Semantic Backpropagation in Genetic Programming

Tomasz Pawlak Bartosz Wieloch Krzysztof Krawiec

Institute of Computing Science
Poznan University of Technology

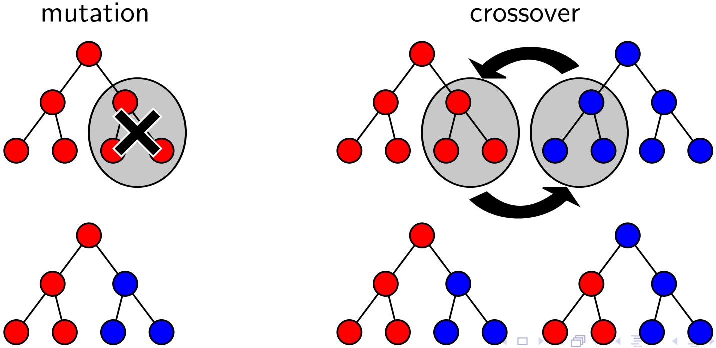
19.03.2013

Outline

Outline

What Is Genetic Programming?

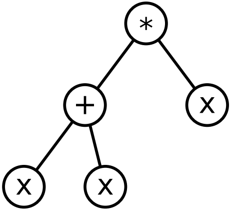
- Goal: produce a computer program that carries out the desired computation
- Means: evolving a population of candidate solutions, with fitness function measuring how solution's computation diverges from the desired one
- Standard search operators:



Semantics of Program

Semantics

- In general: Description of what a program does, i.e. what are the *effects* of execution of an entire program or its constituent components.
- In GP: a *list of outputs* that are actually produced by a program for all training examples (fitness cases).



x	result
-0.5	**0.5**
1.0	**2.0**
1.5	**4.5**
2.0	**8.0**

semantics=**[0.5, 2.0, 4.5, 8.0]**

Use of Semantics in Genetic Programming

Recent GP works on semantics:

- L. Beadle, C. Johnson, *Semantically Driven Crossover in Genetic Programming*, IEEE Press, 2008, pp 111-116,
- N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, E. Galvan-Lopez, *Semantically-based crossover in genetic programming: application to real-valued symbolic regression*, Genetic Programming and Evolvable Machines, 2011, pp 91-119.
- A. Moraglio, K. Krawiec, C. Johnson, *Geometric Semantic Genetic Programming*, Springer, 2012, pp 21-31.
- K. Krawiec, T. Pawlak, *Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators*, Genetic Programming and Evolvable Machines, 2013, pp 31-63.

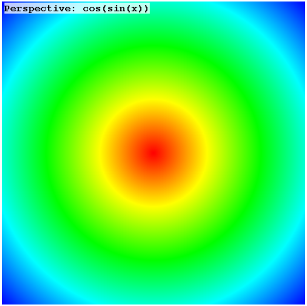
Fitness Landscape

Example:

- Symbolic regression problem,
- Only two fitness cases,
- Target semantics $= [0, 0]$,
- Error function is Euclidean distance,

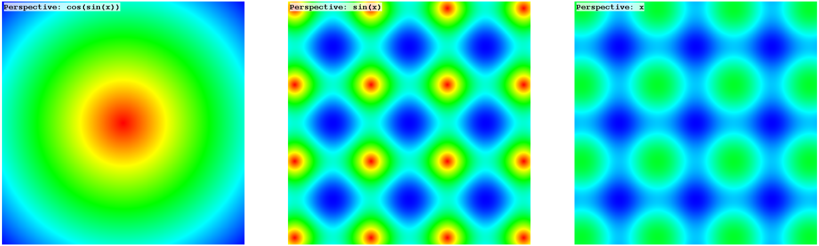
$$\Downarrow$$

Fitness landscape is a cone with vertex in the target semantics.



Fitness Landscape Seen From Different Perspectives

- Program: $\cos(\sin(x))$
  - Decomposable into tree instructions:
  - $\cos(\#), \sin(\#), x$



target: 0 | target: $\frac{\pi}{2} + k\pi, k \in \mathbb{Z}$ | no target!

