EuroGP 2023 Recap

Hot Trends in Genetic Programming

Hengzhe Zhang

Victoria University of Wellington

26/04/2023

Hot Trends

According to the EuroGP 2023 article list, the hot trends in GP mainly involve:

- Adaptive Methods for GP
- GP for Symbolic Regression
- Knowledge Guided GP
- GP for Machine Learning
- GP in Domain Applications

Table of Contents

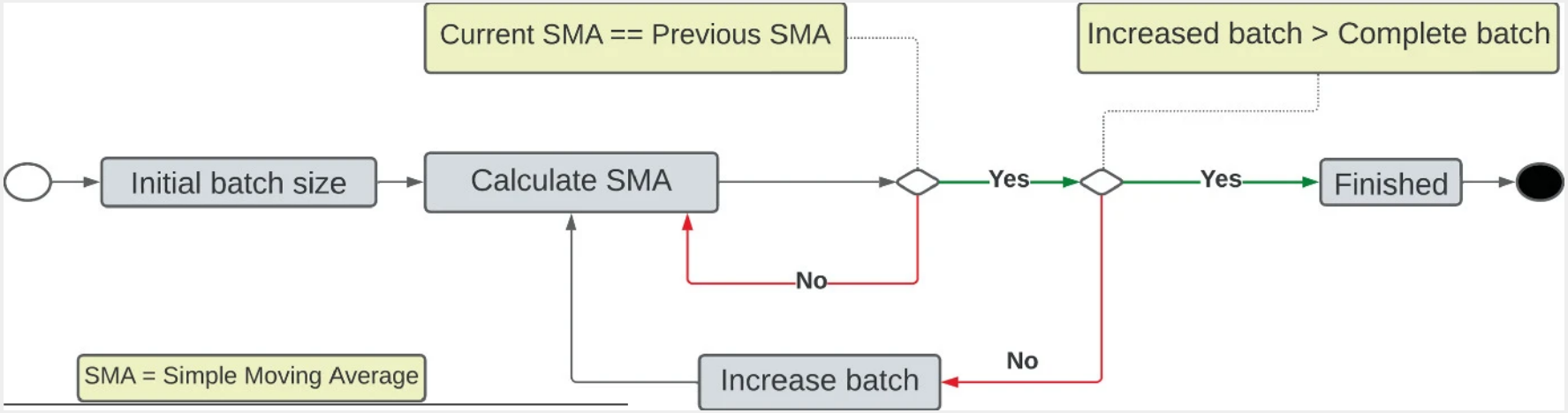
Adaptive Methods for GP

1. Bryan Martins Lima et al. (2023). “Adaptive Batch Size CGP: Improving Accuracy and Runtime for CGP Logic Optimization Flow”. In: *EuroGP 2023*, pp. 149–164
2. José Ferreira et al. (2023). “A Self-Adaptive Approach to Exploit Topological Properties of Different GAs’ Crossover Operators”. In: *EuroGP 2023*, pp. 3–18
3. Pedro Carvalho et al. (2023). “Context Matters: Adaptive Mutation for Grammars”. In: *EuroGP 2023*, pp. 117–132

Adaptive Batch Size

Adaptive Batch Size CGP: Improving Accuracy and Runtime for CGP Logic Optimization Flow[1]

- Initialization: Start with an initial batch size (β) for evaluating individuals in the CGP population.
- Stagnation detection: Track the Simple Moving Average (SMA) of the accuracy over a window of σ generations, and compare it with the previous SMA to detect stagnation.

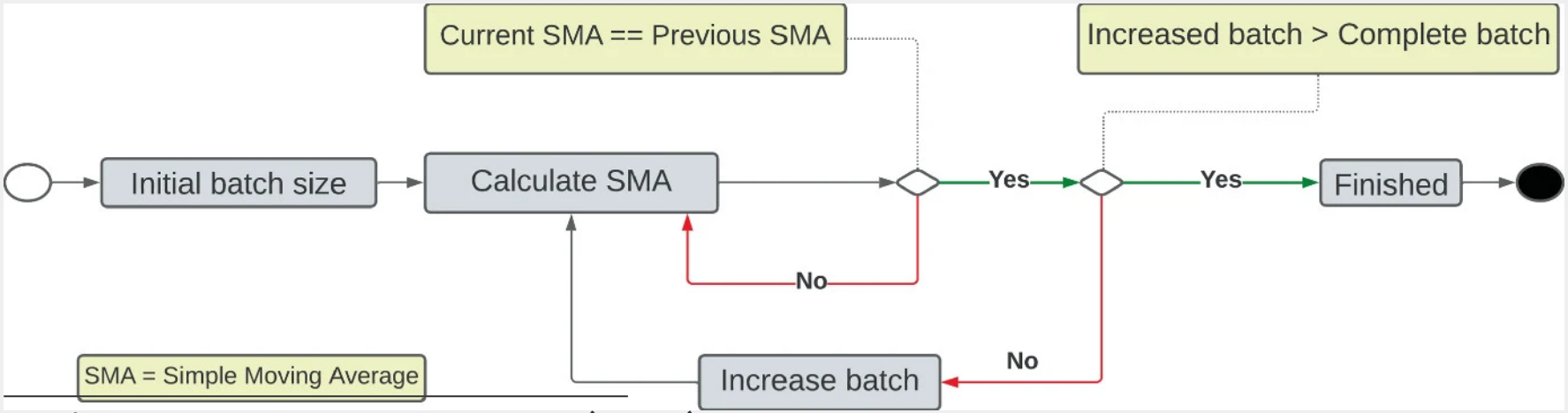


[1]Bryan Martins Lima et al. (2023). "Adaptive Batch Size CGP: Improving Accuracy and Runtime for CGP Logic Optimization Flow". In: *EuroGP 2023*, pp. 149–164.

Adaptive Batch Size

Adaptive Batch Size CGP: Improving Accuracy and Runtime for CGP Logic Optimization Flow[1]

- Batch size adjustment: Increase the batch size by α terms if stagnation is detected, provided it doesn't surpass the total available data.
- Maximum data utilization: If all available data is being used, continue the search using the standard CGP algorithm without further ABS adjustments.



[1]Bryan Martins Lima et al. (2023). "Adaptive Batch Size CGP: Improving Accuracy and Runtime for CGP Logic Optimization Flow". In: *EuroGP 2023*, pp. 149–164.

