Towards Evolving Creative Algorithms: Musical Time Series Forecasting with Tangled Program Graphs

Ali Naqvi[1] and Stephen Kelly[2]

[1]McMaster University, Canada naqvia18@mcmaster.ca
[2]McMaster University, Canada kellys32@mcmaster.ca

Introduction

Artificial Intelligence (AI) has achieved significant milestones in fields governed by clear rules, such as theorem proving and strategy games like chess (Agüera y Arcas, 2017). However, applying AI to music composition presents a unique challenge due to the subjective nature of evaluating creative work; the difficulty in designing a fitness function capable of measuring the creativity and uniqueness of solutions remains an unsolved issue (McCormack, 2005). Even in the realm of generative machine learning, the focus has been modelling large datasets and producing plausible recreations rather than generating novel works or simulating the creative process itself.

Music is widely regarded as a product of evolution, with theories on the cultural evolution of music (Savage, 2019) and ethnomusicology studies (Berger and Stone, 2019) suggesting music and culture co-evolve. This insight leads to the proposition that genetic programming could foster innovative compositions by modelling evolutionary processes. Specifically, an evolutionary search for computer programs has great potential for *exploratory creativity* through the discovery of new programs, and transformational creativity by automatically encapsulating reusable code modules (e.g. functions) and performing a meta-search at the function/module level. A bi-level search of this nature may facilitate *insight*, a fundamental aspect of creative problem solving (Colin et al., 2016).

In this context, we propose using Tangled Program Graphs (TPG) (Kelly et al., 2021a), a highly modular genetic programming framework, to evolve music-generating agents that require a fraction of the computational and storage demands typically associated with Deep Neural Networks. TPGs have yet to be explored for solving multivariate time series forecasting problems, making this an exciting and relatively untapped area of research.

Methodology

MIDI Music Representation

To represent the MIDI file as a multivariate time series that can also be reversed back to MIDI, we identified the most important features that need to be included in a time step. Each time step consists of the *offset*: the time location of the note, *duration*: the held length of the note, and the *pitch*: the note. To make the time location of the notes stationary such that the data can be normalized, we use the first-order difference between notes (i.e. the change in seconds between one note and another). Expression symbols such as dynamics of pitches are omitted, as our focus is on generating note sequences rather than reproducing subtleties of a particular performance. We hypothesize that this representation will be suitable for multivariate time series forecasting. However, in this initial study we will focus solely on predicting pitch sequences.

The representation of the MIDI file was validated by comparing two MIDI files from the MidiFind system (Xia et al., 2014). We performed a subjective listening assessment post-encoding and decoding of the input MIDI file and detected no significant data loss.

Tangled Program Graphs

Tangled Program Graphs (TPGs) are a genetic programming framework initially designed for Reinforcement Learning (RL)(Kelly and Heywood, 2018; Kelly et al., 2021a). While Recurrent Deep Learning Networks are prevalent for time series forecasting, TPGs have shown comparable competence at a fraction of the computational and storage cost(Desnos et al., 2021). TPG agents are represented by teams of programs where each program is a sequence of instructions that operate on sensory inputs and internal memory registers and return two values: a bid value and an action value. The team's *prediction* at each time step is the action value of the highest bidder. Teams and programs are stored in separate populations and co-evolved, effectively performing a bi-level search in which programs learn the context in which their output is appropriate, and teams learn which programs work well together collectively, (see Figure 1). Finally, each program's action output can either be a computed scalar prediction (s1) or a pointer to another team (i.e. an action delegation). This property allows evolution to build team hierarchies, or *program graphs*.

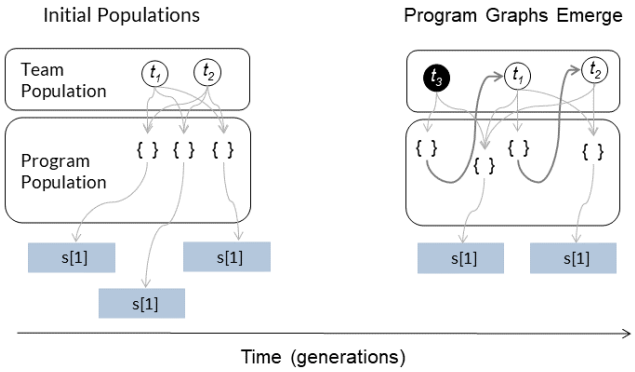

Figure 1: Illustration of multi-level TPG structure. Detailed description can be found in (Kelly and Heywood, 2018).

