Investigating Memetic Scheduling Strategies for Distributed Neuroevolution

Diana Velychko
dv6943@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Travis Desell
tjdvse@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Abstract

Asynchronous evolutionary algorithms (AEAs) accelerate neuroevolution but are affected by evaluation-time bias, where networks that train quickly re-enter the population sooner and disproportionately influence evolutionary outcomes. In this work, we investigate memetic scheduling strategies that dynamically determine how much backpropagation should be done by generated genomes over the course of an asynchronous neuroevolution algorithm to see if this can mitigate evaluation-time bias under a strict 1-hour wall-clock budget per run. We compare constant, randomized, and exponentially scaled memetic backpropagation scheduling strategies across multiple time-series forecasting datasets and target variables. Our experiments find an interesting negative result – while it was expected that increasing the amount of backpropagation over time would allow a search to break out of local minima and mitigate time bias, instead we show that exponentially increasing training schedules consistently underperform. We further show that different forecasting targets find better solutions with different memetic schedules, highlighting the no free lunch nature of the problem. We do, however, find a correlation between average improvement per backpropagation epoch (given a varying constant number of backpropagation epochs) and schedules performing well, which can help guide memetic schedule selection.

CCS Concepts

• **Mathematics of computing** → **Evolutionary algorithms**; • **Computing methodologies** → **Neural networks**.

Keywords

Neuroevolution, Asynchronous Evolutionary Algorithms, Evaluation Time Bias, Training-Time Allocation, Time-Series Forecasting

ACM Reference Format:
Diana Velychko and Travis Desell. 2026. Investigating Memetic Scheduling Strategies for Distributed Neuroevolution. In *Genetic and Evolutionary Computation Conference (GECCO '26), July 13–17, 2026, San Jose, Costa Rica.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3795095.3805165

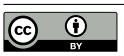
1 Introduction

The choice of a neural network architecture can be a tedious and time-consuming process due to the vast number of possible layer types, their combinations, and associated design decisions. Neuroevolution (NE) offers an automated solution to this problem by eliminating the need for manual architecture design by directly evolving network topologies alongside their parameters through mutation and selection. This involves maintaining a population of candidate networks, evaluating each on the target task, and applying crossover and mutation to drive the search toward better-performing topologies.

There are multiple neuroevolution strategies. CoSyNE [6] and CMA-ES-based [7] neural weight search optimize parameters for fixed architectures. Other approaches, such as NEAT [22] and HyperNEAT [21] focus on evolving feedforward and indirectly encoded networks with topology innovation and speciation. Since models differ in complexity, their training times vary. As a consequence, in synchronous generational evolutionary algorithms, all processes wait for the slowest evaluation to finish in generational evolutionary algorithms, causing idle computational time. This issue can be addressed with the introduction of utilizing asynchronous, steady-state, evolutionary algorithms instead.

As an example, the EXACT [4] algorithm introduced asynchronous, steady-state evolutionary search for convolutional neural networks, which was extended by EXALT to support long short-term memory (LSTM) neural networks. Building on these advances, EXAMM [16] brings this asynchronous neuroevolution paradigm to a variety of recurrent neural networks (RNNs) for time-series forecasting. EXAMM employs a scalable asynchronous master–worker architecture, where a main process maintains the evolving population and coordinates variation and selection, while multiple worker processes independently evaluate candidate genomes on available processors [16]. This asynchronous design solves the issue of workers remaining idle, and is naturally load balanced, however, it is susceptible to evaluation-time bias [11, 19, 20]. Typically, each candidate is trained for a fixed number of epochs before evaluation regardless of its complexity. As a result, asynchronous evolutionary algorithms (AEAs) favor smaller or shallower networks, as they are evaluated faster and thus repopulate the pool of genomes more frequently (see Figure 1). This skews selection toward simpler models with shorter training times, potentially overlooking more complex architectures that require longer optimization but might yield superior performance.

Prior work consistently shows that AEAs are inherently sensitive to heterogeneous evaluation times [11]. Evaluation-time bias not only alters search trajectories but also shifts evolutionary pressure toward individuals that evaluate quickly rather than those with higher fitness potential. Although several mitigation strategies have been proposed in the literature, none fully resolve the underlying tension between asynchronous efficiency and unbiased selection dynamics.



GECCO '26, San Jose, Costa Rica

ACM ISBN 979-8-4007-2487-9/2026/07
https://doi.org/10.1145/3795095.3805165

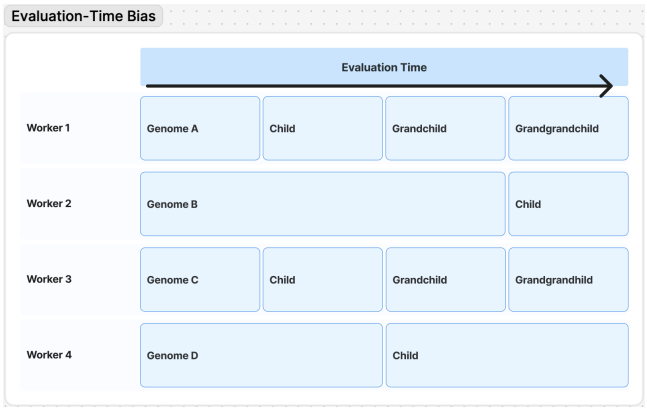


Figure 1: Evaluation Time Bias Schematic

In neuroevolution frameworks, this issue is particularly pronounced because evaluation time is directly tied to model complexity and training duration. More complex architectures require more backpropagation epochs to reach competitive performance, while simpler models complete evaluation sooner and consequently exert greater influence on the population. Using a fixed number of training epochs therefore compounds evaluation-time bias: it systematically favors shallow or low-capacity networks, reduces exploration of high-cost regions of the search space, and may prevent the discovery of superior architectures that require longer optimization.

In this work, we introduce the idea of memetic schedules, which explicitly control the number of backpropagation epochs allocated to each genome over the course of evolution, hypothesizing that introducing controlled randomness (e.g., sampling from variable epoch ranges) or gradually ramping up training epochs over time can mitigate evaluation-time bias and improve both fairness and performance in asynchronous neuroevolution. To our knowledge, utilizing memetic schedules to dynamically determine how much backpropagation to perform over the course of a search has not yet been studied. In particular, we aim to answer these questions:

(1) *Does gradually increasing the number of epochs allocated to genomes over the course of evolution improve the performance of the best discovered model?* We expect that allowing more exploration by beginning with shorter training durations, allowing the search to break out of fast-evaluating local minima, can lead to better-found solutions.
(2) *Does allowing selected genomes to train for longer enable the population to better explore high-capacity architectures and ultimately produce a more competitive set of solutions?* We expect there to be a cost-benefit tradeoff where longer training times give better measures of a network's fitness; however, this comes at the cost of fewer networks being able to be evaluated.

Our experiments find an interesting negative result – while we expected that increasing the amount of backpropagation over time would allow a search to break out of local minima and mitigate time bias, instead we show that exponentially increasing training schedules consistently underperform. Instead, our results show that different forecasting targets perform better with different memetic schedules, with memetic scheduling performance depending more strongly on per-epoch learning efficiency. We also show that the optimal best network size also correlates to better performing memetic schedules - smaller optimal networks perform better with more backpropagation per genome, while larger networks perform better with less. Overall, our results show the benefit of providing multiple memetic schedules as options and give insight into how to select a well performing schedule for a given task.

2 Related Work

Asynchronous evolutionary algorithms (AEAs) have been widely studied in the context of distributed and parallel optimization, where evaluation latencies vary across individuals. Prior work consistently shows that evaluation-time heterogeneity is not merely a practical inconvenience but a structural factor that alters selection dynamics.

The seminal analyses of asynchronous steady-state EAs demonstrated [11, 19, 20] that variation in evaluation duration systematically skews reproductive opportunities. In AEAs, individuals that complete evaluation sooner re-enter the population earlier, increasing their chances of selection regardless of fitness. Experimental studies confirmed that this bias persists across a range of test functions and selection schemes, and it influences convergence speed, population diversity, and the likelihood of reaching high-quality solutions.

Later work examined variants that blend asynchronous and generational structures [20]. These studies showed that partial synchronization does not eliminate evaluation-time bias; even quasi-generational schemes retain disproportionate influence from faster-evaluating individuals. This reinforces the view that bias arises from the interaction between evaluation latency and steady-state replacement rather than from any specific algorithmic detail.

Beyond artificial benchmarks, heterogeneous evaluation costs naturally occur in domains where the cost of assessing an individual depends on its structure or the computational demands of simulation. Research on multi-objective AEAs with variable-cost evaluations shows a consistent pattern [8, 24]: regions of the search space associated with low computational cost are explored more intensively, while high-cost regions are under-sampled. As a result, the algorithm implicitly optimizes a combined objective of fitness and evaluation speed.

