A Semantic-based Hoist Mutation Operator for Evolutionary Feature Construction in Regression

Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, Mengjie Zhang

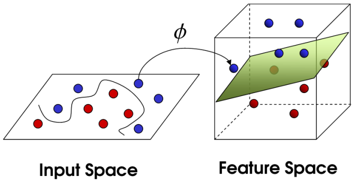
Victoria University of Wellington

15/07/2024

Table of Contents

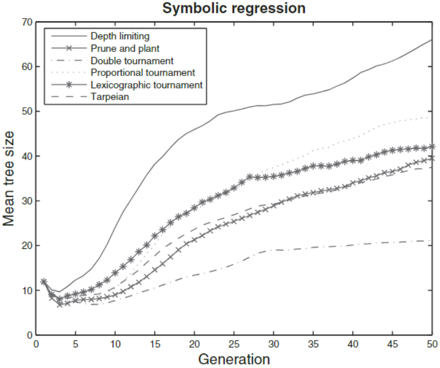
- **Feature construction** is a critical task in the machine learning pipeline.
- Constructs m high-order features $\Phi = \{\phi_1(X), \ldots, \phi_m(X)\}$ to **improve prediction accuracy**.
- **Genetic programming (GP)** is popular for feature construction due to its flexible representation and gradient-free search mechanism.

(a) Feature Construction on Linear Regression



(b) New Feature Space

- **Bloat** refers to the increase in the size of GP trees without a corresponding improvement in fitness.
- Several hypotheses explain bloat: **hitchhiking, defense against crossover, removal bias**, and the **nature of the program search space**.
- Bloat can trap GP in local optima and **reduce model interpretability**.



Growth of program size

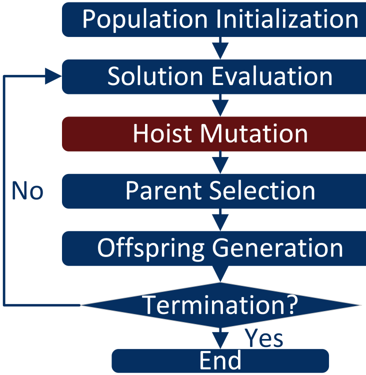
Bloat Control Methods

- Various methods have been proposed to control bloat in GP:
  - **Parsimony pressure**
  - **Dynamic depth limit**
  - **Prune and plant**
  - **Multi-objective methods**
  - **Program simplification**
- Prune and plant (PAP) is effective but may **disrupt informative components**.

- **Semantics** in GP refers to the outputs of GP individuals.
- **Semantic GP** uses semantics to guide evolution, **improving search performance and population diversity**.

New Algorithm

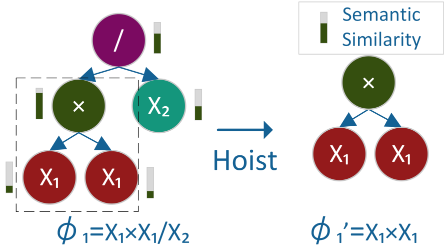
- **SHM operator** reduces the size of GP trees in feature construction.
- Five steps: **population initialization**, **solution evaluation**, **hoist mutation**, **parent selection**, and **offspring generation**.



The workflow of SHM-GP

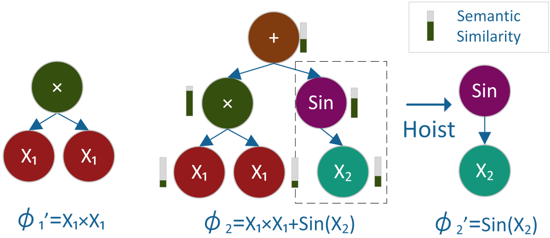
Steps:

- Measure **cosine semantic similarity** between the semantics of each subtree $\psi(X)$ and target Y.
- Hoist the subtree with the **highest semantic similarity** to form a new GP tree.
- For multi-tree GP, apply this operator to all GP trees $\psi \in \Phi$.



An example of the semantic-based hoist mutation operator

- **Hash-based checking strategy** prevents generating repetitive features $\psi_1 = \psi_2$.



An example of hash-based checking.

- Theorem: The **generalization loss** of the constructed model can be bounded by:

$$L_{\text{exp}}(\phi) \leq (1 - \theta^2) \left(1 - \sqrt{p - p \ln p + \frac{\ln n}{2n}} \right)_+^{-1} \tag{1}$$

- Shows that the hoisted subtree generalizes **at least as well as the original tree**.

Experimental Settings

Datasets

- Experiments conducted on **98 regression datasets** from Penn Machine Learning Benchmark (PMLB).
- Datasets are **standardized before training**.