Agents are shown a sliding window of 50 values at a time and rely on TPG's temporal memory mechanisms (Kelly et al., 2021a) to learn mental models of the environment from which to predict future values.

Recursive Forecasting We use the recursive forecasting paradigm in which agents are *primed* with 50 samples and then evaluated on how well they predict the next 50 samples through feedback by connecting the agent's output back to its input (Kelly et al., 2020). This methodology supports sequence predictions up to any horizon, and opens the possibility of creative sequence generation well beyond the horizon used for training. Indeed, our test procedure measures how well agents recursively predict 100 samples, twice the horizon used during training (see Figure 3).

Experiments

Experimental setup We use "Ave Maria" by Bach/Gounod as a test piece and compare the utility of three fitness functions: Mean Squared Error (MSE) (Botchkarev, 2019), Pearson Correlation (Haut et al., 2023), and U of Theil (Lima et al., 2016). 20 evolutionary repeats with unique random seeds are conducted for each fitness function. In order to compute the fitness, we wait until the end of each recursive forecasting episode to gather all the predictions and compare them to their targeted values. This is because the U of Theil and correlation require more than the current values to be effective; U of Theil is normalized by a random walk forecast error (Lima et al., 2016) and correlation requires the mean of its window of inputs (Haut et al., 2023). The parameters used for TPG generally follow previous studies (Kelly et al., 2021b).

Results To compare the fitness functions on a common scale, we used MSE as the standard metric. When running 20 tests per fitness function, U of Theil produced the best-performing agent on the testing data (see Figure 2). On average, however, U of Theil performed close to MSE. Not surprisingly, the correlation metric resulted in the worst MSE. However, the combination of correlation with a post-evolution alignment step is effective in symbolic regression tasks (Haut et al., 2023) and may also be worth investing in time series forecasting.

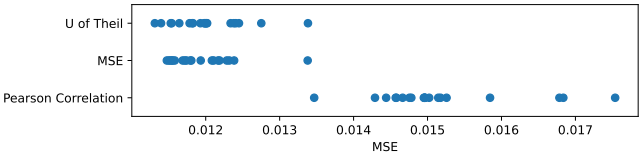

Figure 2: Test performance of single-best agents from 20 repeats with each fitness functions. Test forecasts are measured with MSE for the purpose of comparison.

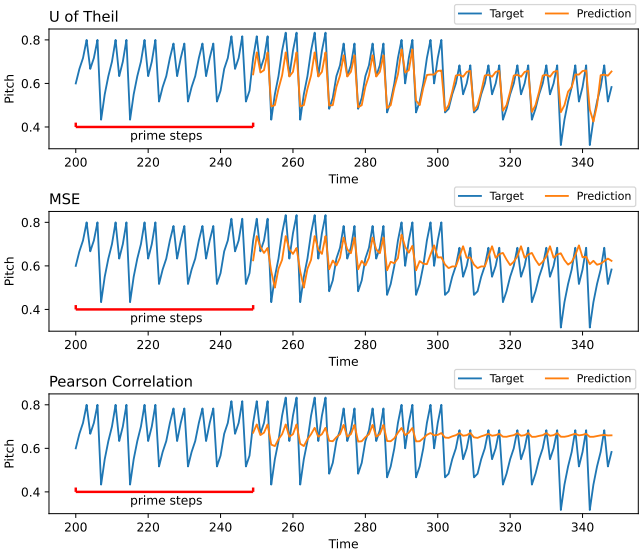

Figure 3: Comparison of recursive forecasting behaviours discovered using three distinct fitness functions.

As can be seen in a sample of test results (see Figure 3), agents trained with U of Theil predict changes in the pitch sequence that occur beyond the 50-step prime input (See the change at step 300).