Adaptive Batch Size

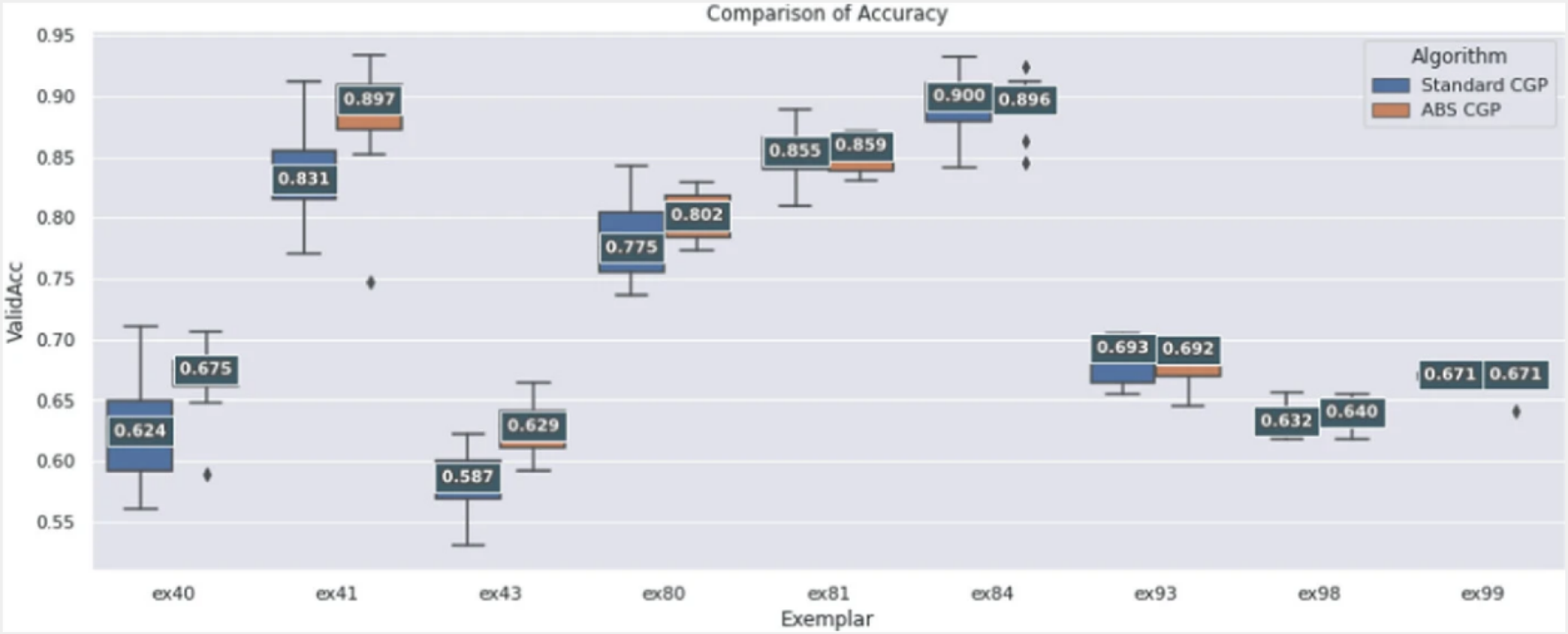


Figure: Accuracy of Standard CGP and ABS CGP

Adaptive Batch Size

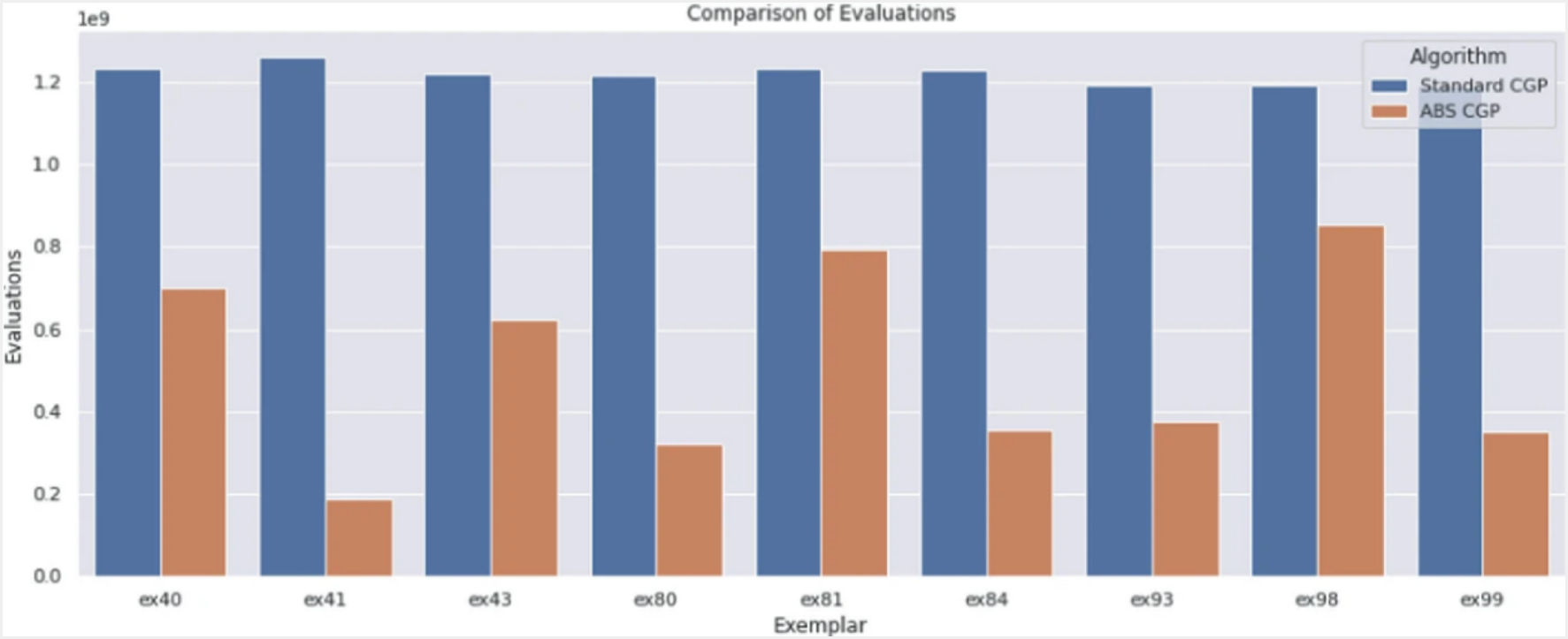
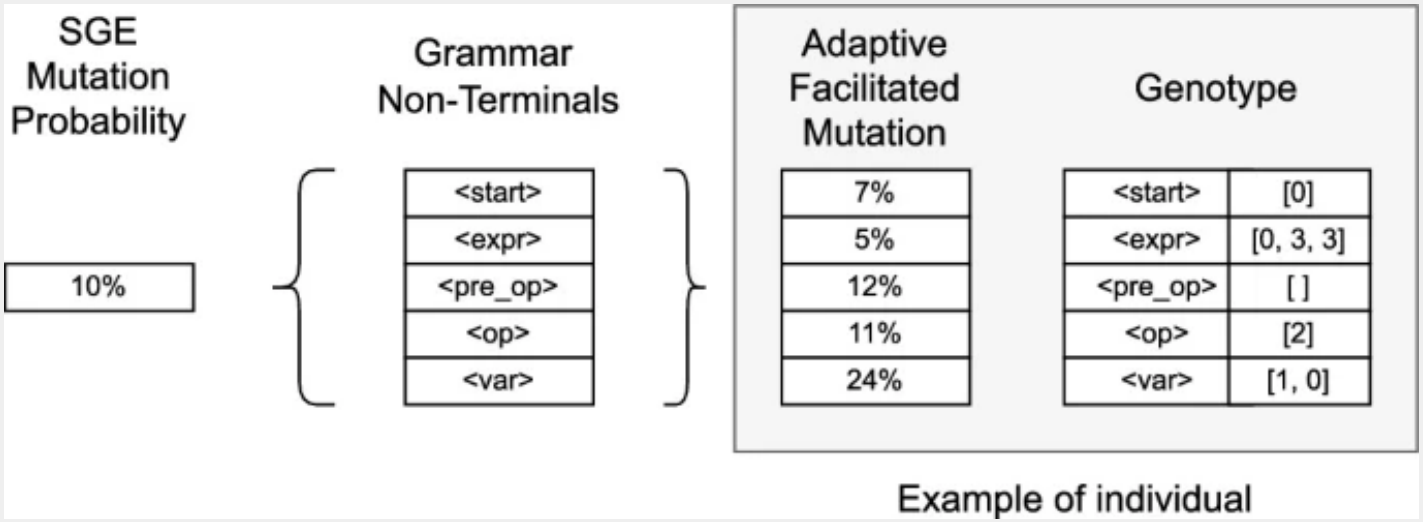


