Theory-Laden Design of Mutation-Based Geometric Semantic Genetic Programming for Learning Classification Trees

Andrea Mambrini[1], Luca Manzoni[2], Alberto Moraglio[1]

University of Birmingham, Birmingham UK
Università degli Studi di Milano-Bicocca, Milan, Italy

21th June 2013

Outline

- Preliminaries (Runtime Analysis of RLS on Generalized OneMax)
- Traditional Genetic Programming: why is it **hard** to analyse?
- Geometric Semantic Genetic Programming (GSGP): why it is **easy** to analyse
- Theory-laden design of mutation operators for GSGP learning classification trees

Generalized OneMax Problem

- box-constrained integer problem
- fitness of a solution (to maximise): number of matching elements with a particular target vector (traditionally a string consisting of all 1s)

Example

- Size of the problem n=5
- Target vector $\overline{X} = 11111$
- fitness of an individual X: $\sum_{i=1}^{5} 1 - HD(x_i, \overline{x}_i)$ (e.g. $X = 13124$ has fitness equal to 2).

RLS Algorithm

Algorithm - Random Local Search (RLS)

1. Inizialise P_0 with a vector of symbols $x \in OS^n$, where OS is a finite set of symbols
2. Create the offspring x' by changing one element of the vector at random
3. Select the fittest between x' and x for survival
4. Repeat from point 2 until stopping condition applies

From theory:
Expected runtime of RLS on (Generalized) OneMax: $O(|OS|n \log n)$

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 operation on the syntax of the tree
- 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.

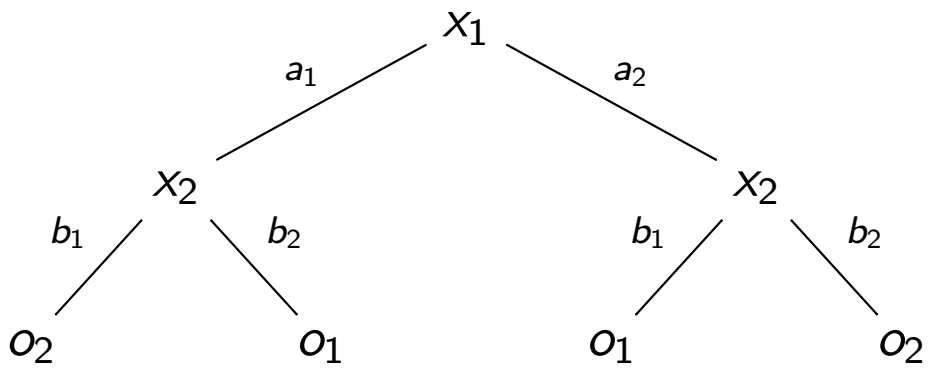
Black-box learning of classification trees

Which classification tree represents the following classification table?

