Geometric Semantic Genetic Programming (GSGP): theory-laden design of variation operators

Andrea Mambrini

University of Birmingham, Birmingham UK

Research skills
15th January 2014

Genetic Programming

Genetic programming is an evolutionary algorithm-based methodology to evolve functions (or computer programs).

- Individuals are functions represented for example as parse trees
- Mutation operators traditionally perform operations on the syntax of the trees
- The fitness function measures the input-output behaviour (semantic) of the function, usually comparing it with a target behaviour that is aimed to be reached.

Research question

How can we get **guarantees** on the convergence time of genetic programming (GP)?

Black-box boolean function learning

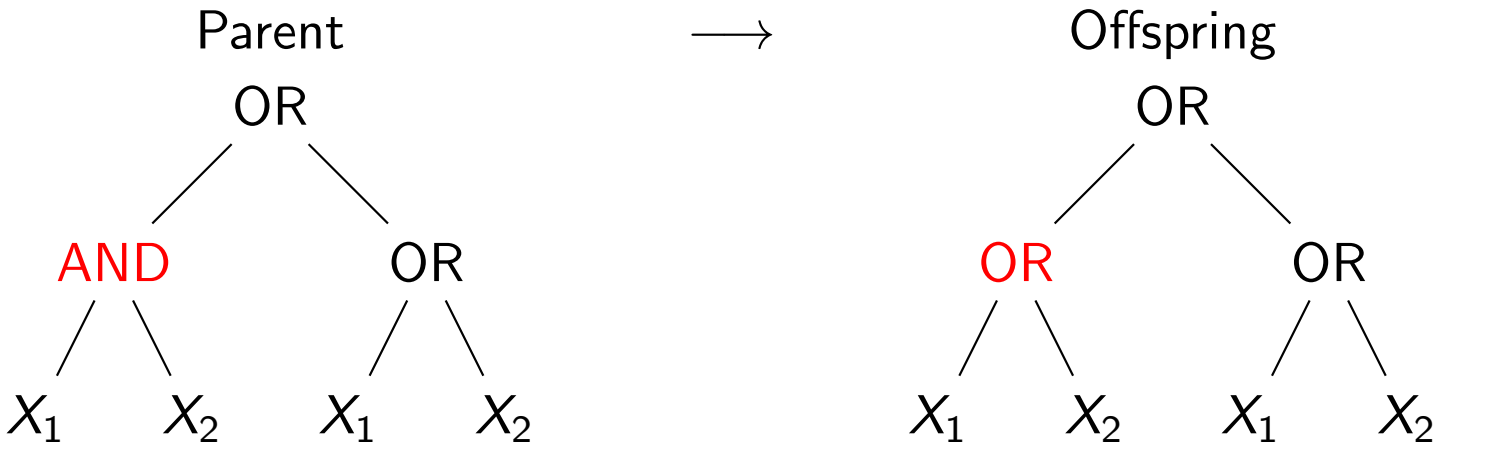
Which boolean expression produces the required output behaviour?

X_1	X_2	Output
1	1	1
1	0	0
0	1	1
0	0	0

Black-box setting:

- We don't have access to the truth table
- We can give our candidate function $f(X_1, X_2)$ to an oracle that will answer us the number of matched output rows.

