Exploration of Evolutionary Algorithm design, and other solutions, as the means for weights adjustment of a Neural Network

Stefan Hristov

ABSTRACT

In this paper different strategies and parameters for an Evolutionary Algorithm are evaluated on their effectiveness to evolve a good solution as the means for tuning the weights of a neural network. The goal was to find a good combination which produces consistently solution with good fitness on both training and testing data. The research established that in the given domain, population size of 50, 5 hidden neurons, 1 child produced through uniform crossover, 2 parents selected with roulette-wheel selection, worst individual replacement strategy and mutated through a dynamic customized mutation produces the best solutions. A side effect of this algorithm is signs of over fitting. The research also proposes potential test and solutions to develop the algorithm further and fix some of the draw backs.

1 INTRODUCTION

For this coursework the task was to research and implement an algorithm which can be used to train a neural network in order to land 8 space ships in a small 2D game.

The neural network is with 5 inputs, 3 outputs and only one hidden layer. The number of nodes is one of the variables. The fitness evaluation was part of a java jar file and changes were not permitted.

The approach researched in this paper was Evolutionary Algorithm (EA). The goal was to find optimal strategies and parameters for the implementation of the EA which yield best results "consistently".

Evolutionary algorithms are biologically inspired techniques that incorporate concepts such as selection, reproduction, recombination and mutation.

2 APPROACH

The next paragraphs outline the research and comparison done of different strategies for each stage of the EA. The initialization of the population was done randomly using a seed produced at run time. Replacement of members from the population was done so that if a new member (a child) has better fitness than the worst in the population, it will take its place. The activation function is a hyperbolic tangent function.

2.1 Strategies

2.1.1 Selection. Selection is the stage at which solutions are selected from the population to be parents for the next generation. Their genetic material is used in recombination process known as crossover. It's importance is to keep gene diversity in order to be able to explore the solution domain. With inappropriate selection strategy, the population might converge too early and loose valuable solutions.

The initial approach was chosen to be **Tournament selection** due to its efficiency and simple implementation [6][4]. Furthermore Zhong [8] shows that Tournament selection performs better than **roulette wheel** in similar conditions to out problem. On the basis of this research, Hypothesis I is that tournament selection performs better than roulette wheel in our problem. The tournament size should be relatively low, otherwise it leads to higher expected loss of diversity [1].

In **Tournament selection** a random sample of individuals is selected from the population and moved to a tournament pool. The number of individuals is one of the variables (tournament size) and it is responsible for the selection pressure. The fittest person from the tournament pool is selected to be one of the parents. The process is repeated till the desired number of parents is reached. This paper's implementations ensures the selected parents are not the same solution.

In **Roulette-wheel selection** each solution has a chance of being selected proportionate to its fitness. Since in our problem the improvement of fitness produces lower fitness, the calculations were done with invert fitness values.

2.1.2 Crossover. Crossover, also known as recombination, is the stage of selecting and combining the genes from the selected parents. The significance of crossover is that it allows for recombination of those genes and therefore it comprises a solution with the best genes available in the population.

The first implementation of a crossover strategy is **uniform** crossover. This type of crossover randomly selects the parent from which each gene should come from and therefore ignores any bias for keeping adjacent genes together. It was chosen since it allows the fastest exploration of genes in the gene pool (population).

An alternative solution was deviated from this paper's [5] suggestion for keeping **neuron relevant** information together. It was considerate due to its similarity in searching of a solution for control system on a robot. In this crossover, the weight information and bias for each neuron in the hidden layer and the output layer was kept together. During the recombination, genes will be exchanged between the parents only in terms of relative to a neuron. An example is where all the weights and bias for neuron one come from parent one and all the weights and bias for output one comes from parent two.

Other strategies such as **single-point** and **k-point** cross over were not considerate as potential solutions since the neuron related approach is practically a k-point crossover. That being said, random single-point or random k-point cross over could be a viable solution. Unfortunately due to time constrains, those two approaches were not tested.

An experiment was conducted in order to decide which of the two approach is better suited for our problem.

2.1.3 *Mutation.*

Mutation is genetic operator which promotes gene diversity in the population but also ensures the chance of exploring a solution in the domain of our problem. That is done by mutating genes into new, previously not present in our gene-pool. In other words, mutation helps the algorithm to escape local minima/maxima in order to continue exploring the search domain. Mutation can also help perfect a solution by mutating genes with very small amount and rather than escaping this solution, it can improve it.

Mutation is comprised of mutation rate (the chance for each gene to be mutated) and mutation change (the amount with which each gene is mutated).

The initial approach was to keep the mutation rate and mutation change (mutation step) constant throughout the training and test different values.