Figure: Number of evaluations between ABS CGP and Standard CGP

Adaptive Mutation for GP

Context Matters: Adaptive Mutation for Grammars[1]

1. Initialize a population of individuals, each carrying a mutation array containing mutation probabilities for each non-terminal.
2. Determine which non-terminals to mutate based on the mutation probabilities.

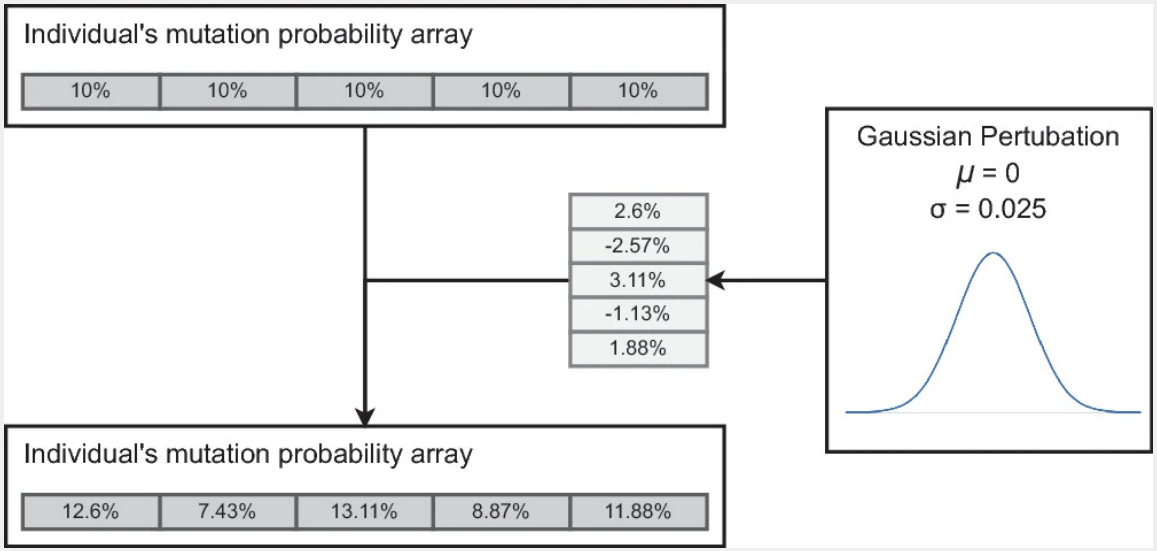


[1] Pedro Carvalho et al. (2023). "Context Matters: Adaptive Mutation for Grammars". In: *EuroGP 2023*, pp. 117–132.

Adaptive Mutation for GP

Context Matters: Adaptive Mutation for Grammars[1]

1. During each generation, adjust the mutation probabilities in the mutation array using a random value sampled from a Gaussian distribution with mean 0 and a configurable standard deviation (σ).
2. During crossover, offspring inherit the mutation array from their fittest parent.



[1]Pedro Carvalho et al. (2023). "Context Matters: Adaptive Mutation for Grammars". In: *EuroGP 2023*, pp. 117–132.

