Minimizing the EXA-GP Graph-Based Genetic Programming Algorithm for Interpretable Time Series Forecasting

Anonymous Author(s)

ABSTRACT

In this work we show that when the majority of all trainable constants are eliminated and an appropriate seed computational graph in the form of using a parameter's value at time t as the forecast for the parameter's value at time $t + 1$ is used, the Evolutionary eXploration of Augmenting Memory Models (EXA-GP) graph-based genetic programming (GGP) algorithm can produce time series forecasting (TSF) equations that are vastly more simple and interpretable than the original implementation without heavily compromising on predictive ability, allowing this minimal version of EXA-GP (EXA-GP-MIN) to be a powerful tool for explainable time series forecasting. EXA-GP-MIN is compared to EXA-GP and EXAMM, a full blown neuroevolution algorithm for evolving recurrent neural networks for TSF, on a suite of six real world benchmark problems, with MIN-EXA-GP showing the best forecasting ability on four of the six benchmarks with significantly more interpretable genetic programs.

CCS CONCEPTS

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

KEYWORDS

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

ACM Reference Format:

Anonymous Author(s). 2018. Minimizing the EXA-GP Graph-Based Genetic Programming Algorithm for Interpretable Time Series Forecasting. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Time series forcasting (TSF) is an essential tool used in a diverse set of domains and industries. This has spawned a wide range of approaches, including statistical methods such as ARMIA and VAR [4], to machine learning methods based on recurrent neural networks (RNNs) [5] and more recently even transformer based models [19]. Given the complexity of architecting neural networks for time series forecasting, neuroevolution [11] and neural architecture search [13] have also become popular.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Woodstock '18, June 03–05, 2018, Woodstock, NY

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00

https://doi.org/10.1145/1122445.1122456

Unfortunately, neural network models tend to be black box systems which make them difficult to explain and interpret. In part, because of this, genetic programming (GP) has emerged as a powerful method for designing explainable AI systems [1, 2, 9], with Cartesian and graph based genetic programming (CGP and GGP, respectively) in particular being developed for TSF [17].

Recently, a new GGP algorithm called the Evolutionary eXploration of Augmenting Genetic Programs (EXA-GP) was proposed. EXA-GP replaces the library of recurrent memory cells in the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm [11] with a set of standard genetic programming operations, with the goal of evolving more explainable genetic programs for time series forecasting. EXA-GP creates and evolves computational graphs in the form of RNN's, and leverages the fact that its set of simple GP operations are differentiable by using backpropagation to train a complex network of additive and multiplicative constants in the form of weights and biases. While EXA-GP in its current form has shown to create more explainable genetic programs that perform comparably to state of the art memory cell networks at TSF, they still are lacking in interpretability.

In order to minimize the complexity of equations generated, and to further investigate the trade off between predictive capability and model interpretability, this work examines modifying EXA-GP into a minimal version (EXA-GP-MIN) when nearly all trainable constants are eliminated. Using six real world, noisy, multivariate time series forecasting benchmark problems, we evaluate EXA-GP-MIN against its original implementation, as well as the more complex EXAMM neuroevolution algorithm. Results show that when model constants are restricted to a an additive bias in the sum operation and a multiplicative bias in the product operation, and when seeded with the appropriate computational graph, EXA-GP-MIN is not only capable of producing significantly more interpretable genetic programs with minimal constants, but also, that for four out of the six benchmarks it even provides more accurate forecasts.

2 EXAMM, EXA-GP AND EXA-GP-MIN

EXAMM is a genetic algorithm that evolves recurrent neural network architectures for TSF [11]. It can select neurons from a suite of complex memory cells (LSTM, GRU, MGU, UGRNN and Δ-RNN cells), simple nodes with standard activation functions (tanh, sigmoid, identity), edges, and potentially deep recurrent edges as building blocks for RNN evolution. Through a series of mutation and crossover operations, computational graphs in the form of RNN's are evolved and trained in parallel on various real world data sets. The result is a powerful methodology for generating black box RNN's that optimize for predictive power over interpretability.

In order to design computational graphs that were more explainable, the Evolutionary eXploration of Augmenting Genetic Programs (EXA-GP) was proposed [10], which modifies the nodes available in EXAMM to include simple, differentiable GP operations with trainable constants (weights and biases), removing the complex memory cells. In doing this, EXA-GP leverages the performance of EXAMM, such as the use of backpropagation and multiple islands with repopulation events [8], to create a powerful GGP algorithm which utilizes GP operations such as *sin*, *cos*, *tanh*, *sigmoid*, *inverse*, *sum* and *product*. EXA-GP produces computational graphs that can be trained with backpropagation, as the error signal can be propagated backwards through these nodes, which allows for more effective use of ephemeral constants.

