Improving Distributed Neuroevolution Using Island Extinction and Repopulation[⋆]

Zimeng Lyu, Joshua Karns, AbdElRahman ElSaid, Mohamed Mkaouer, and Travis Desell

Rochester Institute of Technology, Rochester, NY 14623, USA
zimenglyu@mail.rit.edu, josh@mail.rit.edu, aae8800@rit.edu, mwmvse@rit.edu, tjdvse@rit.edu

Abstract. Neuroevolution commonly uses speciation strategies to better explore the search space of neural network architectures. One such speciation strategy is the use of islands, which are also popular in improving the performance of distributed evolutionary algorithms. However, islands may experience stagnation, which prevents their convergence towards better solutions and can result in wasted computation. This work evaluates utilizing an island extinction and repopulation mechanism to avoid premature convergence using Evolutionary eXploration of Augmenting Memory Models (EXAMM), an asynchronous island based neuroevolution algorithm that progressively evolves recurrent neural networks (RNNs). In island extinction and repopulation, all members of the worst performing island are erased periodically and repopulated with mutated versions of the global best RNN. This island based strategy is additionally compared to NEAT's (NeuroEvolution of Augmenting Topologies) speciation strategy. Experiments were performed using two different real-world time series datasets (coal-fired power plant and aviation flight data). With statistical significance, results show that in addition to being more scalable, this island extinction and repopulation strategy evolves better global best genomes than both EXAMM's original island based strategy and NEAT's speciation strategy. The extinction and repopulation strategy is easy to implement, and can be generically applied to other neuroevolution algorithms.

Keywords: NeuroEvolution · Neural Architecture Search · Extinction · Repopulation · Recurrent Neural Networks · Time Series Prediction.

1 Introduction

Neuroevolution (NE), or the evolution of artificial neural networks (ANNs), has been widely applied as a neural architecture search strategy for a variety of

⋆ This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Combustion Systems under Award Number #FE0031547 and by the Federal Aviation Administration and MITRE Corporation under the National General Aviation Flight Information Database (NGAFID) award.

machine learning problems, including image classification, natural language processing, reinforcement learning and time series data prediction [32, 23]. As the complexity of the tasks ANNs are trained to solve increases, manually designing the network becomes impossible, especially when they may need to be optimized for multiple criteria such as cost, latency, power consumption, and accuracy. NE provides a way to evolve ANNs in large and high dimensional space without prior knowledge, searching through the growing number of ANN building blocks, such as activation functions, memory cells, convolutional filter, and feature map types, while at the same time determining network topology.

NE algorithms tend to be computationally expensive in that candidate neural networks need to be trained or otherwise evaluated to determine their fitness. Because of this, most NE algorithms are distributed in order to make progress more quickly. The use of islands, as a common strategy in distributed evolutionary algorithms, has been shown to potentially provide significant speedup beyond distribution, as islands can evolve independently with smaller populations as different species more quickly, with the periodic transfer best found solutions [1]. However, if we look into how species evolve, we find that different species converge and evolve at different speeds. Some species show premature convergence and can become stuck at local optima. In this work, we take inspiration from *extinction* and *repopulation* mechanisms, which have shown to speed up evolution and speciation in the real world [25] as well as in EAs [11, 20, 15, 37] and apply them to distributed NE algorithms.

This work presents a novel extinction and repopulation strategy that repopulates poorly performing islands by first removing all the genomes in the island and then repopulating it with random mutations of the global best genome. Experiments explore how the frequency of extinction and the number of random mutations applied to the global best genome affect the island based evolution strategy. This was done using the Evolutionary eXploration of Augmenting Memory Models (EXAMM) [26] algorithm that evolves deep Recurrent Neural Networks (RNN) for time series data prediction. We further implemented NEAT's speciation strategy in EXAMM, so it could be fairly compared as a benchmark. To test the robustness of this strategy, we used two real world, non-seasonal, large scale time series data sets from aviation data and a coal-fired power plant. Results show that the new extinction and repopulation based strategies outperform baseline EXAMM and NEAT's speciation strategy with statistical significance.

2 Related Work

According to the history of biological evolution, extinction plays an important role in the process of evolution [25, 8, 7]. It will erase the species that are not suited for their niches and create opportunities for new species to emerge. Similarly in EAs, research conducted by Greenwood *et al.* [11] and Krink *et al.* [20] show that applying mass extinction can enhance the performance in evolutionary search. A mass extinction mechanism was also shown to significantly improve hybrid particle swarm optimizer model performance [39], and more recently, Lehman *et al.* proved using extinction events is an effective mechanism for divergent search algorithms [21].

In addition to extinction, the biological concepts of migration and repopulation are also applied in EAs to improve performance. Grefenstette *et al.* have investigated replacing a percentage of the population with randomly generated individuals [12]. De Falco *et al.* take the inspiration of biological invasion and migrate genomes into other subpopulations to compete with native genomes [4]. Hernandez *et al.* replace a fraction of the population with selective repopulation [15], and Wan *et al.* take genomes generated from "elite clusters" to randomly replace individuals in the population [37].

In terms of speciation for NE, NEAT (Evolving Neural Networks through Augmenting Topologies) [31] presented one of the first speciation strategies for NE, where genomes speciate by tracking historical genes and measuring the distance between new genomes and an existing species. This has been extended with Natural Evolution Speciation for NEAT (NENEAT) [18], which replaces NEAT's speciation with a cladistic strategy, where all the genomes in the same species share a subset of nodes. Trujillo *et al.* speciate evolutionary robotics in the behavior space [35], whereas NEAT and NENEAT speciate genomes in topology space. Hadjiivanov *et al.* also investigated a complexity-based speciation strategy, which groups genomes by their number of hidden neurons [13].

