

Grammatical Evolution Mapping for Semantically-Constrained Genetic Programming

Alcides Fonseca, Paulo Santos, Guilherme Espada and Sara Silva

Grammar-Guided Genetic Programming

Attribute Grammar Evolution

[Cruz05]

CFG-GP

[Whigham95]

Typed Genetic Programming

Refined-Typed GP

[Fonseca20]

Strongly-Typed GP

[Montana95] + [Binard08]

More expressive power



Faster Synthesis

Grammar-Guided Genetic Programming

Attribute Grammar
Evolution

[Cruz05]

CFG-GP

[Whigham95]

Typed Genetic Programming

Refined-Typed GP

[Fonseca20]

Strongly-Typed GP

[Montana95] + [Binard08]

More expressive
power

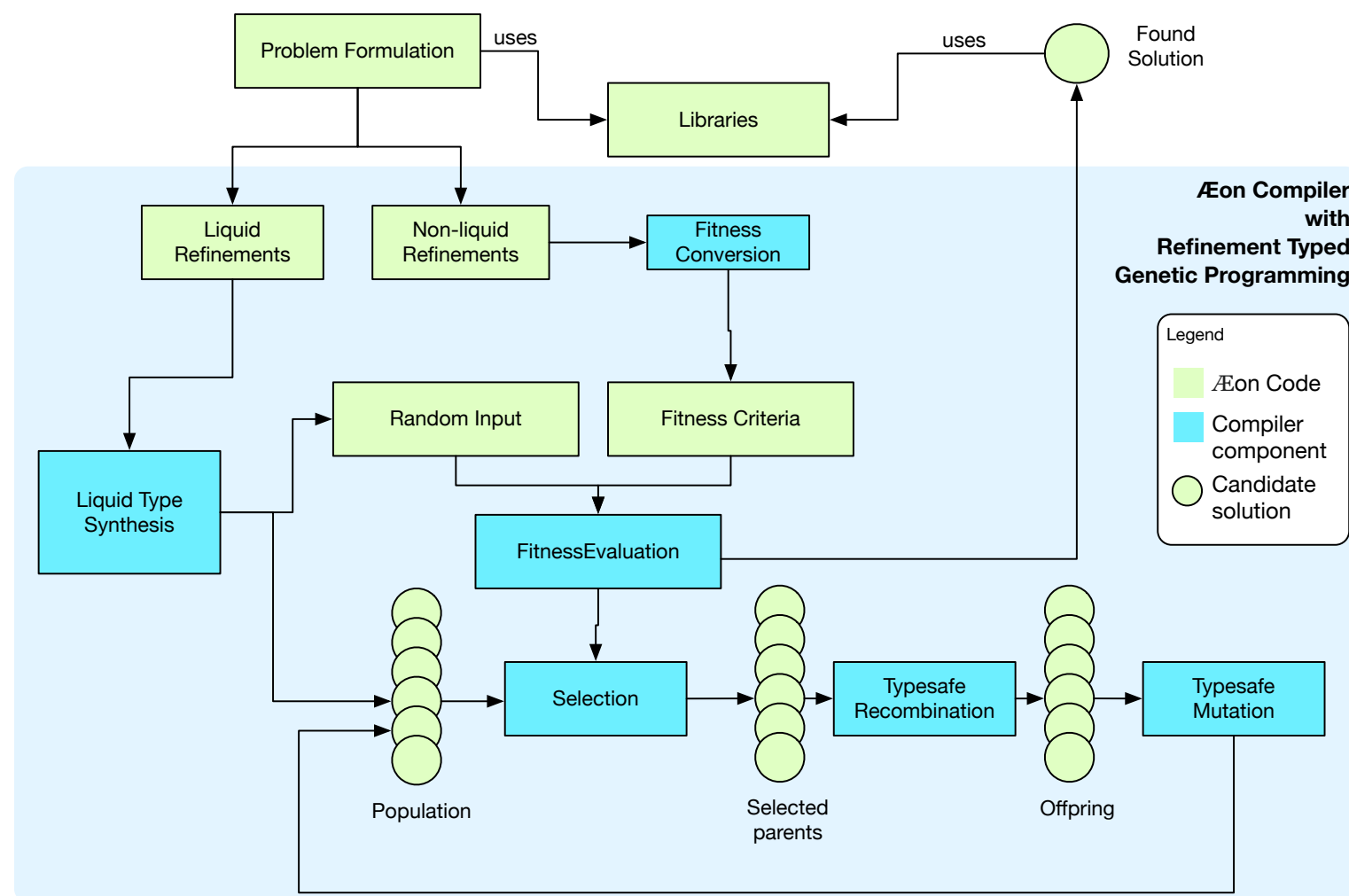
Faster Synthesis



SMT-based Program Synthesis

Fitness Function via Property-Based Testing

```
def f(i: Int, j: Int) -> k: Int | k > i+j and isPrime(k) {  
  ??  
}
```



Fonseca, Alcides, Paulo Santos, and Sara Silva. "The Usability Argument for Refinement Typed Genetic Programming." *International Conference on Parallel Problem Solving from Nature*. Springer, Cham, 2020.