Evaluation Protocol

- Each dataset is divided into **training and test sets** (80:20 ratio).
- R^2 **score** metric is used to evaluate test performance.
- **30 independent runs** with different random seeds for reliable conclusions.

- Seven GP approaches with bloat control methods are compared:
  - **Standard GP with Depth Limit**
  - **Double Tournament Selection (DTS)**
  - **Tarpeian**
  - **Prune and Plant (PAP)**
  - α**-MOGP**
  - **TS-S**
  - **Dynamic Subtree Approximation (DSA)**

Results

- **SHM operator** maintains competitive performance to standard GP on most datasets, and even better on some.

Statistical comparison of **test R^2 score** for different bloat control methods. ("+","$\sim$", and "-" indicate using the method in a row is better than, similar to or worse than using the method in a column.)

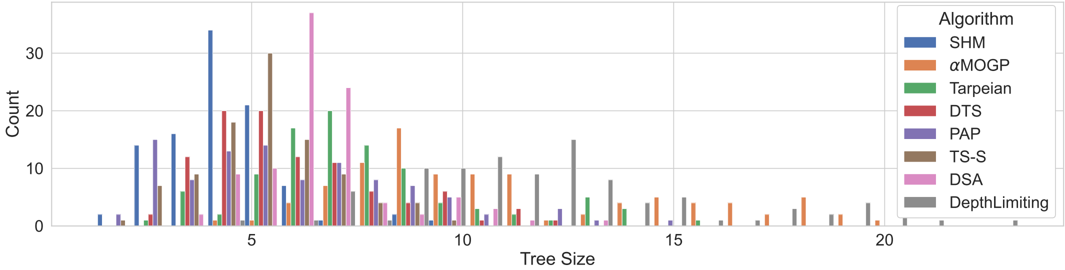
	αMOGP	Tarpeian	DTS	PAP	TS-S	DSA	DepthLimiting
SHM	29(+)/68($\sim$)/1(-)	13(+)/81($\sim$)/4(-)	35(+)/57($\sim$)/6(-)	46(+)/49($\sim$)/3(-)	27(+)/65($\sim$)/6(-)	19(+)/74($\sim$)/5(-)	13(+)/80($\sim$)/5(-)
αMOGP	—	2(+)/78($\sim$)/18(-)	11(+)/79($\sim$)/8(-)	40(+)/48($\sim$)/10(-)	5(+)/76($\sim$)/17(-)	1(+)/86($\sim$)/11(-)	3(+)/75($\sim$)/20(-)
Tarpeian	—	—	22(+)/73($\sim$)/3(-)	44(+)/52($\sim$)/2(-)	11(+)/80($\sim$)/7(-)	7(+)/89($\sim$)/2(-)	4(+)/91($\sim$)/3(-)
DTS	—	—	—	32(+)/60($\sim$)/6(-)	4(+)/87($\sim$)/7(-)	1(+)/73($\sim$)/24(-)	6(+)/64($\sim$)/28(-)
PAP	—	—	—	—	2(+)/57($\sim$)/39(-)	1(+)/58($\sim$)/39(-)	4(+)/51($\sim$)/43(-)
TS-S	—	—	—	—	—	6(+)/85($\sim$)/7(-)	12(+)/62($\sim$)/24(-)
DSA	—	—	—	—	—	—	8(+)/81($\sim$)/9(-)

- **SHM operator** significantly reduces tree size on all datasets.

Statistical comparison of **tree size** for different bloat control methods. ("+","∼", and "-" indicate using the method in a row is better than, similar to or worse than using the method in a column.)

	αMOGP	Tarpeian	DTS	PAP	TS-S	DSA	DepthLimiting
SHM	98(+)/0(∼)/0(-)	98(+)/0(∼)/0(-)	61(+)/37(∼)/0(-)	55(+)/40(∼)/3(-)	58(+)/35(∼)/5(-)	94(+)/4(∼)/0(-)	98(+)/0(∼)/0(-)
αMOGP	—	0(+)/5(∼)/93(-)	0(+)/0(∼)/98(-)	0(+)/2(∼)/96(-)	0(+)/1(∼)/97(-)	0(+)/0(∼)/98(-)	39(+)/51(∼)/8(-)
Tarpeian	—	—	0(+)/20(∼)/78(-)	0(+)/38(∼)/60(-)	0(+)/34(∼)/64(-)	5(+)/45(∼)/48(-)	98(+)/0(∼)/0(-)
DTS	—	—	—	32(+)/32(∼)/34(-)	29(+)/37(∼)/32(-)	47(+)/47(∼)/4(-)	98(+)/0(∼)/0(-)
PAP	—	—	—	—	25(+)/44(∼)/29(-)	49(+)/27(∼)/22(-)	98(+)/0(∼)/0(-)
TS-S	—	—	—	—	—	60(+)/30(∼)/8(-)	98(+)/0(∼)/0(-)
DSA	—	—	—	—	—	—	98(+)/0(∼)/0(-)

- **SHM operator** significantly reduces tree size on all datasets.