X_1	X_2	O
a_1	b_1	o_1
a_1	b_2	o_2
a_2	b_1	o_1
a_2	b_2	o_2

$IS_1 = \{a_1, a_2\}$
$IS_2 = \{b_1, b_2\}$
$OS = \{o_1, o_2\}$

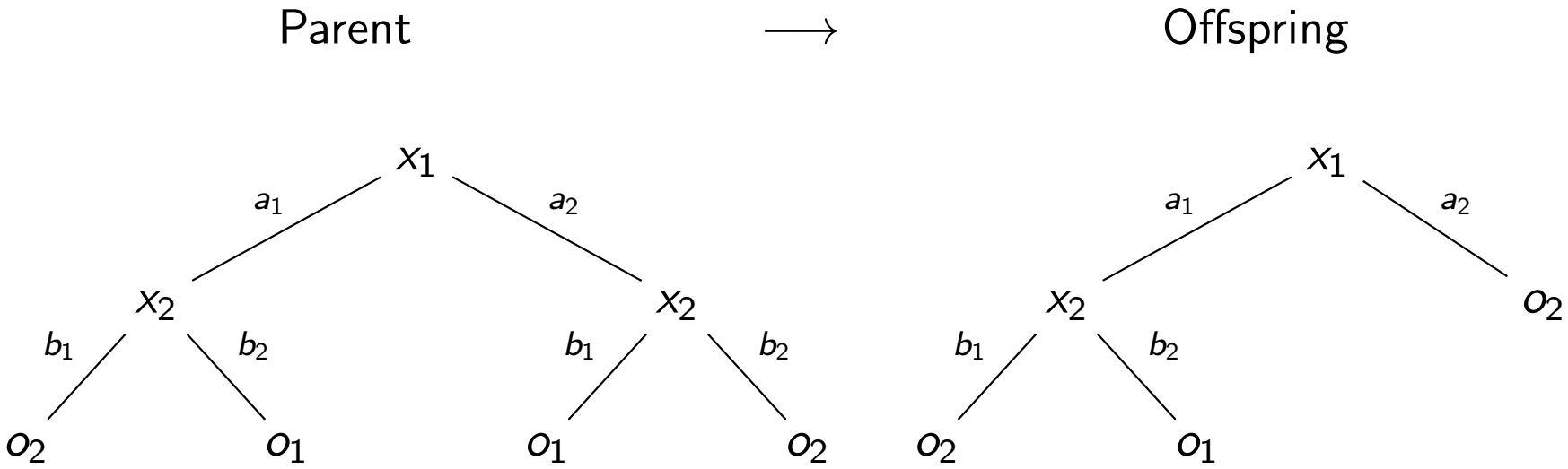
$\rightarrow$



Black-box setting:

- We don't have access to the classification 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.

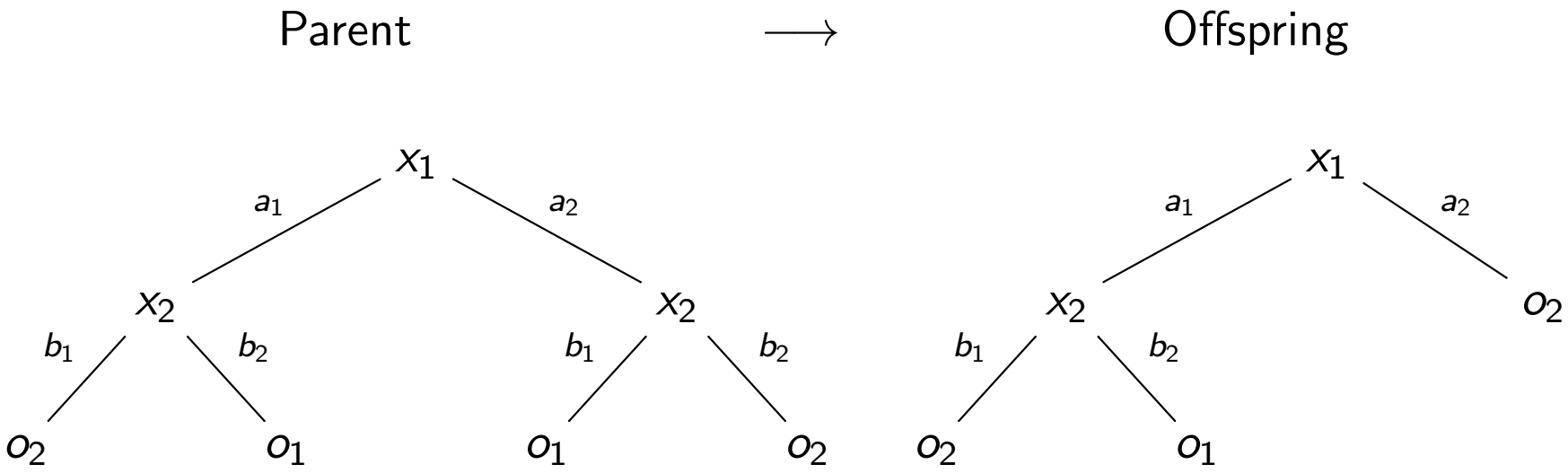
Traditional Genetic Programming



- Starts from a random classification tree.
- Variation operators produce offspring by syntactic manipulation of parent trees

X_1	X_2	O	Parent	Offspring
a_1	b_1	$\mathbf{o_1}$	o_2	o_2
a_1	b_2	$\mathbf{o_2}$	o_1	o_1
a_2	b_1	$\mathbf{o_1}$	o_1	o_2
a_2	b_2	$\mathbf{o_2}$	o_2	o_2
Fitness			2	1

Traditional Genetic Programming



- Starts from a random classification tree.
- Variation operators produce offspring by syntactic manipulation of parent trees

Why is GP 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? (genotype-phenotype mapping)

Aim

The aim is to design GP operators such that:

- it is easy to deal with them from a runtime point of view (i.e. the genotype-phenotype mapping is *easy*)
- we can easily get runtime results for classes of problems
- we can use those results to design better operators (e.g provably good parameters setting)

Semantic Genetic programming

Semantic variation operator

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

How to implement a semantic operator?