Outline

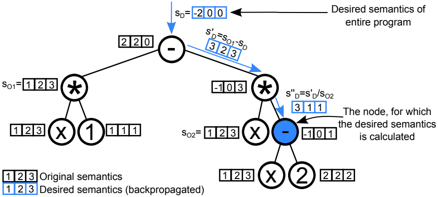
Assumptions

The objective: Propagate the semantic target backwards through the program tree, so that it defines a subgoal for a subproblem.

- Input:
  - The program p (tree-based representation),
  - The target semantics s_D ,
  - The chosen node p' of the program p.
- Output:
  - Desired semantics $s_D(p')$ for p'.

The algorithm

- Determine a path from the program root to p'.
- Starting from the root node, for each instruction I on the path, do recursively:
  - Determine inverse instruction I^{-1} to p w.r.t. child node p_c, which is next on the path,
  - Execute p^{-1} to compute desired semantics $s_D(p_c)$ for the child node p_c,
  - Stop when recursion reaches the chosen node ($p_c \equiv p'$)



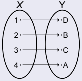
Common problems

Important observation

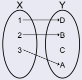
Most instructions **are not invertible**!

The reason

- In order to instruction be invertible for any output, it must implement bijection.



- In order to invert particular execution of instruction, it must implement injection.



Possible cases

1. Instruction is invertible:
   - $I: y \leftarrow x + c \implies I^{-1}: x \leftarrow c - y.$
2. Instruction is ambiguously invertible:
   - $I: z \leftarrow x^2 \implies I^{-1}: x \in \{-\sqrt{z}, \sqrt{z}\},$
   - $I: z \leftarrow \sin(x) \implies I^{-1}: x \leftarrow \arcsin(z) + 2k\pi, k \in \mathbb{Z}.$
3. Instruction is non-invertible:
   - $I: z \leftarrow e^x \implies I^{-1}: \forall_{z \in \mathbb{R}^-} x \leftarrow X$ (NaN, inconsistent).
4. Argument of instruction is ineffective:
   - $I: z \leftarrow 0 \times x \implies I^{-1}: x \leftarrow ?$ (don't care).

Possible cases

1. Instruction is invertible:
   - $I: y \leftarrow x + c \implies I^{-1}: x \leftarrow c - y$.
2. Instruction is ambiguously invertible:
   - $I: z \leftarrow x^2 \implies I^{-1}: x \in \{-\sqrt{z}, \sqrt{z}\}$,
   - $I: z \leftarrow \sin(x) \implies I^{-1}: x \leftarrow \arcsin(z) + 2k\pi, k \in \mathbb{Z}$.
3. Instruction is non-invertible:
   - $I: z \leftarrow e^x \implies I^{-1}: \forall_{z \in \mathbb{R}^-} x \leftarrow X$ (NaN, inconsistent).
4. Argument of instruction is ineffective:
   - $I: z \leftarrow 0 \times x \implies I^{-1}: x \leftarrow ?$ (don't care).

Possible cases

1. Instruction is invertible:
   - $I : y \leftarrow x + c \implies I^{-1} : x \leftarrow c - y.$
2. Instruction is ambiguously invertible:
   - $I : z \leftarrow x^2 \implies I^{-1} : x \in \{-\sqrt{z}, \sqrt{z}\},$
   - $I : z \leftarrow \sin(x) \implies I^{-1} : x \leftarrow \arcsin(z) + 2k\pi, k \in \mathbb{Z}.$
3. Instruction is non-invertible:
   - $I : z \leftarrow e^x \implies I^{-1} : \forall_{z \in \mathbb{R}^-} x \leftarrow X$ (NaN, inconsistent).
4. Argument of instruction is ineffective:
   - $I : z \leftarrow 0 \times x \implies I^{-1} : x \leftarrow ?$ (don't care).