Other than strategies to divide genomes into different species, evolutionary rules such as mutation, crossover, weight initialization, and distance functions, can be used to drive speciation and improve EA performance. Verbancsics *et al.* investigated the effect of crossover and mutation on the NE speciation strategies [36]. Mathias *et al.* explored the use of extinction for path finding GAs in a continuous environment [24]. Sun *et al.* applied a variable length gene encoding to avoid the network depth constraint for solving complex problems [34]. Krčah *et al.* modified NEAT's fitness evaluation rule by changing the capacity of species dynamically [19]. Lastly, instead of using objective functions to measure the fitness of a genome, Lehman *et al.* instead drove search using behavior novelty [22].

In contrast to these strategies, the extinction and repopulation presented in this paper is easily adaptable to any NE or EA method utilizing islands, and does not require fitness modifications or the calculation of expensive distance metrics, which in many cases need to be evaluated against all other individuals or species within the population, which can significantly degrade performance in a distributed EA. In fact, when comparing to NEAT's speciation strategy, this issue became apparent as we were limited in the number of processors we could scale to (see section 4.4). The strategy is also easier to use and tune, only requiring users to specify the frequency of extinction and repopulation events.

3 Methodology

3.1 Evolutionary eXploration of Augmenting Memory Models

This work utilizes the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm [26] to explore the extinction and repopulation of islands. EXAMM evolves progressively larger RNNs through a series of mutation and crossover (reproduction) operations. Mutations can be edge-based: *split edge*, *add edge*, *enable edge*, *add recurrent edge*, and *disable edge* operations, or work as higher-level node-based mutations: *disable node*, *enable node*, *add node*, *split node* and *merge node*. The type of node to be added is selected uniformly at random from a suite of simple neurons and complex memory cells: Δ-RNN units [27], gated recurrent units (GRUs) [2], long short-term memory cells (LSTMs) [16], minimal gated units (MGUs) [40], and update gate RNN cells (UGRNNs) [3]. This allows EXAMM to select for the best performing recurrent memory units. EXAMM also allows for *deep recurrent connections*, which enables the RNN to directly use information beyond the previous time step. These deep recurrent connections have proven to offer significant improvements in model generalization, even yielding models that outperform state-of-the-art gated architectures [5]. EXAMM has both a multithreaded implementation and a Message Passing Interface (MPI) implementation for distributed use on high performance computing resources. To the authors' knowledge, these capabilities are not available in other neuroevolution frameworks capable of evolving RNNs, which is the primary reason EXAMM was selected for this work.

EXAMM uses an asynchronous island based evolution strategy with a fixed number of islands n, each with an island capacity m. During the evolution process, islands go through two phases: *initialized*, and *filled*. During the *initialization* phase, each island starts with one seed genome, which is the minimal possible feed-forward neural network structure with no hidden layers, with the input layer fully connected to the output layer. Worker processes repeatedly request genomes to evaluate from the master process using a work stealing approach.

On receiving a genome, the worker then evaluates its *fitness*, calculated as mean squared error (MSE) on a validation data set after being trained by stochastic back propagation through time (BPTT). When reported back to the master process, if the island is not full, it is inserted into the island, or if the *fitness* is better than the worst genome in that island, it will replace the worst genome. The master generates new genomes from islands in a round-robin manner, by doing one random mutation on randomly selected genomes from an island until that island reaches its maximum capacity m, and its status becomes *filled*. When all islands are *filled*, they repopulate through inter-island crossover, intra-island crossover and mutation operations. *Intra-island crossover* selects two random genomes from the same island, and the child gets inserted back to where its parents come from. *Inter-island crossover* selects the first parent at random from the target island, and the second parent is the best genome from another randomly selected island. As islands are distinct sub-populations and evolve independently, the only chance for the islands to exchange genes is through *inter-island crossover*.

The weights of the seed genome, generated during the *initialization* phase, are initialized uniformly at random between -0.5 and 0.5, or by the Kaiming [14] or Xavier [9] strategies. After this, RNNs, generated through mutation or crossover, reuse parental weights, allowing the RNNs to train from where the parents left off, *i.e.*, *"Lamarckian" weight initialization*. Mutation operations may add new nodes or/and edges that are not present in the parent, and these are initialized using a normal distribution of the average μ and variance σ^2 of the best parent's weights. During crossover, in the case where an edge or node exists in both parents, the child weights are generated by recombining the parents' weights. Given a random number $-0.5 <= r <= 1.5$, a child's weight w_c is set to $w_c = r(w_{p2} - w_{p1}) + w_{p1}$, where w_{p1} is the weight from the more fit parent, and w_{p2} is the weight from the less fit parent. This allows the child weights to be set along a gradient calculated from the weights of the two parents, allowing for informed exploration of the weight space of the two parents.

3.2 EXAMM Island Repopulation and Extinction

While investigating the performance of the EXAMM algorithm, it was observed that islands do not converge at the same speed, and some are stagnant. A naive approach to repopulation would be to erase the prematurely converged island and restart from scratch, however, given that the other islands will have well-developed genomes, it might be impossible for the restarted island to ever catch up. Further, it would involve re-examining the preliminary regions of the search space. Taking inspiration from nature, most new species are not directly evolved from a single-celled organism. In common cases, a group of organisms evolves in a certain direction to adapt to a new niche, and eventually new species emerge. With this as motivation, we utilize the idea of immigrating existing genomes to the worst island for repopulation. In addition, we opt for using mutations on these immigrating genomes to bring innovation through the evolution process, allowing them to potentially further explore new niches.

