Aparna Gopalakrishnan

1004692941

Research opportunity program (CSC299)  
final report

**Contents**

1. Abstract
2. Introduction
3. Methods
4. Results and Discussion
   1. Description of Model
   2. Observations
5. Future Steps
6. References

**Abstract**

Modelling different phenomena such as the Baldwin effect1 and Lamarckian learning is useful in evolutionary algorithms to solve a wide array of optimization problems. Through this project, results regarding the effect of mutation rate and selection methods on efficiency in convergence in a small population of 50 individuals were confirmed. The interaction between evolution and learning has been studied from a molecular perspective2 as well as from a computational perspective to understand the processes, benefits, and costs involved. This interaction is relevant in the field of artificial intelligence as it improves our understanding of efficiency in adaptation in dynamic environments. Classical models implement genetic changes, however, behaviour such as learning can potentially influence fitness, survival, and thus, evolution. The aim of this project was to model the Baldwin effect in plants, as well as explore different plant-specific parameters of reproduction like self-reproduction and the lack of decision and classification problem for plants.

**Introduction**

The concepts of Evolutionary Algorithms (EAs), artificial neural networks (ANNs), and Evolutionary Artificial Neural Networks (EANNs) are introduced below as background material for this project, followed by an explanation of the Standard Genetic Algorithm (SGA) used in the project.

**Evolutionary Algorithms (EAs)** are population-based stochastic search algorithms developed from the principles of evolution as a tool for problem-solving, design, and optimization of those problems which cannot be solved by more traditional design and optimization tools. EAs model natural selection, by having specific steps (initialization, selection, genetic operators – crossover and mutation, termination) that closely parallel their biological counterpart. There exist four main EA types:

1. Evolutionary Strategies
2. Evolutionary Programming
3. Genetic Algorithms
4. Genetic Programming

They share similar basic design but differ in a few parameters like preferred selection strategies and relative importance of genetic operators.3

**Artificial neural networks (ANNs)** are computational models that are inspired by biological neural networks, which model receiving, processing, and transmitting information similar to the structure and functions of biological neural networks. ANNs are composed of processing units (similar to neurons in a biological neural network) which have weighted connections (similar to neurotransmitters among neurons) between them. ANNs have three different layers:

1. **Input layer** through which inputs are fed into the model
2. **Hidden layers** which are used to process the inputs received from the input layers
3. **Output layer** which makes available the data after processing

The weighted connections between these layers are tasked with extracting useful and required information and features from the input data, which is done through a combination of parameters such as:

1. Activation function which decides if the output of the neuron will be active (similar to biological soma, which activate the output only if there are strong enough stimuli on the input)
2. Propagation function which aggregates outputs from many other neurons as input to another neuron
3. Connection weights which improves accuracy of the network

Some types of ANNs include Feedforward Neural Network, Convolutional Neural

Network (CNN), and Recurrent Neural Network (RNN), but this list is in no way exhaustive of the various types of ANNs available.

ANNs have the advantage of adaptability, much like the human brain, to learn and model non-linear and complex relationships between inputs and outputs. They also have the ability to predict and generalize on previously unseen data after learning and training from initial data.

**Evolutionary ANNs (EANNs)** are a special class of ANNs in which forms of adaptation include learning and evolution. EANNs can adapt to the environment as well as to changes in the environment. In a broader sense, they can also be considered as adaptive systems that can modify their architectures and learning rules suitably without human intervention. Combining evolution and learning with artificial neural networks (ANNs) can lead to more intelligent systems, compared to ANNs or Evolutionary Algorithms (EAs) alone.

In EANNS, evolution is used in adapting and training the neural network’s connection weights, architectures, and learning rules. This can be understood as “learning to learn” in ANNs and automatic ANN design in without human intervention.4

In using these varying structures of neurons or population of individuals, the underlying goal is to find the best solution possible to a certain problem (optimization, classification etc.). In both these structures, learning is involved, i.e., training in ANNs to improve accuracy of prediction or classification when the target answer is known or improving fitness of individuals in response to environmental change or stressors.