Alternative approach was found in MATERIAL DYNARDO paper [2], which proposed a solution to a problem with similar search graph to the coursework. The reason behind this approach is when the algorithm is stuck in a local minima/maxima, the mutation will adapt so it can speed up the process of escaping it. In order to achieve that we first must calculate the Fitness Frequency Distribution (FFD) of the best fitness (BFF).

$$BFF = \frac{1}{D}\sum_{i=1, i\neq b}^{N} \delta_i, \qquad BFF \in [0,1) \tag{1}$$

where

$$\delta_i = \begin{cases} 1, & t_i \in [t_b - \epsilon, t_b + \epsilon], \\ 0, & otherwise, \end{cases} \tag{2}$$

and D is the size of the population, t_b is the best fitness value, t_i is the fitness value of i-th point and ϵ is the bandwidth of the class interval. It is calculated as a percentage of the value of t_b.

$$\epsilon = t_b \cdot r, \qquad r \in (0,1] \tag{3}$$

The paper also suggests value for r as 5%.

Mutation Rate was then updated in a similar fashion to the suggested implementation with small alteration. Instead of an average mutation rate, this implementation uses a constant value represented with p_{const} in the equation.

$$p_m = p_{const} + \mu \tag{4}$$

and

$$\mu = 3\sigma\left(\frac{\overline{f}^L}{\xi}\right), \qquad \mu \in [0, 3\sigma], \tag{5}$$

where $\overline{f}^L$ is an average BFF for L generations, ξ is a linking coefficient and σ is the standard deviation of the population's fitness. As recommended, the linking coefficient was set between 1 and 2.

The mutation change for this approach was a Gaussian distribution.

$$g_i^{t+1} = g_i^t + N(0,\sigma), \tag{6}$$

where $N(0,\sigma)$ is the normal distribution with mean of zero and the standard deviation σ.

The parameters used in this implemenations are as follows: $r = 0.05, p_{const} = 0.1, L = 50$ and $\xi = 1.2$.

The final approach was inspired from the previous approach from MATERIAL DYNARDO's paper. The mutation rate was changed dynamically before each mutation. It begins at 5% and it increases with 1% for each evaluation which did not find a better solution than the already best known solution.

$$p_m = p_{const} + E \cdot 0.01, \tag{7}$$

where p_{const} is set to 0.05 and E is the number of evaluations completed since the last update of the fittest individual.

The mutation change is similar to the previous implementation but the standard deviation is set 5 times the best fitness from the population.

$$g_i^{t+1} = g_i^t + N(0,\sigma), \tag{8}$$

where

$$\sigma = 5 \cdot t_b \tag{9}$$

2.2 Parameters

2.2.1 *Hidden Neuron Nodes.*

The default value was 5, which is equal to the inputs to the network and was found to produce adequate results.

2.2.2 *Population Size.*

According to this paper [3] in problems where selection of individuals tent to get stuck in local optimum with a high fitness, similar to our problem, large population size can be harmful. Online forum suggests, "At CEC2013, a presenter said that Storn and Price recommended a population size of 10 times the number of dimensions". For the initial testing, population size of 50 was selected.

2.2.3 *Selection Pressure.*

In the case of the tournament selection, the pool size is a variable which controls the selection pressure. As recommended from literature, two low selection pressures, 2 and 10% of population size, were selected and compared.

2.2.4 *Mutation rate.*

For the constant mutation rate, the value of 10% was chosen.

2.2.5 *Mutation change.*

The mutation change was chosen to be 0.3 based on few runs.

2.2.6 *Crossover children.*

Since we have a limit of 20000 evaluations, it was decided that all crossover implementations will be producing a single child as a offspring in order to promote more crossover stages and therefore higher exchange rate of genes.

3 EXPERIMENTS AND ANALYSIS

Test were conducted consecutively and all 10 runs were documented. The following values were set as default values and were used unless specified differently.

Neuron Nodes	5
Population Size	50
Mutation Rate	0.1
Mutation Change	0.2
Activation Function	tanh
Replacement	Replace the individual with the worst fitness

Table 1: Default Settings

3.1 Selection

Different selection operands were tested against each other with the default settings and uniform crossover. First, the selection pressure was tested with tournament size of 2 (minimum) and 5 (10% of population size).

	Training Data	Testing Data
Mean	0.104719	0.239617
Median	0.112658	0.263071
Standard Deviation	0.033043	0.064766

Table 2: Tournament Size 2

	Training Data	Testing Data
Mean	0.125420	0.257714
Median	0.127737	0.265718
Standard Deviation	0.015051	0.032082

Table 3: Tournament Size 5

	Training Data	Testing Data
p value	0.104363	0.462262

Table 4: T-test between Tournament Sizes 2 and 5

The results are not conclusive and since $p > 0.05$ they are considered insignificant. Regardless, it seems but the mean and median are slightly lower with tournament size 2 which supports the hypothesis that higher tournament size would lead to premature convergence. Therefore, tournament size 2 was the preferred parameter.