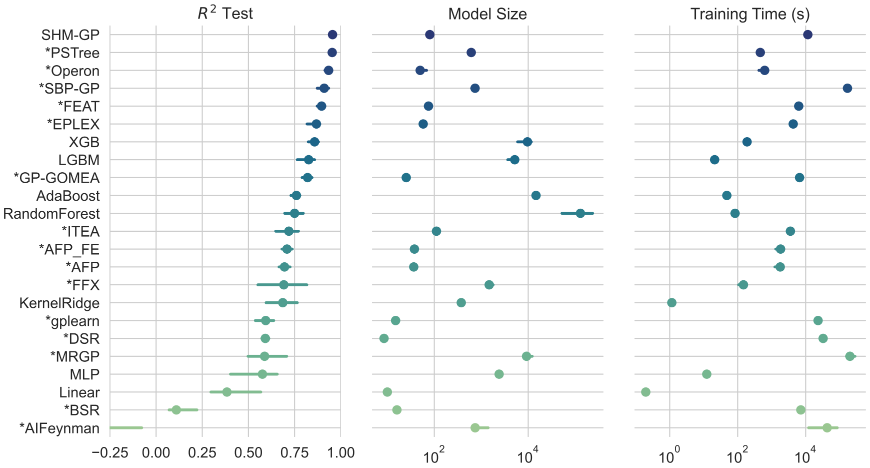


The distribution of tree size for different bloat control methods.

Friedman's rank of R^2 test scores and average tree sizes.

Algorithm	R^2 Rank	P-Value	Size Rank	P-Value
SHM	3.05 (1)	-	1.48 (1)	-
Tarpeian	3.88 (2)	1.9e-02	5.42 (6)	0.0e+00
DepthLimiting	3.96 (3)	1.9e-02	7.83 (8)	0.0e+00
DSA	4.07 (4)	1.1e-02	4.48 (5)	0.0e+00
TS-S	4.55 (5)	7.3e-05	3.12 (2)	4.0e-06
αMOGP	5.01 (6)	1.1e-07	7.17 (7)	0.0e+00
DTS	5.11 (7)	2.3e-08	3.14 (3)	4.0e-06
PAP	6.37 (8)	0.0e+00	3.35 (4)	2.6e-07

Overall Comparisons

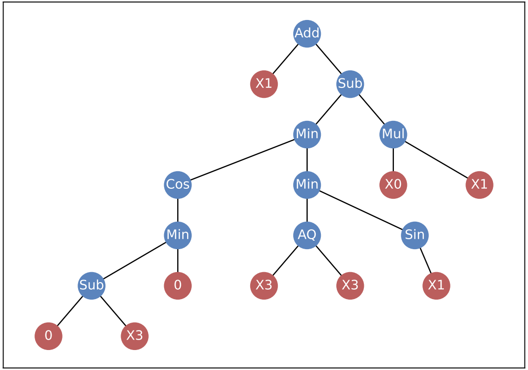


R^2 scores, model sizes and training time of 23 algorithms on 120 regression problems.

Example Models

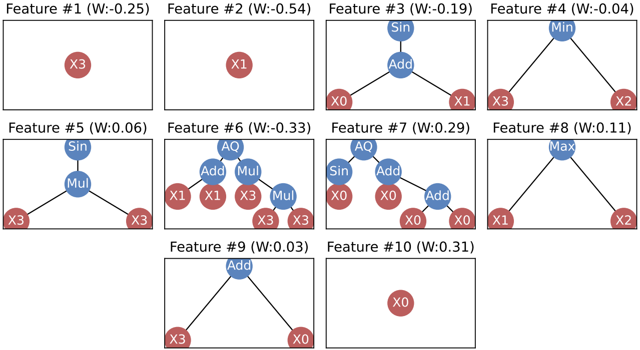
- Features constructed using Standard GP are **complex**.

Feature #1 (W:-0.08)



An example of a constructed feature based on standrad GP with depth limiting

- Features constructed using SHM-GP are **simple and interpretable**.



An example of constructed features based on the semantic-based hoist mutation operator

Conclusions

Conclusions

- **SHM operator** effectively controls bloat and improves predictive performance.
- **Extensive experiments** validate the superiority of SHM over other bloat control methods.

Thanks for listening!

Email: Hengzhe.zhang@ecs.vuw.ac.nz

GitHub Project: https://github.com/hengzhe-zhang/EvolutionaryForest