While these trainable constants contribute significantly to EXA-GP's ability to generate effective solutions, it comes at a cost to graph interpretability. The EXA-GP node library consisting of simple GP operations does allow for computational graphs to be exported as functions, however, EXA-GP still has a tendency to create complex solutions beyond the scope of practical interpretation (as an example, see Figure 2).

In an effort to move towards more interperatable computational graphs, in developing EXA-GP-MIN[1], we eliminated from EXA-GP all trainable multiplicative and additive constants (edge weights and node biases) except for two types. The first is a trainable bias term b added to the sum operation, and the second is a trainable bias term b that scales the product operation. Given $x_1, ..., x_n$ as n inputs to an operation, EXA-GP-MIN operations are now:

$$sin(x_1, ..., x_n) = sin(x_1 + ... + x_n) \tag{1}$$
$$cos(x_1, ..., x_n) = cos(x_1 + ... + x_n) \tag{2}$$
$$tanh(x_1, ..., x_n) = tanh(x_1 + ... + x_n) \tag{3}$$
$$sigmoid(x_1, ..., x_n) = \frac{1}{1 + e^{x_1 + ... + x_n}} \tag{4}$$
$$inverse(x_1, ..., x_n) = \frac{1}{x_1 + ... + x_n} \tag{5}$$
$$sum(x_1, ..., x_n) = (x_1 + ... + x_n) + b \tag{6}$$
$$product(x_1, ..., x_n) = (x_1 * ... * x_n) * b \tag{7}$$

All edges in the EXA-GP-MIN computational graphs which had trainable weights in EXAMM and EXA-GP are held fixed at 1.0 in EXA-GP-MIN (and thus can be eliminated from the computational graph). The biases in the sum and multiply operations are initialized using the Xavier method [3], and are the only parameters trained using backpropagation. With these modifications, the Lamarkian approach for weight inheritance [7] was no longer necessary and is eliminated.

EXA-GP and EXAMM are typically seeded with a fully connected graph that expresses the output parameter as the simple sum of the input parameters and a potential activation function. This works in practice when many trainable constants are present, however, when nearly all weights and biases have been eliminated, EXA-GP has trouble finding optimal solutions given this seed type. Further, when provided the fully connected seed type, this new version of EXA-GP fails to even converge upon the $\hat{x}_{t+1} = x_t$ trivial solution, which simply uses the previous value of the parameter as its forecast. This is significant, as while this is a trivial solution, it can be extremely challenging for TSF methods to reach or beat. This is especially the case for highly noisy data, as it is the optimal solution when the time series is purely random. Highlighting this issue, in prior work on EXA-GP, it was shown that other state-of-the art GP and CGPAN methods for TSF get stuck at the trivial solution and are unable to find better solutions [10].

In order to mitigate this issue, EXA-GP is instead provided a seed graph of $\hat{x}_{t+1} = x_t$, with other potential inputs present but not connected. Given the initial seed computational graph, this new version of EXA-GP quickly finds novel and better performing graphs for TSF.

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. All data for these datasets was pre-normalized using min-max scaling within the range $[0 - 1]$. These datasets, including the preprocessed normalized versions, have been made publicly available for reproducibility and further study[1]. Two parameters for forecasting were selected from each, which were *Supplementary Fuel Flow* and *Main Flame Intensity* from the coal burner dataset, *Pitch* and *Engine 1 Cylinder Head Temperature 1 (E1 CHT1)* from the aviation dataset, and *Average Converter Torque (Cm_avg)* and *Average Power (P_avg)* from the wind turbine dataset.

4 RESULTS

Figure 1 shows the performance of EXA-GP-MIN to EXA-GP and EXAMM (note the logarithmic y-axis). In previous work [10], three modern GP frameworks (Jenetics [18], EC-KitY [14, 15], GPLearn [6, 12, 16]) and RCGPANN from the CGP-Library [17] were evaluated on these benchmarks and all failed to find solutions improving over the trivial $\hat{x}_{t+1} = x_t$ solution (denoted as the dotted red line on each plot), so these results represent state of the art results on these benchmarks.

Results for EXA-GP-MIN are impressive, finding the best found forecasters across four of the six benchmarks (`Supp_Fuel_Flow`, `Cm_avg`, `E1_CHT1`, and `P_avg`). EXA-GP-MIN was also able to find a GP improving on the trivial solution for `E1_CHT1` which all previous methods were not able to do. These results do show significant improvements in forecasting (even though the mean squared error values are small) as they are across significantly long validation datasets (typically thousands of time steps). We also see a significant result of seeding EXA-GP-MIN with the trivial solution - across all benchmarks it was able to improve over the trivial solution and not get stuck in local minima prior to this.