Adaptive Mutation for GP

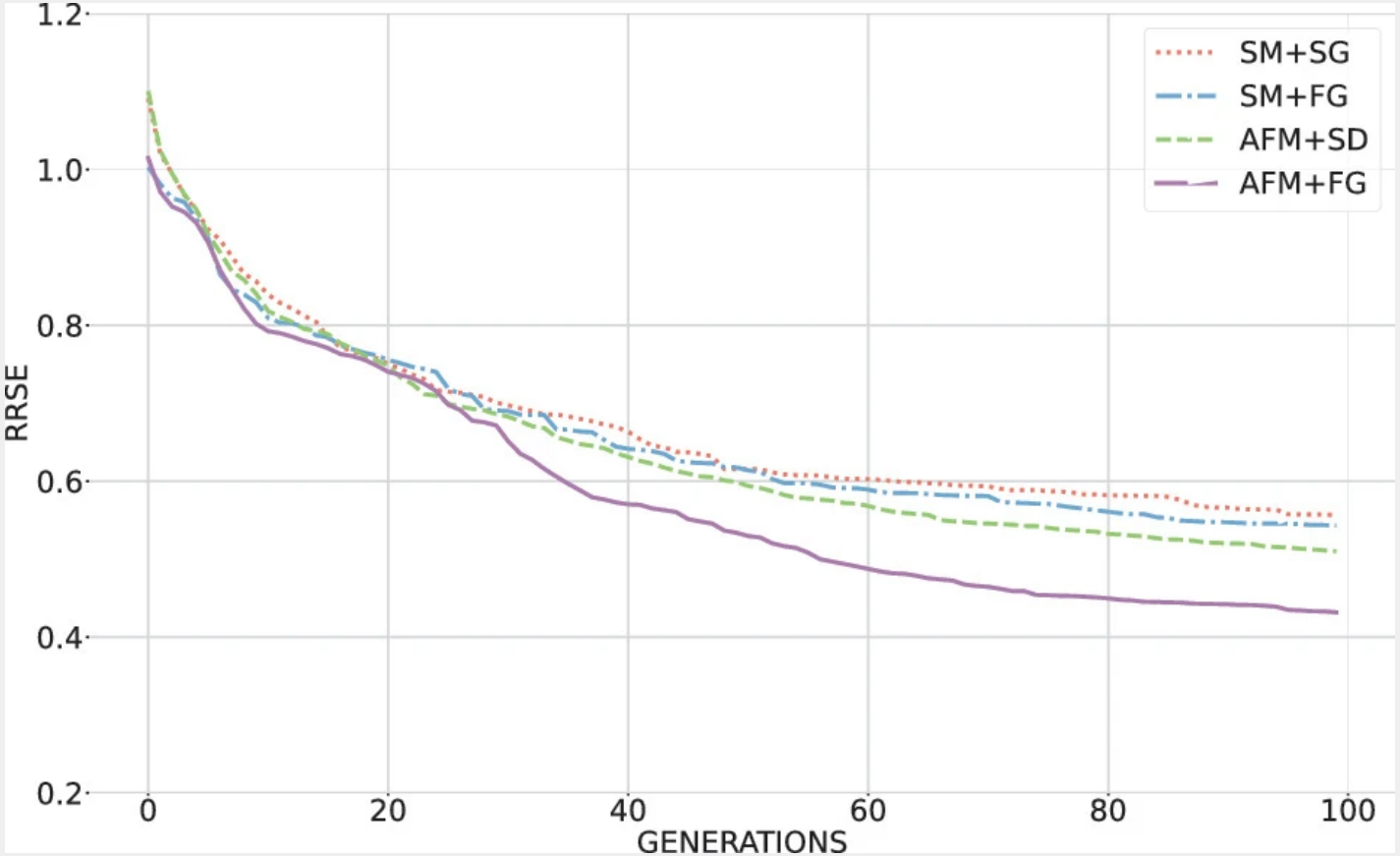


Figure: The mean best fitness of 100 runs for the Pagie polynomial.