Possible cases

1. Instruction is invertible:
   - $I: y \leftarrow x + c \implies I^{-1}: x \leftarrow c - y$.
2. Instruction is ambiguously invertible:
   - $I: z \leftarrow x^2 \implies I^{-1}: x \in \{-\sqrt{z}, \sqrt{z}\}$,
   - $I: z \leftarrow \sin(x) \implies I^{-1}: x \leftarrow \arcsin(z) + 2k\pi, k \in \mathbb{Z}$.
3. Instruction is non-invertible:
   - $I: z \leftarrow e^x \implies I^{-1}: \forall_{z \in \mathbb{R}^-} x \leftarrow X$ (NaN, inconsistent).
4. Argument of instruction is ineffective:
   - $I: z \leftarrow 0 \times x \implies I^{-1}: x \leftarrow ?$ (don't care).

Solution

When inversion of instruction is:

- Ambiguous: Store only one value (of many possible),
- Impossible (non-invertible): mark element as inconsistent
- Ineffective: mark element as 'don't care'

Outline

RDO Mutation

Given one parent program p:

- Choose randomly a mutation node p',
- Backpropagate target semantics t to the mutation node p' to obtain desired semantics $s_D(p')$ of p',
- Find a procedure p_L that commits the smallest error w.r.t. $s_D(p')$,
- Replace p' with p_L.

Approximately Geometric Semantic Crossover (AGX)

Given two parent programs p_1, p_2:

- Compute corresponding semantics $s(p_1), s(p_2)$ of p_1, p_2,
- Compute midpoint s_m between semantics $s(p_1), s(p_2)$,
  - e.g. $s_m = (s(p_1) + s(p_2))/2$ for numerical semantics,
- For each parent $p \in \{p_1, p_2\}$:
  - Choose with uniform distribution w.r.t. tree depth a crossover node p',
  - Backpropagate semantics s_m to the crossover node p' to obtain desired semantics $s_D(p')$ of p',
  - Search a procedure p_L committing the smallest error w.r.t. $s_D(p')$,
  - Replace p' with p_L.

Applicability of Operators

Representation of semantics	Knowledge on target: Semantics	Knowledge on target: Fitness value	Knowledge on target: No knowledge
Object in normed vector space	RDO, AGX	AGX	AGX
Object in vector space	RDO, AGX[a]	AGX[a]	AGX[a]
Object in metric space	RDO	—	—
Object from a set without space	RDO	—	—
No semantics, syntax only	—	—	—

[a] Although in general vector space we cannot check if a point lies between two other points, we still can combine two points. Consequently AGX can operate in this space, however with no guarantee that the calculated desired semantics of the offspring is geometrically between semantics of its parents.

Outline

A Static Library

All possible programs built upon given set of instructions, filtered for semantic uniqueness.

Example

- Instructions: $\{+, -, \times, /, \sin, \cos, \exp, \log, x\}$,
- Max tree depth: 4,
- Total no. of programs: 269217, unique: 108520.

A Population-based Library

- Genetic Programming is population-based algorithm!
- Use all subprograms of all programs in population as a library.
- Library evolves with solutions.

Comparison of Libraries

	Static library	Population-based library
Time of build	Once, before run	Every generation
No. of unique procedures	Constant	Variable
Semantic diversity	Guaranteed	May converge
Can produce new semantics	No	Yes

Semantic Diversity

All possible programs:

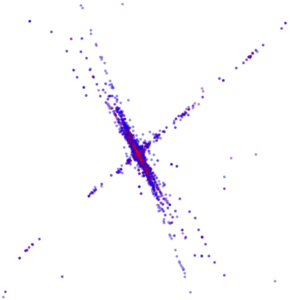
- Instructions: $\{+, -, \times, /, sin, e^x, x\}$,
- Max tree depth: 4.