Using our proposed strategy, the EXAMM island repopulation strategy has now three phases: *initialization*, *filled*, and *repopulation*. The *initialization* phase is the same as original EXAMM. However, after all the islands become *filled*, we introduce periodic extinction mechanisms to the worst performing island. At the time of an extinction mechanism, all the islands are ranked based on their best genome's fitness, and all the genomes in the worst island are removed. Then this depopulated island moves into the *repopulation* phase. During this phase, new genomes for the island are generated by randomly performing m mutations on the global best genome until the island is full and goes back to *filled* status. To handle the asynchronous RNN evaluation in EXAMM, when a worker processes return trained RNNs generated from before the extinction mechanism, they are not added to the repopulating island and instead immediately removed. Through this, these periodic extinction mechanisms encourage further diversity in the entire population.

To repopulate an island, there are two ways EXAMM uses to find and erase the worst performing island: 1) the worst island can be repopulated at any extinction event, and 2) if an island becomes the worst island again after being repopulated, it has to wait for e other extinction events before it can become extinct and be repopulated again. The difference between the two strategies is that the second gives the newly repopulated island more time to evolve. As a repopulated island might need more time to evolve and find new well performing genomes, if the extinction mechanisms keep erasing the worst island regardless of its recent repopulation, the same island might end up being repeatedly repopulated. On the other hand, mutated global best genomes can perform better or worse than the original ones, especially when more than one mutation are applied together. If an island has not caught up with the rest of the population by the next extinction mechanism, it may have become stagnant in different local optima.

3.3 NEAT Speciation

To compare to a benchmark strategy, we utilized the speciation strategy from the popular Neuro-Evolution of Augmenting Topologies (NEAT) [31]. Newer versions of NEAT, such as HyperNeat [33] were not used because they cannot easily be applied to recurrent neural networks, especially those with modern memory cells. Further, HyperNEAT still utilizes NEAT's speciation strategy to generate its compositional pattern producing networks as opposed to a different strategy. Instead of using an island strategy, NEAT organizes genomes into small sub-populations, or species. New genomes are inserted into the first species in which the distance δ between the new genome and a random genome inserted from last generation is less than threshold δ_t. The distance is calculated using a distance function, δ:

$$\delta = \frac{C_1 E}{N} + \frac{C_2 D}{N} + C_3 \overline{W} \tag{1}$$

where E and D are the excess and disjoint genes between two genomes, and $\overline{W}$ is the weight difference of matching genes. $c1$, $c2$, and $c3$ are hyperparameters adjusting the weight of those factors and N is the number of genes in the larger genome.

NEAT does not limit the number of species or the species capacity. The species size is controlled by *explicit fitness sharing* [10]. A genome's adjusted fitness f'_i is calculated by:

$$f'_i = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))} \tag{2}$$

When distance between two genomes i and j exceeds a threshold δ_t, sh is set to 0, sh is 1 otherwise [30]. Genomes who have a high adjusted fitness f'_i are removed. If the best fitness of a species does not improve in 15 generations, this species loses the ability to reproduce. If the entire population does not improve for 20 generations, then only the top 2 species are allowed to reproduce.

4 Results

4.1 Data Sets

Two datasets were utilized to test the varying speciation strategies[1]. The first comes from a coal-fired power plant, and the second comes from a selection of 10 flights worth of data from the National General Aviation Flight Information Database (NGAFID). Both datasets are multivariate, with 12 and 31 parameters, respectively, non-seasonal, and the parameter recordings are not independent. Furthermore, they are very long – the aviation time series range from 1 to 3 hours worth of per-second data, while the power plant data consists of 10 days worth of per-minute readings. *Main flame intensity* was chosen as the prediction parameter from the coal data set, and *pitch* was chosen as the prediction parameter from the flight data set.

4.2 Hyperparameter Settings

Each EXAMM run used 10 islands, each with a maximum capacity of 10 genomes. EXAMM was then allowed to evolve and train $20,000$ genomes (RNNs) through its neuroevolution process. New RNNs were generated via mutation at a rate of 70%, intra-island crossover at a rate of 20%, and inter-island crossover at a rate of 10%. 10 out of EXAMM's 11 mutation operations were utilized (all except for *split edge*), and each was chosen with a uniform 10% chance. EXAMM generated new nodes by selecting from simple neurons, Δ-RNN, GRU, LSTM, MGU, and UGRNN memory cells uniformly at random. Recurrent connections could span any time-skip generated randomly between $\mathcal{U}(1, 10)$.

In related work, EXAMM has been shown to significantly outperform standard NEAT [6]. We attribute this to the fact that EXAMM can create nodes from a library of recurrent memory cells, has additional node level mutations, uses a Lamarckian/epigenetic weight inheritance strategy, and trains RNNs via stochastic gradient descent and back propagation through time (BPTT). On the other hand, NEAT only deploys edge-level mutations and has a rather simple evolutionary strategy to assign weights to networks. Additionally, NEAT was not designed for large scale parallelism, and uses a synchronous strategy for iteratively generating new populations. Due to this, we implemented NEAT's speciation strategy within the EXAMM framework to compare the speciation strategies without confounding effects from other algorithmic details.