Adaptive Mutation for GP

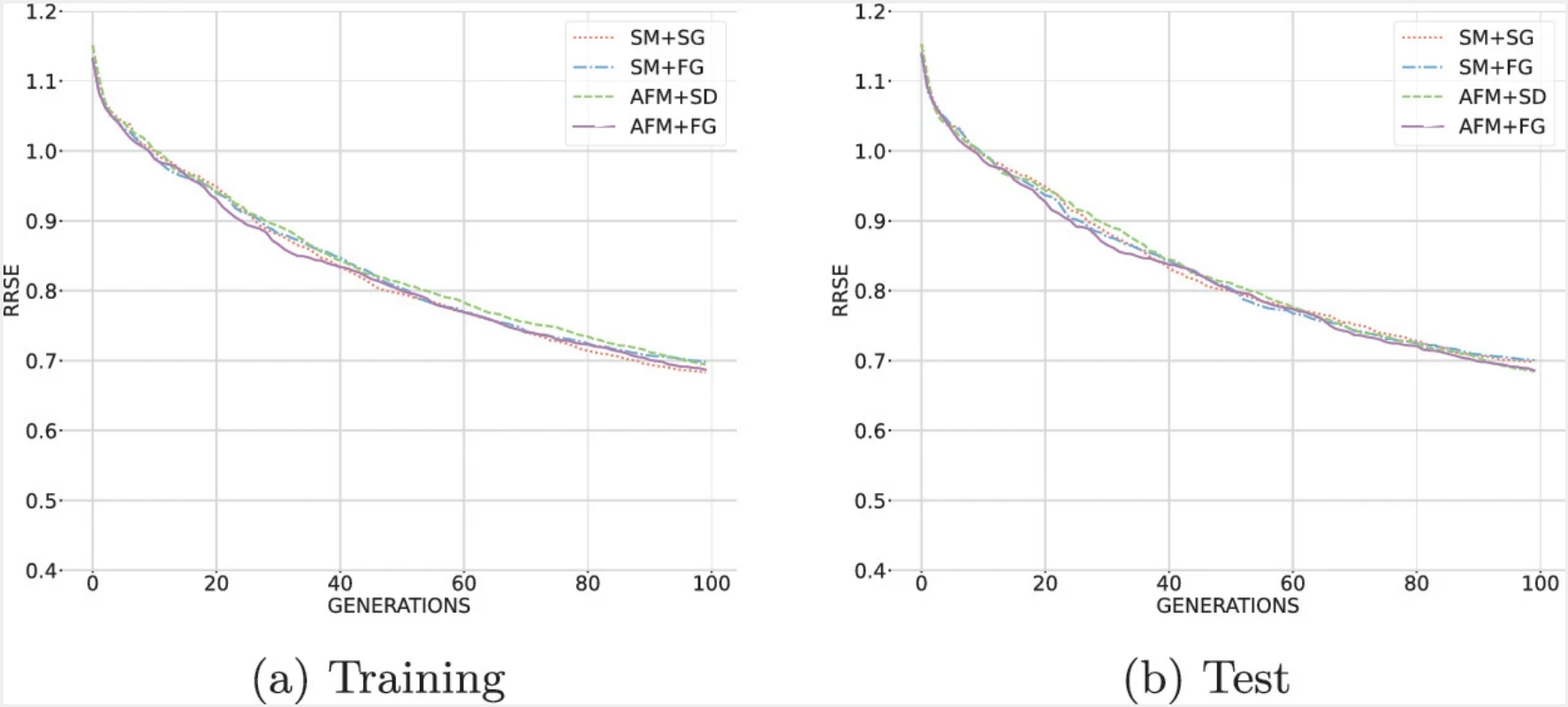


Figure: The mean best fitness of 100 runs for the Boston Housing dataset.

Genetic Programming for SR

1. David Wittenberg and Franz Rothlauf (2023). “Small Solutions for Real-World Symbolic Regression Using Denoising Autoencoder Genetic Programming”. In: *EuroGP 2023*, pp. 101–116
2. Ting Hu, Gabriela Ochoa, and Wolfgang Banzhaf (2023). “Phenotype Search Trajectory Networks for Linear Genetic Programming”. In: *EuroGP 2023*, pp. 52–67
3. Alessandro Leite and Marc Schoenauer (2023). “Memetic Semantic Genetic Programming for Symbolic Regression”. In: *EuroGP 2023*, pp. 198–212

Small Solutions for Real-World Symbolic Regression Using Denoising Autoencoder Genetic Programming[1]

Step 1: Model Building

1. Select promising candidate solutions to form a training set *X*
2. Split training set into training and validation sets
3. Apply Levenshtein tree edit denoising strategy with chosen corruption strength (edit percentage p)
4. Train DAE-LSTM on denoised solutions to reconstruct original (uncorrupted) solutions
5. Calculate reconstruction error and update model parameters using gradient descent
6. Apply early stopping based on the validation set

[1] David Wittenberg and Franz Rothlauf (2023). "Small Solutions for Real-World Symbolic Regression Using Denoising Autoencoder Genetic Programming". In: *EuroGP 2023*, pp. 101–116.

Step 2: Model Sampling[1]

1. Select random candidate solutions from the training set
2. Apply the same denoising strategy as during model building
3. Propagate denoised solutions through the trained DAE-LSTM to generate new offspring candidate solutions
4. Ensure generated offspring solutions are syntactically valid

Step 3: Evaluate and Update

1. Evaluate fitness of offspring population using the problem-specific fitness function
2. Update parent population by selecting the best individuals from parent and offspring populations

[1] Zhi-Hui Zhan et al. (2022). "Learning-aided Evolution for Optimization". In: *IEEE Transactions on Evolutionary Computation*.

DAE-GP

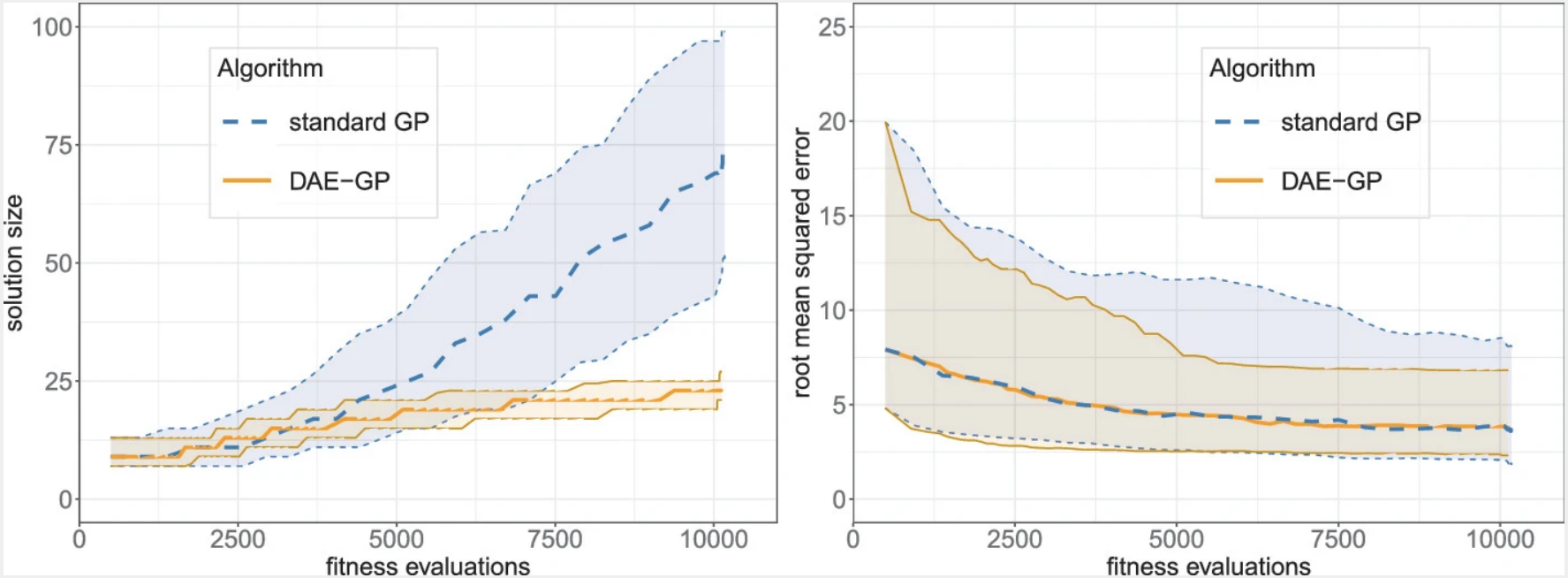


Figure: Median solution size (left) and median fitness on test set (right) of best candidate solution over fitness evaluations across all datasets.