A Standard Genetic Algorithm (SGA) was used to model the interaction between learning and evolution in plants. A brief explanation of the algorithm is as follows:

No

Yes

Stop

**Methods**

In my implementation of the Standard Genetic Algorithm, the first generation was initialized randomly (real values in ranges mentioned in Table 1). For crossover, alternate bits of the parents were swapped to produce offspring, i.e., 2-point crossover for 4-bit individuals, 4-point crossover for 8-bit individuals, and 8-point crossover for 16-bit individuals. In particular, for 8-bit individuals, a detailed explanation of the crossover mechanism can be found below:

Parent 1 Offspring 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | q | c | s | e | u | g | w |

Crossover

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| p | q | r | s | t | u | v | w |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| p | b | r | d | t | f | v | h |

Parent 2 Offspring 2

This was conducted until the population for the next generation was produced. Next, mutation was conducted by setting the mutation rate to 0.1%, 1%, or 5%, where the rate represents the probability of a random change (mutation) occurring. Mutation was conducted by choosing a random integer in [0, (phenotype length=4, 8, 16) - 1] and generating a new value for this bit in the range [-1, 1]. For example, for 8-bit individuals, a random integer is generated in [0, 7], say 3, and the phenotype bit of the individual at this position is changed to a new random real number in [-1, 1].

In my model, the influence of three parameters was tested:

1. Mutation rate:
   1. Low (0.1%)
   2. Moderate (1%)
   3. High (5%)
2. Parent population size percentage – number of individuals from previous generation used to produce offspring generation:
   1. Low (10%)
   2. Moderate (20-50%)
   3. High (60-100%)
3. Tournament size percentage – number of individuals in a tournament such that “best” or most fit individual in this set of individuals chosen for crossover:
   1. Low (10%)
   2. Moderate (20-50%)
   3. High (60-100%)

**Results and Discussion**

Articles by Ellefsen,5 and Suzuki and Arita6 articles provided as starting material.

* Ellefsen modelled the Baldwin Effect in a population of individuals resembling animals using Reinforcement Learning (RL), and more specifically, used Downing’s Script-Based Approach to Evolving Neural Networks (SEVANN)7 system to implement the neural network. In this experiment, an individual was made to choose between red and green food, one being nutritious and the other poisonous. The fitness of the individual was calculated in such a manner to ‘punish’ the individual for eating the poisonous food, or give this action a low fitness score, and on the other hand, ‘reward’ the individual for eating the nutritious food, by giving this action a higher fitness score.   
    
  The environmental change/stress in this model was the switch made, i.e. if the first iteration of the experiment denoted the green food as nutritious and the red food as poisonous, then the switch would result in the green food being poisonous and the red food being nutritious. It is to be noted that consumption of the poisonous food does not kill the individual but reduces its fitness score. The experiment also tested the plasticity of the individual when this switch was made multiple times in one lifetime, versus when it was made multiple times over many generations.
* Suzuki and Arita approached this topic from a game theory perspective and used the Iterated Prisoner’s Dilemma (IPD) as a model to discover the optimal strategy used by an individual in response to environmental stress. The Meta-Pavlov learning method was used to compare strategies adopted in globally-interacting and locally-interacting agents to understand the impact of the scale of interaction in the emergence of the Baldwin Effect.

Plant-based options discussed in this project are as follows:

1. Defence-priming in plants i.e. memory (plasticity) of plants after, say, an insect attack over a lifetime versus over multiple generations
2. Impact on photosynthetic rate and/or transpiration rate in response to a change in heat and moisture8

The main challenge in this project was to convert a classification problem experiment designed for individuals resembling animals to a non-classification, non-decision problem for individuals resembling plants, due to the lack of availability of such clear-cut classification and decision problems in the context of learning in plants. Both these options required modelling of the Baldwin Effect using a specific mathematical model for the respective plant processes (if available), resulting in a loss of abstraction. This secondary option of using a mathematical model to evaluate plant growth8 (fitness function) was not used, as it would become too specific.