Efforts to counteract this effect include modifying parent-selection frequency [8], incorporating fairness adjustments in replacement, and designing mechanisms that explicitly account for evaluation delays. These approaches generally improve balance but do not remove the underlying structural advantage of faster-evaluating individuals.

3 Methodology

3.1 Base Algorithm

Our experiments are conducted using the Evolutionary eXploration of Augmenting Memory Models (EXAMM) framework [14], as it utilizes an asynchronous, steady-state island-based distributed neuroevolution strategy that has been widely studied as a method for designing efficient recurrent neural networks, with a number of

advanced optimizations. It utilizes Lamarckian weight inheritance to significantly reduce the amount of backpropagation required for every generated neural network [14], can evolve deep recurrent connections that improve on or outperform memory cell architectures [5], utilizes a distributed island-based strategy with extinction and repopulation events to prevent stagnation of the search process [15], and can co-evolve training hyperparameters while performing neural architecture search and training [12, 23]. As EXAMM trains each genome for a specified number of backpropagation iterations, it makes for an ideal system to evaluate different memetic scheduling strategies.

Similar to NEAT [22], EXAMM provides edge level mutations to aid in the search process: *add edge*, *add recurrent edge*, *split edge*, *enable edge* and *disable edge*. EXAMM also provides novel node level mutations which significantly speed up the search process [16]: *add node*, *split node*, *merge node*, *enable node* and *disable node*; as well as a *clone* operation to allow a child to further refine weights starting from their parents. Lastly, as EXAMM utilizes a distributed island approach [15], which involve two crossover operations: *intra-island crossover*, where both parents are selected from the same island, and *inter-island crossover*, where both parents are selected from different islands. In this work, EXAMM method performs inter-island crossover 10% of the time, intra-island crossover 20% of the time, and mutation the remaining 70% of the time (which are standard settings), where the mutation operation used is selected uniformly at random from the above 11 mutation methods.

3.2 Memetic Backpropagation Scheduling

To address this work's research questions, three different memetic scheduling strategies were developed, which allow us to determine if ramping up the backpropagation epochs (slowly, quickly or linearly) can help the search process improve, or alternatively if using randomly allocated within-bounds (but not continually increasing) backpropagation epochs allows better search exploration through variation but enough exploitation by randomly allowing more backprop evaluations, or if simply using a constant strategy with different amounts of epochs can provide more reliable improvement. Additionally, for the randomized strategies, we investigated if using backpropagation epochs closer to zero or closer to our maximum allowed (32) would be more effective.

To explore the effect of these different memetic scheduling strategies that determine how much backpropagation to perform on generated genomes throughout the search process, we retain all evolutionary operators from EXAMM and modify only how the number of training epochs (*bp*_iterations) is utilized for each genome evaluation. We investigate three different types of epoch allocation memetic schedules: constant, random, and exponential.

3.2.1 Constant Number of Epochs (Const). The baseline configuration uses a fixed number of epochs per evaluation:

$$bp\_\text{iterations} \in \{0, 1, 2, 4, 8, 16, 32\}.$$

Each value defines a separate control condition.

3.2.2 Random Number of Epochs (Rand). The number of epochs is sampled uniformly based on a given range between provided upper and lower limits. This randomization tests whether variance in training depth can better improve evolutionary exploration.

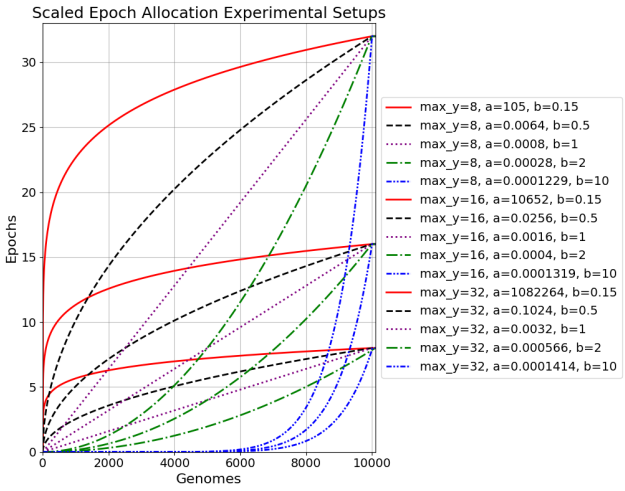


Figure 2: Scaled Epoch Allocation Experimental Setups

3.2.3 Increasing Number of Epochs (Exponential). The number of epochs increases as the number of evaluated genomes grows:

$$\text{bp\_iterations} = \min\left(\text{bp}_{\max},\ a \cdot x^b + \text{bp}_{\min}\right),$$

where x is the total number of genomes produced so far, $a = 1/\text{number\_of\_inserted\_genomes}$ is the slope, b is the exponent, and $bp_{\max}$ caps the maximum number of epochs.

The maximum limit allows us to see how well the approach is performing compared to the constant equivalent. Figure 2 presents the variations of the exponential strategy utilized in this work. Note this equation can be tuned for a variety of behaviors. By setting $b = 1$, it behaves linearly; with $b < 1$ we see logarithmic growth, and with $b > 1$ we see exponential growth.

4 Experimental Setup

To assess the effectiveness of the proposed approaches, we have conducted the following sets of experiments spanning all datasets and target variables. All runs were allocated exactly one wall-clock hour, making total genome throughput an implicit dependent variable. Our approach includes a variety of experimental setups (summarized in Table 1):

- constant approaches spanning 0 to 32, i.e. 0, 1, 2, 4, 8, 16, 32;
- upper and lower limit random approaches (which uniformly at random selected between 32 and a lower limit, and between 0 and a higher limit, respectively).
- exponential approaches that can be separated into 3 groups:
  - exponential growth which starts slow and then increases quickly, $b > 1$;
  - logarithmic growth which starts quickly and tapers off, $b < 1$;
  - linear growth, $b = 1$.

  With variants of each of these groupings generated for three different maximum values of 32, 16, 8 and adjusted to hit the maximum value after 10 000 genomes, as shown in Figure 2.

Backprop. strategy	Description	Experimental setup
Const	The baseline configuration uses a fixed number of backpropagation epochs for every genome evaluation.	$bp\_iterations \in \{0, 1, 2, 4, 8, 16, 32\}$
Random	The number of epochs is sampled uniformly from a range comparative to the corresponding baseline value.	Upper Limit Set: [16; 32] [24; 32] [28; 32] [30; 32] [31; 32]; Lower Limit Set: [0; 1] [0; 2] [0; 4] [0; 8] [0; 16] [0; 32].
Exponential	The number of epochs increases as the number of evaluated genomes grows, following a scaling formula. $b = 1$ is a linear increase.	A grid search where $b \in \{0.15, 0.5, 1.0, 2, 10\}$; $max \in \{8, 16, 32\}$ $a \in \{105, 0.0064, 0.0008, 0.00028, 0.0001229\}$ $a \in \{10652, 0.0256, 0.0016, 0.0004, 0.0001319\}$ $a \in \{1082264, 0.1024, 0.0032, 0.000566, 0.0001414\}$

Table 1: Memetic Backpropagation Epoch Allocation Strategy

We conducted a comprehensive experimental study covering all combinations of proposed memetic scheduling strategies (7 constant, 6 lower bound random, 5 upper bound random, and 6 logarithmic grown, 3 linear growth and 6 exponential growth for 33 strategies in total) across the 6 forecasting targets. Experiments were repeated 10 times per each target and schedule. In total this resulted in 1980 experiments. We additionally ran 10 repeated experiments for each of the constant strategies where the MSE improvement from backpropagation was tracked for each genome for another 420 experiments.

All experiments were performed on a high-performance cluster composed of Intel SkyLake CPUs. For the experiments comparing the performance of the different memetic scheduling strategies, each run was distributed across 18 processes and was allocated an hour run time. The experiments which evaluated backpropagation improvement utilized 54 processes each, also with an hour run time to collect more backpropagation improvement data, which allowed for more genomes to be evaluated given the same time budget.

Experiments used standard EXAMM settings. Nodes were generated randomly from a library of simple neurons (*tanh* activation function), Δ-RNN units [17], gated recurrent units (GRUs) [2], long short-term memory cells (LSTMs) [9], minimal gated units (MGUs) [26], and update-gate RNN cells (UGRNNs) [3]. A consistent set of hyperparameters was selected from prior literature on EXAMM which manually tuned them. EXAMM used 10 islands, each with a population size of 10. Backpropagation (BP) used the Adam optimizer with a learning rate of 0.001 and defaults for all other parameters ($\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 1 \times 10^{-8}$).