NEAT typically generates 150 genomes per generation, and if a species has not improved its best fitness within 15 generations, it will be disabled and not allowed to procreate. It will further disable the entire population except for the top 2 species if the whole population has not found a new best fitness within 20 generations. To convert NEAT's generation based strategy to EXAMM's asynchronous strategy, which does not have explicit generations, species were instead

[1] These data sets are made publicly available at EXAMM GitHub repository: https://github.com/travisdesell/exact/tree/master/datasets/ for reproduction of these results

disabled if they did not improve after 2250 new genomes were inserted (the same number of total genomes as 15 generations of 150 genomes), and all species except the top 2 were disabled if the best found fitness did not improve after 3000 genomes were inserted. The hyperparameters used for NEAT's speciation strategy were $c_1 = 1$, $c_2 = 1$, $c_3 = 0.4$, and the fitness threshold was set to $\delta_t = 0.6$ for the coal dataset, and $\delta_t = 0.4$ for the flight dataset. The c_1, c_2 and c_3 hyperparameters are the standard NEAT values, however the δ_t values were hand tuned to ensure good speciation. The NEAT runs were highly sensitive to δ_t and we found higher values resulted in all genomes clustering to the same species, and lower values resulted in each genome having its own species.

For both EXAMM and NEAT, all RNNs were locally trained for 10 epochs via stochastic gradient descent (SGD) and using back propagation through time (BPTT) [38] to compute gradients, all using the same hyperparameters. RNN weights were initialized by EXAMM's Lamarckian strategy (described in [26]), which allows child RNNs to reuse parental weights, significantly reducing the number of epochs required for the neuroevolution's local RNN training steps. SGD was run with a learning rate of $\eta = 0.001$ and used Nesterov momentum with $\mu = 0.9$. For the memory cells with forget gates, the forget gate bias had a value of 1.0 added to it (motivated by [17]). To prevent exploding gradients, gradient scaling [28] was used when the norm of the gradient exceeded a threshold of 1.0. To combat vanishing gradients, gradient boosting (the opposite of scaling) was used when the gradient norm was below 0.05. These parameters have been selected as they were recommended in previous papers about the EXAMM algorithm.

4.3 Experimental Design

Due to the stochastic nature of our experiments, we performed 20 repeats for each NEAT and EXAMM experiment on the coal and flight data sets. For EXAMM, we compared the baseline strategy (islands without extinction mechanism) to the two variations of the extinction strategy, one allowing repeated repopulations and the other not. For these strategies, extinction frequencies of 1000 and 2000 generated genomes were evaluated, and during the repopulation process, we allowed the global best genome (at the time of the extinction mechanism) to be mutated m times, with $m = 0$, 2, 4, or 8, before being inserted into the repopulated island. This resulted in a total of 680 experiments, 20 for NEAT, 20 for baseline EXAMM, and 320 for the 2 extinction strategies, 2 extinction frequencies, and 4 mutation values for each of the 2 datasets. For the experiments which disallowed repeated extinction for $e = 2$ events.

Various experiments were performed to get an understanding of how the frequency of extinction mechanism affected performance, *i.e.*, did having more frequent extinction mechanisms prevent repopulated islands from catching up and improving on the global best solution? Additionally, the two extinction strategies allowed us to determine the impact of allowing islands to be repeatedly made extinct, to see if they needed even more time to become well performing. Finally, modifying the mutation rates was done to provide an idea of how much exploration needed to be performed when repopulating the islands, to allow them to find new potentially better areas in the search space.

4.4 Computing Environment

Results were gathered using Rochester Institute of Technology's research computing systems. This system consists of 2304 Intel® Xeon® Gold 6150 CPU 2.70GHz cores and 24 TB RAM, with compute nodes running the RedHat Enterprise Linux 7 system. All EXAMM baseline and EXAMM speciation strategies experiments utilized 180 cores. Since the NEAT speciation strategy is implemented in the EXAMM framework, and the EXAMM master process is responsible for generating and inserting genomes, whereas worker processes are only responsible for stochastic back propagation training and evaluate the fitness of genomes, all the genome distances and explicate fitness sharing evaluations were done in the master process. Utilizing NEAT's speciation strategy presented a speed bottleneck at the master process when using a larger number of cores. Therefore, the NEAT runs were limited to 72 cores, as adding additional cores did not improve runtime.

4.5 Repopulation Strategy Evaluation

Figures 1 and 2 present the performance across the 20 repeated experiments for NEAT speciation and the EXAMM variations. The solid line shows the average of the global best genomes across the 20 experiments, and the filled in area shows the range between the min and max. The test results show that the EXAMM extinction and repopulation strategies perform better than baseline EXAMM algorithm across all tests, with the NEAT speciation strategy performing worse than baseline EXAMM. On average, in three of the coal plant test cases and two of the four flight test cases, 2 mutations resulted in the best performing genomes. For the other test cases, applying 4 or 8 mutations found the best performing genomes, and all those test cases come from non-repeated repopulations for both datasets, which proves that innovations need more time to evolve and become better. The results also suggest that adding some, but not too much variance to the global best genome for island repopulation allowed the strategies to best find new regions of the search space to improve performance.

As a further investigation, Table 1 presents Mann–Whitney U test *p*-values comparing the best genomes of the 20 repeats from the various strategies to the best genomes from the 20 repeats of baseline EXAMM. *p*-values in bold represent statistically significant differences with $\alpha = 0.05$, showing that the results of the varying mutation strategies have a statistically significant difference from EXAMM, which similarly has a statistically significant difference from NEAT speciation.

