NeuroEvolution of Augmenting Topologies (NEAT)

Steuerung eins Mario Jump & Run Spiels

Jan Urfei
Bonn-Rhein-Sieg University of Applied Sciences
Grantham-Allee 20
53757 Sankt Augustin, Germany
jan.urfei@inf.h-brs.de

Tim Lügger
Bonn-Rhein-Sieg University of Applied Sciences
Grantham-Allee 20
53757 Sankt Augustin, Germany
tim.luegger@inf.h-brs.de

ABSTRACT

Short summary: in **very few** sentences, describe what was asked, what you did and what the results were

1. GENERAL NOTES

- Do not state things you cannot confirm either by literature or experiment. This is science, not black magic.
- Normalize your data and resulting statistical descriptors (like RMSE, μ)
- Save your experimental results to disk **before you visualize**. You **will** change your plots quite a lot after gathering the data.
- Don't submit reports with many pages containing just a single figure. Scrolling is terrible for our health.
- **Separate** training and test data. Don't use test data in your algorithm to learn or make decisions, they are only allowed to test the end result. If you need a separate sample set to make decisions during training or optimization, create a third (validation) set. Only your test data will really tell you how good your algorithm is.

1.1 Algorithm Parametrization

Develop an actual strategy, preferably on a reduced but similar problem. You can reduce the number of samples, the targetted number of time steps your controller runs in a simulation, or any other non-destructive problem reduction method. If your algorithm has two paramers you need to adjust, it should be no problem to take 5 *sensible* values per parameter and compare all combinations. For stochastic algorithms, like evolutionary approaches, make sure you **repeat your experiments** at least 5-10 times, depending on the amount of randomness. Since your algorithm makes ”random” changes, just comparing single runs does not give you a good estimation on its performance. Do **not** pick your values such that they only confirm the values you **want** to use.

1.2 Structure

1. **Assignment Description**: first provide a brief description of the assignment that was handed out to you. This includes a description of the data.
2. **Approach**: describe the algorithm
3. **Experiments**: describe your experimental setup, algorithm parameters, data preprocessing first. Then include the results and a discussion thereof.
4. **Conclusion**: any conclusions about the algorithm, bugs, future work.

1.3 Visualization

- Label your axes
- Use logarithmic scale when appropriate
- Use descriptive captions below your figures
- Add legends if necessary
- Make fontsizes large enough, linewidths thick enough to be readible in the final report.
- When making comparisons, make sure the results are either in the **same** graph or graphs are plotted next to (or close to) each other.
- **In short: make sure people can read your figures**

2. ASSIGNMENT

In this assignment, we implement NEAT according to the original publication [1]. The compatibility threshold for speciation is increased/decreased depending whether the number of species $\leq$ or $\geq$ the target number of species.

NEAT is then applied to create a time sequence predictor for a heart rate dataset that is provided by the university. We find the best hyperparameters (like number of subpopulations etc.). We then compare NEAT to ESP in terms of **convergence time** and **testing error**.

12 datasets are provided, containing the information as shown in Table 1. Using power data, collected from a cyclist in a lab situation, NEAT will be used to predict the heart rate for the next time step[1].

data	timestep	heart rate	power
unit	t	bpm	W
type	-	output	input

Table 1: Heart rate dataset

3. APPROACH

We extend NEAT by including the evolution of the activation function, hoping to produce *leaner* networks that generalize well and take less training[2].

Algorithm 1 HeteroNEAT

```
c: ← mutRate, crossRate, numSpecies, actFcns /* Configuration */
Input: evalFcn
function HeteroNEAT(Parameters, Input)
    P: ← initPop(n) minimal networks
    for generation g do
        N: ← expressNetwork(P)
        F: ← evalFcn(N)
        P: ← (P, N, F) /* Assign fitness /*
        S: ← speciate(P,c)
        for species s in S)
            s′(1) ← getElite(s)
            cull(s,c) /* If species is large enough */
            p: ← tournamentSelect(s)
            crossover(p,c) /* Combine matching connections */
            mutate(p,c)
        end for
    end for
end function
```

4. EXPERIMENTS

Experiments are repeated 10 times. By the way, this is not enough for a description.

4.1 Parameterization

This would include a description of all parameter tuples and a description on how you reduced the problem to provide small and fast runs for this extensive comparison. Results would be shown in boxplots, as these provide you with the median and 25% and 75% percentile for every parameter tuple. This makes comparison easy, as you can plot multiple boxplots next to each other, similar as in Figure 4.

