EXA-GP: Unifying Graph-Based Genetic Programming and Neuroevolution for Explainable Time Series Forecasting

Jared Murphy
jmm2188@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Devroop Kar
dk7405@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Joshua Karns
josh@mail.rit.edu
Rochester Institute of Technology
Rochester, New York, USA

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

ABSTRACT

This work introduces Evolutionary eXploration of Augmenting Genetic Programs (EXA-GP), which converts the EXAMM neuroevolution algorithm into a graph-based genetic programming (GGP) algorithm by swapping out its library of neurons and recurrent memory cells for genetic programming (GP) operations. This enables EXA-GP to utilize the full suite of optimizations provided by EXAMM, such as distributed and multi-threaded execution, island based populations with re-population events, as well as Lamarckian weight inheritance and backpropagation for optimization of constant values. Results show that EXA-GP's evolved genetic programs achieve the same levels of accuracy as recurrent neural networks (RNNs) evolved by EXAMM, while at the same time being more explainable. EXA-GP's genetic programs are also computationally less complex than EXAMM's RNNs, suggesting that GP operations may actually be more effective than gated recurrent memory cells for time series forecasting. We also show that EXAMM and EXA-GP outperform state-of-the-art GP and recurrent CGPANN algorithms which struggle on these benchmarks.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Genetic programming**.

KEYWORDS

time series forecasting, graph-based genetic programming, neuroevolution, recurrent neural networks

ACM Reference Format:
Jared Murphy, Devroop Kar, Joshua Karns, and Travis Desell. 2024. EXA-GP: Unifying Graph-Based Genetic Programming and Neuroevolution for Explainable Time Series Forecasting. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3638530.3654349

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0495-6/24/07.
https://doi.org/10.1145/3638530.3654349

1 INTRODUCTION

Time series forecasting (TSF) plays a crucial role in various fields and industries, providing invaluable insights for decision-making and planning. By analyzing historical data and identifying patterns, time series forecasting enables organizations to anticipate future trends, fluctuations, and changes in different variables over time. Recurrent Neural Networks (RNNs) have emerged as powerful tools in the field of TSF [7], offering unique advantages in capturing sequential dependencies and temporal patterns within data. Unlike traditional feed forward neural networks, RNNs utilize connections that span time, enabling them to retain information from previous time steps, making them particularly effective in modeling time series data. Their ability to process sequential information and adapt to changing patterns makes RNNs invaluable for a wide range of time-dependent forecasting tasks, contributing significantly to more accurate and reliable predictions in various domains.

Unfortunately, RNNs, as with most neural networks, are black boxes and it is difficult to understand why they make their predictions. Gate memory cells exacerbate this problem as they can store values and experience phase shifts based on a build up of historical data. While there have been some attempts make RNNs more explainable [6, 9, 12, 19, 23], this still remains a challenge.

On the other hand, Genetic programming (GP) has also emerged as a powerful and adaptive approach in the domain of TSF. Unlike RNNs, GP evolves mathematical expressions by leveraging evolutionary algorithms, allowing for adaptability and flexibility. Due to their use of more simplistic mathematical expressions and operations, GP has been shown to be a powerful method for designing explainable AI systems [4, 5, 11].

Graph-based Genetic Programming (GGP) [1, 17] has also shown promise as a more advanced version of GP, which allows for the evolution of directed acyclic graphs, instead of the tree structures commonly used by standard GP. GGP and neuroevolution (the use of evolutionary algorithms to design and/or training neural networks) are highly similar, in that they both evolve computational graphs, as neural networks and genetic programs are both examples of computational graphs. This has been proven out by Cartesian Genetic Programming of Artificial Neural Network (CGPANN) methods, which replace the GP operations of standard GGP with ANN neurons [8, 20, 21].

This work takes the opposite approach from CGPANN methods, replacing the library of recurrent memory cells in the Evolutionary

eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm [13] with a set of standard genetic programming operations, with the goal of evolving more explainable genetic programs for time series forecasting. Using six real world, noisy, multivariate time series forecasting benchmark problems, we evaluate EXA-GP against EXAMM, three state-of-the-art GP libraries (Jenetics [22], EC-KitY [15, 16] and GPLearn [10, 14, 18]) and a recurrent CGPANN from the CGP-Library [21]. Results show that EXA-GP can evolve genetic programs that are as accurate as RNNs evolved by EXAMM, and that it can do this in a similar amount of time, while finding more explainable structures that are computationally more efficient. Additionally, the benchmark problems highlight limitations in existing GP and CGPANN algorithms, which are unable to beat out the trivial (yet challenging to beat) TSF solution of using the previous value as the predicted value. Our findings indicate that this is due to their inability to efficiently optimize constant values which can strongly represent relations between parameters in TSF problems.