- We are using GP for **Non-Deterministic and unmodeled** Program Synthesis
- We are using Liquid Program Synthesis for Improving GP

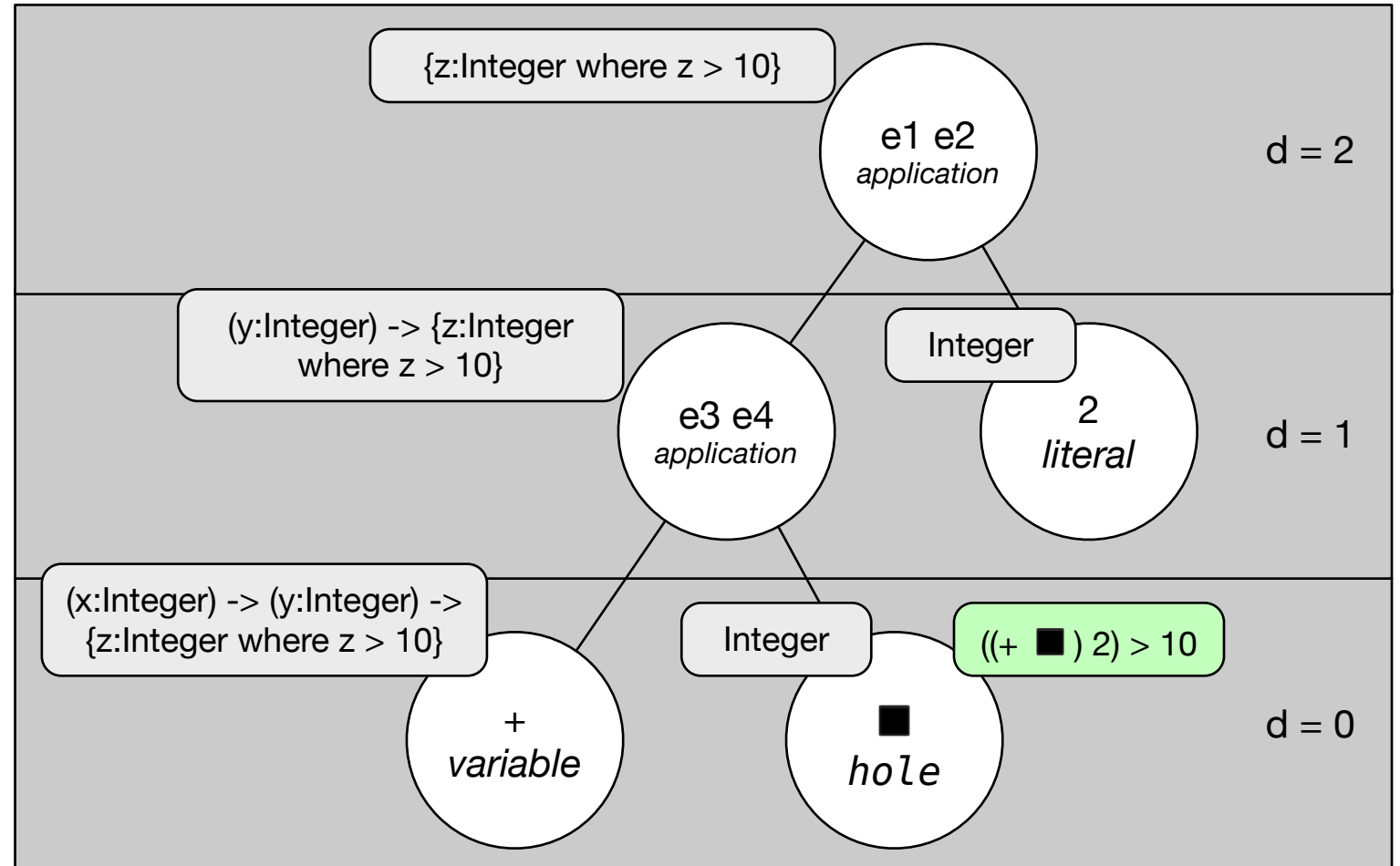
TREE INITIALIZATION

$\langle t \rangle ::= 1$
 $\quad | \quad x$
 $\quad | \quad \langle t \rangle \langle t \rangle$
 $\quad | \quad \lambda x \mapsto \langle t \rangle$
 $\quad | \quad \text{if } \langle t \rangle \text{ then } \langle t \rangle \text{ else } \langle t \rangle$