Both of these problems above do not have a target answer to be achieved, and hence traditional neural networks could not be used, as training the individuals would be difficult. Instead, Standard Genetic Algorithms (SGAs) were more suitable as these algorithms could be used for optimization of problems for which the correct/absolute answer was unknown prior to the start of the experiment.

Professor Kai Olav Ellefsen was contacted regarding his article, and details regarding the fitness function used and other parameters, and Professor Keith L. Downing was contacted for further details regarding SEVANN. Professor Ellefsen provided insight regarding his experiment as well as gave me details regarding existing Python modules to implement this experiment (2, 3 listed below). Abstraction of learning through the following models was explored:

* 1. Own implementation of a Standard Genetic Algorithm (SGA)
  2. NEAT-Python (NeuroEvolution of Augmenting Topologies)9
     + Was not chosen due to the incompatibility of the nature of the problem and the parameters offered by NEAT-Python
  3. DEAP (Distributed Evolutionary Algorithms in Python)10
     + Did not have time to fully implement a model using DEAP

**Description of Model**

The details regarding the parameters chosen for my own implementation of SGA is listed in Table 1:

Table 1: Description of parameters and values chosen

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Chosen Values** | **Comments** |
| Phenotype length | 4, 8, 16 |  |
| Phenotype values | Version 1: Real values in [-10, 10] for genetic bits, real values [0, 1] for learned bits |  |
| Version 2: Real values in [-1, 1] for all bits |
| In both versions, alternate bits for benefit and cost. |
| Fitness function = Benefit - Cost | Version 1: Sigmoid for benefit, Rectified Linear Unit (ReLu) for cost | In both versions, cost functions were appropriately chosen so as to not provide a negative value, as subtraction of a negative value would lead to increase of total fitness function, which should not be allowed. |
| Version 2: Sigmoid |
| Population size | 50 | Advantage of SGA is lost in larger populations11 |
| Selection Methods12, 13 | * Linear Rank Selection (LRS) * Exponential Rank Selection (ERS) * **Tournament Selection (TOS) (used in final version)** | These selection methods are suitable for negative and non-negative fitness values, and with appropriate parameters, will avoid premature convergence to local optima. |
| Mutation rate | Low (0.1%), Moderate (1%), high (5%) |  |
| Number of generations | 50, 100, 150, 300, 1000, 3000, 5000 |  |
| Termination criteria | * Number of generations * Fitness threshold | Number of generations (50, 100, 150) was used as a termination criterion during testing phase. Observations in the next section have been given using fitness threshold termination criterion. |
| Iterations | 50 | Observations shown in the next section have been averaged over 50 iterations |

**Observations**

The model tested the influence of three parameters: phenotype length, parent population size percentage, and tournament size percentage on the rate and efficiency of convergence in a small population of 50 individuals. The summary of observations is given in Table 2:

Table 2: Summary of Observations Regarding Convergence

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter choices** | | | **Observations on convergence** | **# generations to convergence** |
| **PH (bit-length)** | **P, T (%)** | **M (%)** |
| 4, 8, 16 | 10, 10 | 0.1, 1, 5 | * No or inefficient convergence * Decrease in average fitness * Observed in Fig. 1 | Not applicable |
| 4, 8 | 20-50, 20-50 | 0.1 | * Convergence to local optima due to low mutation rate * Observed in Fig. 2 | 200-2000 |
| 4, 8 | 60-100, 60-100 | 0.1 | * Quicker convergence in population of 4-bit individuals * Observed in Fig. 3 | 50-600 |
| 4, 8 | 20-50, 20-50 | 1 | * Quick convergence * Observed in Fig. 4 (PH=4), Fig. 5 (PH=8) | 30-250 |
| 4, 8 | 60-100, 60-100 | 1 | * Quicker convergence in population of 4-bit individuals * Observed in Fig. 6 | 10-50 |
| 4, 8 | 20-50, 20-50 | 5 | * Quicker convergence in population of 4-bit individuals * Best individual outperforms population in many generations * Observed in Fig. 7 | 10-200 |
| 4, 8 | 60-100, 60-100 | 5 | * Very quick convergence * Sharp increase in average fitness in the first 10-15 generations * Observed in Fig. 8 | 10-50 |
| 16 | 20-50, 20-50 | 0.1 | * Slower rate of increase in average fitness * Observed in Fig. 9 | 1500-3000 |
| 16 | 60-100, 60-100 | 0.1 | * Quicker convergence * Observed in Fig. 10 | 600-2000 |
| 16 | 20-50, 20-50 | 1 | * Slower rate of increase in average fitness for lower tournament size * Observed in Fig. 11.1, Fig. 11.2 | 100-600 |
| 16 | 60-100, 60-100 | 1 | * Quicker convergence * Observed in Fig. 12 | 100-200 |
| 16 | 20-50, 20-50 | 5 | * Quicker convergence * Observed in Fig. 13 | 20-200 |
| 16 | 60-100, 60-100 | 5 | * Sharp increase in average fitness within the first 15-20 generations * Very quick convergence * Observed in Fig. 14 | 20-100 |

*PH = Phenotype length, P = Parent population size percentage, T = Tournament size percentage, M = Mutation rate*

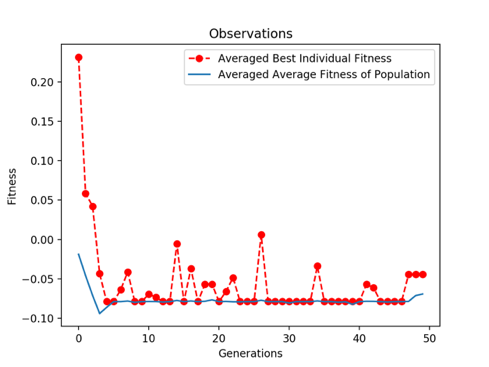
* Combinations of low (0-10%), moderate (20-50%), and high (60-100%) parent population size and tournament size were tested to compare the effect of each combination on the loss of diversity in population, and consequently, the rate of convergence of population.
* Regardless of parent population size percentage and mutation rate, low tournament size percentage (10%) is not advisable as it provides very little to no advantage to the fitter individuals in the population. On the other hand, very high advantage to the fitter individuals, i.e., higher tournament size percentages (80-100%), as the probability of always choosing the best individual in the population is high, leading to quick loss in diversity of population and convergence to local optimum.
* With the combination of higher parent population size and tournament size percentages (60-100%), quick convergence is due to the best individual having a higher probability of being chosen (seen by an initial sharp increase in average fitness) for crossover leading to quick loss in diversity in the population and convergence towards local optimum.
* Moderate mutation rate, parent population, and tournament size percentages (20-40%) allow for slower convergence but avoid premature convergence toward the local optima.
* Extreme mutation rates are not suggested, as very low mutation rates allow for minimal exploration of the search space, and very high mutation rates result in inefficient convergence.
* Combination of low parent population size percentage and tournament size percentage leads to no or inefficient convergence as seen in Fig. 1 below:

Fig. 1: Observations for PH=8 P=10%, T=10%

* Low mutation rate (0.1%) leads to very quick convergence, convergence to local optimum, loss in diversity in population, and stagnation in the fitness of population as can be seen in Fig. 2, 3, and 9 below:
* Combination of moderate M%, P%, T% provides sufficient advantage to the fitter individuals in the population. This combination allows for slower convergence, but more efficient convergence, regardless of phenotype length, as can be seen over 1000s of generations in Fig. 4, 5, 11.1, and 11.2 below. This combination with high M% led to quicker convergence as can be seen in Fig. 7 and 13 below. In particular, the difference between combinations of lower values (20%) and higher values (40-50%) for P%, T% can be observed in Fig. 11.1 and 11.2, and the rate of convergence in each.