DAE-GP

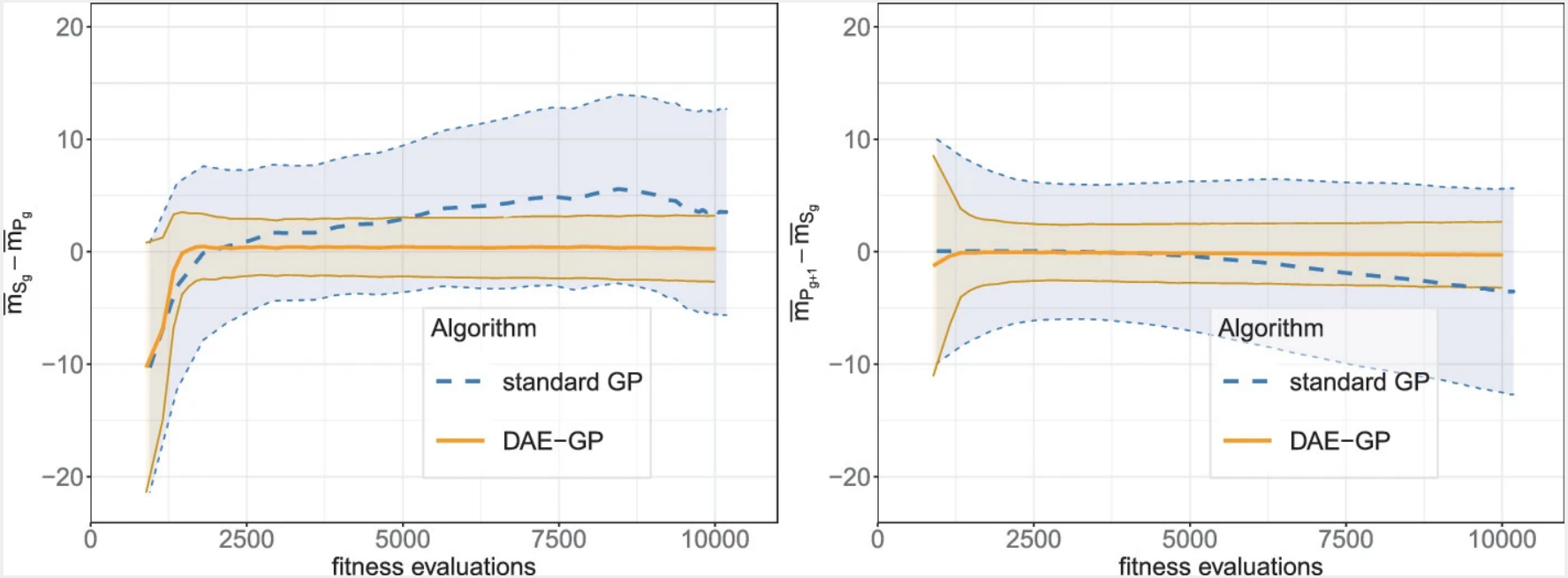


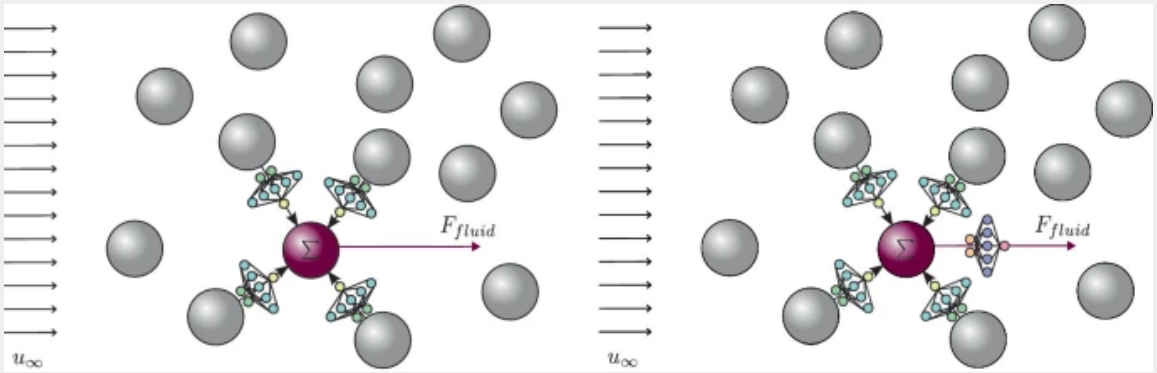
Figure: Bias of selection step (left) and variation step (right) on solution size measured by average differences in solution size *m* between populations *P* over fitness evaluations across all datasets.

Knowledge Guided GP

1. Julia Reuter et al. (2023). “Graph Networks as Inductive Bias for Genetic Programming: Symbolic Models for Particle-Laden Flows”. In: *EuroGP 2023*, pp. 36–51
2. Michał Kowalczykiewicz and Piotr Lipiński (2023). “Grammatical Evolution with Code2vec”. In: *EuroGP 2023*, pp. 213–224

Graph Networks as Inductive Bias for Genetic Programming: Symbolic Models for Particle-Laden Flows[1]

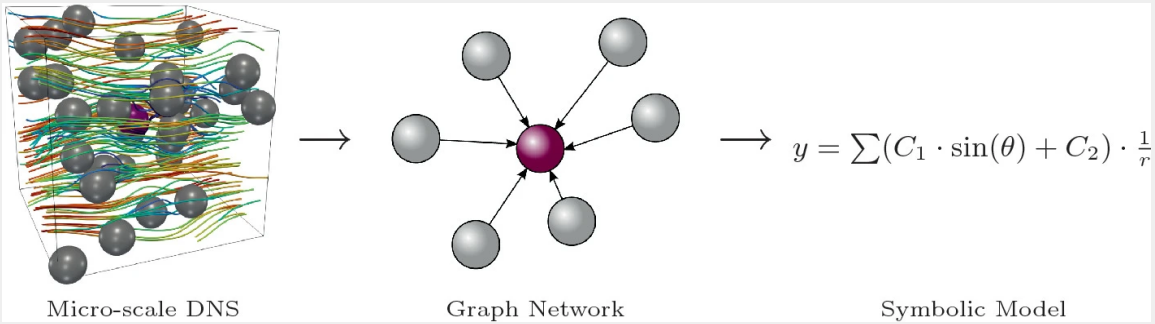
1. Choose one of the two GN structures: $y = g(x)$ (predicting the target variable by summing the edge messages) or $y = f(g(x))$ (predicting the target variable by applying a function on the summed edge messages).
2. Train a Graph Network (GN) on the input data (positions, velocities, etc. of the particles).