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

By construction : Operate on the syntax of the tree in a way that by construction the semantic criterion is satisfied

A semantic mutation operator

Definition

Semantic point mutation: Given a parent classifier `T`, returns the offspring classifier `T' = IF CONDR THEN OUTR ELSE T` where `CONDR` is a random condition which is true only for a single input combination of all input variables, and `OUTR` is a random output symbol.

Example ($n = 2$, $IS_1 = \{a_1, a_2, a_3\}$, $IS_2 = \{b_1, b_2\}$, $OS = \{o_1, o_2, o_3, o_4\}$):

Parent: `T = IF` $(X_1 == a_1)$ `THEN` o_2 `ELSE` o_3.

Random condition: `CONDR =` $(X_1 == a_1$ `AND` $X_2 == b_1)$, `OUTR=`o_4

Offspring: `T' = IF` $(X_1 == a_1$ `AND` $X_2 == b_1)$ `THEN` o_4

`ELSE` `(IF` $(X_1 == a_1)$ `THEN` o_2 `ELSE` o_3`)`

X_1	X_2	O	$P(T)$	$P(T')$
a_1	b_1	o_2	$\mathbf{o_2}$	$\mathbf{o_4}$
a_2	b_1	o_1	o_3	o_3
a_3	b_1	o_3	o_3	o_3
a_1	b_2	o_2	o_2	o_2
a_2	b_2	o_4	o_3	o_3
a_3	b_2	o_1	o_3	o_3

Semantic point mutation

X_1	X_2	O	$P(T)$	$P(T')$
a_1	b_1	o_2	$\mathbf{o_2}$	$\mathbf{o_4}$
a_2	b_1	o_1	o_3	o_3
a_3	b_1	o_3	o_3	o_3
a_1	b_2	o_2	o_2	o_2
a_2	b_2	o_4	o_3	o_3
a_3	b_2	o_1	o_3	o_3

Effect on the output vector (semantic):

- Select a row at random (randomly generating `CONDR`)
- Force the output of the function on that row (the row where `CONDR` is true) to `OUTR`

Basically at each generation we force one element from a vector of symbols (the output vector) aiming to reach a desired vector configuration...

that reminds of RLS on Generalized OneMax!