Semantics:

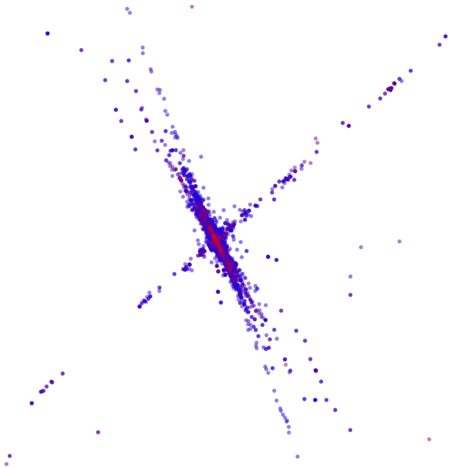
- 20 points distributed equidistantly in range $[-5, 5]$,
- Programs filtered according to semantic uniqueness.

Visualization:

- Reduction to $2D$ by PCA,
- Red: the smallest (i.e. single node) programs,
- Blue: the longest (i.e. 15 nodes) programs.



Semantic Diversity



Conclusion

The space is mostly empty.

Outline

RDO Setup

- Population-based library
- Operators:
  - M — canonical mutation,
  - X — canonical crossover,
  - RDO — RDO mutation,
- Operators applied:
  - individually, and
  - in every combination of two of them
    (probability varying from 0.1 to 0.9 with step 0.1)
- Benchmarks:
  - Ten symbolic regression problems,
  - Ten Boolean problems.

Benchmarks

	Target program (expression)	Vars	Range
F03	$x^5 + x^4 + x^3 + x^2 + x$	1	$[-1; 1]$
F04	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	1	$[-1; 1]$
F05	$\sin(x^2)\cos(x) - 1$	1	$[-1; 1]$
F06	$\sin(x) + \sin(x + x^2)$	1	$[-1; 1]$
F07	$\log(x + 1) + \log(x^2 + 1)$	1	$[0; 2]$
F08	$\sqrt{x}$	1	$[0; 4]$
F09	$\sin(x) + \sin(y^2)$	2	$[0.01; 0.99]$
F10	$2\sin(x)\cos(y)$	2	$[0.01; 0.99]$
F11	x^y	2	$[0.01; 0.99]$
F12	$x^4 - x^3 + y^2/2 - y$	2	$[0.01; 0.99]$

Benchmarks

Problem	Instance	Bits	Fitness cases
even parity	PAR4	4	16
	PAR5	5	32
	PAR6	6	64
multiplexer	MUX6	6	64
	MUX11	11	2048
majority	MAJ5	5	32
	MAJ6	6	64
	MAJ7	7	128
comparator	CMP6	6	64
	CMP8	8	256

Results — Friedman ranks

success ratio

Setup	Rank
M+RDO 0.7	**8.63**
M+RDO 0.3	8.78
X+RDO 0.5	**8.90**
M+RDO 0.5	9.15
X+RDO 0.4	9.20
X+RDO 0.8	9.23
M+RDO 0.4	9.25
X+RDO 0.6	9.75
M+RDO 0.6	9.88
X+RDO 0.3	9.95
X+RDO 0.7	9.95
M+RDO 0.8	10.08
M+RDO 0.2	10.65
X+RDO 0.9	11.15
M+RDO 0.9	11.20

Setup	Rank
X+RDO 0.2	11.70
RDO 1.0	**13.40**
X+RDO 0.1	14.28
M+RDO 0.1	14.58
X 1.0	**20.55**
X+M 0.1	**21.30**
X+M 0.2	22.53
X+M 0.3	23.10
X+M 0.4	23.55
X+M 0.5	23.85
X+M 0.6	24.53
X+M 0.7	25.73
X+M 0.8	25.85
M 1.0	**27.18**
X+M 0.9	27.18