5 Datasets

This work utilized three multivariate datasets, with two forecasting targets per dataset. This makes for challenging real-world forecasting as the data is noisy and non-seasonal. All data was pre-normalized using min-max scaling within the range $[0 - 1]$, and the data is publicly available for reproducibility and further study[1]. The first dataset targets were *Main Flame Intensity (Main_Flm_Int)* and *Supplementary Fuel Flow (Supp_Fuel_Flow)* from a coal-fired power plant, with per-minute data taken from 12 burners across

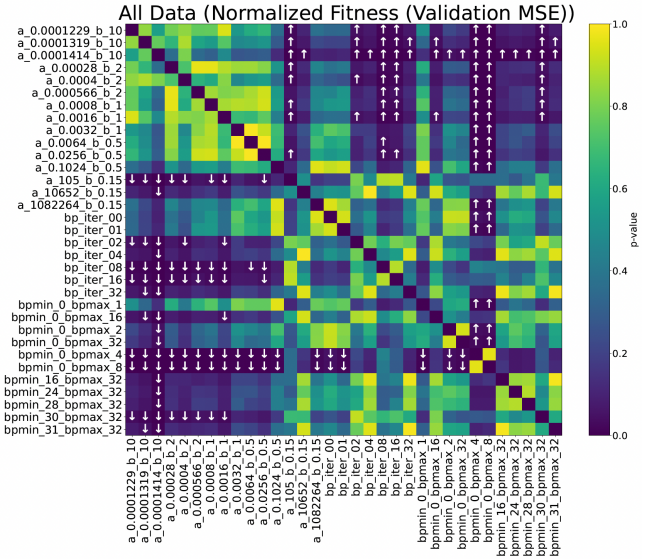


Figure 3: MSE vs Memetic Scheduling Strategy

10 days. The second dataset targets were *Pitch* and *Engine 1 Cylinder Head Temperature 1 (E1 CHT1)* from per-second flight data recorder readings from multiple Cessna 172 flights. This data comes from the National General Aviation Flight Information Database (NGAFID) [13, 25], which is a web portal for collecting, viewing and analyzing general aviation flight data [10]. The last dataset targets were *Average Converter Torque (Cm_avg)* and *Average Power (P_avg)* from wind turbine sensor data logged every 10 minutes. This data was collected between 2017 and 2020 by ENGIE's La Haute Borne open data wind farm[2].

6 Results

6.1 Scheduling Strategy Impact on Fitness and Complexity

To investigate the effects of the different memetic scheduling strategies on neuroevolution performance, we first explored if there

[1]EXAMM source code and benchmark datasets are publicly available at https://github.com/travisdesell/exact

[2]https://opendata-renewables.engie.com

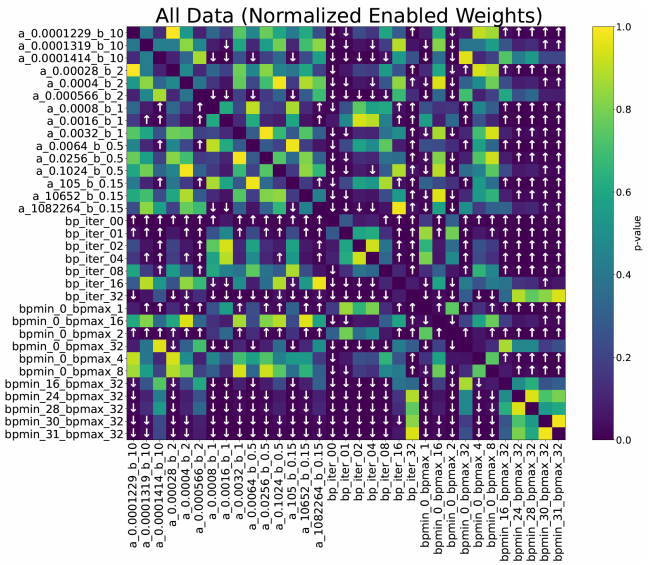


Figure 4: Model Complexity vs Memetic Scheduling Strategy

were any common trends across datasets. Figures 3 and 4 show the Mann-Whitney U-Test values for z-score normalized (per forecasting target) MSE values and total enabled weights of the best found genome from each experiment. To enhance interpretability of the pairwise Mann–Whitney U test heatmaps, statistically significant cells ($\alpha = 0.05$) are augmented with directional markers that indicate the direction of the effect. Specifically, for each significant comparison, we compute the median value for both experimental setups and annotate the heatmap cell with an arrow pointing downward (↓) if the row setup exhibits a lower median value than the column setup, and upward (↑) if it exhibits a higher median value.

These Mann–Whitney U-test p-values conclude that there is a statistically significant difference between the evaluation results achieved. Interestingly, the constant and random schedules perform better than the logarithmic, linear, and exponential schedules in the majority of cases. Additionally, random[0,4] and random[0,8] consistently perform statistically the best on average, and they also outperform some of the constant and other random schedules. Somewhat expectedly, the schedules with a consistently high number of backpropagation epochs, e.g., random[31,32] and const-32, find smaller networks, and similarly const-0, const-1 and random[0,1] find larger networks, statistically. Appendix A.3 presents heatmaps for each forecasting target separately, showing somewhat similar behavior to the overall size correlations.

However, when effects are analyzed for each forecasting target separately (see Figure 13 for heatmaps and Figure 6 for boxplots showing result ranges), the overall trends do hold across targets. For example, *Main_Flm_Int* shows that const-01 performs the best with statistical significance (in contrast to the overall results). Similarly, *Supp_Fuel_Flow* does well with const-01 and const-02; while *Cm_avg* performs well with const-08.

Data	Target	Strategy		MSE	Weights
Wind	P_Avg	const	bp = 16	0.003020	105.0
		const	bp = 08	0.003061	124.0
		const	bp = 08	0.003079	115.0
		rand	[0; 16]	0.003083	94.0
		const	bp = 32	0.003089	103.0
	Cm_avg	const	bp = 01	0.001472	145.0
		const	bp = 08	0.001502	125.0
		exp	a = 105; b = 0.15	0.001514	144.0
		const	bp = 02	0.001514	166.0
		rand	[0; 2]	0.001535	112.0
Coal	Flm_Int	rand	[24; 32]	0.000486	97.0
		rand	[0; 16]	0.000489	83.0
		rand	[0; 2]	0.000504	98.0
		rand	[0; 2]	0.000507	189.0
		rand	[0; 16]	0.000527	106.0
	Supp_Fuel	rand	[0; 8]	0.000079	109.0
		const	bp = 08	0.000086	80.0
		rand	[30; 32]	0.000092	77.0
		const	bp = 02	0.000093	79.0
		exp	a = 0.0256 b=0.5	0.000094	123.0
C172	Pitch	rand	[0; 4]	0.000211	92.0
		const	bp = 04	0.000251	90.0
		const	bp = 08	0.000257	89.0
		const	bp = 08	0.000268	95.0
		const	bp = 00	0.000271	72.0
	E1_CHT1	const	bp = 08	0.000080	94.0
		const	bp = 32	0.000144	82.0
		const	bp = 32	0.000148	72.0
		const	bp = 04	0.000205	89.0
		rand	[0; 1]	0.000247	82.0

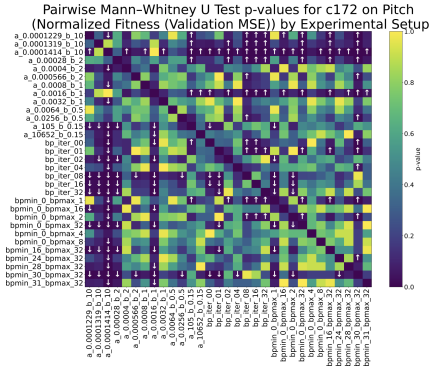
Table 2: Top 5 performing models for each target.

6.2 Model Complexity and Fitness Trends

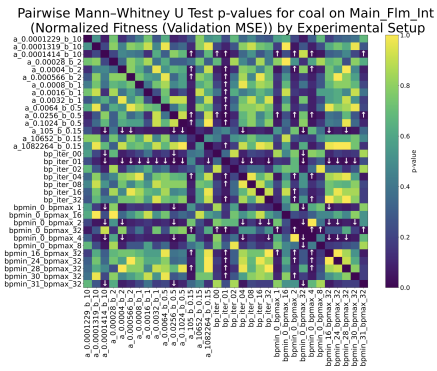
To better understand what may be contributing factors to some strategies performing better on some targets than others, we found the strategies that produced the top 5 best final genomes for each target, which are shown in Table 2. We first analyzed trends related to the relationship between MSE and size, discussed in detail in Appendix A.2 (see Figures 9 and 10), which show that each forecasting target has its own "sweet spot" basin of size for the best-performing networks. Additionally, we find that overall the wind and coal targets trended towards better MSE with larger networks, while aviation did not. We also find that *Cm_avg*, *Main_Flm_Int* and *Pitch* preferred larger network sizes than their counterparts in the same dataset.