To highlight the improvement in interpretability, Figure 2 presents the best found GP for `E1_CHT1` by EXA-GP. While this is technically more explainable than the RNNs evolved by EXAMM, it is a far cry from being interpretable. This can be contrasted to Figure 3 which is the GP evolved by EXA-GP-MIN for `E1_CHT1`. Not only did this significantly improve on forecasting ability, it is significantly more interpretable. Looking at the selected parameters in the GP, it consists of the previous value of `E1_CHT1` added to a combination of `E1_CHT4` (the fourth cylinder head temperature for the same engine), `E1_OilT` (engine 1 oil temperature), `E1_EGT2` and `E1_EGT4` (engine 1 exhaust gasket temperature for the gaskets of cylinders 2 and 4), `OAT` (the outside air temperature), `IAS` (indicated airspeed),

[1]EXA-GP's source code has been made publicly available at REMOVED FOR DOUBLE BLIND REVIEW.

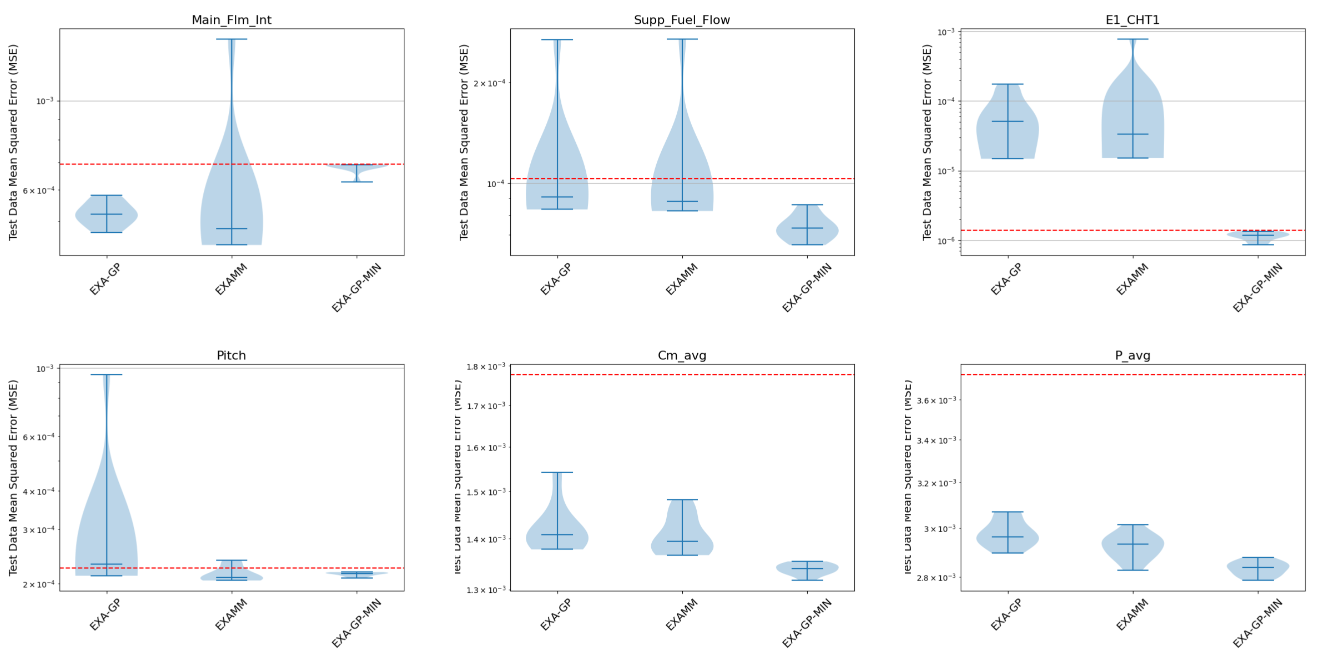

Figure 1: Algorithm performance (Test Dataset MSE) across the 6 benchmark problems. Note that the y-axis is logarithmic.

`LatAC` (lateral acceleration) and `AltGPS` the aircraft's altitude – all of which reasonably would effect the head temperature of an aircraft's engine's cylinders. In particular this parameter is well known to be strongly correlated to other cylinder head temperatures and exhaust gasket temperatures, as well as the outside air temperature.