Conclusion

Our results demonstrate that Tangled Program Graphs show promise for recursive time series prediction in an unpredictable task such as music, even with limited data. However, the choice of fitness function is critical. The U of Theil performs best when the goal is to achieve the optimal value rather than settling for a sub-optimal value. This can be partially attributed to U of Theil benefiting from double the iterations compared to MSE. On the other hand, MSE would be ideal if faster convergence is the goal. Future work will consider the multivariate prediction of complete MIDI music representations.

References

Blaise Agüera y Arcas. 2017. Art in the Age of Machine Intelligence. *Arts* 6, 4 (2017). https://doi.org/10.3390/arts6040018

Harris M. Berger and Ruth M. Stone. 2019. *Theory for Ethnomusicology: Histories, Conversations, Insights*. Routledge, New York ; London.

Alexei Botchkarev. 2019. A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management* 14 (2019), 045–076. https://doi.org/10.28945/4184

Thomas R. Colin, Tony Belpaeme, Angelo Cangelosi, and Nikolas Hemion. 2016. Hierarchical reinforcement learning as creative problem solving. *Robotics and Autonomous Systems* 86 (2016), 196–206.

Karol Desnos, Nicolas Sourbier, Pierre-Yves Raumer, Olivier Gesny, and Maxime Pelcat. 2021. Gegelati: Lightweight Artificial Intelligence through Generic and Evolvable Tangled Program Graphs. In *Workshop on Design and Architectures for Signal and Image Processing (14th edition) (DASIP '21)*. ACM. https://doi.org/10.1145/3441110.3441575

Nathan Haut, Wolfgang Banzhaf, and Bill Punch. 2023. *Correlation Versus RMSE Loss Functions in Symbolic Regression Tasks*. Springer Nature Singapore, Singapore, 31–55. https://doi.org/10.1007/978-981-19-8460-0_2

Stephen Kelly and Malcolm I. Heywood. 2018. Emergent Solutions to High-Dimensional Multitask Reinforcement Learning. *Evolutionary Computation* 26, 3 (09 2018), 347–380. https://doi.org/10.1162/evco_a_00232 arXiv:https://direct.mit.edu/evco/article-pdf/26/3/347/1552375/evco_a_00232.pdf

Stephen Kelly, Jacob Newsted, Wolfgang Banzhaf, and Cedric Gondro. 2020. A modular memory framework for time series prediction. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 949–957. https://doi.org/10.1145/3377930.3390216

Stephen Kelly, Robert J. Smith, Malcolm I. Heywood, and Wolfgang Banzhaf. 2021a. Emergent Tangled Program Graphs in Partially Observable Recursive Forecasting and ViZDoom Navigation Tasks. *ACM Trans. Evol. Learn. Optim.* 1, 3, Article 11 (aug 2021), 41 pages. https://doi.org/10.1145/3468857

Stephen Kelly, Robert J. Smith, Malcolm I. Heywood, and Wolfgang Banzhaf. 2021b. Emergent Tangled Program Graphs in Partially Observable Recursive Forecasting and ViZDoom Navigation Tasks. *ACM Trans. Evol. Learn. Optim.* 1, 3, Article 11 (aug 2021), 41 pages. https://doi.org/10.1145/3468857

Aranildo Lima, Paulo De Mattos Neto, David Silva, and Tiago Ferreira. 2016. Tests With Different Fitness Functions For Tuning Of Artificial Neural Networks With Genetic Algorithms. 1–8. https://doi.org/10.21528/CBIC2011-32.5

Jon McCormack. 2005. Open Problems in Evolutionary Music and Art. In *Applications of Evolutionary Computing*, Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, David Wolfe Corne, Rolf Drechsler, Yaochu Jin, Penousal Machado, Elena Marchiori, Juan Romero, George D. Smith, and Giovanni Squillero (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 428–436.

P.E. Savage. 2019. Cultural evolution of music. *Palgrave Communications* 5 (2019), 16. https://doi.org/10.1057/s41599-019-0221-1

Guangyu Xia, Tongbo Huang, Yifei Ma, Roger Dannenberg, and Christos Faloutsos. 2014. MidiFind: Similarity Search and Popularity Mining in Large MIDI Databases. In *Sound, Music, and Motion*, Mitsuko Aramaki, Olivier Derrien, Richard Kronland-Martinet, and Sølvi Ystad (Eds.). Springer International Publishing, Cham, 259–276.