Next Roulette-wheel selection is compared to Tournament selection with tournament size 2.

	Training Data	Testing Data
Mean	0.069901	0.163306
Median	0.080951	0.171514
Standard Deviation	0.034096	0.062592

Table 5: Roulette-Wheel selection

	Training Data	Testing Data
p value	0.020450	0.020450

Table 6: T-test between Tournament Sizes 2 Roulette-Wheel

Following the findings from Table 5 and Table 11 where the mean and median are lower and $p < 0.05$ it is concluded that the result is significant and Roulette Wheel performs better than Tournament selection in our context with our parameters. That invalidated our initial hypotheses I.

3.2 Crossover

To confirm the findings for the selection strategy, test was carried out for Neuron related crossover with both Tournament selection (tournament size 2) and Roulette-Wheel selection. Since in the previous experiments Uniform crossover was used, it is possible to compare the results.

	Training Data	Testing Data
Mean	0.108142	0.248208
Median	0.123814	0.248676
Standard Deviation	0.033829	0.041726

Table 7: Neuron related crossover with Roulette-Wheel selection

	Training Data	Testing Data
Mean	0.110577	0.232368
Median	0.119445	0.254152
Standard Deviation	0.030165	0.071744

Table 8: Neuron related crossover with Tournament selection (t=2)

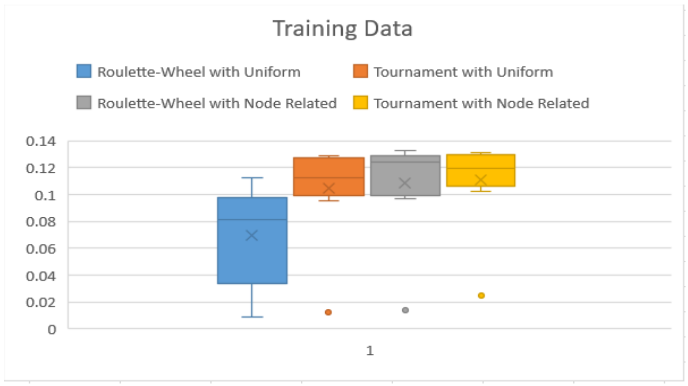

Figure 1: Training data

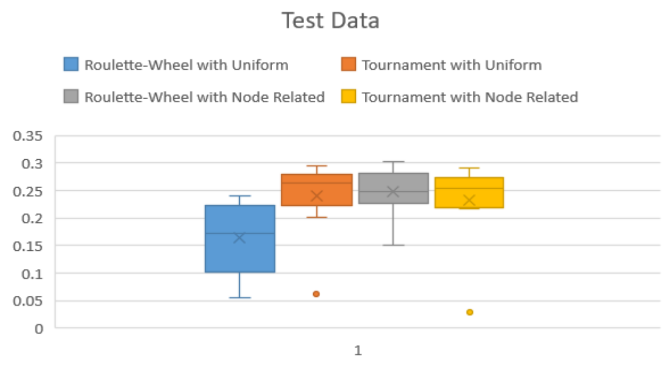

Figure 2: Training data

The results suggest that there is no significant difference between either selection method when used in conjunction with the Neuron related crossover approach. Overall, the results were poor and it seems the approach is not suited for our context or not refined enough.

Roulette-Wheel selection combined with Uniform crossover produced the best results. It was not as consistent but overall it produced lower fitness both on training and testing data.

3.3 Mutation

Previous experiments were all done with a static mutation rate and mutation change. The next experiment is testing the suggested adaptive mutation [2]

	Training Data	Testing Data
Mean	0.098854	0.210924
Median	0.111858	0.222519
Standard Deviation	0.028927	0.061124

Table 9: Adaptive mutation (FFD)

The results show decrease in performance from Table 5. The undesirable results could perhaps be explained by a flow in the algorithm. In equation 5 when the algorithm is stuck in a local minima, the standard deviation will approach 0, which will bring the change of mutation rate to 0 as well. That is opposite of the desired result. There are some concerns with the usefulness of the standard deviation of the fitness for the population in this equation. More research is needed to perhaps improve this approach.

Alternatively, with the custom approach proposed in this paper inspired from the previous one, the tests were carried out.

	Training Data	Testing Data
Mean	0.009327	0.050275
Median	0.006741	0.042918
Standard Deviation	0.007021	0.042145

Table 10: Best Fitness Mutation

	Training Data	Testing Data
p value	0.000058	0.000281

Table 11: T-test between Static and Best Fitness Mutation

Considering the p value there is significant improvement. The mutation strategy seems to be extremely successful at times but it lacks consistency. Relatively often it over fits and performs poorly on testing data. On training data the standard deviation demonstrates that it produces close results and the median suggests that majority of them are lower than the presented average (mean). Unfortunately, there are also some outliers, but their effect is not too severe.

