# Neuro-Guided Graph Search for Symbolic Regression (Student Abstract)

**Piotr Wyrwiński, Krzysztof Krawiec**

Institute of Computing Science, Poznan University of Technology, Poznan, Poland
{piotr.wyrwinski, krawiec}@cs.put.poznan.pl

## Abstract

This study introduces a neurosymbolic approach that performs iterative graph expansion guided by a graph neural network to solve symbolic regression problems. Empirical evaluation demonstrates superior performance of the method compared to baseline algorithms. We also integrate the method with an evolutionary algorithm, which results in further performance improvements.

## Motivations

*Symbolic regression* (SR) (Koza 1994; Schmidt and Lipson 2009) is a generalization of the conventional regression, where both the mathematical formula of the model and its parameters are synthesised from data. Formally, the task is to discover a formula that maps $d$ independent variables $\mathbf{x}_i \in \mathbb{R}^n$, represented as columns in matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d]$, to a dependent variable $\mathbf{y}$, aiming to minimise the approximation error (e.g., Mean Squared Error) on a training dataset $T = \{(\mathbf{x}^{(j)}, y^{(j)})\}$. Solving SR tasks involves joint combinatorial search in the space of symbolically represented formulas and continuous optimization of its parameters. The interplay between these two search spaces is very complex, because qualitative modifications of the formula (e.g. replacement of a variable or mathematical operator) are discontinuous and not differentiable, rendering gradient-based search methods useless. Also, the number of formulas grows exponentially with the depth of the expression, and may easily exhaust memory even for small vocabularies of symbols (SR is NP-hard (Virgolin and Pissis 2022)).

## Method

We present *Neuro gUideD Graph sEarch* (NUDGE), a neurosymbolic algorithm that explores the space of SR models using the guidance obtained from a trained graph neural network (GNN). NUDGE iteratively builds candidate SR models in a bottom-up fashion, gradually merging small models into larger ones. The search is guided by a GNN queried on the graph spanning all SR models generated so far.

NUDGE starts from the initial edgeless digraph $G_0 = (O \cup V, \emptyset)$, where $O$ is the set of mathematical operators and $V$ the set of terminals (variables). It builds $G_{k+1}, k > 0$ by expanding $G_k = (N, E)$ with a 'layer' of new nodes and edges chosen by the GNN: (i) *application nodes* $N_a$, each representing the application of an operation from $O$ to arguments from $N$, and (ii) *value nodes*, which represent the outcomes of those applications. Adding an application node to $N$ is followed by extending $E$ with the edges that capture causal dependencies: one incoming edge from an operation node ($\in O$), $e \geq 1$ edges ($e$ being the operation's arity) incoming from variable, constant, and value nodes created in previous iterations, and one outgoing edge leading to the value node $v$ representing the outcome of the operation.

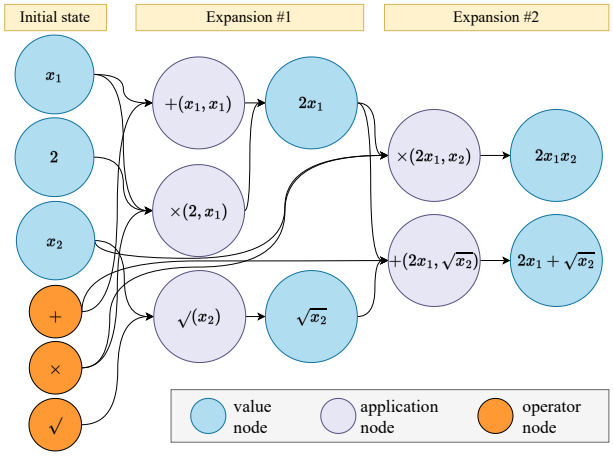


Figure 1: An example of a graph constructed by NUDGE.