6.3 Effect of Backpropagation Improvement

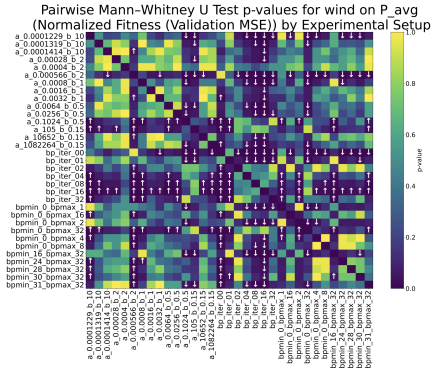
We additionally ran an extra set of experiments where we tracked the improvement it gained from backprop for the duration of evaluation using each of the constant backpropagation schedules (except for 0, which would always have no gain) within an hour wall-clock



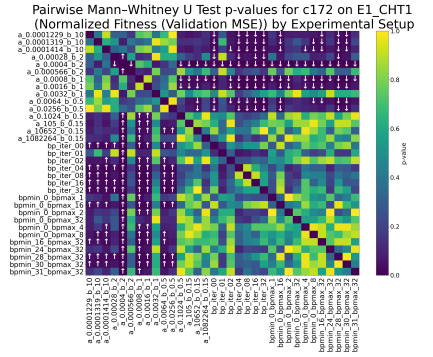
(a) Pitch prediction on Aviation dataset with 18 processes heatmap for MSE



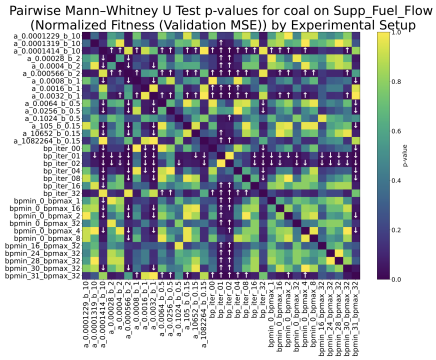
(b) Main_Flm_Int prediction on Coal dataset with 18 processes heatmap for MSE



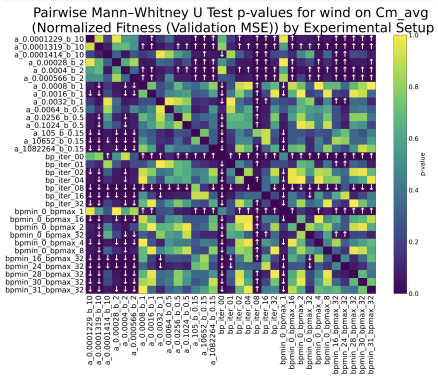
(c) P_avg prediction on Wind dataset with 18 processes heatmap for MSE



(d) E1_CHT1 prediction on Aviation dataset with 18 processes heatmap for MSE



(e) Supp_Fuel_Flow prediction on Coal dataset with 18 processes heatmap for MSE



(f) Cm_avg prediction on Wind dataset with 18 processes heatmap for MSE

Figure 5: Pairwise Mann–Whitney U test p-value heatmaps comparing best found genome MSE distributions across experimental setups

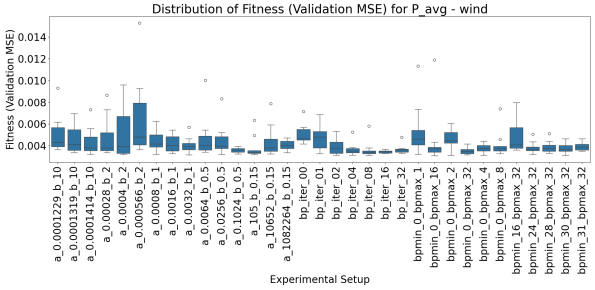
time limit. Figure 7 shows a boxplot for each target and const strategy for how much genomes improved per backpropagation epoch (e.g., the bp-16 lines show the gain for each genome trained for 16 epochs divided by 16). Note that outliers have been removed from these boxplots[3]. From this, we see that different forecasting targets do have ranges where some amounts of backprop on average provide more gain than others.

6.4 Insights on Memetic Scheduling

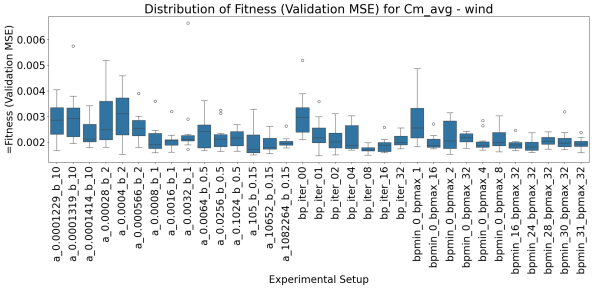
Interestingly, if we take the memetic scheduling strategies which performed best for a given forecasting target, and relate these to what amount of scheduled constant backpropagation epochs gave the most gain on average, we see that there is a good amount of overlap. *Main_Flm_Int* saw the least amount of backpropagation improvement across the board, and also performed the best with strategies that required little to no backpropagation (const-0, const-1, rand[0,2]). For the other forecasting targets, we see a good overlap. *Pitch* saw the most improvement per epoch for higher amounts of backpropagation (8, 16 and 32) and similarly saw const-8, const-16 and rand[30,32] performing well on the boxplots. *P_avg* also saw the best improvements per epoch using 8 or 16 epochs, and similarly saw const-8 and const-16 performing well in its boxplots. *Supp_Fuel_Flow* showed less overlap, with the best improvement per epoch being for 16 epochs, however 2 epochs also performed well. On its boxplots, we see it performing well for const-1 and const-2. *Cm_avg* had the best performance per epoch with 4 epochs, however 16 and 32 were also good. On its boxplots, we see const-8, some exp over curves (including its top-5), and random upper limits (which include 32 bp epochs) performing well.

We also see that for the forecasting targets that preferred larger networks within the dataset, they tended to do better with strategies that performed less backprop. This is somewhat intuitive that larger networks would require more genomes to be generated so that larger structures could be evolved as new neural network components are added through mutations when new genomes are generated.

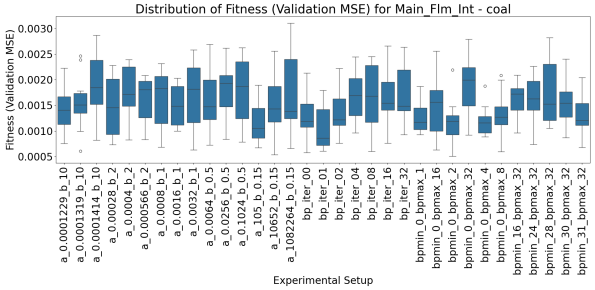
[3]Figure 8 in Appendix A.1 the overall gains (not per epoch) for the different constant strategies with outliers, which is interesting in that the range of gains across all strategies is similar, even if the average gain per epoch is not.



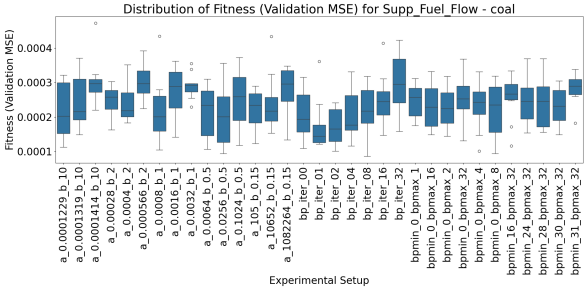
(a) Validation MSE for Wind Average Power target



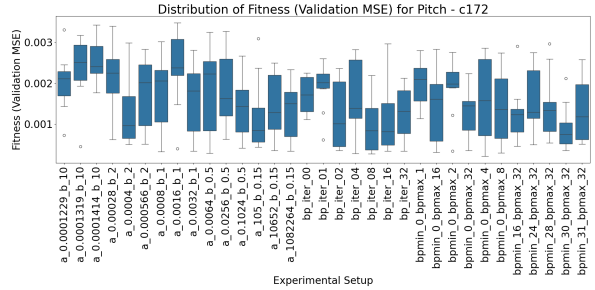
(b) Validation MSE for Wind on Cm_avg target



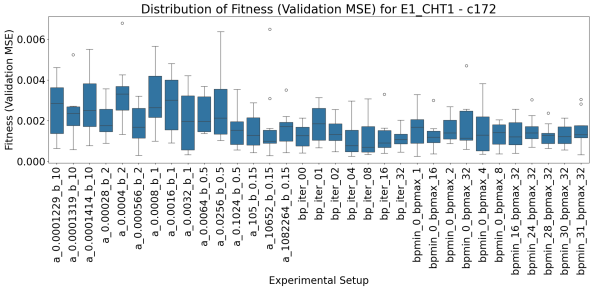
(c) Validation MSE for Coal on Main Flame Intensity



(d) Validation MSE for Coal on Supplemental Fuel Flow



(e) Validation MSE for C172 on Pitch target



(f) Validation MSE for C172 on E1 CHT1 target

Figure 6: Distributions of final best genome validation MSE across all experimental setups.