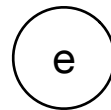
$$\frac{x: T \in \Gamma}{\Gamma \vdash T \xrightarrow{S} x} \quad (\text{STGP-Syn-Var})$$

$$\frac{\Gamma \xrightarrow{T} U \quad \Gamma \vdash U \rightarrow T \xrightarrow{S} e_1 \quad \Gamma \vdash U \xrightarrow{S} e_2}{\Gamma \vdash T \xrightarrow{S} e_1 e_2} \quad (\text{STGP-Syn-App})$$

$$\frac{\text{fresh}(x) \quad \Gamma, x: T \vdash e \xrightarrow{S} U}{\Gamma \vdash T \rightarrow U \xrightarrow{S} \lambda x: T. e} \quad (\text{STGP-Syn-Abs})$$



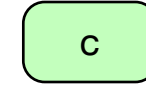
Legend



AST node



Expected Type

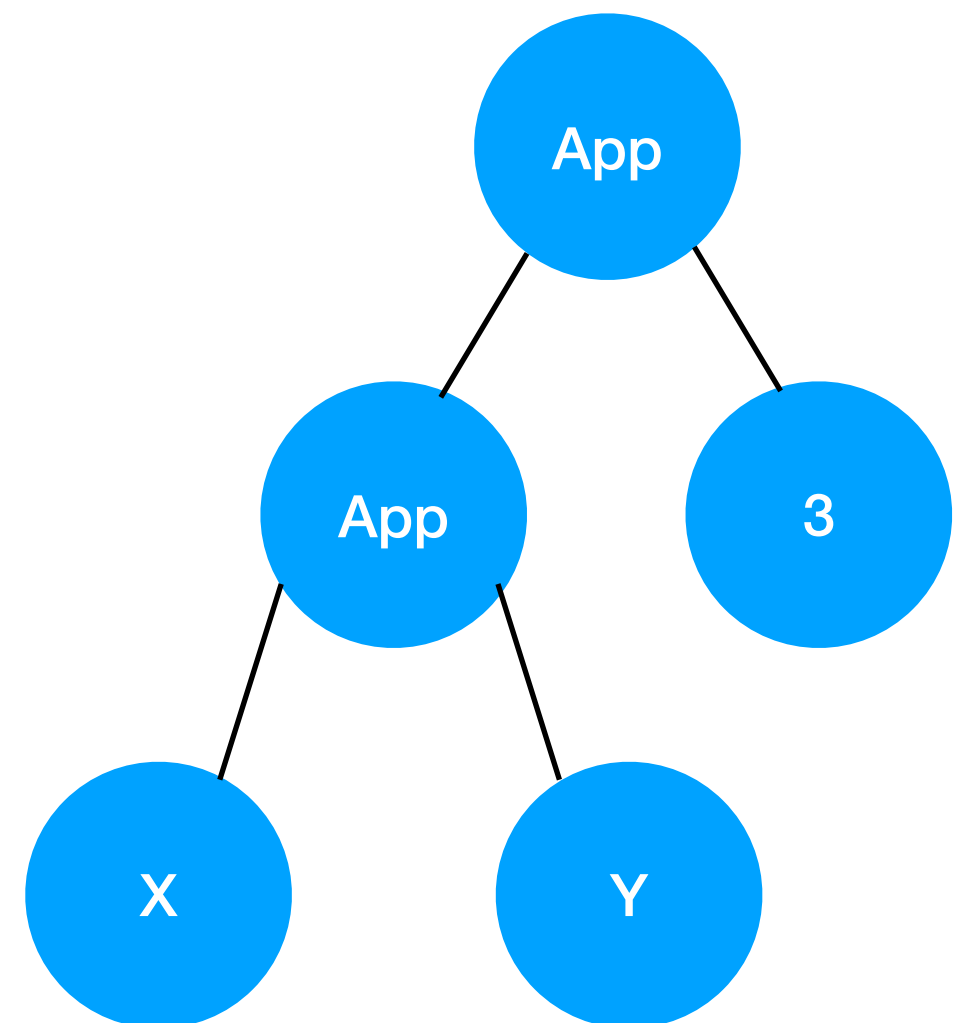


Expected condition

Practical limitation:
Population initialization
takes too long

$\langle t \rangle ::= 1$
 $\quad x$
 $\quad \langle t \rangle \langle t \rangle$
 $\quad \lambda x \mapsto \langle t \rangle$
 $\quad \text{if } \langle t \rangle \text{ then } \langle t \rangle \text{ else } \langle t \rangle$