[1] Julia Reuter et al. (2023). "Graph Networks as Inductive Bias for Genetic Programming: Symbolic Models for Particle-Laden Flows". In: *EuroGP 2023*, pp. 36–51.

Graph Networks as Inductive Bias for Genetic Programming: Symbolic Models for Particle-Laden Flows[1]

1. Use Genetic Programming (GP) to fit symbolic models (Φ'_e and Φ'_n) to the outputs of the edge and node models of the GN.
2. Refit the constants in the symbolic models using a regression algorithm (e.g., Levenberg-Marquardt).
3. Predict the fluid force F_{fluid} using the fitted symbolic models.



[1] Julia Reuter et al. (2023). "Graph Networks as Inductive Bias for Genetic Programming: Symbolic Models for Particle-Laden Flows". In: *EuroGP 2023*, pp. 36–51.

GNN Guided GP

ϕ	Experiment	Equation	GP MSE	GN MSE
0.064	$y = g(x),\ c = 2$	$\sum\left(0.01146r + 0.01146\sin(\theta) - 0.0142\right)\frac{1}{r}$	0.000209	0.000120
	$y = g(x),\ c = 1$	$\sum\left((0.03448r + 0.03448\sin(\theta) - 0.04238)\left(-\log(r)\right)\right)$	0.000188	0.000120
	$y = f(g(x))$	$0.0992\sum\left((r\left(\sin(\theta) - 0.1312\right) - 0.1983)\left(-\log(r)\right)\right) + \bar{u}_x^f - 0.3177$	0.000157	0.000106
0.125	$y = g(x),\ c = 2$	$\sum\left(0.01397\sin(r) + 0.01397\sin(\theta) - 0.01724\right)\frac{1}{r}$	0.000284	0.000173
	$y = g(x),\ c = 1$	$\sum\left(0.00839 + (0.01578\sin(\theta) - 0.01644)\frac{1}{r}\right)$	0.000260	0.000173
	$y = f(g(x))$	$0.0597\sum\left(\left(\sin(\theta) - 0.45368 - \frac{0.12479}{r}\right)e^{-r}\right) - 0.0616$	0.000209	0.000146
0.216	$y = g(x),\ c = 2$	$\sum\bar{u}_x^f\left(\sin(\theta) - 0.57328 - \frac{0.10557}{r}\right)$	0.000316	0.000206
	$y = g(x),\ c = 1$	$\sum\left(0.00944 + (0.01932\sin(\theta) - 0.01982)\frac{1}{r}\right)$	0.000247	0.000206
	$y = f(g(x))$	$0.1166\sum\left(\left(0.17448\sin(\theta) - 0.08318 - \frac{0.01419}{r}\right)\frac{1}{r}\right) - 0.1602$	0.000248	0.000167
0.343	$y = g(x),\ c = 2$	$\sum\left((0.08249\sin(\theta) - 0.07348)\left(-\log(r)\right) + 0.00539\right)$	0.000239	0.000191
	$y = g(x),\ c = 1$	$\sum\left((0.08749\sin(\theta) - 0.07348)\left(-\log(r)\right) + 0.00423\right)$	0.000239	0.000191
	$y = f(g(x))$	$0.3904\sum\left(\left(0.10982e^{\sin(\theta)} - 0.26635\right)\left(-\log(r)\right) + 0.0165\right) - 0.0421$	0.000219	0.000197

Figure: Symbolic models with constants refitted to the original dataset.

Genetic Programming for ML

1. Nicholas Matsumoto et al. (2023). “Faster Convergence with Lexicase Selection in Tree-Based Automated Machine Learning”. In: *EuroGP 2023*, pp. 165–181
2. Zhilei Zhou et al. (2023). “A Boosting Approach to Constructing an Ensemble Stack”. In: *EuroGP 2023*, pp. 133–148
3. Hengzhe Zhang et al. (2023). “MAP-Elites with Cosine-Similarity for Evolutionary Ensemble Learning”. In: *EuroGP 2023*, pp. 84–100

Faster Convergence with Lexicase Selection in Tree-Based Automated Machine Learning[1]

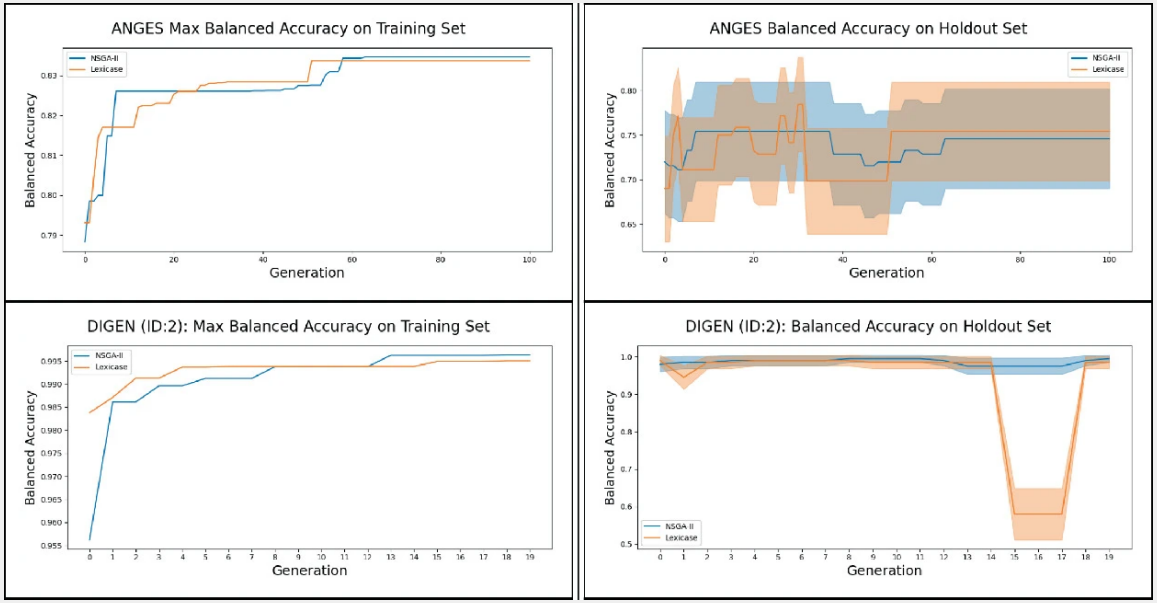


Figure: Left: Best model based on training accuracy. Right: Holdout scores for the best models based on training accuracy.