2 EXAMM AND EXA-GP

Evolutionary eXploration of Augmenting Memory Models (EXAMM) algorithm [13] is a distributed neuroevolution algorithm that evolves progressively larger RNNs for large-scale, multivariate, real-world TSF. EXAMM evolves RNN architectures consisting of varying recurrent connections, simple neurons and memory cells through a series of mutation and crossover (reproduction) operations. Recurrent connections can be deep, spanning multiple time steps, which has shown to significantly improve the RNNs predictive ability. For an in depth presentation of EXAMM we refer the reader to ElSaid *et al.* [2, 3]).

Evolutionary eXploration of Augmenting Genetic Programs (EXA-GP)[1] is able to leverage all this prior research on EXAMM by replacing EXAMM's simple neurons and memory cells with genetic programming operations that are all differentiable: sin, cos, $tanh$, $sigmoid$, $inverse$, sum and $product$. This allows the evolved computational graphs to be trained with backpropagation, as the error signal can be propagated backwards through these nodes. Given the ability to quickly train and optimize the constants (weights) in the evolved computational graphs, we expanded each of these operations by adding a trainable bias term b to each.

3 DATASETS

This work utilizes three challenging, noisy, non-seasonal, multivariate datasets from real world systems (aircraft, wind turbines and a coal fired power plant) as benchmarks for time series forecasting. *Supplementary Fuel Flow* and *Main Flame Intensity* were predicted using the coal burner dataset, *Pitch* and *Engine 1 Cylinder Head Temperature 1 (E1 CHT1)* were predicted from the aviation dataset, and *Average Converter Torque (Cm_avg)* and *Average Power (P_avg)* were predicted from the wind turbine dataset. All data for these was pre-normalized using min-max scaling within the range $[0-1]$. These datasets, including the pre-processed normalized versions, have been made publicly available for reproducibility and further study[1].

[1] EXA-GP's source code and benchmark datasets have been made publicly available at https://github.com/travisdesell/exact.

4 RESULTS

We evaluated EXA-GP against EXAMM [13], three modern GP frameworks (Jenetics [22], EC-KitY [15, 16], GPLearn [10, 14, 18]) and RCGPANN from the CGP-Library [21]. For the GP methods, we examined using only parameters from the previous timestep for forecasting, as well as time windowed versions where they were able to additionally select from any of the input parameters up to 5 time steps in the past (time windowed experiments were titled GPLearn TW, EC-KitY TW, and Jenetics TW). We evolved RNNs/GPs for each of the 6 output parameters across the 3 datasets, repeating each experiment 10 times.

Performance Evaluation. Figure 1 shows violin plots of the mean squared error (MSE) of the evolved GPs and RNNs on the test dataset over all the repeated experiments and benchmark problems. These results highlight one of the major challenges in performing time series on noisy real world data. The dotted red line in each plot represents the naive and trivial solution of using the target parameter prior value as the prediction ($pred(x_t) = x_{t-1}$, i.e., the predicted value of x at time t is simply x at time $(t-1)$. This trivial solution is not only quite challenging to outperform, but it also creates a significant local minimum and valley in the search landscape that any TSF algorithm will need to break out of.

Across all 6 benchmark problems, the EC-KitY and GPLearn libraries always converged to this trivial solution (finding GPs such as $f(x) = inverse(inverse(x))$), except for the `Cm_avg` benchmark, where the time windowed version of EC-KitY was not able to always reach even the trivial solution. RCGPANN also converged to the trivial solution in every benchmark except for the `Supp_Fuel_Flow` benchmark, where it was able to find some better performing recurrent neural networks (but not to the performance of EXA-GP and EXAMM). Jenetics had the widest range of performance (in some cases not even reaching the trivial solution), but did also manage to find some better than trivial solutions for `Supp_Fuel_Flow`. EXAMM and EXA-GP fared much better, finding GPs and RNNs that beat the trivial solution in all benchmarks except for `E1_CHT1`, where none of the methods found a better GP or RNN.

