Schema-based Diversification in Genetic Programming

Bogdan Burlacu[2,3] Michael Affenzeller[1,2]

[1]Institute for Formal Models and Verification, Johannes Kepler Universität Linz

[2]Heuristic and Evolutionary Algorithms Laboratory, University of Applied Sciences Upper Austria

[3]Josef Ressel Center for Symbolic Regression, University of Applied Sciences Upper Austria

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

HEAL

HEURISTIC AND EVOLUTIONARY ALGORITHMS LABORATORY

SymReg

JOSEF RESSEL CENTER FOR SYMBOLIC REGRESSION

Introduction

Genetic programming

- *Diversity* instrumental in algorithm performance
- Balance between the *exploration* and *exploitation* aspects of the search

How to maintain balance? [Goldberg and Deb, 1991]

- Use slow growth ratios to prevent premature convergence
- Use higher growth ratio followed by mutation
- Permit localized differential mutation rates
- Preserve useful diversity via niching, dominance, diploidy

Diversity in Genetic Programming

Selection as the main mechanism to control diversity

- soft brood selection [Altenberg, 1994]
- fitness groups [Rosca, 1995]
- "keep best" selection [Wiese and Goodwin, 1998]
- distance-based [Ekárt and Németh, 2000, O'Reilly, 1997]
- correlative tournament selection [Matsui, 1999]
- offspring selection [Affenzeller and Wagner, 2003]
- lineage selection [Burke et al., 2003]
- self-adaptive selection [Affenzeller and Wagner, 2004]
- "fitness segments" [Yan and Clack, 2006]
- "age layers" [Hornby, 2006]
- "genetic markers" [Burks and Punch, 2015, Burks and Punch, 2016]

Schema-based Diversification

Proposed Approach

- Identify genotypically and phenotypically similar individuals
  - Genetic programming *schemas* as genotypic templates
  - Correlation as phenotypic similarity measure
- Apply localized mutation within groups of similar individuals

Main Steps

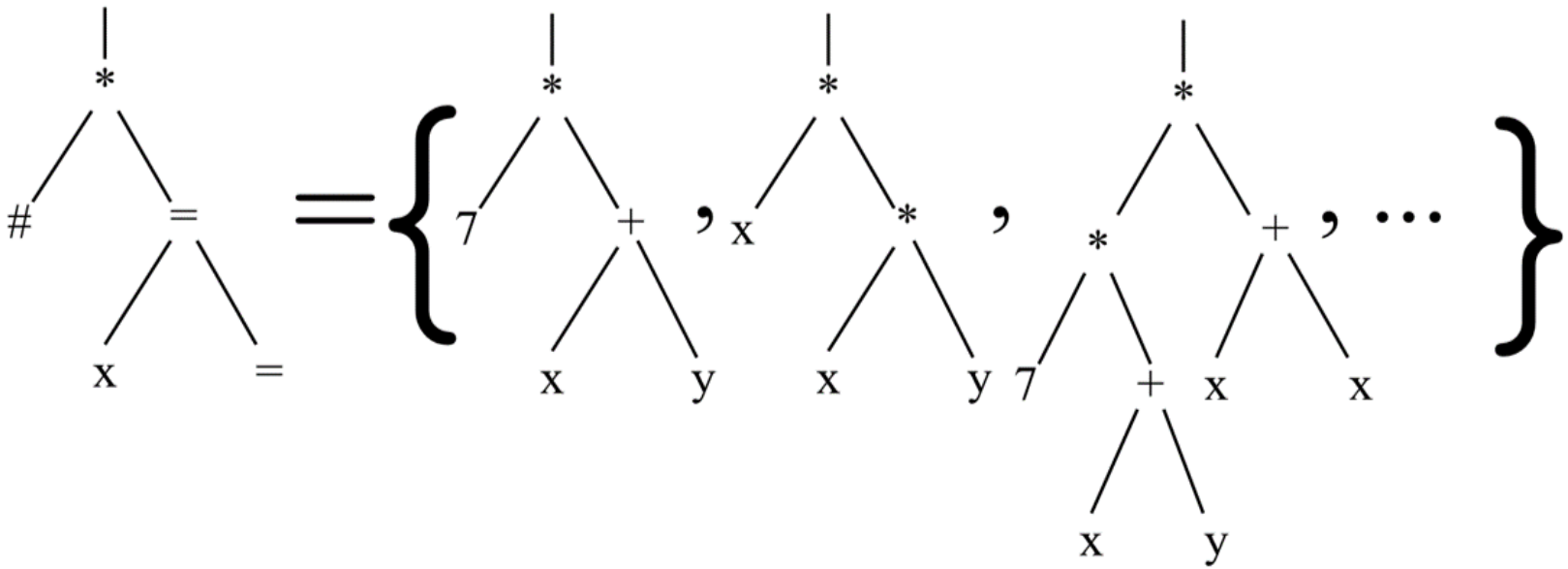
1. Generate relevant schemas and calculate their frequencies
2. Calculate phenotypic similarity within schema groups
3. Apply mutation (larger group $\rightarrow$ more mutation)
4. Re-evaluate mutated individuals