|  |  |  |
| --- | --- | --- |
| Fig. 4: Observations for PH=4, P=40%, T=40%, M=1% | Fig, 5: Observations for PH=8, P=20%, T=20%, M=1% | Fig. 11.1: Observations for PH=16, P=50%, T=40%, M=1% |
| Fig. 11.2: Observations for PH=16, P=20%, T=20%, M=1% | Fig. 7: Observations for PH=8, P=50%, T=40%, M=5% | Fig. 13: Observations for PH=16, P=40%, T=20%, M=5% |

* Combination of higher parent population size percentage and tournament size percentage provides more advantage to the fitter individuals, i.e., higher probability of being chosen for crossover, regardless of phenotype length. This combination leads to quicker convergence as the better individuals are consistently picked for reproduction leading to quick loss in diversity in the population. However, even with this combination, higher mutation rate (5%) leads to slower convergence, seen in Fig. 14. This can be seen in Fig. 6, 8, 10, 12, and 14 below:

|  |  |  |
| --- | --- | --- |
| Fig. 6: Observations for PH=8, P=80%, T=80%, M=1% | Fig. 8: Observations for PH=8, P=80%, T=80%, M=5% | Fig. 10: Observations for PH=16, P=80%, T=80%, M=0.1% |

|  |  |
| --- | --- |
| Fig. 12: Observations for PH=16, P=80%, T=80%, M=1 | Fig. 14: Observations for PH=16, P=80%, T=80%, M=5% |
|  |  |

**Future Steps**

This model could be expanded to include more parameters such as:

* Plant-specific reproduction (self-reproduction)

This type of reproduction could have significant impact on the rate of convergence, as a percentage of population in each generation would be created by crossover of the same individual phenotype.

* Comparison of convergence in a population of globally-interacting agents (implemented by the current model, which is unrealistic) and locally-interacting agents for reproduction
* Introduction of crossover rate, different crossover mechanisms, mutation mechanisms
* Alternative Evolutionary Algorithms (EAs) and/or neural network architectures

**Acknowledgements**

I would like to thank Katharina Braeutigam, Kai Olav Ellefsen, and Keith L. Downing for their interest, insight, and suggestions.

**References**

[1] Baldwin, M. (1896). A New Factor in Evolution. *The American Naturalist,* (30):536-553.

[2] Bräutigam, K. and Cronk, Q. (2018) DNA Methylation and the Evolution of

Developmental Complexity in Plants. 9, 1447

[3] Downing, K. L. <https://www.idi.ntnu.no/emner/it3708/lectures/notes/evolalgs.pdf>

[4] Yao, X. (1999) Evolving Artificial Neural Networks. *Proceedings of the IEEE,*

87(9):1423-1447

[5] Ellefsen, K. O. (2013) Balacing Costs and Benefits of Learning Ability. Conference

Proceedings, *12th European Conference on Artifical Live, At Taorima, Italy*

[6] Suzuki, R. and Arita, T. (2004) Interactions between learning and evolution. The outstanding strategy generated by the Baldwin effect. BioSystems 77:57-71

[7] Downing, K. L. (2010). A Script-Based Approach to Evolving Neural Networks. In *Proceedings of the second Norwegian Artificial Intelligence Symposium*, pages 29-36.

[8] Zee, F. and Bubenheim, D. (1997). Plant Growth Model Using Artificial Neural Networks.

[9] Stanley, K. O. and Miikkulainen, R. (2002). Efficient Evolution of Neural Network Topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC ’02).*

[10] <https://github.com/deap/deap>

[11] Chen, T., Tang K., Chen G., and Yao X. (2018). A Large Population Size Can Be Unhelpful in Evolutionary Algorithms.

[12] Blickle, T. and Lothar, T. (1995). A Comparison of Selection Schemes used in Genetic Algorithms.

[13] Jebari, K. and Madiafi, M. (2013). Selection Methods for Genetic Algorithms