median error

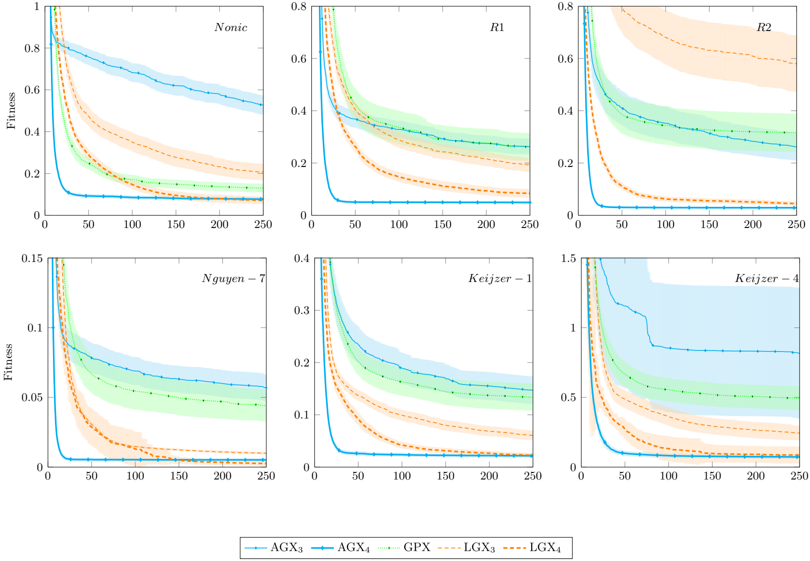
Setup	Rank
M+RDO 0.7	**8.83**
M+RDO 0.6	8.98
M+RDO 0.5	9.00
M+RDO 0.4	9.35
M+RDO 0.8	9.38
X+RDO 0.7	**9.75**
M+RDO 0.3	9.78
X+RDO 0.8	10.05
X+RDO 0.6	10.18
X+RDO 0.5	10.33
X+RDO 0.4	10.35
X+RDO 0.3	10.53
M+RDO 0.9	11.08
M+RDO 0.2	11.50
X+RDO 0.9	11.73

Setup	Rank
X+RDO 0.2	12.45
X+RDO 0.1	12.63
RDO 1.0	**13.18**
M+RDO 0.1	13.25
X 1.0	**20.70**
X+M 0.1	**20.70**
X+M 0.2	20.85
X+M 0.3	22.25
X+M 0.4	22.53
X+M 0.5	23.45
X+M 0.6	23.93
X+M 0.7	25.90
X+M 0.8	25.95
X+M 0.9	26.85
M 1.0	**29.63**

AGX Setup

- Two static libraries:
  - Instructions: $\{+, -, \times, /, \sin, \cos, \exp, \log, x\}$,
  - Max tree depth: $\{3, 4\}$,
  - Total no. of unique programs: $212, 108520$,
- Use of library denoted by index:
  - AGX_3, AGX_4
- Competition:
  - Standard subtree crossover (GPX),
  - Locally Geometric Semantic Crossover (LGX).
- Benchmark:
  - Six univariate symbolic regression problems.

AGX Performance



AGX Success Rate (%)

Problem	AGX_3	AGX_4	GPX	LGX_3	LGX_4
Nonic	0	0	**1**	0	0
R1	0	**1**	0	0	0
R2	0	**1**	0	0	**1**
Nguyen-7	0	**34**	6	0	0
Keijzer-1	0	0	0	0	0
Keijzer-4	0	**21**	0	0	0

Summary

- Semantic backpropagation allows us to transform original problem trough a program structure.
- Operators involving semantic backpropagation achieve significantly better results than traditional ones.

- Outlook
  - We want to combine our efforts to improve the method.
  - We work on modifications of semantic backpropagation and GP operators, that allow us to use more inversions of semantics keeping the computational costs at bay.

For Further Reading I

- K. Krawiec, T. Pawlak.
  Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators.
  Genetic Programming and Evolvable Machines Vol 14, pp 31-63, Springer, 2013.
- (Accepted) K. Krawiec, B. Wieloch.
  Running programs backwards.
  GECCO 2013 Proceedings, ACM, 2013.
- (Accepted) K. Krawiec, T. Pawlak.
  Approximating Geometric Crossover by Semantic Backpropagation.
  GECCO 2013 Proceedings, ACM, 2013.