Schema-based Diversification

Schema Definition

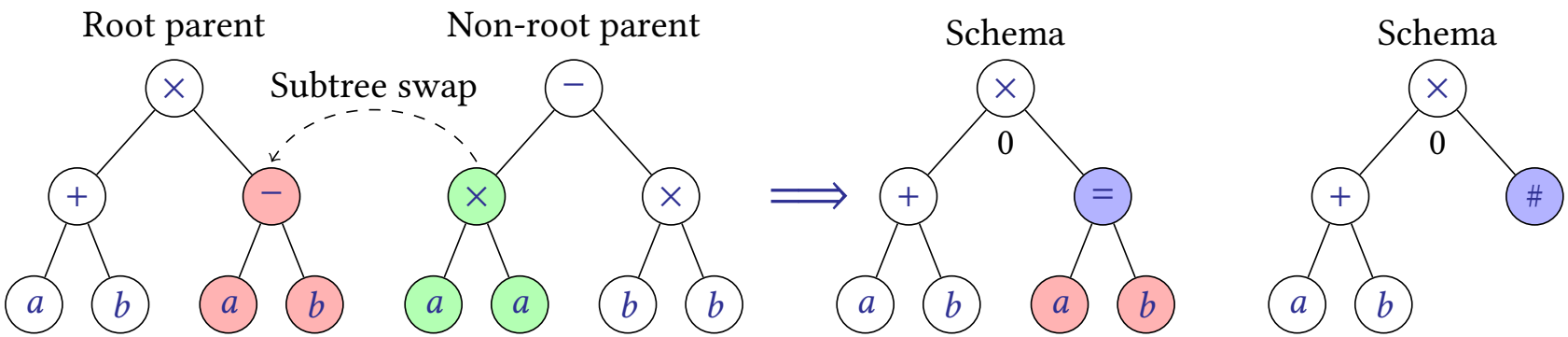
We adopt the definition from [Poli and McPhee, 2003], where schemas are rooted trees with internal nodes from $\mathcal{F} \cup \{=\}$ and leaf nodes from $\mathcal{T} \cup \{=, \#\}$.

- The $=$ symbol matches any symbol of the same arity
- The $\#$ symbol matches any valid subtree



Schema-based Diversification

Schema Generation



- Under selection pressure, fit individuals selected multiple times
- Multiple cutpoint locations in root parent's structure
- Replace symbols at cutpoint locations with = or #
- Several possible replacement strategies

Schema-based Diversification

Schema Matching

- Bottom-up query matching algorithm by [Götz et al., 2009]
- Matches a data tree D (GP individual) against a query tree Q (GP schema)
- Additional restrictions: two nodes are matched if
  - They are on the same tree level
  - Their parent and children nodes also match
  - Commutative symbols matched regardless of child order

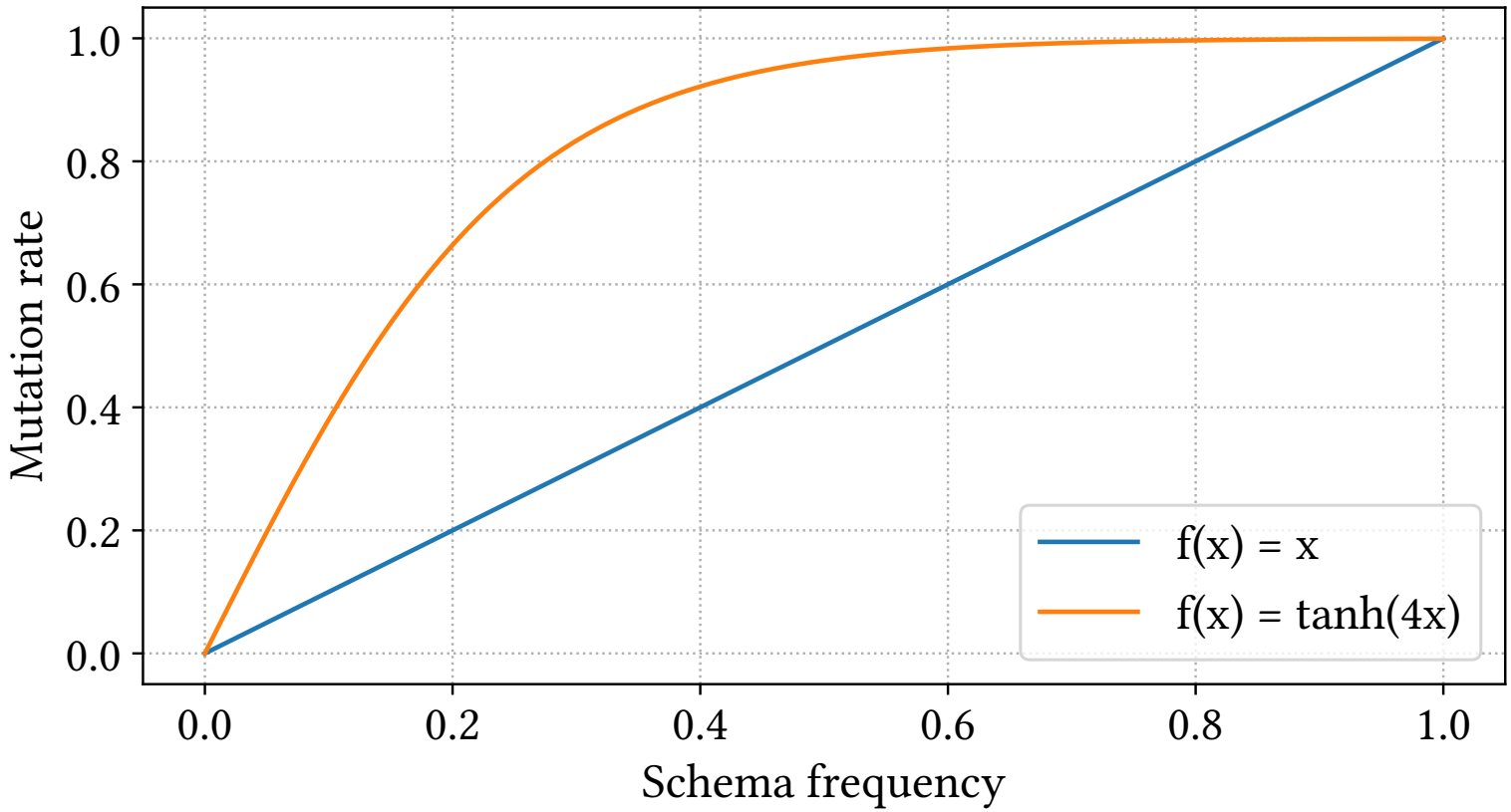
Phenotypic Similarity

- Defined as the R^2 between individual responses on the training data
- Two constant responses considered completely similar
- A constant and non-constant response considered completely dissimilar