Table 1 also provides more detail about the best, average and worst global best genome fitness at the end of the 20 repeated tests for each experiment. From this we can see that in the average cases having a faster extinction frequency

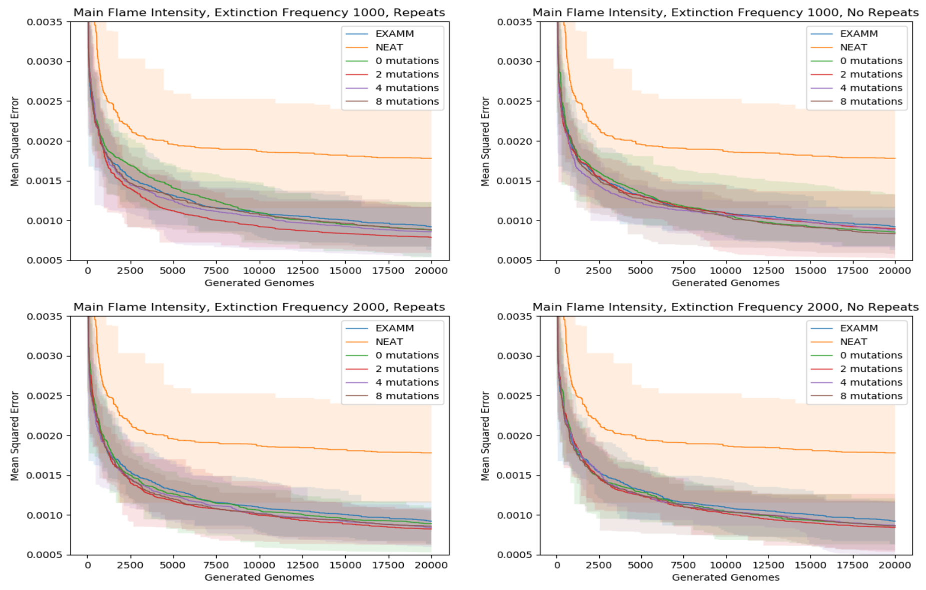

Fig. 1: Convergence rates (in terms of best MSE on validation data) for NEAT speciation and the EXAMM extinction and repopulation strategies predicting *main flame intensity* from the coal fired power plant dataset.

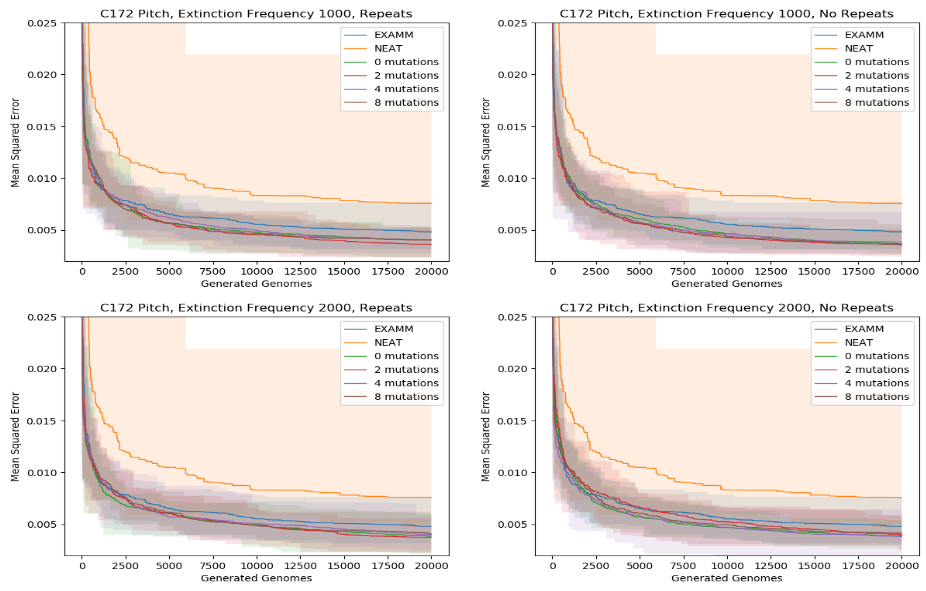

Fig. 2: Convergence rates (in terms of best MSE on validation data) for NEAT speciation and the EXAMM extinction and repopulation strategies predicting *pitch* from the c172 flight dataset.