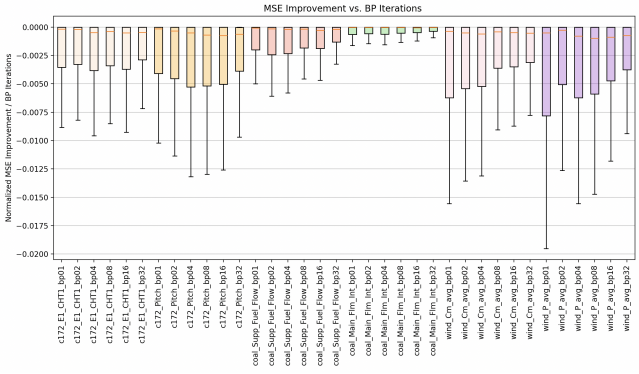


Figure 7: Average MSE improvement per epoch for the varying constant strategies, with outliers removed.

7 Discussion

The results gathered are interesting in that they ran contrary to our expectations for our research questions. Additionally, it seems that no particular memetic scheduling strategy is consistently best across forecasting tasks; however, for given tasks there are indeed clear winners for memetic scheduling. This does align with the no free lunch in machine learning theorem [1], and speaks to the utility of having a suite of memetic schedules to choose from for neuroevolution algorithms, similar to how neural network training frameworks allow for a variety of schedulers for adapting backpropagation hyperparameters.

Instead of a single strategy of memetic scheduling being a universal solution, it appears that the schedules that perform well for a forecasting task are more tied to how many backpropagation epochs provide the best average improvement per epoch during the search. This suggests that the improvement of MSE per epoch

information for a varying range of constant approaches could be used to select which schedules would be a better fit for a task. The average backpropagation improvement experiments also show that there is no direct correlation between the training duration and final performance of the evolved models, but rather that there is a correlation between the average backpropagation improvement and the best-performing strategies.

Our results do provide experimental data to suggest how to select a memetic scheduling strategy. Random schedules statistically perform the strongest, suggesting that they are a good starting point for new forecasting tasks that have not been well studied. They effectively sample the search space and often end up producing models with top performance. If the problem requires a larger network size, then the epoch scheduling strategy should be chosen to complement early exploration, such as a constant strategy with a low number of epochs, a logarithmic strategy with a small maximum limit, and a random lower limit strategy. Additionally, the average backpropagation improvement per epoch for genomes generated with a wide variety of total backpropagation epochs can be tracked to determine the most effective range.

Despite statistical testing showing that exponential strategies largely underperform, logarithmic schedules do reach the top-performing architectures for several dataset target variables, including *Cm_avg* and *Supp_Fuel_Flow*. These targets also benefit from smaller numbers of backpropagation iterations under constant scheduling. This suggests exponential scheduling can potentially be beneficial for tasks where initial exploration with a smaller number of epochs is more effective for fitness improvement. However, results find that relying solely on gradual increase does not improve the performance of the best discovered model; which is an interesting negative result in contrast to our expectations for RQ1.

Lastly, the effect of the model complexity on MSE depends largely on the task at hand. Fewer weights correlate with worse performance for the Coal dataset on both the *Supp_Fuel_Flow* and *Main_Flm_Int* target variables. The wind dataset targets require more weights to perform better: MSE decreases as the number of weights grows larger, confirming the observations from our backprop improvement scatterplots found in Figure 9 in the Appendix. The C172 dataset has an opposite correlation between model complexity and MSE. Top-performing models utilized a higher number of backpropagation iterations which lead to simpler model complexity, including const-32 for *E1_CHT1*, yet it did also include rand[0,1] in its performing models, which is a very low number of backpropagation epochs. Howver, both of these strategies end up with the similar model complexities (72-94 enabled weights). This suggests that the number of weights has a potentially higher impact on the final performance for this dataset than the choice of the strategy.

8 Conclusion and Future Work

This work investigated memetic scheduling strategies for allocating backpropagation epochs in asynchronous neuroevolution under a fixed computational budget under an effort to address evaluation time bias. By comparing constant, randomized, and scaled epoch-allocation strategies across multiple real-world time-series forecasting datasets and targets within the EXAMM framework, we demonstrated that how training effort is distributed over time shapes selection dynamics, model complexity, and final performance.

Experiments were performed across 3 datasets, with two different time series forecasting targets for each. Some targets benefitted from rapid exploration enabled by shallow training with a random lower limit on backpropagation, a low-grade constant number of backpropagation epochs, or logarithmically scaled scheduling with a small cap for the number of epochs, while others required deeper optimization to produce informative selection pressure. Together, these findings reflect the broader principle that no single memetic scheduling strategy dominates across all tasks and that effective asynchronous neuroevolution benefits from allowing access to a diverse set of scheduling options.

Across datasets, constant, randomized, and logarithmic schedules produced the majority of top-performing models. Since a range of strategies was found to be among the best, and overall, no single strategy performed best on average for all targets and datasets, we can conclude that it is useful to explore multiple scheduling strategies when evolving networks for any particular task.

The results further revealed a systematic relationship between training allocation, model complexity, and fitness. Allocating fewer backpropagation epochs enables faster exploration of the search space and supports the discovery of competitive architectures, even as model complexity increases. However, validation-based evaluation and selection of genomes may lead to overfitting and discovery of the architectures that do not improve with finetuning. Scaled schedules revealed a nuanced tradeoff: logarithmic variants occasionally discovered highly competitive genomes, while slow exponential ramp-ups underperformed consistently. These results show that gradual increases in training do not guarantee improved outcomes alone and that the timing and rate of increased training matter more than monotonic growth.

Most interestingly, an analysis of average fitness improvement per backpropagation epoch reveals a correlation between lower final MSE and faster per-epoch learning. This can be used as an effective strategy to choose memetic scheduling directly related to task-specific learning dynamics and offers a principled basis for selecting or adapting schedules during evolution.

Overall, this work shows that evaluation-time bias in asynchronous neuroevolution can be mitigated through simple, lightweight scheduling strategies that reshape evaluation dynamics rather than enforcing uniformly deeper training. Constant and randomized schedules achieve this by promoting efficient exploration and reliable selection, enabling asynchronous algorithms to evolve higher-quality solutions under fixed time budgets. Future work can build on these results by developing adaptive schedulers that leverage per-epoch improvement signals to allocate training dynamically during evolution.

9 Acknowledgments

This research is partially supported by a gift from Lenovo, and used CPU resources provided by Research Computing at Rochester Institute of Technology, and we thank the team for their support [18].

References

[1] Stavros P Adam, Stamatios-Aggelos N Alexandropoulos, Panos M Pardalos, and Michael N Vrahatis. 2019. No free lunch theorem: A review. *Approximation and optimization: Algorithms, complexity and applications* (2019), 57–82.

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[3] Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo. 2016. Capacity and trainability in recurrent neural networks. *arXiv preprint arXiv:1611.09913* (2016).

[4] Travis Desell. 2017. Large scale evolution of convolutional neural networks using volunteer computing. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Berlin, Germany) *(GECCO '17)*. Association for Computing Machinery, New York, NY, USA, 127–128. doi:10.1145/3067695.3076002

[5] Travis Desell, AbdElRahman ElSaid, and Alexander G. Ororbia. 2020. An Empirical Exploration of Deep Recurrent Connections Using Neuro-Evolution. In *The 23nd International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps 2020)*. Seville, Spain.

[6] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. 2008. Accelerated Neural Evolution through Cooperatively Coevolved Synapses. *Journal of Machine Learning Research* 9 (05 2008), 937–965. doi:10.1145/1390681.1390712

[7] N. Hansen and A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9, 2 (2001), 159–195.

[8] Tomohiro Harada. 2025. A frequency-based parent selection for reducing the effect of evaluation time bias in asynchronous parallel multi-objective evolutionary algorithms. *Natural Computing* 24, 2 (01 Jun 2025), 211–225. doi:10.1007/s11047-022-09940-z

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[10] Kelton Karboviak, Sophine Clachar, Travis Desell, Mark Dusenbury, Wyatt Hedrick, James Higgins, John Walberg, and Brandon Wild. 2018. Classifying aircraft approach type in the national general aviation flight information database. In *Computational Science–ICCS 2018: 18th International Conference, Wuxi, China, June 11–13, 2018, Proceedings, Part I 18*. Springer, 456–469.