Schema-based Diversification

Mutation Strategy

- *Goal*: improve diversity without affecting convergence.
- Restrictions: minimum threshold for phenotypic similarity or schema frequency
- Adaptive mutation rates within schema-groups



Schema-based Diversification

Algorithm

Input:

- minimum schema frequency $f_{s_{min}}$
- minimum phenotypic similarity $p_{s_{min}}$

```
1 S ← generate schemas;
2 for every schema s in S do
3     M ← GetMatchingIndividuals(s);
4     f_s ← |M| / |POP_i|;                    // calculate schema frequency
5     p_s ← CalculatePhenotypicSimilarity(M);
6     if f_s > f_smin and p_s > p_smin then
7         apply mutation within M;
8         recalculate fitness for mutated individuals;
```

Practical Advantage

- Can be applied to any existing genetic algorithm flavors
- Will not harm performance (when reasonably configured)

Empirical Analysis

Test Problems

Name	Function	Training	Test
Friedman-2	$f(x_1, \ldots, x_{10}) = 10 \sin(\pi x_1 x_2) + 20(x_3 - 1/2)^2 + 10x_4 + 5x_5 + \mathit{Noise}$	500 rows	5000 rows
Poly-10	$f(x_1, \ldots, x_{10}) = x_1 x_2 + x_3 x_4 + x_5 x_6 + x_1 x_7 x_9 + x_3 x_6 x_{10}$	250 rows	250 rows
Pagie-1	$f(x, y) = \frac{1}{1+x^{-4}} + \frac{1}{1+y^{-4}}$	676 rows	1000 rows
Vladislavleva-4	$f(x_1, \ldots, x_5) = \frac{10}{5+\sum_{i=1}^{5}(x_i-3)^2}$	1024 rows	5000 rows

Test Algorithm

- Offspring Selection Genetic Algorithm (OSGA) [Affenzeller and Wagner, 2003]
- Offspring must outperform their parents (strict OS)
- Selection pressure as adaptive stopping criteria
- 50 repetitions for each configuration

Empirical Analysis

Algorithm Settings

Parameter	Value
Tree initialization	PTC2 [Luke, 2000]
Maximum tree length	50 nodes
Maximum tree depth	12
Population size	500 individuals
Elites	1 individual
Selection	Gender specific selection [Wagner and Affenzeller, 2005] (random & proportional)
Crossover probability	100%
Crossover operator	Subtree crossover
Mutation probability	25%
Mutation operator	Change symbol Single point mutation Remove branch Replace branch
Fitness function	Maximize R^2
Termination criterion	Selection pressure ≥ 100
Terminal symbols	*constant*, *weight* $*$ *variable*
Function symbols	binary functions $(+, -, \times, /)$

Diversification Strategy

Parameter	Value
Min schema frequency	1%
Min semantic similarity	99%
Min schema length	5 nodes
Adaptive mutation rate	True
Mutation rate function	$f(x) = x$ $f(x) = \tanh(4x)$
Schema wildcard set	$W = \{=\}$ $W = \{\#\}$ $W = \{=, \#\}$
Schema generation	Use the best 20% individuals
Schema matching	Ignore numerical constants and variable weights
Termination criterion	Selection pressure ≥ 100 Evaluated solutions $\geq 2 \cdot 10^6$ R^2 fitness value ≥ 0.99