Dataset	Erase Rule	Extinct Freq	Mutation	p-Value	Worst	Avg	Best
Coal	EXAMM	/	/	/	0.00125	0.00099	0.00072
	NEAT	/	/	**7.15e-8**	0.00238	0.00178	0.00115
	Repeat	1000	0	**4.55e-2**	0.00123	0.00088	0.00054
			2	**8.05e-5**	0.00117	**0.00079**	0.00059
			4	**1.58e-2**	0.00116	0.00086	0.00054
			8	**2.06e-2**	0.00123	0.00088	0.00067
		2000	0	5.68e-2	0.00109	0.00089	0.00053
			2	**1.28e-3**	0.00108	0.00082	0.00062
			4	**1.98e-3**	0.00108	0.00084	0.00060
			8	**7.74e-3**	**0.00105**	0.00086	0.00064
	No Repeat	1000	0	2.28e-1	0.00122	0.00093	0.00057
			2	**4.89e-3**	0.00120	0.00083	**0.00051**
			4	6.32e-2	0.00126	0.00089	0.00056
			8	2.20e-1	0.00132	0.00094	0.00070
		2000	0	**9.99e-5**	0.00116	**0.00079**	0.00057
			2	**3.28e-3**	0.00107	0.00084	0.00058
			4	7.39e-2	0.00112	0.00091	0.00065
			8	**3.82e-2**	0.00118	0.00090	0.00064
C172	EXAMM	/	/	/	0.00765	0.00480	0.00316
	NEAT	/	/	**1.3e-6**	0.01725	0.00755	0.00473
	Repeat	1000	0	**5.72e-3**	0.00526	0.00404	0.00229
			2	**3.31e-5**	0.00514	0.00360	0.00236
			4	**2.56e-3**	0.00523	0.00401	0.00282
			8	**5.72e-3**	0.00538	0.00401	0.00242
		2000	0	**2.16e-3**	0.00606	0.00385	0.00223
			2	**1.52e-4**	0.00556	0.00371	0.00234
			4	**1.28e-2**	0.00584	0.00399	0.00252
			8	**1.04e-2**	0.00621	0.00411	0.00216
	No Repeat	1000	0	**3.70e-5**	0.00513	0.00366	0.00281
			2	**3.70e-5**	**0.00497**	**0.00355**	0.00240
			4	**4.18e-4**	0.00672	0.00382	0.00266
			8	**4.64e-5**	0.00554	**0.00355**	0.00258
		2000	0	**5.07e-4**	0.00539	0.00387	0.00257
			2	**3.02e-3**	0.00590	0.00398	0.00246
			4	**1.17e-3**	0.00643	0.00381	**0.00163**
			8	**4.89e-3**	0.00577	0.00411	0.00300

Table 1: Performance of the various strategies for the varying EXAMM experiments, with best values marked bold. Mann–Whitney U test p-values are included comparing EXAMM to NEAT speciation and the different extinction and repopulation strategies. p-values in bold indicate a statistically significant difference with $\alpha = 0.05$.

of 1000 generally provided the best results, providing more evidence that performing extinction and repopulation improves the performance of neuroevolution strategy. Interestingly, the strategy which allowed islands to not be repeatedly erased provided slightly better results in the best case for both the coal and flight data.

To provide more insight into how the repopulation strategies were affecting the islands, Figure 3 and 4 shows an average of the frequency in which islands were repopulated when allowing islands to be repeatedly repopulated or not. For example, the upper left subplot in Figure 3 shows that for predicting main flame intensity, with an extinction frequency of 1000 and 0 mutations to the global best on repopulation, on average 1.25 islands never are repopulated, 3.75 islands are repopulated once, 3 islands are repopulated twice, and so on. Note that when generating 20,000 genomes for the runs, EXAMM removes the worst island 19 times when the extinction frequency is 1000, and 9 times when the

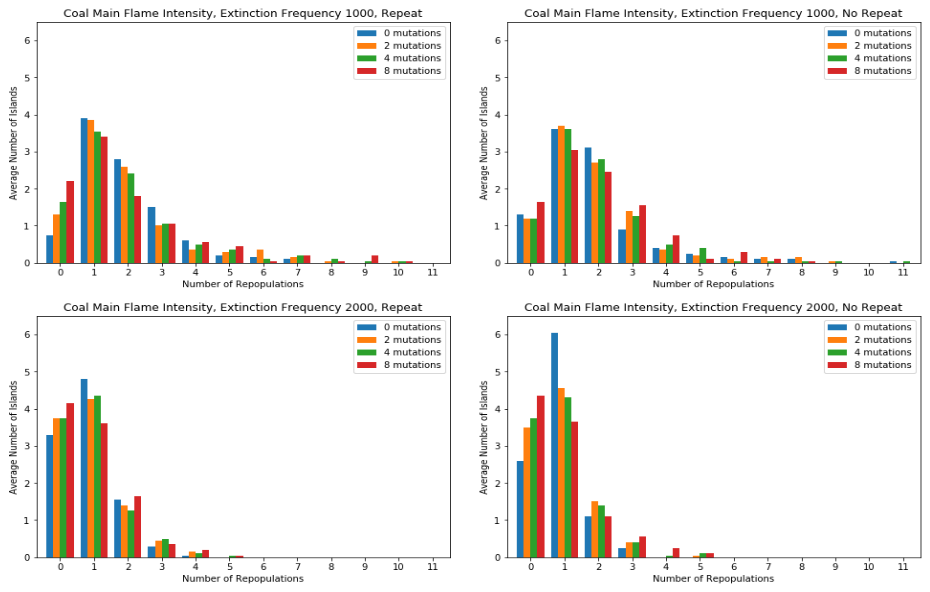

Fig. 3: The average number of times islands were repopulated across the repeated and no repeated experiments using the various EXAMM repopulation strategies on Coal dataset.