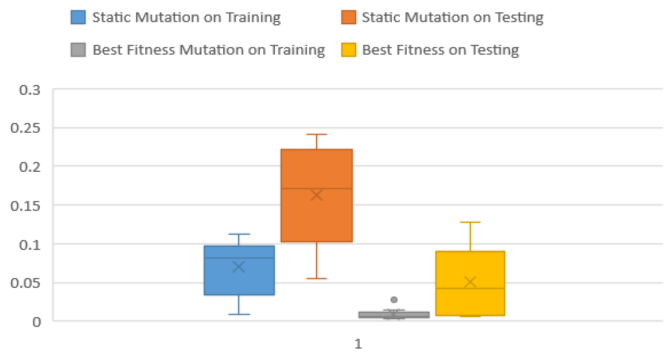

Figure 3: Static vs Best Fitness Mutation

Even though the results are not perfect and there is some over fitting, the standard deviation of the final approach has improved as well as the overall fitness.

An improvement to the testing design could be to use the same seed for the relevant test when comparing two strategies.

4 CONCLUSION

In conclusion, this study found that for population size 50 and 5 hidden neurons in our context, roulette-wheel selection performed better than tournament selection. An attempt to keep neuron relevant genes together during crossover proved to be unsuccessful. In search for improvement for the mutation strategy, a solution was explored which had promising results in the original paper. In our problem it did not perform well, but it had insightful information which was the building block for the proposed ”best fitness” mutation implementation. This approach showed promising results which performed really well but not consistently. Additionally, there was a high occurrence of over fitting.

5 FUTURE WORK

An improvement to over fitting can be to stop the algorithm when it has reached a reasonable level of fitness during training [7]. Other option is to stop training when the fitness on a small set of data, outside of the training one, starts increasing compared to the training fitness. More research is needed in order to implement this approach. This research was also constrained with the data sets and therefore the suggested solution was not implemented, but it is considerate as a viable solution.

Further test that could produce good results or be insightful include increasing the amount of parents during crossover in order to increase the rate at which genes are exchanged and allow for perhaps safer increase in population size. Unfortunately, that has the draw back of increased computational time.

As the size of the problem increases, both tournament selection and proportional roulette wheel selection become susceptible to premature convergence. On the other hand, **Rank-based** selection continues to explore the search space for better solution [6]. Based on this statement, and the option to increase the hidden neurons (which increases the amount of weights to tune and therefore the size of the solution) could prove to be better than both tournament and roulette-wheel selection.

Another place where more work could be done is into developing the implementation of the Fitness Frequency driven mutation strategy. In conclusion of the tests and the implementation, more research and design is needed before the solution could be properly justified.

Finally, apart from improvements in strategies, there are many combinations of parameters which were not explored, such as increase and decrease in neurons, population size, amount of children produced during crossover and functions such as the fitness and replacement.

REFERENCES

[1] T Blickle and L Thiele. 1995. A Comparison of Selection Schemes used in Genetic Algorithms. *TIK Report* (12 1995).

[2] Stephan Blum, Romanas Puisa, Jörg Riedel, and Marc Wintermantel. 2001. Adaptive mutation strategies for evolutionary algorithms. In *The Annual Conference: EVEN at Weimarer Optimierungsund Stochastiktage*.

[3] Tianshi Chen, Ke Tang, Guoliang Chen, and Xin Yao. 2012. A Large Population Size Can Be Unhelpful in Evolutionary Algorithms. *Theoretical Computer Science* 436 (08 2012), 54–70. https://doi.org/10.1016/j.tcs.2011.02.016

[4] David E. Goldberg and Kalyanmoy Deb. 1991. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, Vol. 51. https://doi.org/10.1016/B978-0-08-050684-5.50008-2

[5] Wilfried Elmenreich and Gernot Klingler. 2007. Genetic Evolution of a Neural Network for the Autonomous Control of a Four-Wheeled Robot. https://doi.org/10.1109/MICAI.2007.13

[6] Noraini Mohd Razali and John Geraghty. 2011. Genetic Algorithm Performance with Different Selection Strategies in Solving TSP, Vol. 2.

[7] Adam P. Piotrowski and Jarosffaw Napirkowski. 2013. A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modeling. *Journal of Hydrology* 476 (01 2013), 97fi??111. https://doi.org/10.1016/j.jhydrol.2012.10.019

[8] Jinghui Zhong, Xiaomin Hu, Jun Zhang, and Min Gu. 2005. Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms., Vol. 2. 1115–1121. https://doi.org/10.1109/CIMCA.2005.1631619