Empirical Analysis

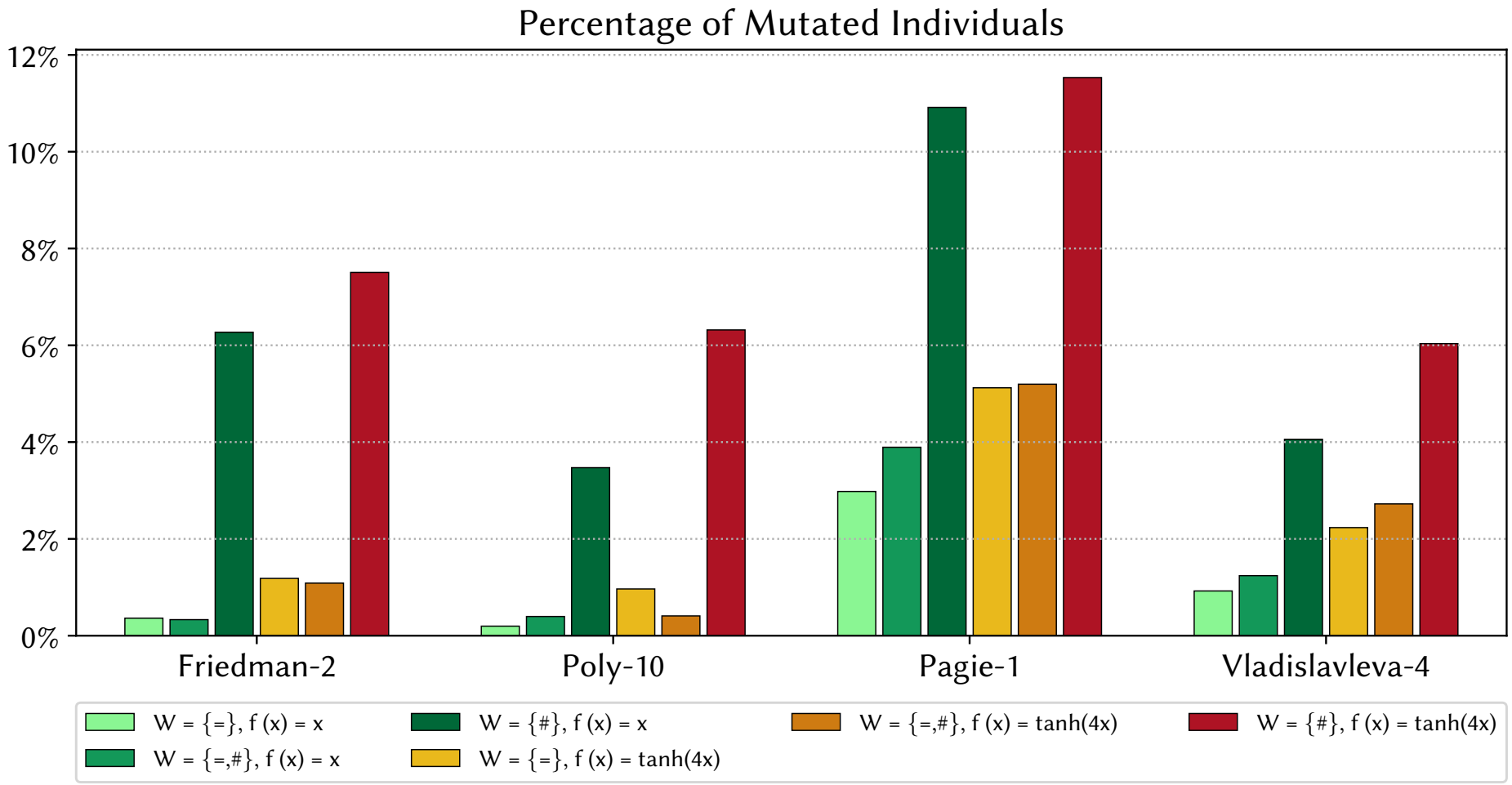
Results (median ± stdev)

Algorithm configuration	Friedman-1	Poly-10	Pagie-1	Vladisl.-4
OSGP	0.837 ± 0.027 0.824 ± 0.172	0.836 ± 0.116 0.780 ± 0.200	0.952 ± 0.027 0.919 ± 0.309	0.831 ± 0.098 0.795 ± 0.280
OSGP-S ($W = \{=\}, f(x) = x$)	0.849 ± 0.029 0.833 ± 0.129	**0.878 ± 0.063** **0.843 ± 0.190**	0.951 ± 0.026 0.919 ± 0.257	0.830 ± 0.083 0.795 ± 0.276
OSGP-S ($W = \{\#\}, f(x) = x$)	**0.862 ± 0.034** **0.859 ± 0.120**	**0.889 ± 0.097** **0.839 ± 0.126**	0.960 ± 0.023 0.911 ± 0.298	0.835 ± 0.066 0.796 ± 0.271
OSGP-S ($W = \{=, \#\}, f(x) = x$)	0.834 ± 0.034 0.815 ± 0.114	0.865 ± 0.131 0.784 ± 0.228	0.971 ± 0.026 0.907 ± 0.311	**0.870 ± 0.093** **0.799 ± 0.222**
OSGP-S ($W = \{=\}, f(x) = \tanh(4x)$)	0.848 ± 0.032 0.847 ± 0.060	**0.894 ± 0.122** **0.869 ± 0.192**	0.969 ± 0.022 0.913 ± 0.333	**0.915 ± 0.104** **0.865 ± 0.234**
OSGP-S ($W = \{\#\}, f(x) = \tanh(4x)$)	**0.862 ± 0.036** **0.841 ± 0.177**	**0.905 ± 0.100** **0.876 ± 0.190**	0.970 ± 0.022 0.919 ± 0.291	0.840 ± 0.090 0.801 ± 0.269
OSGP-S ($W = \{=, \#\}, f(x) = \tanh(4x)$)	0.847 ± 0.030 0.832 ± 0.145	0.843 ± 0.115 0.794 ± 0.177	**0.974 ± 0.024** **0.953 ± 0.298**	**0.877 ± 0.089** **0.798 ± 0.254**

- Limited benefits for problems where finding good constants takes precedence over structure discovery (eg., Poly-10 vs. Pagie-1)
- No best configuration, depends on fitness landscape. Results favor more aggresive mutation rates.
- Overall good results with # wildcards, but higher runtime costs. Reasonable compromise with = wildcards.