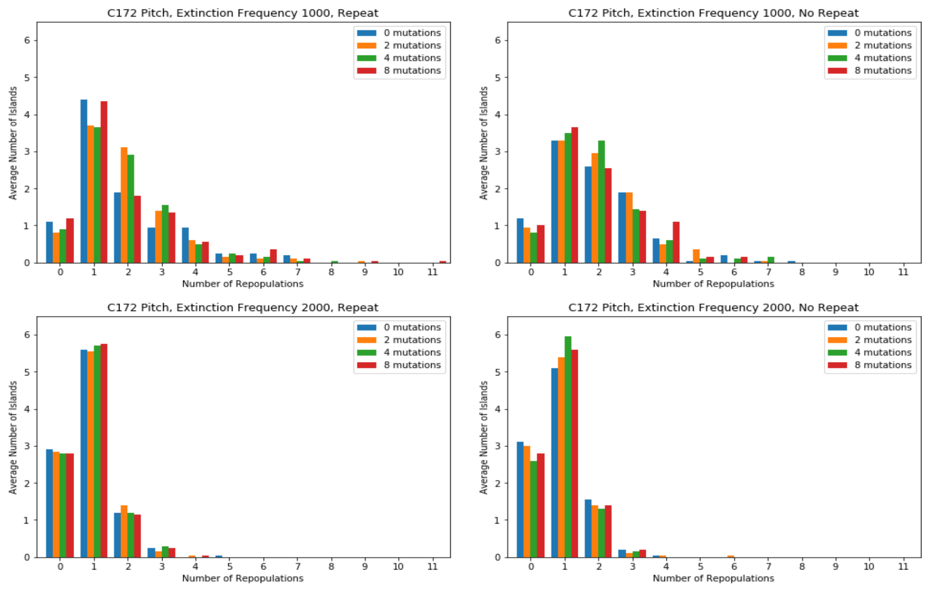

Fig. 4: The average number of times islands were repopulated across the repeated and no repeated experiments using the various EXAMM repopulation strategies on C172 dataset.

extinction frequency is 2000. These figures show that when allowing islands to be repeatedly repopulated, in many cases, the same islands are frequently repopulated, while the others are not. The figures also show a trend that when repeated repopulation is disallowed, the number of islands never repopulated is slightly reduced. Further, when applying more mutations during the repopulating phase, the same islands tend to be repopulated more frequently (*i.e.*, the number of islands that are never repopulated increases). This is most likely due to the fact that having too many mutations brings excessive variety or innovation to the new island population, making the performance of the repopulated island unstable. Interestingly, even in light of this we did not see much benefit from disallowing repeated repopulation (as shown in Tables 1), suggesting that if a repopulated island does not quickly find new better genomes, it does not have a good chance of finding better results if given more time to evolve.

5 Conclusion

This work investigates a novel speciation strategy based on extinction and repopulation mechanisms for island based evolutionary algorithms, applying it to neuroevolution of recurrent neural networks for time series data prediction on two challenging real-world data sets. In this strategy, the worst performing islands periodically experience extinction events and are repopulated with either the global best genome or mutations of it. Two versions of this strategy were implemented, one which allowed islands to be repeatedly repopulated and the other which prevented an island from being repopulated until a specified number of extinction events occurred on other islands. We investigated versions of this strategy with varying extinction frequencies, as well as numbers of mutations to the global best genome.

These mutation strategies were incorporated into the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution project, along with NEAT's speciation strategy as a benchmark comparison to a well-known neuroevolution technique. Results show that the repopulation strategy led to statistically significant improvements over baseline EXAMM, which in turn had large and statistically significant improvements over NEAT's speciation strategy. The repopulation strategies were also found to be more scalable than NEAT's strategy which requires determining the distance of each new genome to all others in the population to perform speciation. While the number of mutations applied to the global genome during repopulation was not significantly correlated with the best performance, in general a lower number (2 or 4) provided the best results. Having more mutations brought more innovation, but was also more unstable, leading to repopulated islands being repeatedly erased. Allowing islands to be repeatedly repopulated had advantages and disadvantages, where repeated repopulation would remove "bad" genomes more quickly, but preventing repeat repopulations protected innovations, giving the repopulated islands more time to evolve. In general, both strategies (allowing and disallowing repeated repopulation) provided statistically significant improvements over not using extinction and repopulation, but interestingly, neither significantly outperformed the other.

This work explored how a different number of mutations combined with different island extinction rules affected the repopulation process. Future work will involve examining other types of island extinction mechanisms, for example erasing multiple islands during an extinction, or controlling extinction mechanisms based on how much an island has improved over a period of time. Other options for repopulation can also be investigated beyond using the global best genome. Future work also includes investigating how to use varying forms of crossover to improve the repopulation algorithm's performance, which will include examining various crossover rules for repopulation, changing genome encoding methods, and redesigning the distance evaluation function. Lastly, it was particularly interesting that preventing and allowing repeated extinction both provided similar improvements, but neither outperformed the other. Developing a strategy that can make use of the best qualities of both may lead to further performance improvements. It should also be noted that while this work was examined in the context of neuroevolution algorithms, it could also be applied to any evolutionary strategy utilizing islands.

Acknowledgements

Most of the computation of this research was done on the high performance computing clusters of Research Computing at Rochester Institute of Technology [29]. We would like to thank the Research Computing team for their assistance and the support they generously offered to ensure that the heavy computation this study required was available.