Each graph node is presented to the GNN via its *node representation*, which is a vector that captures (i) the type of the node, (ii) a one-hot encoded index of the operator (for operator nodes), and (iii) the *value embedding* (for $v$s only). For the latter, we follow (Kamienny et al. 2022) and consider two types of value encoding: a *binary* (BIN) method and a variant of *positional encoding* (POS) commonly used in Transformers (Vaswani et al. 2017). In contrast to our preliminary study (Wyrwiński and Krawiec 2024), where the GNN was queried independently for each example, the current approach utilises a transformer-like *Cross-Example Attention Mechanism*. This mechanism captures not only the complex dependencies between the independent variables and the dependent variable, but also models interactions

across the entire dataset $T$, effectively aggregating information across all examples – similar to sentence classification or set representation (Devlin et al. 2019; Lee et al. 2019).

We adopt a GNN architecture inspired by the Graph Attention Network (GAT) framework (Veličković et al. 2018). The GNN comprises three layers of GAT, each designed to facilitate a round of message passing that updates the state for every node. Each layer consists of four attention heads operating in parallel. The final state vector for each node is passed through a sigmoid-activated output layer and use to partition nodes into those selected for expansion and the rest.

We also devise stateful variants of GAT (GATState) that maintain memory across iterations of graph expansion as a *global graph state* defined as the average of all node states, to capture temporal dependencies across graph expansions. GATStateRNN extends GATState by processing the global graph state with a GRU layer (Chung et al. 2014) before integrating it with node states.

The training set comprises expressions involving 1 to 6 input variables, constants 0, 1, 2, 3, and $\pi$, binary operators $+, -, \times, \div$, and functions $\sqrt{x}, x^2, x^3, \sin, \cos, \log$, and $\exp$. For each SR expression $p$, we generate a set of $n = 50$ data points $T = \{(\mathbf{x}^{(j)}, y^{(j)})\}$, where $j \in [1, n]$. The values of the independent variables $x_i$ are drawn uniformly from the interval $[1, 5]$, and $y^{(j)}$ is calculated as $p(\mathbf{x}^{(j)})$. Each tuple $(p, T)$ thus forms a distinct SR problem instance. With the inclusion of expressions of height up to 6, we compiled 1032 SR problems and randomly divided them into a training set of 522 problems and a test set of 510 problems.

The key aspect of the proposed method is the availability of the target program for each SR problem instance, which allows us to map out the *trajectory* that the graph expansion should follow to construct the desired expression. In training, we iteratively query the GNN, perform expansions and determine which subexpressions are present in the target solution (SR model). This partitions the application nodes into those that should and should not be expanded, which in turn determines the target values for the binary cross-entropy loss function applied to the output of the GNN.

To address NUDGE's challenge of exponential growth of the 'search front', we hybridise it in EvoNUDGE with an evolutionary algorithm (Wyrwiński and Krawiec 2024). Given an SR problem, we first invoke NUDGE and allow it to perform $h$ expansions of the graph. The subexpressions of the final graph $G_h$ produced by NUDGE are then used to populate a *library* $l$, to be later sampled from by the evolutionary search equipped with library-based initialisation and mutation operator. For instance, the graph shown in Fig. 1 would translate into a library comprising subexpressions $x_1$, 2, $x_2$, $2x_1$, $\sqrt{x_2}$, $2x_1x_2$, and $2x_1 + \sqrt{x_2}$. The evolutionary search cycles through fitness-based selection, applies mutation and crossover to generate offspring, and updates the population with the new candidates. As baselines, we use the search operators that are popular in *genetic programming* (GP), i.e. one-point crossover and subtree-replacing mutation. We consider EvoNUDGE variants that use (i) only the library-based initialisation of the population ($I$), (ii) only the library-based mutation (probability 0.2, $M$), (iii) the library-based mutation or the baseline mutation with a 50/50 chance ($MM$), and (iv) all these operators simultaneously ($IM$, $IMM$). The settings of the evolutionary algorithm are the same as in (Wyrwiński and Krawiec 2024).