Due to space constraints, the other evolved GPs are not shown, however Table 1 summarizes the total number of functions and operations used by the best found GPs by EXA-GP and EXA-GP-MIN across the six benchmarks, showing a significant reduction in complexity across all datasets. An additional interesting observation was that EXA-GP-MIN made strong use of the multiply operation (which creates a product of all its inputs multiplied by a bias). This potentially suggests that having multiply nodes may be useful in improving the performance of EXAMM RNNs for TSF.

5 DISCUSSION

This paper introduces a minimal version of the Evolutionary eXploration of Augmenting Genetic Progams (EXA-GP-MIN) algorithm which provides significant advances over the earlier version of EXA-GP. It does this by minimizing the number of trainable parameters coupled with the use of a more intelligent seeding strategy, which utilizes a network initialized to the trivial (but hard to beat, strong local minimum) solution of using the previous parameter values as the forecast, $\hat{x}_{t+1} = x_t$. We show that on a suite of six challenging real world time series forecasting tasks, EXA-GP-MIN not only can provide computational graphs which provide better time series forecasts, even in comparison to EXAMM, an advanced neuroevolution algorithm for recurrent neural networks, but which are also significantly more interpretable.

This work opens up some interesting findings and avenues for future work. In particular, for two of the six benchmarks, EXAMM still provided the best forecasts (and in one case significantly). But in some cases (*e.g.*`Main_Flm_Int`), the TSF problem may be challenging enough that additional complexity is required. Exploring new methods for adaptation to allow EXA-GP-MIN to evolve more complicated layer based operations in a way that still provides interpretability could further advance the algorithm. Additionally, seeding the evolutionary strategy with the trivial solution also provided significant benefit. Adding this capability to EXAMM and EXA-GP to compare if and how it improves their performance would be interesting. Other GGP and CGP methods could also utilize similar strategies in order to evolve better TSF GPs. EXA-GP-MIN was also shown to heavily utilize our multiply node operation, which could also potentially be used in EXAMM to impove its performance. This work also highlights the importance of utilizing backpropagation in graph-based genetic programming algorithms to more effectively learn constants, which in part enables EXA-GP-MIN to achieve its performance even while remaining minimal in the amount of trainable parameters.

REFERENCES

[1] 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.

[2] 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.

[3] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks.. In *Aistats*, Vol. 9. 249–256.

[4] James D Hamilton. 2020. *Time series analysis*. Princeton university press.

	E1_CHT1		Pitch		Main_Flm_Int		Supp_Fuel_Flow		Cm_avg		P_Avg	
	EXA-GP	MIN	EXA-GP	MIN	EXA-GP	MIN	EXA-GP	MIN	EXA-GP	MIN	EXA-GP	MIN
tanh	2	0	1	0	2	0	3	0	2	0	1	0
cos	0	0	1	1	3	0	0	0	1	0	4	0
sin	1	0	0	1	2	2	0	0	1	1	0	0
sigmoid	1	1	1	0	0	0	0	1	2	0	1	0
inverse	0	0	0	2	1	0	0	0	1	0	2	0
+	91	9	68	13	105	20	44	45	69	10	104	3
×	90	12	87	9	136	40	49	6	74	27	127	28