Runtime

Search equivalence

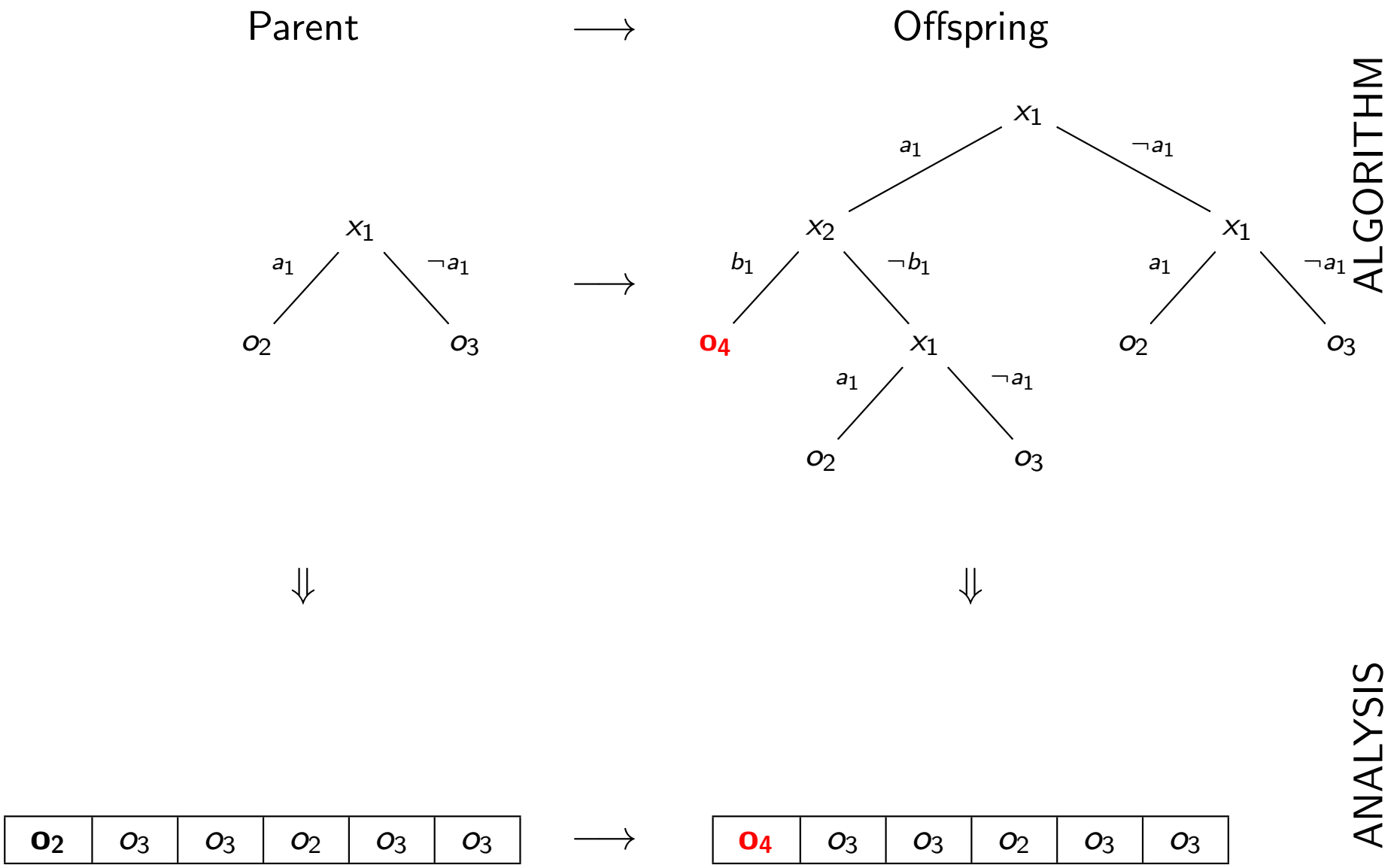
Genetic Programming using semantic operators solving **ANY** black-box classification tree learning problem

⇳ **is equivalent to** ⇳

an Evolutionary Algorithm (evolving a vector of symbols) solving Generalized OneMax

The search outputs a tree, but the runtime analysis can be done on the EA!

Equivalence



Runtime

Search equivalence

Genetic Programming using semantic operators solving **ANY** black-box classification tree learning problem

⇳ **is equivalent to** ⇳

an Evolutionary Algorithm (evolving a vector of symbols) solving Generalized OneMax

The search outputs a tree, but the runtime analysis can be done on the EA!

Theorem

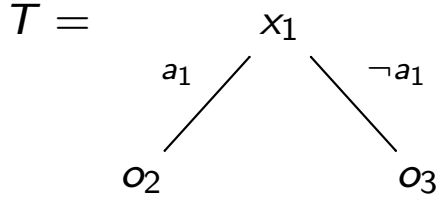
Geometric Semantic Genetic Programming solves the black-box classification tree learning problem in expected time $T = O(|OS|N \log N)$, where N is the length of the output vector (i.e. the number of rows of the classification table) and $|OS|$ is the size of the output set.

Issues

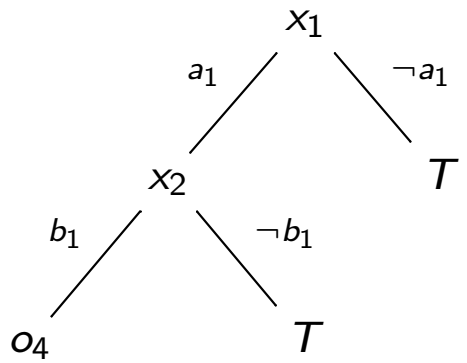
The proposed operator still suffers from the following issues:

1. A fair choice of the problem size is the number of input variables n. Since $N = \prod_{i=1}^{n} |IS_i| > (IS_{\min})^n$, the runtime (which is $O(|OS|N \log N)$) is actually exponential in the problem size.
2. In the way we designed the semantic mutation operator the size of the offspring tree grows exponentially.