Traditional Genetic Programming

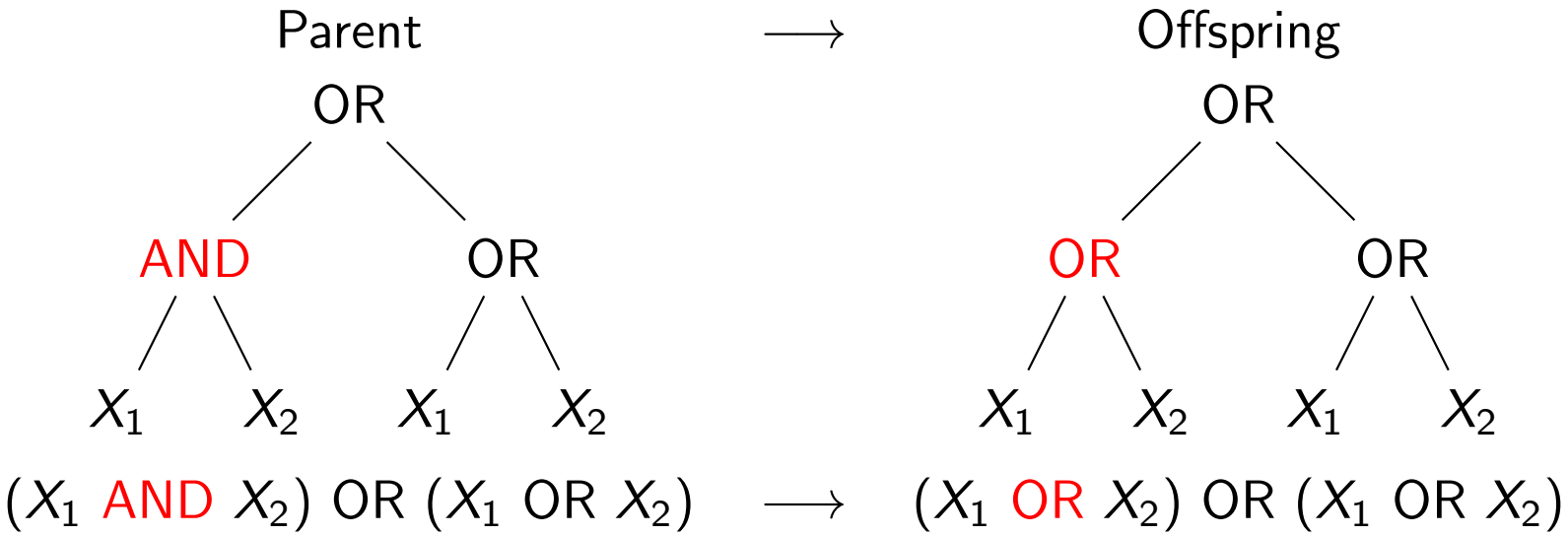


$(X_1 \text{ AND } X_2) \text{ OR } (X_1 \text{ OR } X_2) \longrightarrow (X_1 \text{ OR } X_2) \text{ OR } (X_1 \text{ OR } X_2)$

- Expressions are represented by trees.
- Variation operators produce offspring by syntactic manipulation of parent trees

X_1	X_2	Target Output	Parent's Output
1	1	**1**	1
1	0	**0**	1
0	1	**1**	1
0	0	**0**	0
Fitness			3