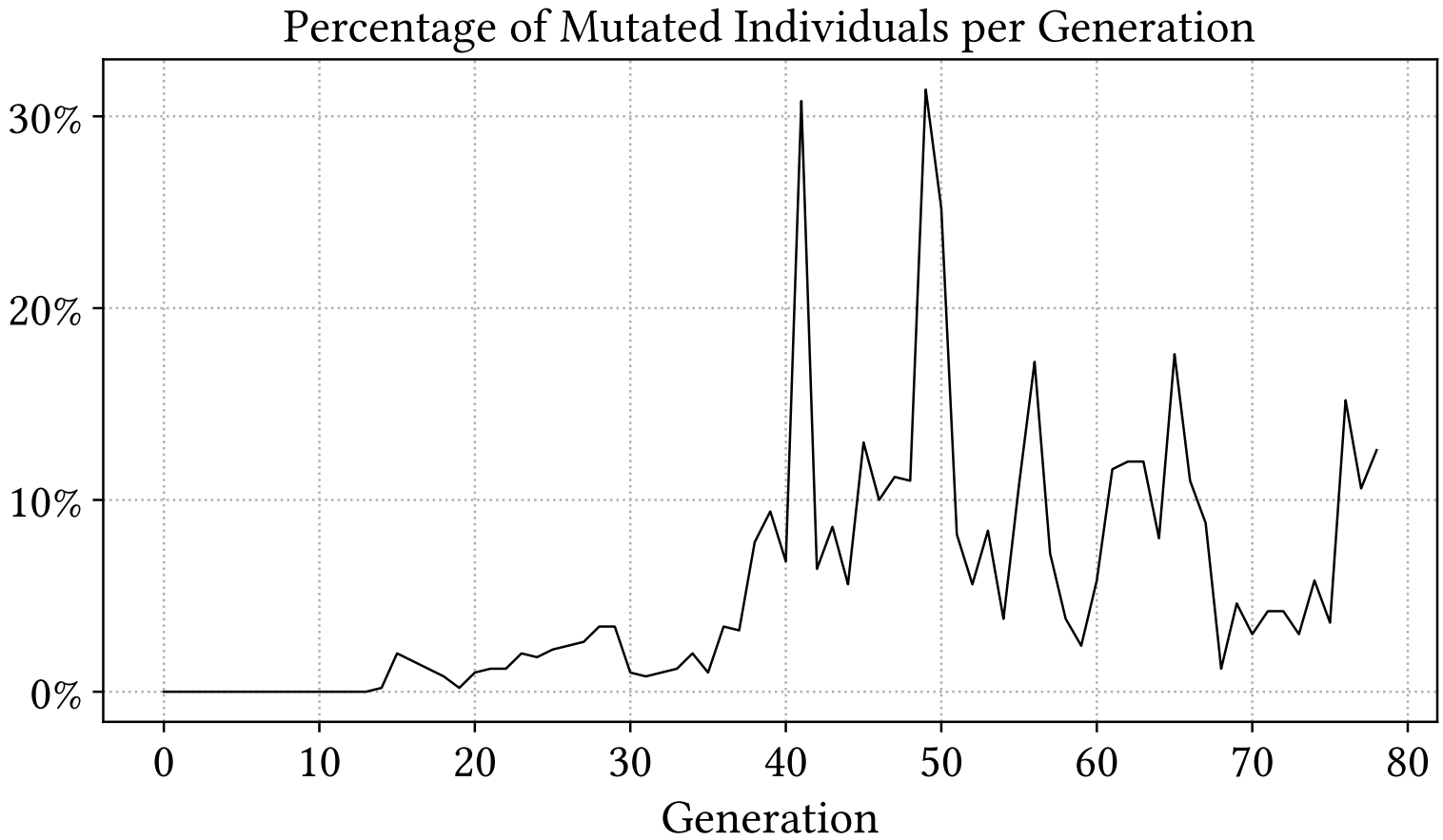
Empirical Analysis

Mutation Overhead



Empirical Analysis

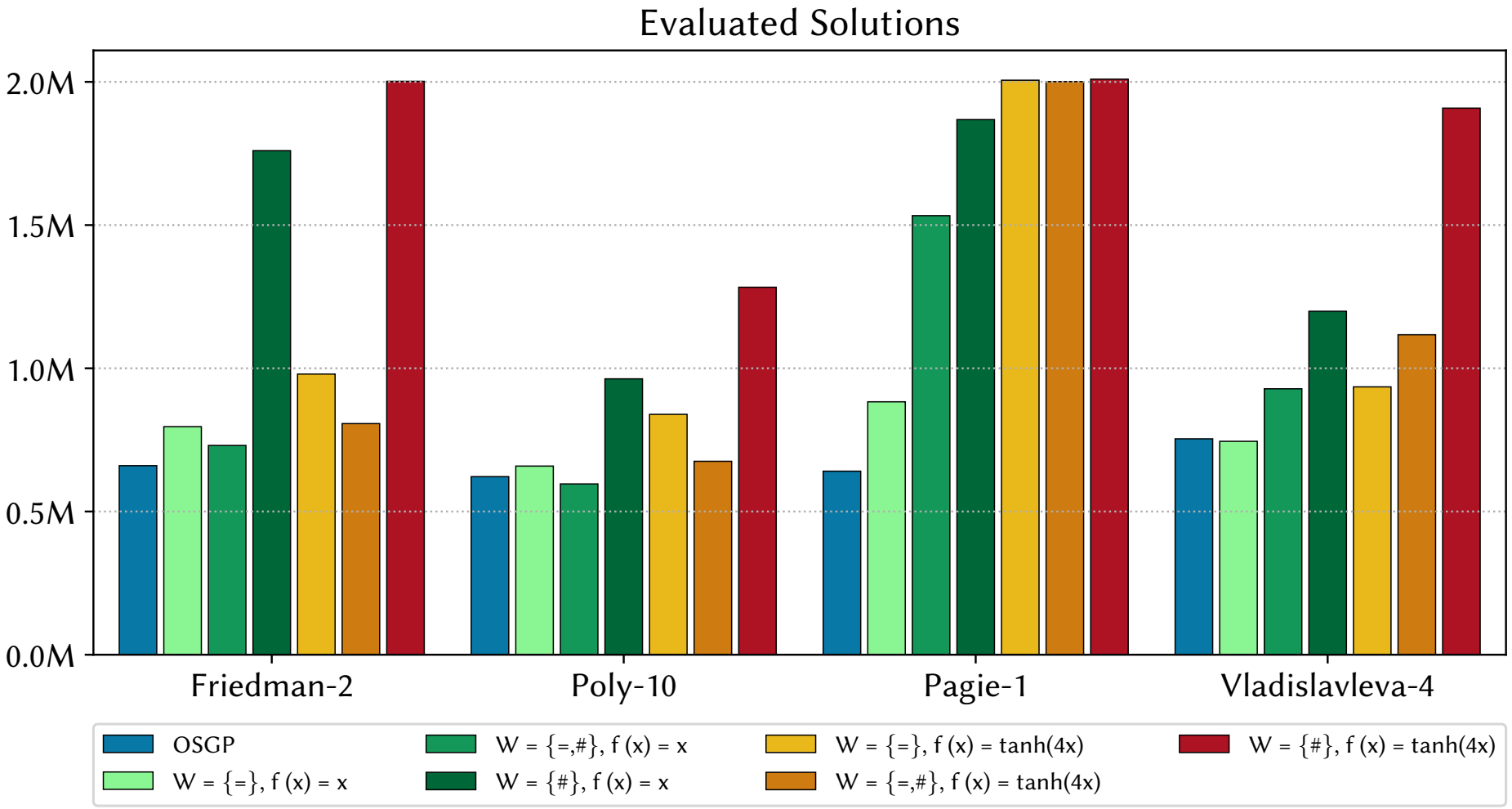
Mutation Overhead



$W = \{=\},\ f(x) = \tanh(4x)$

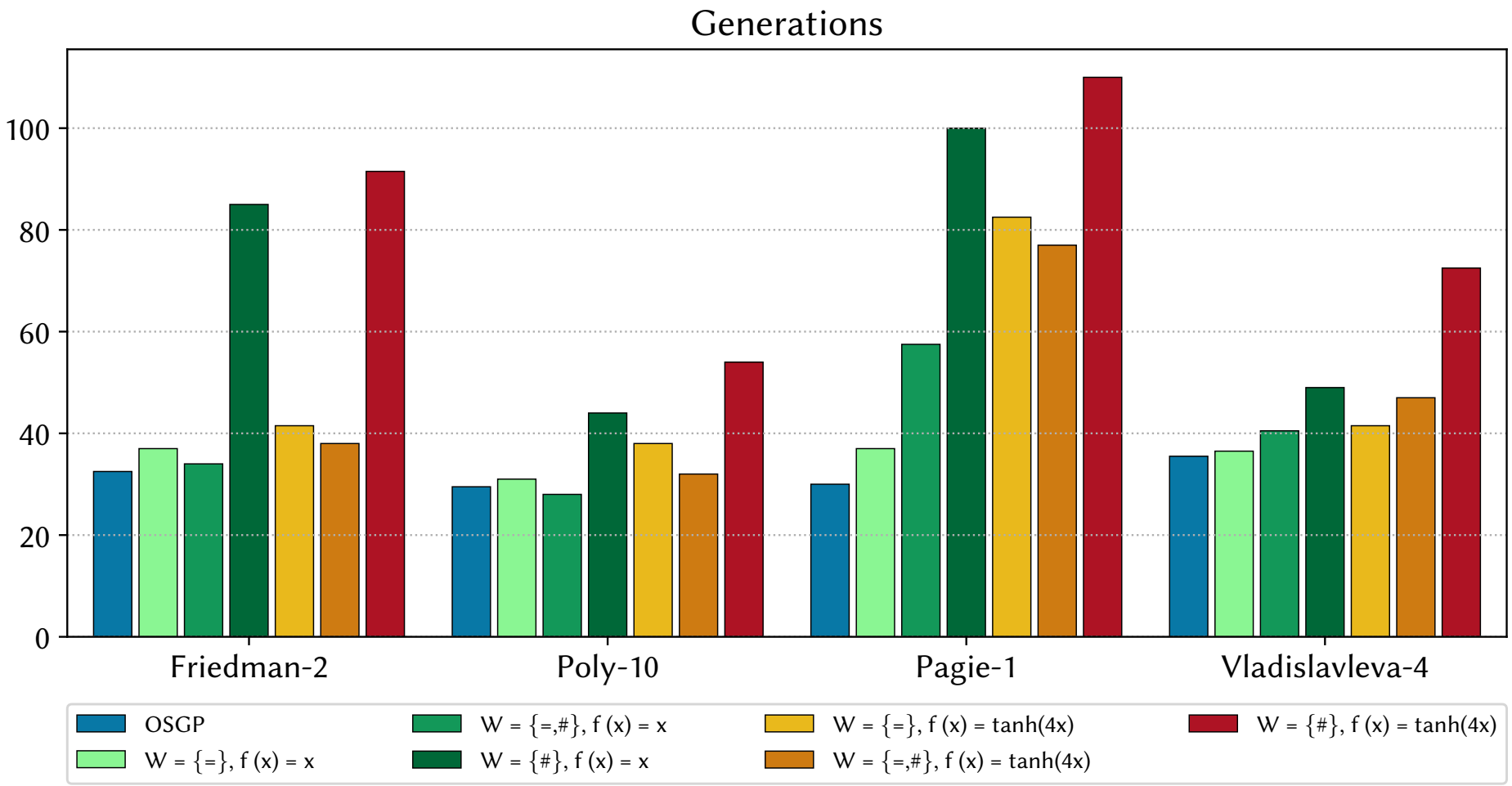
Empirical Analysis

Evaluation Overhead



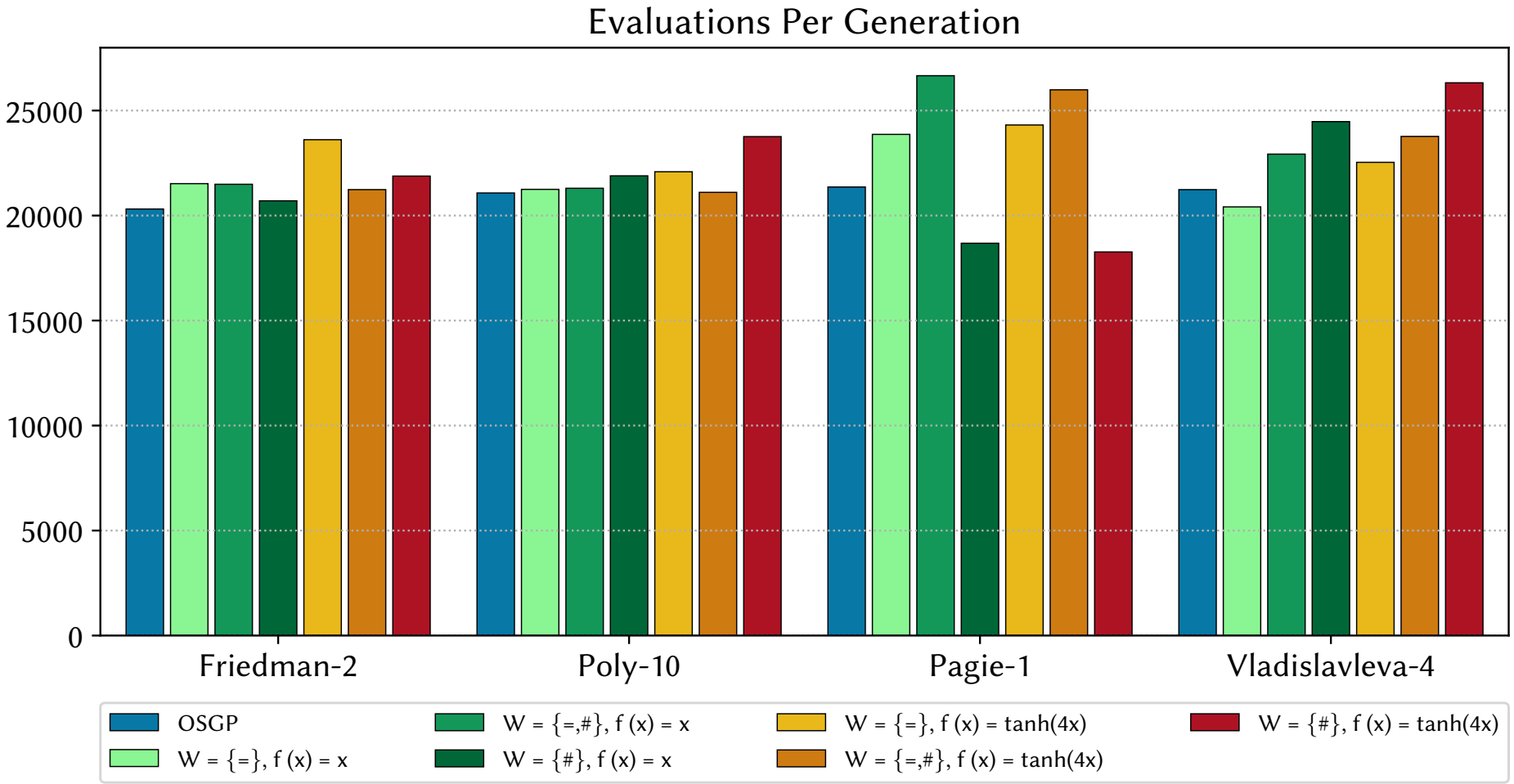
Empirical Analysis

Number of Generations



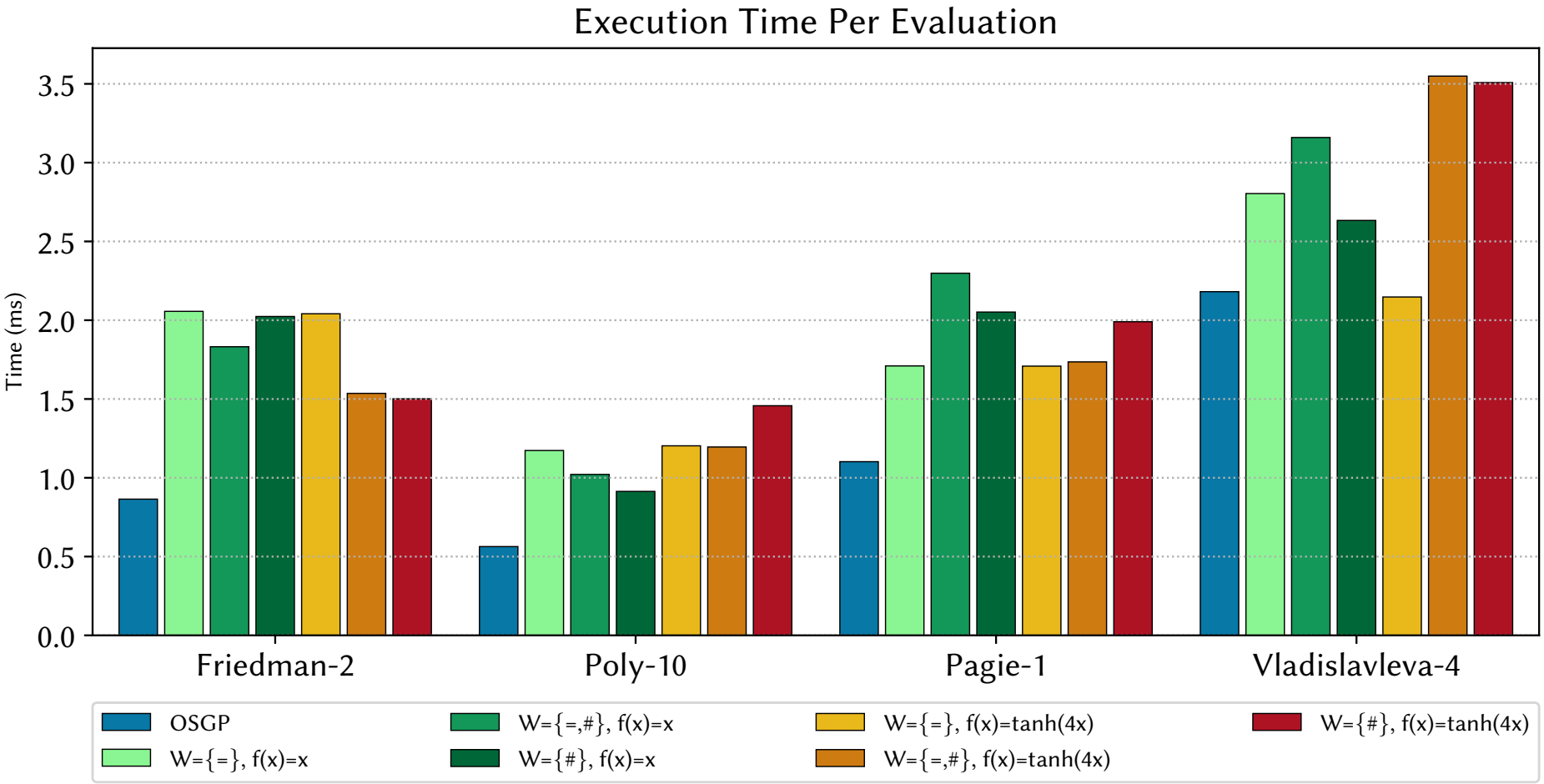
Empirical Analysis

Evaluation Overhead per Generation



Empirical Analysis

Runtime Overhead



Discussion

Benchmark Results

- OSGP-S overall better than OSGP
- Different degrees of effectiveness for schema diversification
- Extra mutation decreases active selection pressure, increases # of generations
- Small effect of $=$-schemas on mutation rates and # of evaluations
- $\{\#\}$-schema frequency indicates common top-level tree structures

Computational Overhead

- Low overhead in terms of evaluated solutions per generation
- Increased CPU-time due to schema matching (becomes small relative to increasing # of training samples)
- Implementation not currently optimized for speed

Conclusion

Schema-based Diversification

- Introduces pattern matching to GP
- Able to identify common structural templates
- Does not alter the behaviour of standard GP operators