Parent



Offspring



Polynomial-sized training set

Problem The runtime $O(|OS|N \log N)$ is exponential in the problem size n (since $N = \prod_{i=1}^{n} |IS_i|$).
Actually, in practice, the training set is *small*. Particularly we can assume it to have polynomial size w.r.t. the number of input variables.
$\tau = \text{poly}(n)$.

X_1	X_2	X_3	O
a_1	b_1	c_1	o_1
a_1	b_1	c_2	o_2
a_1	b_2	c_1	o_1
a_1	b_2	c_2	o_2
a_2	b_1	c_1	o_1
a_2	b_1	c_2	o_2
a_2	b_2	c_1	o_1
a_2	b_2	c_2	o_2

This transforms the equivalent problem on output vectors into a "sparse" version of OneMax.
Unfortunately, using the point operator, the time to reach the optimum is still exponential.

Incomplete conditions

Incomplete minterms force mutation of blocks of the output vector instead of single elements.

X_1	X_2	X_3
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_1
a_1	b_2	c_2
a_2	b_1	c_1
a_2	b_1	c_2
a_2	b_2	c_1
a_2	b_2	c_2

$X_2 = b_1 \wedge X_3 = c_1$
$X_2 = b_2 \wedge X_3 = c_1$
$X_2 = b_1 \wedge X_3 = c_2$
$X_2 = b_2 \wedge X_3 = c_2$

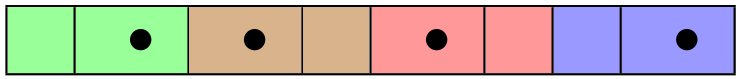
We force more elements at each generation.

Block mutation

Definition

Fixed Block Mutation (FBM):

- Select randomly a set V of $v < n$ variables
- At each generation draw an incomplete condition `CONDR` comprising all the fixed v variables
- Use `CONDR` for the forcing mutation



$N = 8,\ v = 2 \rightarrow b = 4$

Setting v

Assuming $\tau = O(n^c)$, setting $v = \lceil (2c+1) \log_{r_m}(n) \rceil$

$$\Downarrow$$

$$\text{runtime: } O(|OS| b \log b) = O(|OS| r_M n^{(2c+1)\log_{r_m}(r_M)} \log n)$$

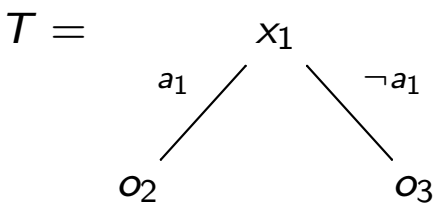
$$p = O(1 - 2/n)$$

r_m, r_M: respectively minimum and maximum input size across all variables

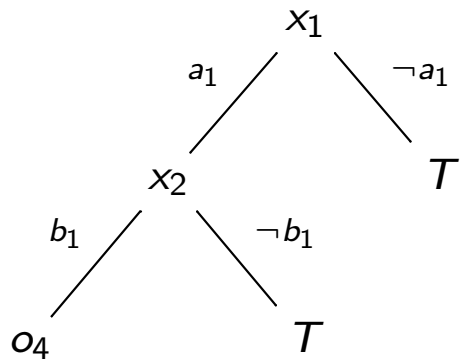
A shorter operator

Problem In the way we designed the semantic mutation operator the size of the offspring tree grows exponentially.

Parent



Offspring



Solution (IFBM)

Design the mutation operator such that the offspring tree always have one specific variable for each level (it is possible to do that defining some insertion rules for new nodes and subtrees)

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
- We can use those theoretical results to design good semantic GP operators (i.e. we designed a block mutation operator (IFBM) which reach the optimum in polynomial time with high probability keeping the size of the classification tree polynomial).

Future work:

- Design GSGP operators for other settings (i.e. Classifier with real inputs and regression trees)
- Study the generalization ability of GSGP as a machine learning tool
- Test GSGP on real-world problems

Thank you!