References

1. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. Evolutionary Computation, IEEE Transactions on **6**(5), 443–462 (2002)
2. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
3. Collins, J., Sohl-Dickstein, J., Sussillo, D.: Capacity and trainability in recurrent neural networks. arXiv preprint arXiv:1611.09913 (2016)
4. De Falco, I., Della Cioppa, A., Maisto, D., Scafuri, U., Tarantino, E.: Biological invasion–inspired migration in distributed evolutionary algorithms. Information Sciences **207**, 50–65 (2012)
5. Desell, T., ElSaid, A., Ororbia, A.G.: 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 (April 2020)
6. ElSaid, A., Ororbia, A.G., Desell, T.J.: Ant-based neural topology search (ants) for optimizing recurrent networks. In: International Conference on the Applications of Evolutionary Computation (Part of EvoStar). pp. 626–641. Springer (2020)

7. Fuqua, L.M., Bralower, T.J., Arthur, M.A., Patzkowsky, M.E.: Evolution of calcareous nannoplankton and the recovery of marine food webs after the cretaceous-paleocene mass extinction. Palaios **23**(4), 185–194 (2008)
8. Gallala, N., Zaghbib-Turki, D., Arenillas, I., Arz, J.A., Molina, E.: Catastrophic mass extinction and assemblage evolution in planktic foraminifera across the cretaceous/paleogene (k/pg) boundary at bidart (sw france). Marine Micropaleontology **72**(3-4), 196–209 (2009)
9. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Aistats. vol. 9, pp. 249–256 (2010)
10. Goldberg, D.E., Richardson, J., et al.: Genetic algorithms with sharing for multimodal function optimization. In: Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms. pp. 41–49. Hillsdale, NJ: Lawrence Erlbaum (1987)
11. Greewood, G., Fogel, G.B., Ciobanu, M.: Emphasizing extinction in evolutionary programming. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 1, pp. 666–671. IEEE (1999)
12. Grefenstette, J.J., et al.: Genetic algorithms for changing environments. In: Ppsn. vol. 2, pp. 137–144. Citeseer (1992)
13. Hadjiivanov, A., Blair, A.: Complexity-based speciation and genotype representation for neuroevolution. In: 2016 IEEE Congress on Evolutionary Computation (CEC). pp. 3092–3101. IEEE (2016)
14. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
15. Hernández, A., Botello, S., et al.: Repairing normal edas with selective repopulation. Applied Mathematics and Computation **230**, 65–77 (2014)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
17. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: International Conference on Machine Learning. pp. 2342–2350 (2015)
18. Knapp, J.S., Peterson, G.L.: Natural evolution speciation for neat. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 1487–1493. IEEE (2019)
19. Krčah, P.: Effects of speciation on evolution of neural networks in highly dynamic environments. In: International Conference on Learning and Intelligent Optimization. pp. 425–430. Springer (2012)
20. Krink, T., Thomsen, R.: Self-organized criticality and mass extinction in evolutionary algorithms. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546). vol. 2, pp. 1155–1161. IEEE (2001)
21. Lehman, J., Miikkulainen, R.: Enhancing divergent search through extinction events. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 951–958 (2015)
22. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary computation **19**(2), 189–223 (2011)
23. Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G.: A survey on evolutionary neural architecture search. arXiv preprint arXiv:2008.10937 (2020)
24. Mathias, H.D., Ragusa, V.R.: An empirical study of crossover and mass extinction in a genetic algorithm for pathfinding in a continuous environment. In: 2016 IEEE Congress on Evolutionary Computation (CEC). pp. 4111–4118. IEEE (2016)

25. Newman, M., Roberts, B.W.: Mass extinction: Evolution and the effects of external influences on unfit species. Proceedings of the Royal Society of London. Series B: Biological Sciences **260**(1357), 31–37 (1995)
26. Ororbia, A., ElSaid, A., Desell, T.: Investigating recurrent neural network memory structures using neuro-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 446–455. GECCO '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3321707.3321795, http://doi.acm.org/10.1145/3321707.3321795
27. Ororbia II, A.G., Mikolov, T., Reitter, D.: Learning simpler language models with the differential state framework. Neural Computation **0**(0), 1–26 (2017). https://doi.org/10.1162/neco_a_01017, https://doi.org/10.1162/neco_a_01017, pMID: 28957029
28. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning. pp. 1310–1318 (2013)
29. Rochester Institute of Technology: Research computing services (2019). https://doi.org/10.34788/0S3G-QD15, https://www.rit.edu/researchcomputing/
30. Spears, W.: Speciation using tag bits. Handbook of Evolutionary Computation (1995)
31. Stanley, K., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
32. Stanley, K.O., Clune, J., Lehman, J., Miikkulainen, R.: Designing neural networks through neuroevolution. Nature Machine Intelligence **1**(1), 24–35 (2019)
33. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artificial life **15**(2), 185–212 (2009)
34. Sun, Y., Xue, B., Zhang, M., Yen, G.G.: Evolving deep convolutional neural networks for image classification. IEEE Transactions on Evolutionary Computation (2019)
35. Trujillo, L., Olague, G., Lutton, E., De Vega, F.F.: Discovering several robot behaviors through speciation. In: Workshops on Applications of Evolutionary Computation. pp. 164–174. Springer (2008)
36. Verbancsics, P., Stanley, K.O.: Evolving static representations for task transfer. Journal of Machine Learning Research **11**(May), 1737–1769 (2010)
37. Wan, J., Chu, P., Jiao, Y., Li, Y.: Improvement of machine learning enhanced genetic algorithm for nonlinear beam dynamics optimization. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **946**, 162683 (2019)
38. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE **78**(10), 1550–1560 (1990)
39. Xie, X.F., Zhang, W.J., Yang, Z.L.: Hybrid particle swarm optimizer with mass extinction. In: IEEE 2002 international conference on communications, circuits and systems and West Sino expositions. vol. 2, pp. 1170–1173. IEEE (2002)
40. Zhou, G.B., Wu, J., Zhang, C.L., Zhou, Z.H.: Minimal gated unit for recurrent neural networks. International Journal of Automation and Computing **13**(3), 226–234 (2016)