Table 1: A count of the different functions and operations used in the best evolved GPs by EXA-GP and EXA-GP-MIN.

```
H0 = tanh((E1_CHT1 * -2.896758) + (LatAc * 0.707578) + (WndDr * 0.017360) + (Roll * 0.232571)
    + (E1_OilT * 0.177733) + (VSpd * 0.017354) + (E1_CHT2 * 0.089848) + (E1_CHT4 * 0.082369)
    + (E1_EGT4 * 0.026312) + (VSpdG(t - 1) * -1.940293) + (AltGPS(t - 1) * 0.137703)
    + (AltB(t - 1) * -0.232408) + (WndSpd(t - 1) * -0.056200) + (E1_FFlow(t - 1) * 0.013382)
    + (AltAGL(t - 1) * -0.008922) + (E1_OilP(t - 1) * 0.081047) + (BaroA(t - 1) * 0.041858)
    + (IAS(t - 1) * 0.058362) + (E1_CHT2(t - 1) * 0.101083) + (OAT(t - 1) * 0.098187)
    + (Roll(t - 1) * 0.078951) + (E1_OilT(t - 1) * 0.091487) + (FQtyR(t - 1) * 0.082324)
    + (AltMSL(t - 1) * 0.123922) + (H0(t - 1) * 0.417875) + 0.257591)
H1 = (LatAc * 0.355767) + (WndDr * -0.835061) + (E1_OilT * 0.025584) + (E1_EGT4(t - 1) * -0.225284)
    + (IAS(t - 1) * -0.865645) + (E1_OilP(t - 1) * 0.019302) + -1.119541
H2 = sigmoid((AltGPS * -1.741688) + (E1_CHT3 * 0.368240) + (TAS * -0.598604) + (E1_EGT2(t - 1) * -1.069433)
    + (E1_OilP(t - 1) * 0.055360) + (E1_CHT1(t - 1) * 0.008021) + -1.743055)
H3 = sin((AltAGL * 1.571885) + (AltGPS(t - 1) * 0.137199) + (E1_EGT1(t - 1) * -0.342481) + (Pitch(t - 1)
    * 0.286015) + (VSpdG(t - 1) * 0.331632) + (TAS(t - 1) * 0.056812) + (H0(t - 1) * 0.042179) + 1.263766)
E1_CHT1(t + 1) = tanh((AltAGL * -0.377750) + (AltB * -0.118642) + (AltGPS * 0.031915) + (AltMSL * 0.124256)
    + (BaroA * -0.140886) + (E1_CHT1 * 1.365681) + (E1_CHT2 * -0.326045) + (E1_CHT3 * -0.316029) + (E1_CHT4
    * 0.003660) + (E1_EGT1 * -0.115915) + (E1_EGT2 * 0.034150) + (E1_EGT3 * -0.057577) + (E1_EGT4 * -0.424495)
    + (E1_FFlow * -0.142562) + (E1_OilP * 0.001731) + (E1_OilT * 0.273468) + (E1_RPM * 0.283138)
    + (FQtyL * 0.071098) + (FQtyR * 0.018481) + (GndSpd * -0.129576) + (IAS * 0.470138) + (LatAc * -0.193395)
    + (NormAc * -0.319939) + (OAT * -0.211911) + (Pitch * 0.119369) + (Roll * -0.026550) + (TAS * -0.003014)
    + (VSpd * 0.314015) + (VSpdG * -0.342445) + (WndDr * 0.382570) + (WndSpd * -0.016100) + (H0 * -0.558617)
    + (H1 * 0.040181) + (H3 * -0.022892) + (AltMSL(t - 1) * 0.037889) + (OAT(t - 1) * 0.207904)
    + (E1_RPM(t - 1) * -0.012689) + (VSpd(t - 1) * 0.023599) + (VSpdG(t - 1) * -0.062903)
    + (Roll(t - 1) * 0.085487) + (E1_EGT4(t - 1) * 0.121840) + (LatAc(t - 1) * 0.150719) + (E1_EGT3(t - 1)
    * 0.071013) + (E1_CHT4(t - 1) * 0.079659) + (H0(t - 1) * 0.079597) + (H2(t - 1) * 0.259861) + 0.462119)
```

Figure 2: The best genetic program evolved by the EXA-GP algorithm on the `E1_CHT1` benchmark.

```
H0 = sigmoid(AltGPS + E1_EGT2 + LatAc + TAS + E1_RPM(t - 1) + 0.120529)
H1 = E1_OilT * H0 * IAS(t - 9) * OAT(t - 4) * LatAc(t - 2) * E1_OilT(t - 3) * -0.019811
H2 = OAT * E1_EGT4 * H0 * E1_EGT4(t - 8) * Pitch(t - 10) * E1_EGT2(t - 10) * 0.040420
E1_CHT1(t + 1) = E1_CHT1 + H1 + H2 + H1(t - 5)
```

Figure 3: The best genetic program evolved by the EXA-GP-MIN algorithm on the `E1_CHT1` benchmark, which significantly outperforms the best found EXA-GP genetic program on the same task.

[5] 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.

[6] 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).

[7] Zimeng Lyu, AbdElRahman ElSaid, Joshua Karns, Mohamed Mkaouer, and Travis Desell. 2021. An Experimental Study of Weight Initialization and Lamarckian Inheritance on Neuroevolution. *The 24th International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps)* (2021).

[8] 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).

[9] 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).

[10] Jared Murphy, Devroop Kar, Zimeng Lyu, and Travis Desell. 2024. EXA-GP: Unifying Graph-Based Genetic Programming and Neuroevolution for Explainable Time Series Forecasting. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*.

[11] 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

[12] 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.

[13] Hojjat Rakhshani, Hassan Ismail Fawaz, Lhassane Idoumghar, Germain Forestier, Julien Lepagnot, Jonathan Weber, Mathieu Brévilliers, and Pierre-Alain Muller. 2020. Neural architecture search for time series classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

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

[15] 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

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

[17] 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.

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

[19] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 11121–11128.