[1] Nicholas Matsumoto et al. (2023). "Faster Convergence with Lexicase Selection in Tree-Based Automated Machine Learning". In: *EuroGP 2023*, pp. 165–181.

GP in Domain Applications

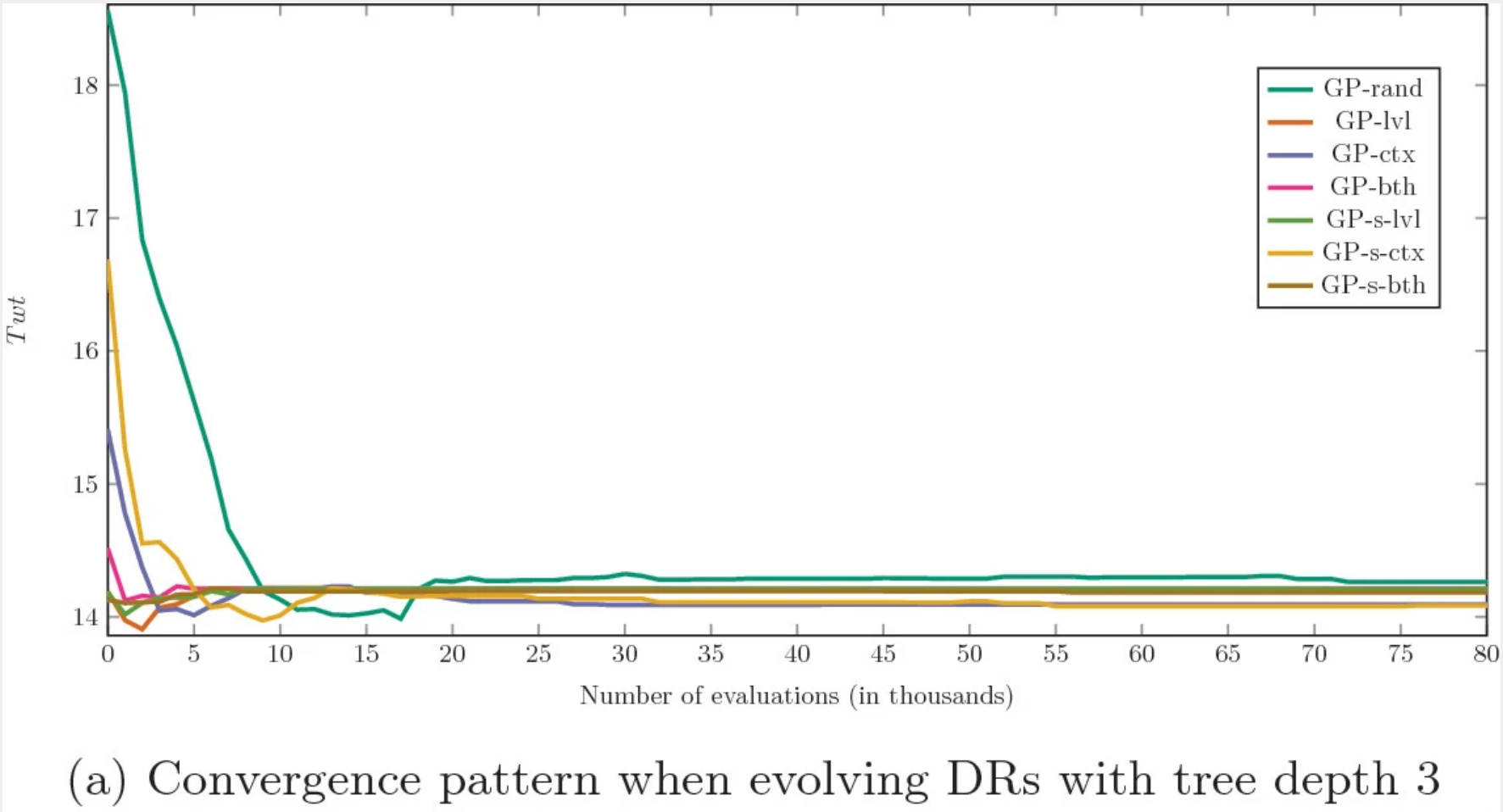
1. Iliya Miralavy and Wolfgang Banzhaf (2023). “Spatial Genetic Programming”. In: *EuroGP 2023*, pp. 260–275
2. Marko Đurasević, Francisco Javier Gil-Gala, and Domagoj Jakobović (2023). “To Bias or Not to Bias: Probabilistic Initialisation for Evolving Dispatching Rules”. In: *EuroGP 2023*, pp. 308–323
3. Matteo De Carlo et al. (2023). “Interacting Robots in an Artificial Evolutionary Ecosystem”. In: *EuroGP 2023*, pp. 339–354

To Bias or Not to Bias: Probabilistic Initialisation for Evolving Dispatching Rules[1]

1. The algorithm introduces a bias in selecting certain primitives at certain positions based on their frequency in the best individuals from previous experiments.
2. Level-based probability is calculated by considering the frequency of each primitive at each level of the tree.
3. Conditional probability takes into account the context, which consists of the parent and sibling nodes.

[1] Marko Đurasević, Francisco Javier Gil-Gala, and Domagoj Jakobović (2023). "To Bias or Not to Bias: Probabilistic Initialisation for Evolving Dispatching Rules". In: *EuroGP 2023*, pp. 308–323.

GP IN DFJSS



(a) Convergence pattern when evolving DRs with tree depth 3

Figure: Convergence patterns when evolving DRs with various initialisation methods

Conclusion

The hot trends at EuroGP 2023 mainly involve research in:

- Developing Adaptive Methods for GP
- Applying Neural Networks in GP
- Applying GP in Machine Learning
- Exploring Domain Applications using GP

In summary, these trends focus to promote the fusion and development of genetic programming, evolutionary computation and machine learning to solve problems in practical applications.

Thanks for listening!