Table 1 shows the average and best performance (test data MSE across the 10 repeated experiments for each benchmark) for EXAMM compared to EXA-GP along with p-values for the Mann-Whitney U-Test of statistical significance. EXA-GP found the overall best solution in half the benchmarks and had the best average performance in another half. However, only one of the benchmarks in which it performed better was statistically significant, and in one other it performed worse with statistical significance. *These findings are quite impressive given that EXAMM utilized its full library of recurrent memory cells and that EXA-GP was able to find comparable solutions using only GP operators.*

Evolved Genetic Programs and Recurrent Neural Networks. For most benchmarks the best found solutions were equal to or not better than the trivial solution, e.g., $pred(x_t) = inverse(inverse(x_{t-1}))$, except for the `Supp_Fuel_Flow` benchmark. The best GP from Jenetics without a time window was:

$$f(x) = inverse(inverse((Supp\_Fuel\_Flow * cos(Supp\_Fuel\_Flow)) * cos(Supp\_Fuel\_Flow))) \quad (1)$$

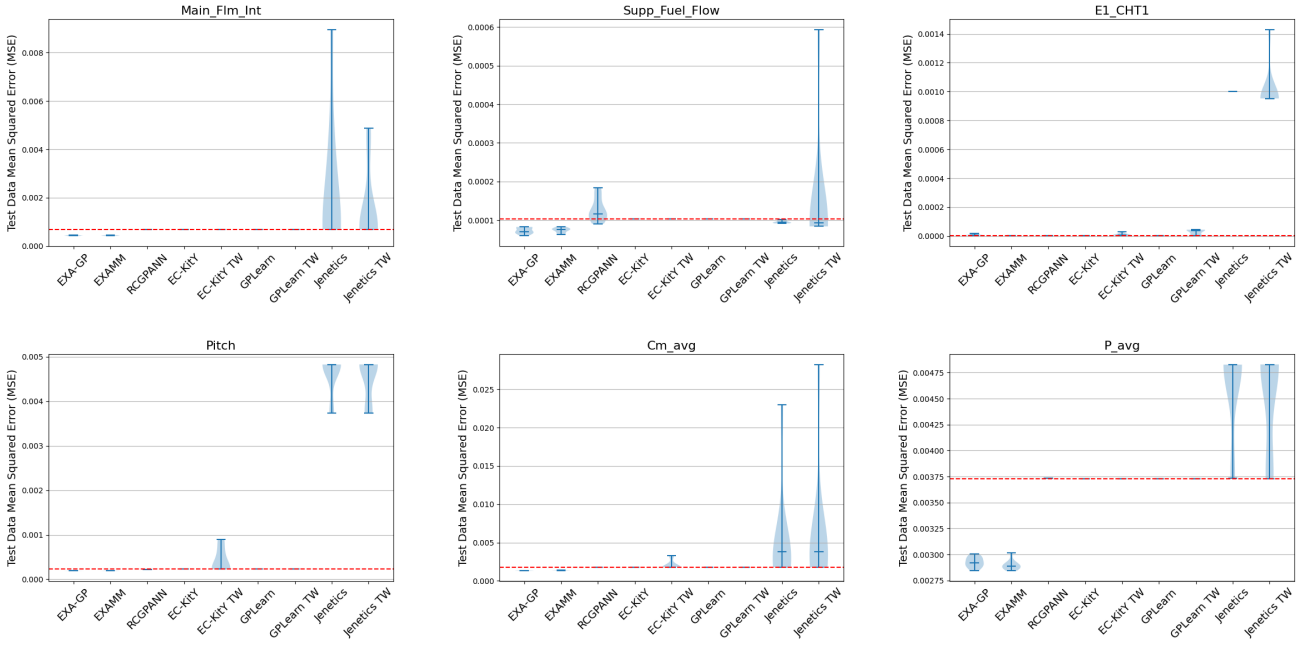

Figure 1: Algorithm performance (Test Dataset MSE) across the 6 benchmark problems.

	Avg		Max		
Dataset	EXA-GP	EXAMM	EXA-GP	EXAMM	p-value
Main_Flm_Int	**0.0004416634**	0.0004435806	**0.000423561**	0.000426996	0.79134
Supp_Fuel_Flow	**7.304e-05**	7.402e-05	**5.95e-05**	6.29e-05	0.96985
E1_CHT1	6.902e-06	**2.836e-06**	2.81e-06	**2e-06**	**0.00728** ↑
Pitch	0.0001921876	**0.0001914784**	0.000189603	**0.00018716**	0.52037
Cm_avg	**0.00134658**	0.001370535	**0.00130777**	0.00134235	**0.02113** ↓
P_avg	0.002926976	**0.002898748**	0.00284333	**0.00284371**	0.21229