[11] Joshua Karns and Travis Desell. 2025. Evaluation Time Bias in Asynchronous Evolutionary Algorithms: A Replication Study and a Novel Mitigation Strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference* (NH Malaga Hotel, Malaga, Spain) *(GECCO '25)*. Association for Computing Machinery, New York, NY, USA, 13–21. doi:10.1145/3712256.3726458

[12] Amit Dilip Kini, Swaraj Sambhaji Yadav, Aditya Shankar Thakur, Akshar Bajrang Awari, Zimeng Lyu, and Travis Desell. 2023. Co-evolving recurrent neural networks and their hyperparameters with simplex hyperparameter optimization. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. 1639–1647.

[13] Aidan P LaBella, Joshua A Karns, Farhad Akhbardeh, Travis Desell, Andrew J Walton, Zechariah Morgan, Brandon Wild, and Mark Dusenbury. 2022. Optimized flight safety event detection in the national general aviation flight information database. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 1570–1579.

[14] Zimeng Lyu, AbdElRahman ElSaid, Joshua Karns, Mohamed Mkaouer, and Travis Desell. 2021. An Experimental Study of Weight Initialization and Lamarckian Inheritance on Neuroevolution. In *Applications of Evolutionary Computation*, Pedro A. Castillo and Juan Luis Jiménez Laredo (Eds.). Springer International Publishing, Cham, 584–600.

[15] Zimeng Lyu, Joshua Karnas, AbdElRahman ElSaid, Mohamed Mkaouer, and Travis Desell. 2021. Improving Distributed Neuroevolution Using Island Extinction and Repopulation. *The 24th International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps)* (2021).

[16] Alexander Ororbia, Ahmed Ahmed Elsaid, and Travis Desell. 2019. Investigating Recurrent Neural Network Memory Structures using Neuro-Evolution. arXiv:1902.02390 [cs.NE] https://arxiv.org/abs/1902.02390

[17] Alexander G. Ororbia II, Tomas Mikolov, and David Reitter. 2017. Learning Simpler Language Models with the Differential State Framework. *Neural Computation* 0, 0 (2017), 1–26. arXiv:https://doi.org/10.1162/neco_a_01017 doi:10.1162/neco_a_01017 PMID: 28957029.

[18] Rochester Institute of Technology. 2019. Research Computing Services. doi:10.34788/0S3G-QD15 Accessed 2026-01-23.

[19] Eric O. Scott and Kenneth A. De Jong. 2015. Evaluation-Time Bias in Asynchronous Evolutionary Algorithms. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Madrid, Spain) *(GECCO Companion '15)*. Association for Computing Machinery, New York, NY, USA, 1209–1212. doi:10.1145/2739482.2768482

[20] Eric O. Scott and Kenneth A. De Jong. 2016. Evaluation-Time Bias in Quasi-Generational and Steady-State Asynchronous Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (Denver, Colorado, USA) *(GECCO '16)*. Association for Computing Machinery, New York, NY, USA, 845–852. doi:10.1145/2908812.2908934

[21] Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. 2009. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life* 15, 2 (2009), 185–212. doi:10.1162/artl.2009.15.2.15202

[22] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evol. Comput.* 10, 2 (June 2002), 99–127. doi:10.1162/106365602320169811

[23] Aditya Shankar Thakur, Akshar Bajrang Awari, Zimeng Lyu, and Travis Desell. 2023. Efficient Neuroevolution Using Island Repopulation and Simplex Hyperparameter Optimization. In *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1837–1842.

[24] Mouadh Yagoubi, Ludovic Thobois, and Marc Schoenauer. 2011. Asynchronous Evolutionary Multi-Objective Algorithms with heterogeneous evaluation costs. In *2011 IEEE Congress of Evolutionary Computation (CEC)*. 21–28. doi:10.1109/CEC.2011.5949593

[25] Hong Yang and Travis Desell. 2022. A Large-Scale Annotated Multivariate Time Series Aviation Maintenance Dataset from the NGAFID. arXiv:2210.07317 [cs.LG] doi:10.48550/arXiv.2210.07317

[26] Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. 2016. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing* 13, 3 (2016), 226–234.

A Appendix

A.1 Overall Backpropagation Improvements with Outliers

Figure 8 presents boxplots for the actual backpropagation improvement values for each genome generated in the constant experiments. This is particularly interesting as we see similar possible ranges of improvement for genomes across all constant backpropagation strategies 1 through 32. Figure 7 divides each of these values by the number of backpropagation epochs in the strategy (so it shows average per-epoch improvement) and removes outliers. Figure 8 can explain why we see varying ranges on the average per-epoch improvement values for the different const strategies, as it has removed the outliers, which means for the strategies with a lower average improvement, they had more genomes come back with less improvement even if the overall range was similar.

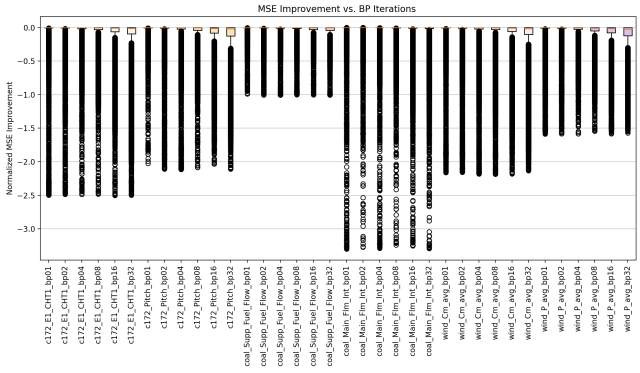


Figure 8: MSE improvements for the varying constant strategies with outliers, unscaled by total backpropagation epochs evaluated per genome.

A.2 MSE vs Size Analysis

Figure 9 presents scatter plots showing the relationship between model size and fitness (MSE) for each forecasting target. Polynomial regressions are shown against the data, which are not particularly significant. The best found genome is highlighted with a red star in each scatterplot. One thing which is noticeable is that each forecasting target has a particular basin for a model size where the best performing genomes are found, which is why there is not a particular linear trend.

We note that for wind, *Cm_avg*'s best genome is found with significantly more weights than *P_avg* (~150 vs ~100). Similarly, for coal, *Supp_Fuel_Flow*'s best genome is found at around ~75 weights vs ~100 for *Main_Flm_Int*. Aviation is similar to coal, with *E1_CHT1* having the best MSE at ~75 weights, and *Pitch* at ~90 weights. We notice for all models this best found genome is typically near the center of the bottom of the cluster of points in the scatter plot, apart from *Cm_avg*, which suggests that each forecasting target has a "sweet spot" for model size.

Figure 10 colors the points in the scatterplot by strategy type, where the groupings are constant strategies, random lower limit, random upper limit, linear, exponential (upper curve) and logarithmic (lower curve). We find that for the wind dataset, all these strategies tend towards lower MSE values with larger networks (minus the basin effect). For coal, however, we find that the exponential upper curve strategies trend towards worse fitness with larger networks, while the rest do not. This correlates with the results found which measured backpropagation improvement, as candidate models for the coal dataset do not improve much with any number of iterations compared to the results on other datasets. Lastly for aviation, we find all the strategies trend worse with larger networks.

A.3 U-Test Heatmaps for Network Size

Figure 11 shows heatmaps of the significance of each memetic scheduling strategy compared to each other in relation to the size of the models they generated. Overall, these show similar trends as the combined heatmap; however, each forecasting target does have its own set of strategies, which generate larger or smaller networks. Interestingly, for coal, const-0 shows statistically larger networks; however, this is not as much the case for the other datasets. *Pitch* is also interesting in that the middle-range constant strategies produce statistically significantly larger networks (which may be due to *Pitch* performing better with larger networks). However, overall, the random upper bound and constant strategies with more backpropagation, as expected, produce smaller networks.

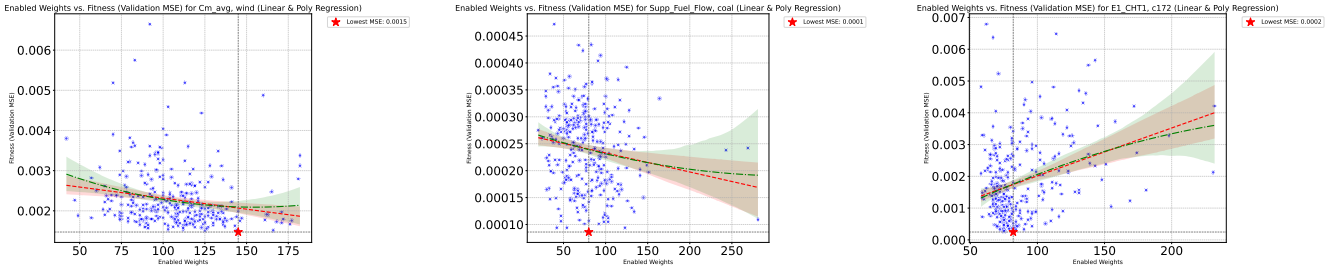
A.4 Boxplots for Network Size

Figure 12 shows the model complexities in relation to presented scheduling strategies separately for all target variables. Expectedly, there is a clear downward trend in enabled weights for the constant strategies as the number of epochs grows. Same downward trend in model complexity is observed with random upper limit strategies as the lower bound increases (as the upper bound stays 32) and with exponential strategies as the maximum limit increases. As it comes to logarithmic, linear, and exponential strategies that are dependent on b-value, the complexity of the model tends to decrease as the b-value increases.