| Method | | $k=2$ | $k=3$ | $t=60s$ |
|---|---|---|---|---|
| GAT$^\star$ | $BIN$ | **11.20** | 13.75 | 14.93 |
| GAT | $BIN$ | 9.04 | 11.79 | 12.57 |
| GATState | $BIN$ | 10.61 | **14.54** | **15.32** |
| GATStateRNN | $BIN$ | 9.04 | 11.59 | 12.38 |
| GAT | $POS$ | 11.00 | 12.77 | 14.34 |
| GATState | $POS$ | 10.81 | 13.36 | 14.73 |
| GATStateRNN | $POS$ | 10.81 | 13.36 | 14.93 |

Table 1: Success rates (percentage of successful runs) for various number of graph expansions $k$ or time budget $t$.

| Method | | $h=1$ | $h=2$ | $h=3$ |
|---|---|---|---|---|
| GP | | 21.65 | 22.68 | 21.65 |
| GATState | $BIN$ | 4.12 | 7.22 | 7.22 |
| EvoNUDGE | $I$ | 22.68 | 23.71 | 23.71 |
| | $M$ | 27.84 | 27.84 | 28.87 |
| | $MM$ | 27.84 | 27.84 | 27.84 |
| | $IM$ | 28.87 | 25.77 | 27.84 |
| | $IMM$ | 30.93 | 25.77 | 28.87 |

Table 2: Success rates on the AI Feynman suite of benchmarks for various heights $h$ of the subtrees.

## Results

Table 1 presents the performance of NUDGE configurations when tackling the 510 test problems from our generated database of SR problems. We compare the GAT$^\star$ baseline (Wyrwiński and Krawiec 2024) and six configurations that combine BIN and POS with GAT, GATState, and GATStateRNN. A success is defined as achieving test-set MSE $< 10^{-10}$. GATState fares best, corroborating the usefulness of incorporating the state from the previous iterations.

We evaluated the out-of-sample efficacy of NUDGE and EvoNUDGE (with GATState (BIN) configuration) by applying them to the AI Feynman suite of regression problems (Udrescu and Tegmark 2020) comprising equations extracted from the *Feynman Lectures on Physics* (Feynman, Leighton, and Sands 2010). Table 2 summarises the success rates on this suite, including two baselines: the GATState (BIN) (best in Table 1), and the purely evolutionary approach (GP), which starts from a randomly initialised population and does not benefit from NUDGE's guidance. The results indicate that EvoNUDGE brings evident benefits. The odds of synthesizing a correct SR model are not only systematically much higher than those for standalone NUDGE, but also better than for the purely evolutionary GP. This holds for all combinations of NUDGE-informed search operators, though the configurations involving the mutation operators ($M$ and $MM$) benefit the most, suggesting that it pays off to not only 'prime' the initial set of candidate SR models, but also to continue the guidance throughout the evolutionary search.

## Acknowledgments

This research was supported by the statutory funds of Poznan University of Technology and the Polish Ministry of Science and Higher Education, grants no. 0311/SBAD/0740 and 0311/SBAD/0752, and by National Science Centre, Poland, grant no. 2024/53/N/ST6/03961.

## References

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Feynman, R. P.; Leighton, R. B.; and Sands, M. 2010. *The Feynman lectures on physics; New millennium ed.* New York, NY: Basic Books. Originally published 1963-1965.

Kamienny, P.-A.; d'Ascoli, S.; Lample, G.; and Charton, F. 2022. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35: 10269–10281.

Koza, J. R. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4: 87–112.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; and Teh, Y. W. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, 3744–3753. PMLR.

Schmidt, M.; and Lipson, H. 2009. Distilling free-form natural laws from experimental data. *science*, 324(5923): 81–85.

Udrescu, S.-M.; and Tegmark, M. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16): eaay2631.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Virgolin, M.; and Pissis, S. P. 2022. Symbolic Regression is NP-hard. *Transactions on Machine Learning Research*.

Wyrwiński, P.; and Krawiec, K. 2024. Guiding Genetic Programming with Graph Neural Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 551–554.