Traditional Genetic Programming

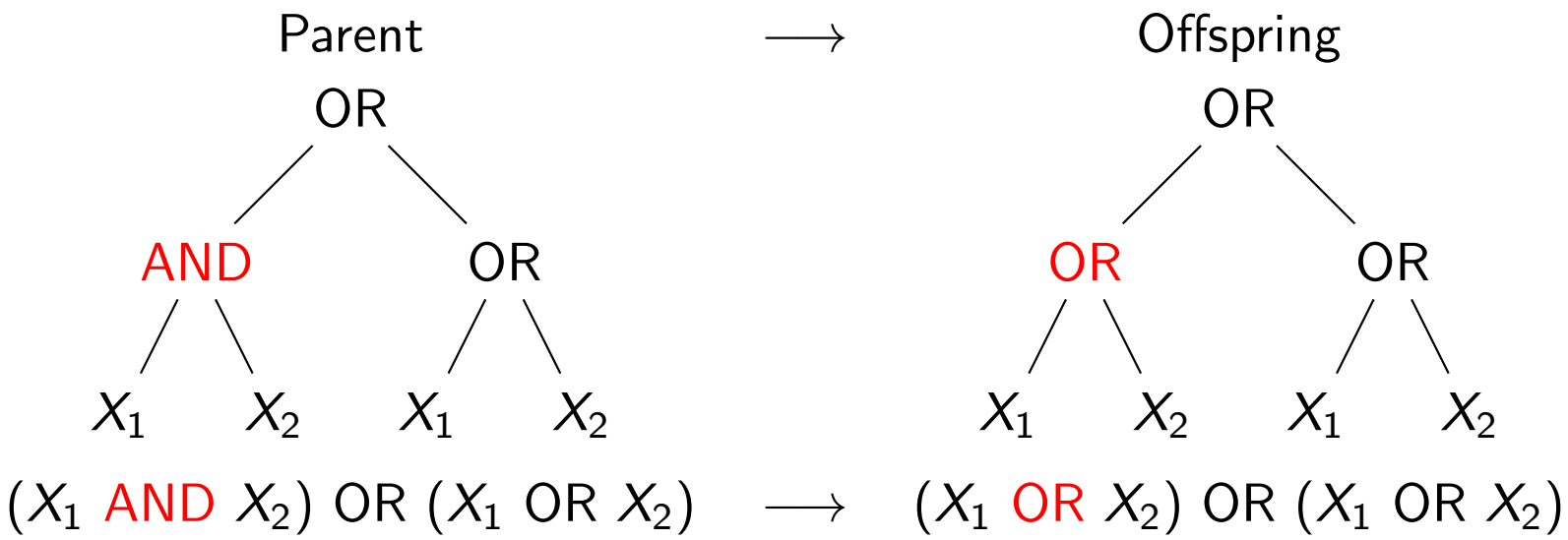


- Expressions are represented by trees.
- Variation operators produce offspring by syntactic manipulation of parent trees

Why GP is hard to analyse (and thus to efficiently design)?

- The fitness depends on the behaviour (semantic) of the function to evolve, while variation operators act on the syntax.
- How a syntactic variation operator affects the semantic of the expression?

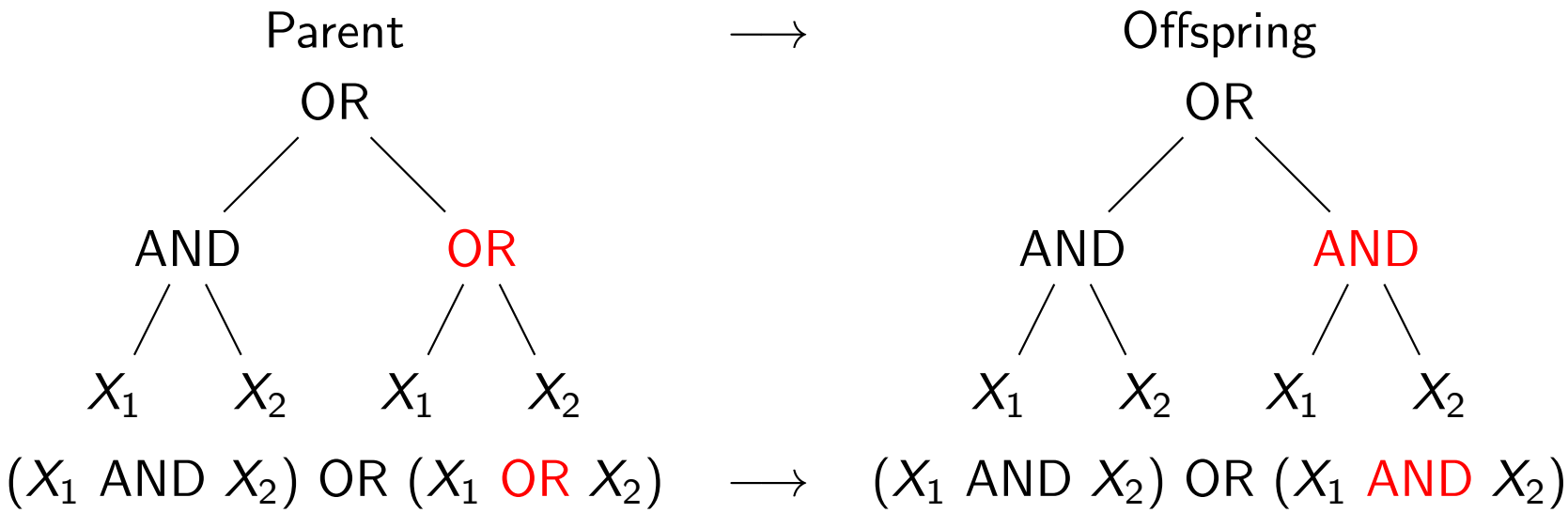
Traditional Genetic Programming



X_1	X_2	Output	Parent	Offspring
1	1	1	1	1
1	0	0	1	1
0	1	1	1	1
0	0	0	0	0

In this case the variation operator **hasn't produced any change** in the semantic of the function.