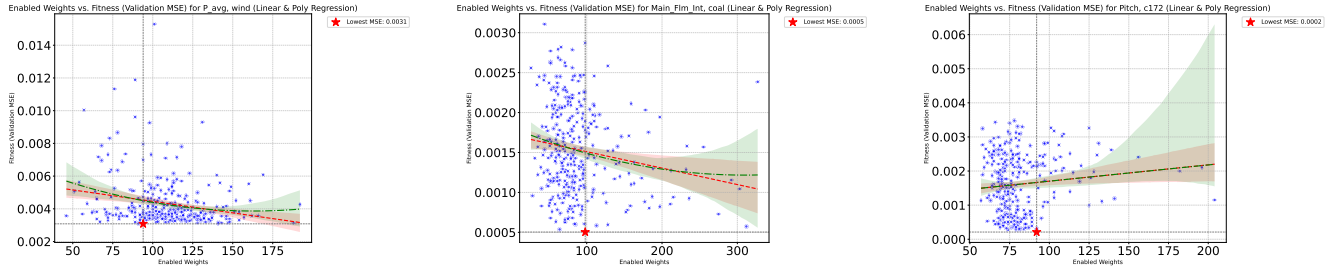
A.5 Architectures of the best global genomes

The final genome architecture of the best global genomes for each of the 6 target variables for 3 datasets. Interestingly, the architecture that performed the best on Pitch doesn't have any hidden layers, which shows that the combination of feedforward and recurrent connections is enough to produce a highly effective neural network. On the contrary, the number of weights is high (equal to 92), which is surprising and counterintuitive at first glance, highlighting the difference between human-designed neural networks and the ones evolved with genetic algorithms. Overall, architectures of the best global genomes had a low number of hidden neurons (up to 3) but a large number of connections.

Received ; revised ; accepted

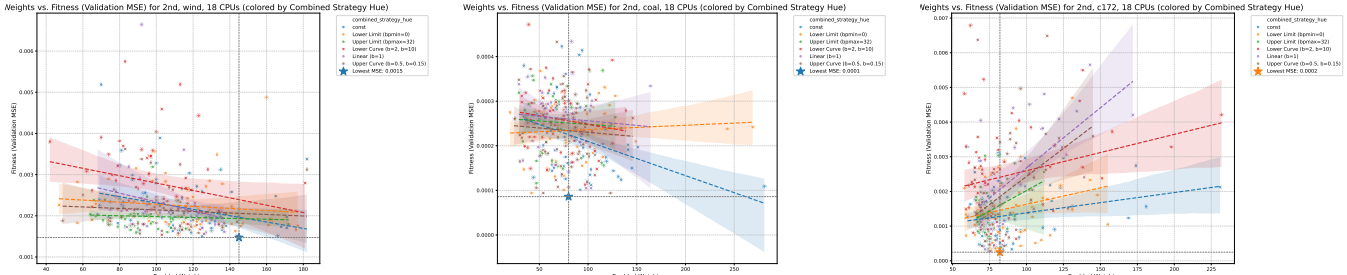


(a) Cm_avg, Wind (b) Supp_Fuel_Flow, Coal (c) E1_CHT1, C172

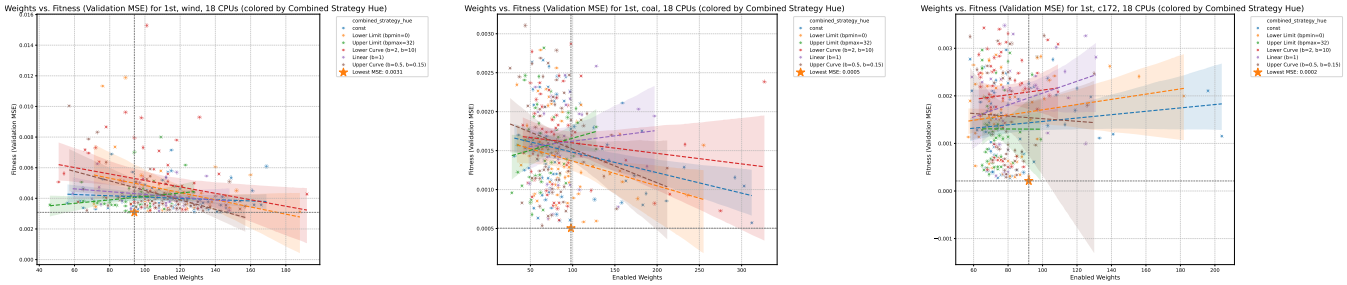


(d) P_avg, Wind (e) Main_Flm_Int, Coal (f) Pitch, C172

Figure 9: Relationship between enabled weights and validation MSE across all strategies for each target.

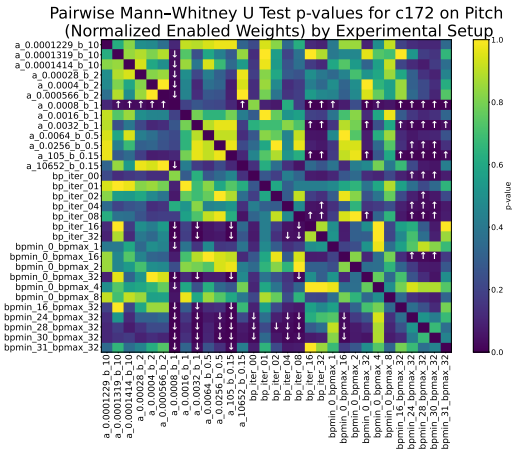


(a) Cm_avg, Wind (b) Supp_Fuel_Flow, Coal (c) E1_CHT1, C172

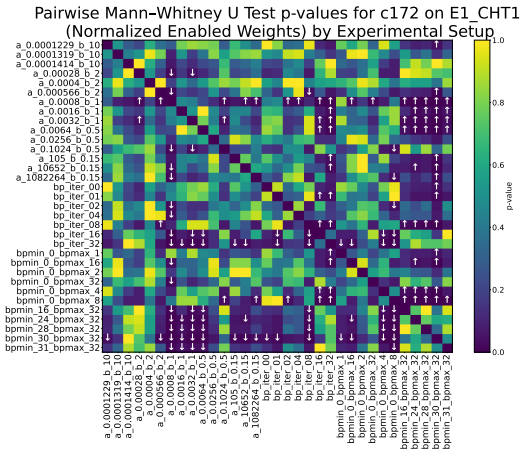


(d) P_avg, Wind (e) Main_Flm_Int, Coal (f) Pitch, C172

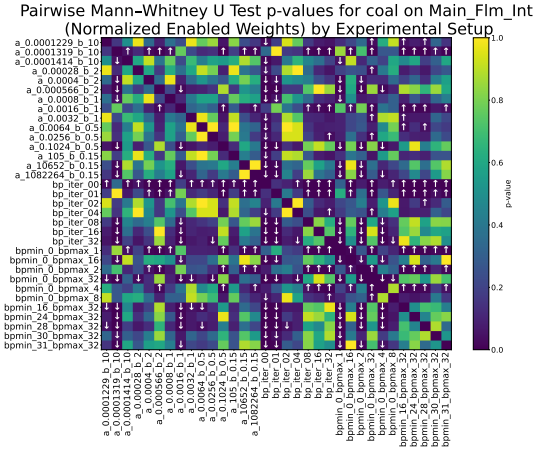
Figure 10: Enabled weights vs. validation MSE for the combined strategy setting. Points are colored by strategy class, and dashed lines indicate per-strategy linear trends