- Multi-level strategy combining existing concepts:
  - *Structural*: schemas as structural templates
  - *Semantic*: phenotypic similarity within schema groups
  - *Hereditary*: schema generation method based on hereditary relationships
  - *Adaptive*: differential mutation rates based on schema frequency

Future Work

- Improved schema generation
- Persist schemas over generations
- Hybridize with other algorithms
- Additional benchmarks

Affenzeller, M. and Wagner, S. (2003).

A Self-adaptive Model for Selective Pressure Handling within the Theory of Genetic Algorithms, pages 384–393.

Springer Berlin Heidelberg, Berlin, Heidelberg.

Affenzeller, M. and Wagner, S. (2004).

Sasegasa: A new generic parallel evolutionary algorithm for achieving highest quality results.

Journal of Heuristics, 10(3):243–267.

Altenberg, L. (1994).

The evolution of evolvability in genetic programming.

In Kinnear, Jr., K. E., editor, *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press.

Burke, E., Gustafson, S., Kendall, G., and Krasnogor, N. (2003).

Is increased diversity in genetic programming beneficial? an analysis of lineage selection.

In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 2, pages 1398–1405 Vol.2.

Burks, A. and Punch, W. (2016).

An analysis of the genetic marker diversity algorithm for genetic programming.

Genetic Programming and Evolvable Machines, pages 1–33.

Burks, A. R. and Punch, W. F. (2015).

An efficient structural diversity technique for genetic programming.

In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 991–998, New York, NY, USA. ACM.


Ekárt, A. and Németh, S. Z. (2000).
A metric for genetic programs and fitness sharing.
In *Proceedings of the European Conference on Genetic Programming*, pages 259–270, London, UK, UK. Springer-Verlag.


Goldberg, D. E. and Deb, K. (1991).
A comparative analysis of selection schemes used in genetic algorithms.
In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann.


Götz, M., Koch, C., and Martens, W. (2009).
Efficient algorithms for descendant-only tree pattern queries.
Inf. Syst., 34(7):602–623.


Hornby, G. S. (2006).
Alps: the age-layered population structure for reducing the problem of premature convergence.
In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 815–822, Seattle, Washington, USA. ACM Press.


Luke, S. (2000).
Two fast tree-creation algorithms for genetic programming.
IEEE Transactions on Evolutionary Computation, 4(3):274–283.


Matsui, K. (1999).
New selection method to improve the population diversity in genetic algorithms.
In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 1, pages 625–630. IEEE.

O'Reilly, U.-M. (1997).
Using a distance metric on genetic programs to understand genetic operators.

Poli, R. and McPhee, N. F. (2003).
General schema theory for genetic programming with subtree-swapping crossover: Part I.
Evolutionary Computation, 11(1):53–66.

Rosca, J. (1995).
Entropy-driven adaptive representation.
In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 23–32. Morgan Kaufmann.

Wagner, S. and Affenzeller, M. (2005).
Sexualga: Gender-specific selection for genetic algorithms.
In *Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*, number 9, pages 76–81, Orlando, United States of America.

Wiese, K. and Goodwin, S. D. (1998).
Keep-best reproduction: A selection strategy for genetic algorithms.
In *Proceedings of the 1998 ACM Symposium on Applied Computing*, SAC '98, pages 343–348, New York, NY, USA. ACM.

Yan, W. and Clack, C. D. (2006).
Behavioural GP diversity for dynamic environments: an application in hedge fund investment.
In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 2, pages 1817–1824, Seattle, Washington, USA. ACM Press.