Table 1: EXA-GP vs. EXAMM Performance and Whitney U-Test of Significance

	Avg. # Nodes			Avg. # Edges			Avg. # Weights		
Dataset	EXA-GP	EXAMM	p-value	EXA-GP	EXAMM	p-value	EXA-GP	EXAMM	p-value
Main_Flm_Int	8.9	**7.1**	0.10124	101.5	**70.7**	0.07544	**110.4**	125.4	0.90969
Supp_Fuel_Flow	**4.1**	6.9	**0.02302** ↓	**40.8**	62.0	**0.03742** ↓	**44.9**	114.1	**0.00281** ↓
E1_CHT1	**6.1**	6.6	0.90798	**75.6**	89.2	0.18538	**81.7**	134.6	**0.00247** ↓
Pitch	**3.9**	4.8	0.19557	**68.9**	83.3	0.12017	**72.8**	106.6	**0.01265** ↓
Cm_avg	**6.4**	8.0	0.39721	**69.1**	84.9	0.04475	**75.5**	138.8	**0.00033** ↓
P_avg	**4.4**	5.2	0.30186	**56.3**	66.1	0.34775	**60.8**	97.8	**0.01996** ↓

Table 2: EXA-GP vs. EXAMM Generated computational Graph Sizes and Mann-Whitney U-Test of Significance

And for Jenetics with a time window:

$$f(x) = tanh(cos(Supp\_Fuel\_Flow^{2.0}) * Supp\_Fuel\_Flow) \quad (2)$$

RCGPANN did find a somewhat simple recurrent neural network for this benchmark, which had a single recurrent connection. After reviewing the best GP's from the other GP libraries, we noted that constants were almost never used, even when the GP's did not converge to the trivial solution. We also noted that networks provided by EXAMM and EXA-GP have significantly more edges (most likely due to their ability to find better weights by using backpropagation), and slightly more nodes overall. This provides more evidence that the other GP libraries likely lack effective methods for optimizing

constants, thus inhibiting their ability to break out of the hard local minimum of the trivial solution and generate well performing TSF.

From our further investigation of the computational graphs evolved by EXAMM and EXA-GP, Table 2 presents the average number of nodes, edges and total weights (trainable parameters), as well p-values from the Mann-Whitney U-Test of statistical significance comparing the two methods. It was found that, on average, for 5 of the 6 benchmarks EXA-GP had less nodes and edges, however in only one case was this statistically significant. Most interestingly, we found that the total number of weights in the computational graphs evolved by EXA-GP *were less than those evolved by EXAMM in 5 out of 6 benchmarks to a statistically significant degree*. We find these results to be quite interesting and contrary to what was expected. The gated recurrent memory cells in the EXAMM library are considered by many to be state-of-the-art when designing recurrent neural networks and are often used in the field of TSF. We would expect that these would be able to better learn the TSF problem and be able to provide solutions with fewer weights. However, *EXA-GP was able to evolve networks that were computationally more efficient, with less trainable parameters, using only simple GP operators*. This suggests that GP operations are potentially more effective than gated recurrent memory cells at solving the problem of TSF, assuming that weights and computational graphs are well optimized.

5 DISCUSSION

In this work we show how, with minor modifications, an advanced neuro-evolution algorithm can become a graph based genetic programming algorithm. In particular, we show that the genetic programs (GPs) evolved by EXA-GP have comparable predictive ability as those evolved by EXAMM, with statistical significance, and that these GPs can be evolved in a similar amount of time (no difference, with statistical significance, however slightly faster on average). Most interestingly, we find that with statistical significance, the GPs evolved by EXA-GP are *computationally more efficient* than those evolved by EXAMM, requiring fewer trainable weights to reach the same levels of accuracy as EXAMM's recurrent neural networks. This has potential ramifications for the design of neural networks and computational graphs for TSF, as it indicates that simplistic GP operators may more appropriately represent the problem, as opposed complicated gated recurrent memory cells (such as LSTMs, GRUs, etc). Finally, we find that the reason that the GP and CGPANN methods struggle on the challenging TSF benchmark problems of this work is most likely due to their poor ability to find and optimize their ephemeral constants. By incorporating backpropagation or other methods to better evolve these constants, GP-based methods could significantly improve their performance.

REFERENCES

[1] Timothy Atkinson, Detlef Plump, and Susan Stepney. 2018. Evolving graphs by graph programming. In *European Conference on Genetic Programming*. Springer, 35–51.

[2] AbdElRahman ElSaid, Joshua Karnas, Zimeng Lyu, Daniel Krutz, Alexander G Ororbia, and Travis Desell. 2020. Neuro-Evolutionary Transfer Learning through Structural Adaptation. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 610–625.

[3] AbdElRahman ElSaid, Joshua Karns, Zimeng Lyu, Daniel Krutz, Alexander Ororbia, and Travis Desell. 2020. Improving neuroevolutionary transfer learning of deep recurrent neural networks through network-aware adaptation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 315–323.