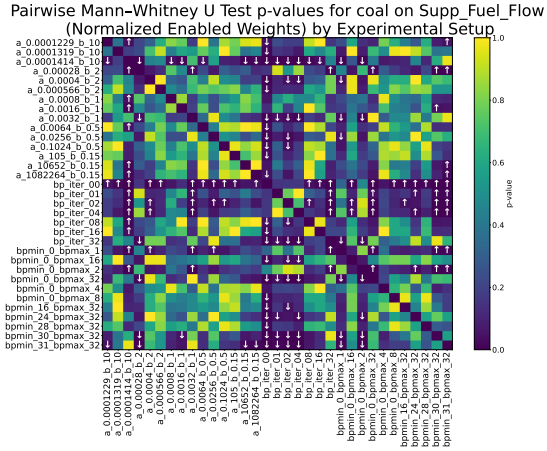
(a) Enabled weights, Cm_avg, Wind



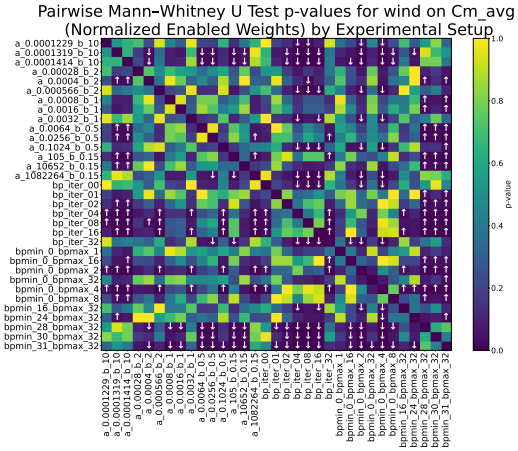
(b) Enabled weights, P_avg, Wind



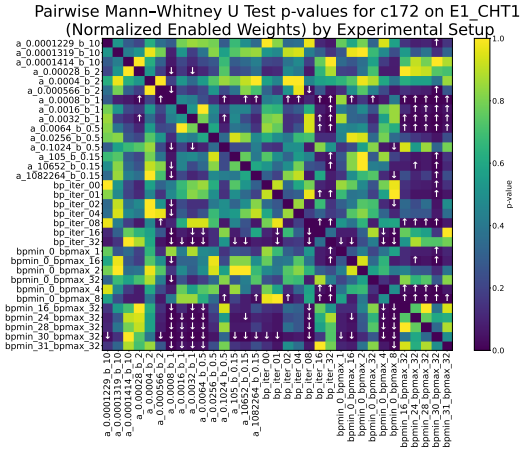
(c) Enabled weights, Supp_Fuel_Flow, Coal



(d) Enabled weights, Main_Flm_Int, Coal

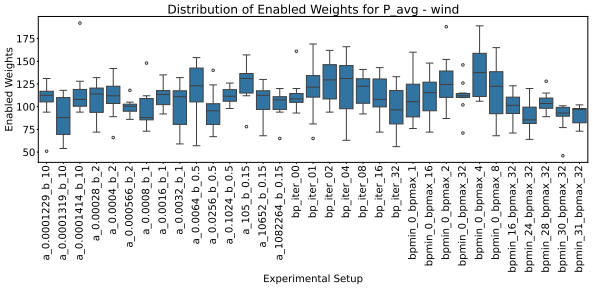


(e) Enabled weights, Pitch, C172

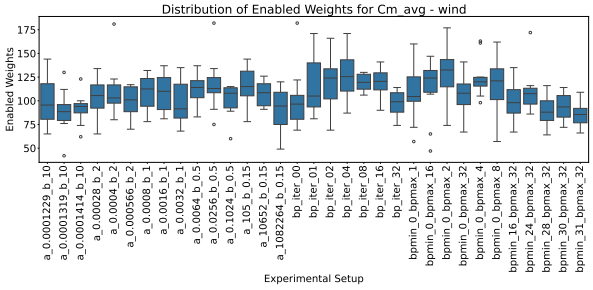


(f) Enabled weights, E1_CHT1, C172

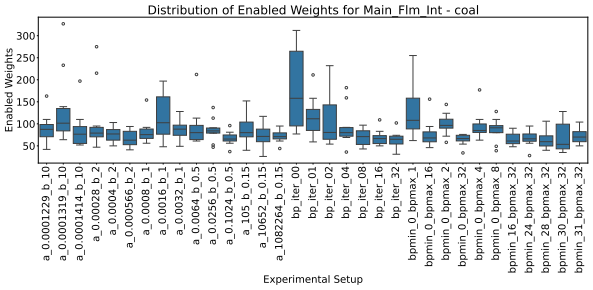
Figure 11: Pairwise Mann–Whitney U test p-value heatmaps comparing enabled-weights distributions across experimental setups



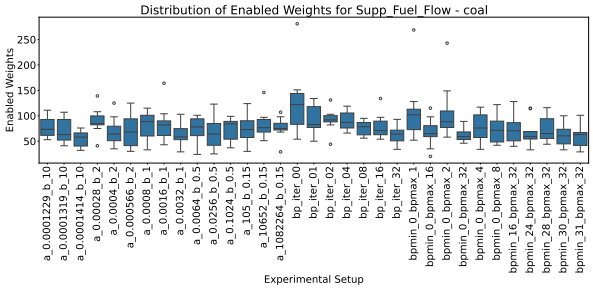
(a) Enabled weights for Wind on Average Power target



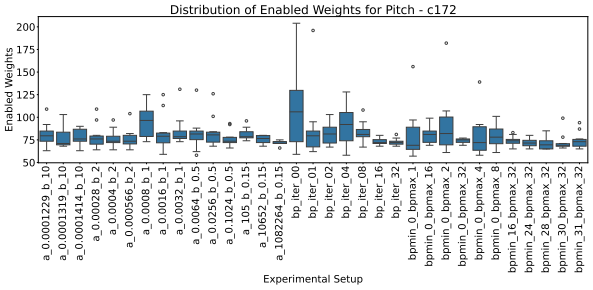
(b) Enabled weights for Wind on Cm_avg target



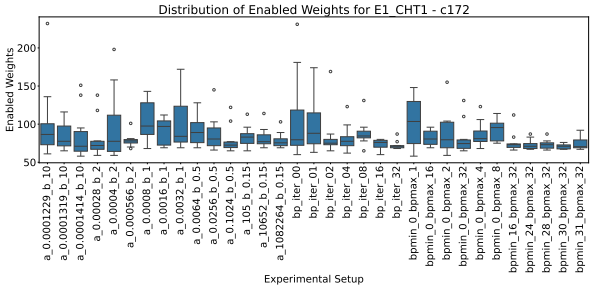
(c) Enabled weights for Coal on Main Flame Intensity



(d) Enabled weights for Coal on Supplemental Fuel Flow

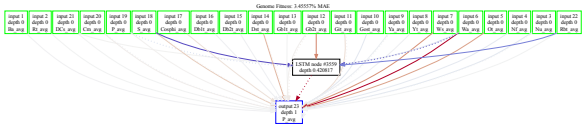


(e) Enabled weights for C172 on Pitch

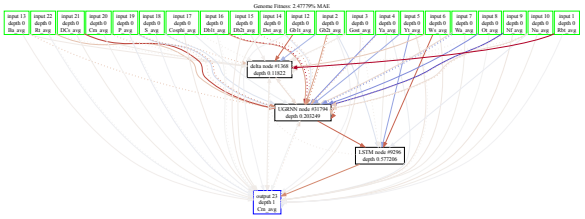


(f) Enabled weights for C172 on E1 CHT1

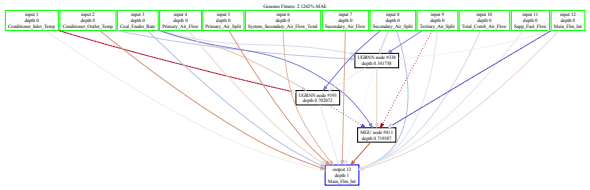
Figure 12: Distributions of final best genome size (number of enabled weights) all experimental setups.



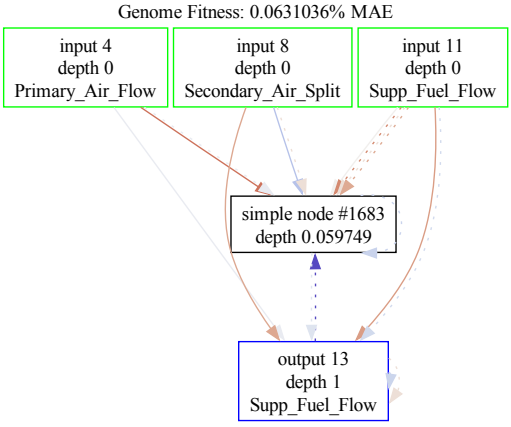
(a) P_avg best global genome architecture



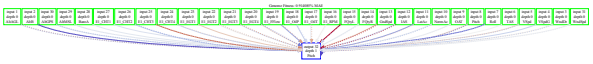
(b) Cm_avg best global genome architecture



(c) Main_Flm_Int best global genome architecture



(d) Supp_Fuel best global genome architecture



(e) Pitch best global genome architecture

(f) E1_CHT1 best global genome architecture

Figure 13: Pairwise Mann–Whitney U test p-value heatmaps comparing best found genome MSE distributions across experimental setups