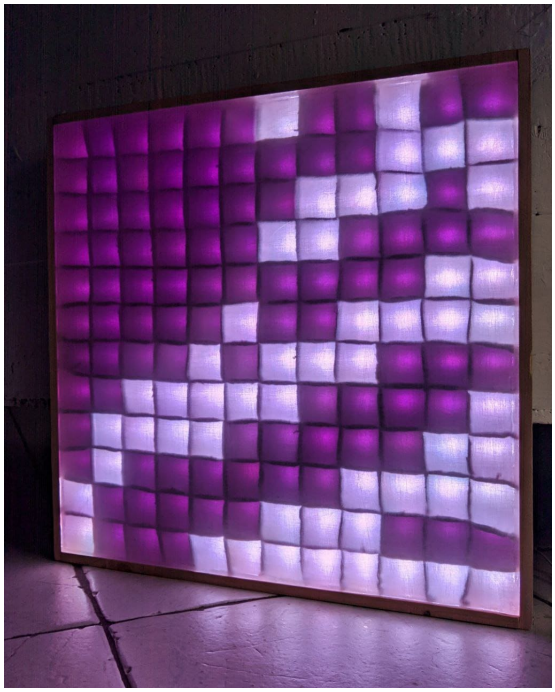

Fig. 4. Photo of the DIY LED panel with the developed project in action

The key aspect learned from this project is the impact of the fitness function on the behaviour of the observed cellular automata system: the result will strongly change depending on the selected parameters to build the fitness function and on the chosen mathematical formulation of it. Such mechanism could be used to design, by only adjusting a fitness formula, specific case behaviours for the cellular automata system applied to a specific problem or study case, making the parametrization management simple.

A potential implementation for this algorithm could be the possibility to investigate multiple sub-optimal solutions selected during the evolutionary computation phase: a possible idea of implementation could be the presence of a ”Hall of Fame” for previous optimal solutions to be displayed depending on the selection of the user.

A. Notes

The project is publicly hosted in a Github repository that can be found at ”https://github.com/emil-fazzi/Neuro-Evolved_CA”

References

[1] Stanley, Kenneth O. (2017-07-13). *Neuroevolution: A different kind of deep learning*. O’Reilly Media. Retrieved 2017-09-04.