[4] Benjamin P Evans, Bing Xue, and Mengjie Zhang. 2019. What's inside the black-box? a genetic programming method for interpreting complex machine learning models. In *Proceedings of the genetic and evolutionary computation conference*. 1012–1020.

[5] Leonardo Augusto Ferreira, Frederico Gadelha Guimarães, and Rodrigo Silva. 2020. Applying genetic programming to improve interpretability in machine learning models. In *2020 IEEE congress on evolutionary computation (CEC)*. IEEE, 1–8.

[6] Warren Freeborough and Terence van Zyl. 2022. Investigating explainability methods in recurrent neural network architectures for financial time series data. *Applied Sciences* 12, 3 (2022), 1427.

[7] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. 2021. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting* 37, 1 (2021), 388–427.

[8] Maryam Mahsal Khan, Arbab Masood Ahmad, Gul Muhammad Khan, and Julian F Miller. 2013. Fast learning neural networks using cartesian genetic programming. *Neurocomputing* 121 (2013), 274–289.

[9] Hyon Hee Kim, Young Seo Lim, Seung-In Seo, Kyung Joo Lee, Jae Young Kim, and Woon Geon Shin. 2021. A Deep Recurrent Neural Network-Based Explainable Prediction Model for Progression from Atrophic Gastritis to Gastric Cancer. *Applied Sciences* 11, 13 (2021), 6194.

[10] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H Moore. 2021. Contemporary symbolic regression methods and their relative performance. *arXiv preprint arXiv:2107.14351* (2021).

[11] Yi Mei, Qi Chen, Andrew Lensen, Bing Xue, and Mengjie Zhang. 2022. Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation* (2022).

[12] Sara Mirzavand Borujeni, Leila Arras, Vignesh Srinivasan, and Wojciech Samek. 2023. Explainable sequence-to-sequence GRU neural network for pollution forecasting. *Scientific Reports* 13, 1 (2023), 9940.

[13] Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. 2019. Investigating Recurrent Neural Network Memory Structures Using Neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. ACM, New York, NY, USA, 446–455. https://doi.org/10.1145/3321707.3321795

[14] Patryk Orzechowski, Paweł Renc, William La Cava, Jason H Moore, Arkadiusz Sitek, Jaroslaw Wąs, and Joost Wagenaar. 2022. A Comparative Study of GP-based and State-of-the-art Classifiers on a Synthetic Machine Learning Benchmark. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 276–279.

[15] Moshe Sipper, Tomer Halperin, Itai Tzruia, and Achiya Elyasaf. 2022. EC-KitY: Evolutionary Computation Tool Kit in Python. https://www.eckity.org/.

[16] Moshe Sipper, Tomer Halperin, Itai Tzruia, and Achiya Elyasaf. 2023. EC-KitY: Evolutionary computation tool kit in Python with seamless machine learning integration. *SoftwareX* 22 (2023), 101381. https://www.sciencedirect.com/science/article/pii/S2352711023000778

[17] Léo Françoso DP Sotto, Paul Kaufmann, Timothy Atkinson, Roman Kalkreuth, and Márcio Porto Basgalupp. 2020. A study on graph representations for genetic programming. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 931–939.

[18] Trevor Stephens. 2015. gplearn: Genetic Programming in Python, with a scikit-learn inspired API. http://gplearn.readthedocs.io/.

[19] Qingxiong Tan, Mang Ye, Andy Jinhua Ma, Baoyao Yang, Terry Cheuk-Fung Yip, Grace Lai-Hung Wong, and Pong C Yuen. 2020. Explainable uncertainty-aware convolutional recurrent neural network for irregular medical time series. *IEEE Transactions on Neural Networks and Learning Systems* 32, 10 (2020), 4665–4679.

[20] Andrew James Turner and Julian Francis Miller. 2013. Cartesian genetic programming encoded artificial neural networks: a comparison using three benchmarks. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. 1005–1012.

[21] Andrew James Turner and Julian Francis Miller. 2017. Recurrent cartesian genetic programming of artificial neural networks. *Genetic Programming and Evolvable Machines* 18 (2017), 18–212.

[22] Franz Wilhelmstötter. 2021. Jenetics. *URL: http://jenetics.io* (2021).

[23] Bin Zhou, Shengnan Du, Lijuan Li, Huaizhi Wang, Yang He, and Diehui Zhou. 2021. An explainable recurrent neural network for solar irradiance forecasting. In *2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 1299–1304.