Traditional Genetic Programming



X_1	X_2	Output	Parent	Offspring
1	1	1	1	1
1	0	0	1	0
0	1	1	1	0
0	0	0	0	0

In this case it has produced a **big changes** on the truth table.

OneMax Problem

- box-constrained binary problem
- fitness of a solution (to maximise): number of matching bits with a particular fixed bit-string (traditionally a string consisting of all 1s)

Example

- Size of the problem n=5
- Target string $\overline{X} = 11111$
- fitness of an individual X: $\sum_{i=1}^{5} 1 - HD(x_i, \overline{x}_i)$ (e.g. $X = 10110$ has fitness equal to 3).

RLS Algorithm

Algorithm - Random Local Search (RLS)

1. Inizialise P_0 with a bitstring $x \in \{0,1\}^n$
2. Create the offspring x' by flipping one bit at random
3. Select the fittest between x' and x for survival
4. Repeat from point 2 until stopping condition applies

Runtime of RLS on OneMax:

- An individual with fitness k (thus having k 1s) is mutated to a better individual if one zero is flipped. This happens with probability $p = \frac{n-k}{n}$.
- The expected time to get to the optimal solution is thus: $E[T] = \sum_{k=1}^{n} \frac{1}{p} = \sum_{k=1}^{n} \frac{n}{n-k} = O(n \log n)$.

Why so easy? Because it is clear how a mutation affect the fitness.

Can we do the same with GP?

Semantic Genetic programming

Semantic variation operator

- acts on the syntax of an expression (given a parent expression produce as offspring another expression)
- **guarantee** some semantic features for the offspring (e.g. a semantic mutation might produce an offspring whose truth table always differs just by one row from the parent)

How to implement a semantic operator?

By trial & error use standard operators and reject offspring that do not conform to the semantic requirement → wasteful

By construction[1] : Operate on the syntax of the expression in a way that by construction the semantic criterion is satisfied

[1] A. Moraglio, K. Krawiec, C. Johnson – *Geometric Semantic Genetic Programming* – 5th Workshop on Theory of Randomized Search Heuristics, 2011

A semantic mutation operator[2]

Definition

Bit-flip point mutation: Given a parent function $P : \{0,1\}^n \to \{0,1\}$ the mutation returns the offspring boolean function $P' = \left(P \wedge \overline{M}\right) \vee \left(M \wedge \overline{P}\right)$, where M is a random minterm of all input variables.

Example:

Random minterm: $\mathrm{M} = X_1 \wedge \overline{X_2}$

Parent: $\mathrm{P} = (X_1 \wedge X_2) \vee (X_1 \vee X_2)$

Offspring: $O = (P \wedge \overline{M}) \vee (\overline{P} \wedge M)$

X_1	X_2	Output	M	Parent	Offspring
1	1	1	0	1	1
1	0	0	1	1	0
0	1	1	0	1	1
0	0	0	0	0	0

[2] A. Moraglio, A. Mambrini, and L. Manzoni – *Runtime Analysis of Mutation-Based Geometric Semantic Genetic Programming on Boolean Functions* – Foundations of Genetic Algorithms (FOGA 2013)

Semantic Bit-flip point mutation

X_1	X_2	Output	M	Parent	Offspring
1	1	1	0	1	1
1	0	0	1	1	0
0	1	1	0	1	1
0	0	0	0	0	0

Effect on the output vector (semantic):

- Select a row at random (randomly generating M)
- Flip the output of the function on that row (the row where M is true)

Basically at each generation we flip one bit from a bitstring (the output vector) aiming to reach a desired bit configuration...

that reminds RLS on OneMax!