4.2 NEAT vs ESP

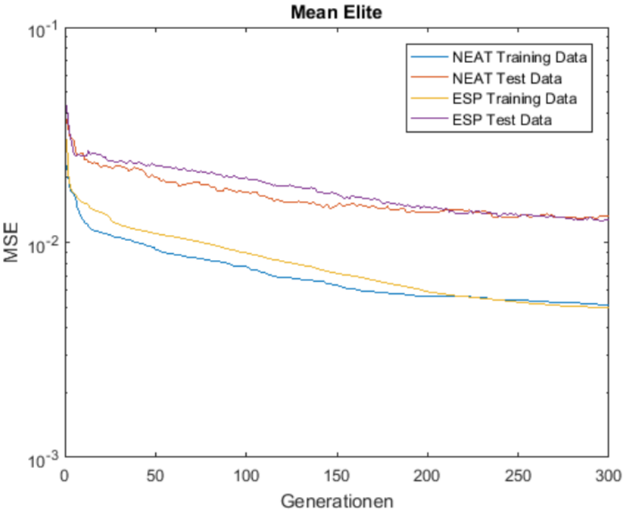

Figure 1: Development of average mean square error, comparing NEAT and ESP

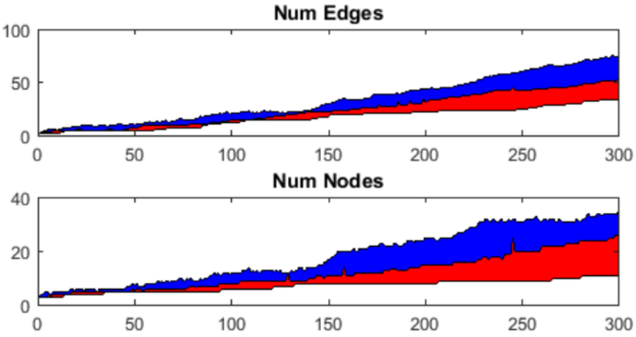

Figure 2: Development of min/mean/max number of nodes and edges. *Legend is missing*!

[1] Please also describe **why** you would use NEAT

[2] This would be something you need to show in your results!

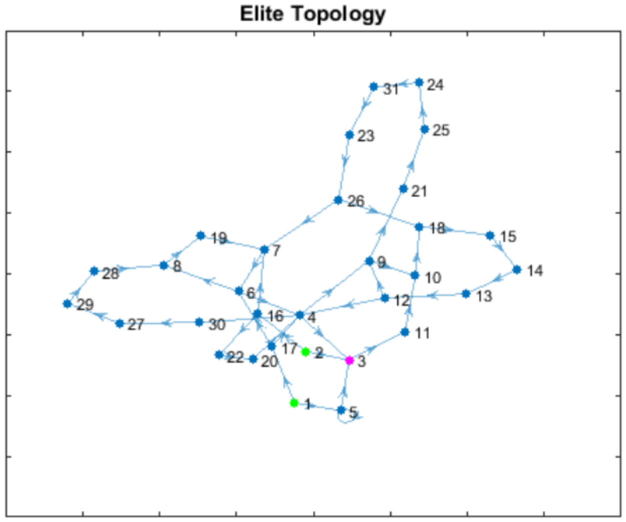

Figure 3: Elite topology. Nodes are assigned their IDs. Green are input nodes, purple are output nodes.

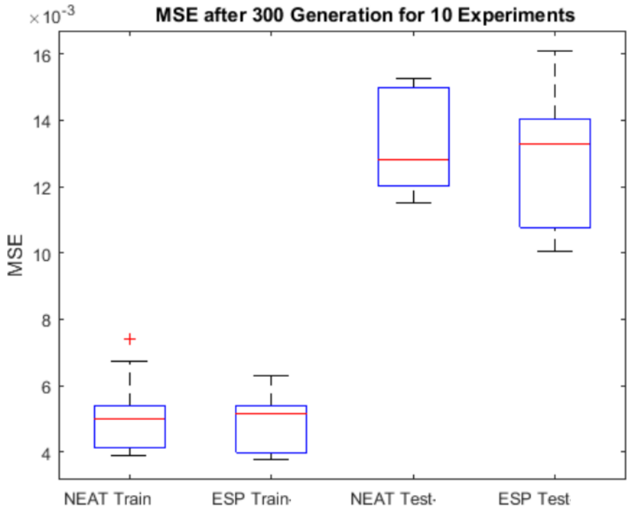

Figure 4: Comparing MSE of NEAT and ESP

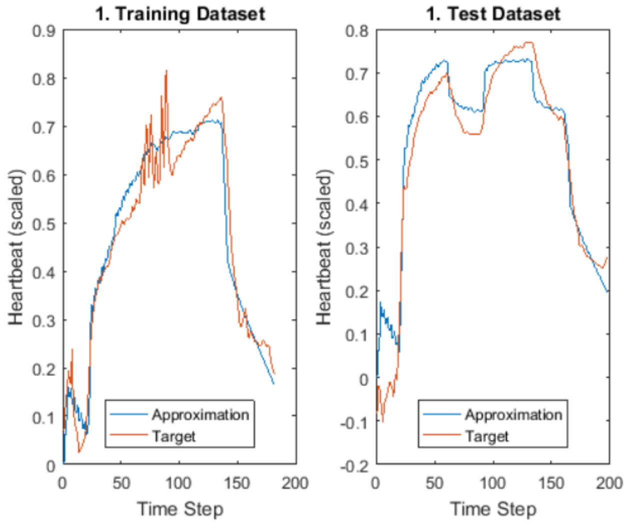

Figure 5: Approximated heart rates after 300 generations

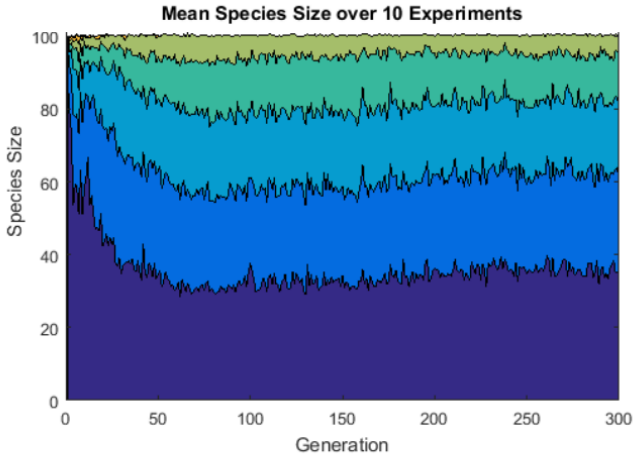

Figure 6: Development of species over time, averaged over 10 generations

5. CONCLUSION

6. REFERENCES

[1] K. O. Stanley and R. Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127, 2002.