Runtime

Search equivalence

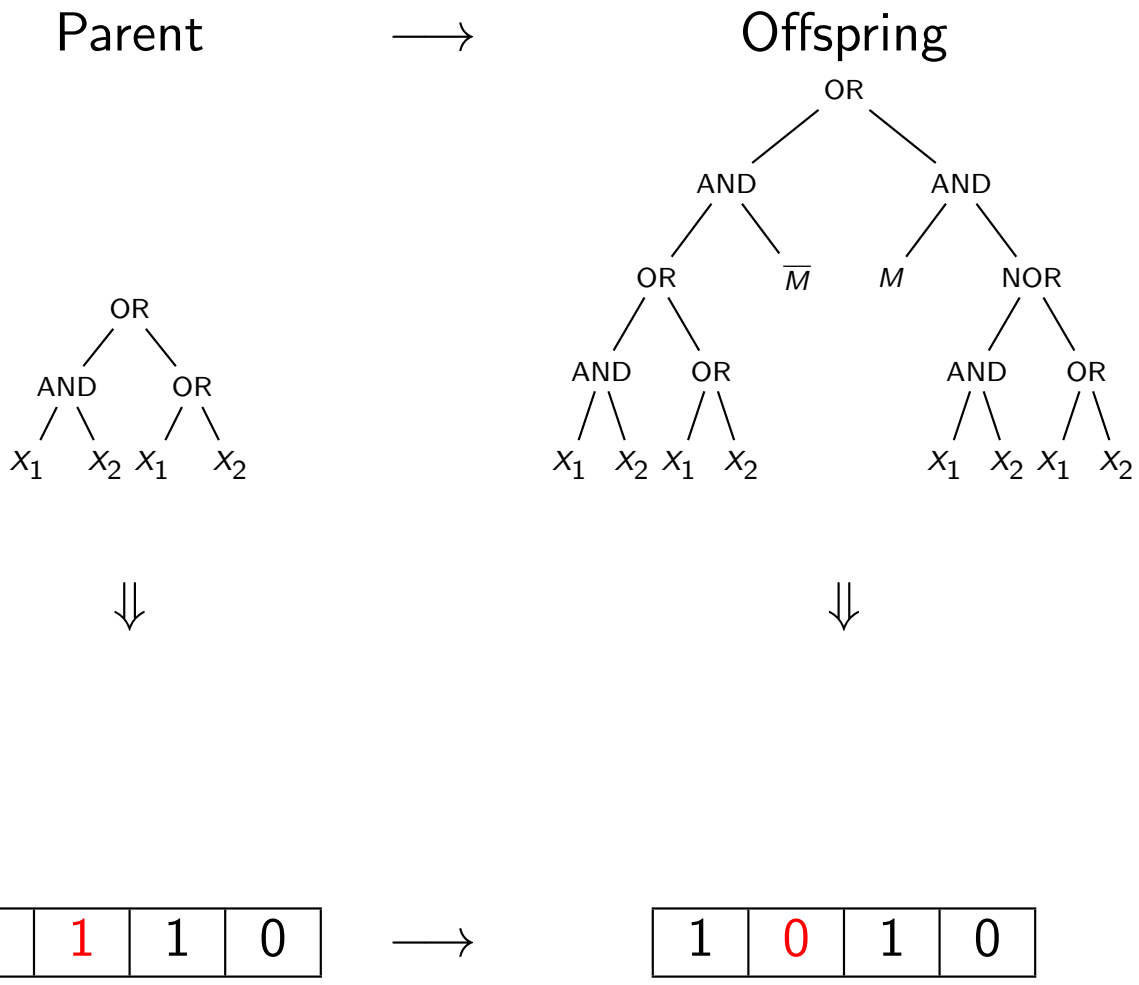
Genetic Programming using semantic operators solving **ANY** black-box boolean function learning problem

$\Updownarrow$ **is equivalent to** $\Updownarrow$

a Genetic Algorithm (evolving bitstrings) solving OneMax

The search outputs a tree (i.e., a boolean expression), but the runtime analysis can be done on the GA!

Equivalence



Runtime

Search equivalence

Genetic Programming using semantic operators solving **ANY** black-box boolean function learning problem

$\Updownarrow$ **is equivalent to** $\Updownarrow$

a Genetic Algorithm (evolving bitstrings) solving OneMax

The search outputs a tree (i.e., a boolean expression), but the runtime analysis can be done on the GA!

Theorem

Geometric Semantic Genetic Programming solves the black box boolean function learning problem in time $T = O(N \log N)$, where N is the length of the output vector (i.e. the number of rows of the truth table).

Conclusion

Take home messages:

- Geometric Semantic Genetic Programming makes the search performed by GP on the space of functions (trees) equivalent to that performed by a GA on the output space.
- It is thus possible to easily get runtime results for GP reusing runtime results for GA thus:
  - Get guarantees of the convergence time of the algorithm
  - Improve the operators' design getting insight from the theory (theory-laden design)

Thank you!