2	7	2	0	2	1	0	3
App	App	Var	X	Var	Y	Var	3

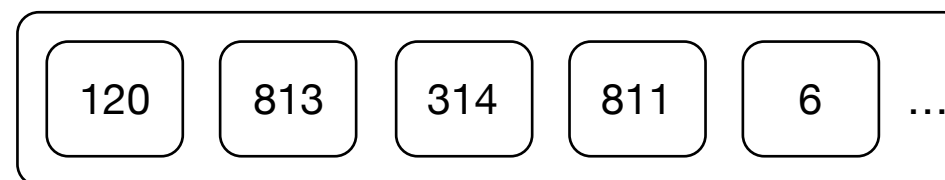


Is Y of the right type?

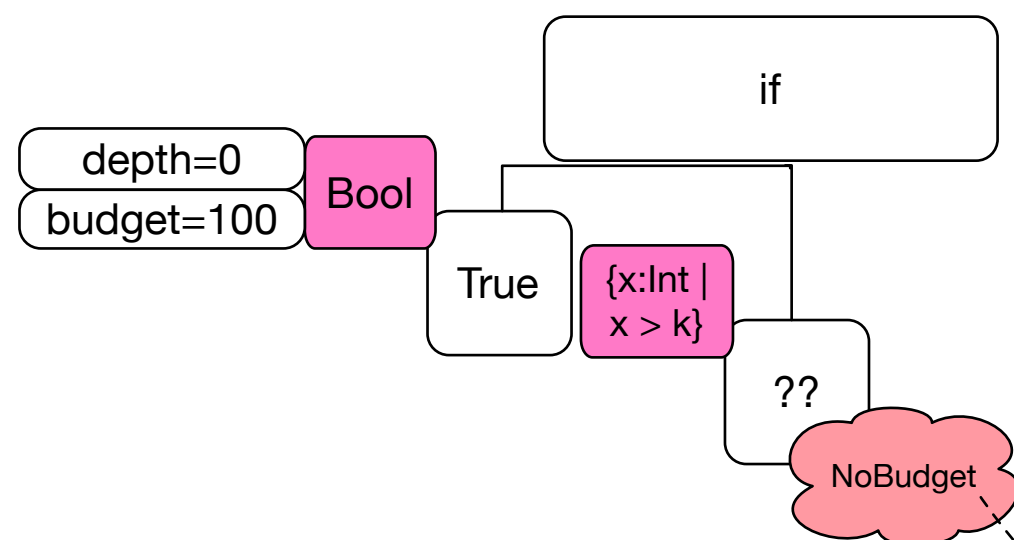
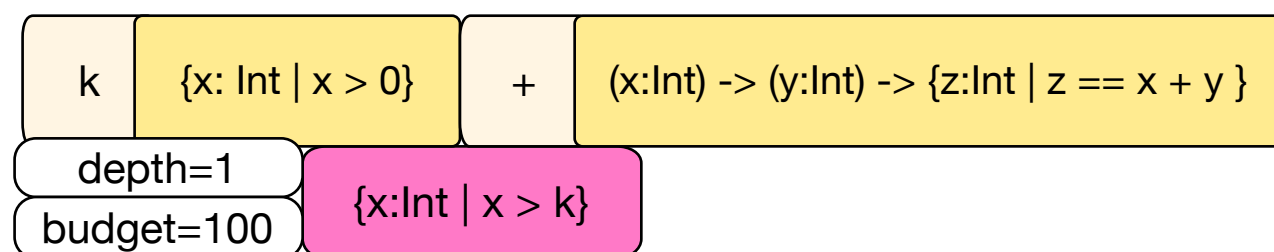
Backtrack!

```
def term_synthesis(ctx, type, d):  
    options = []  
    if ctx.has_lit_of_type(type):  
        options.add(lit)  
    if ctx.has_var_of_type(type):  
        options.add(var)  
    if d > 0:  
        options.add(app)  
        if type.is_abstraction():  
            options.add(abs)  
        options.add(ifthenelse)  
  
    while options and has_budget():  
        checkpoint()  
        try:  
            return follow_rule(choice(options))  
        except NoBudget:  
            rollback()  
    raise NoBudget()
```

Individual



$$813 \% 300 = 213$$



d = 1		
literal	100	0
var	100	100
abs /1	100	
app /2	100	100
if /3	100	200
		300

- 10%

Goal: To synthesize an initial population of 1000 elements of type:

$\{x:\text{Int} \mid x > 0 \ \&\& \ x < 1000\}$

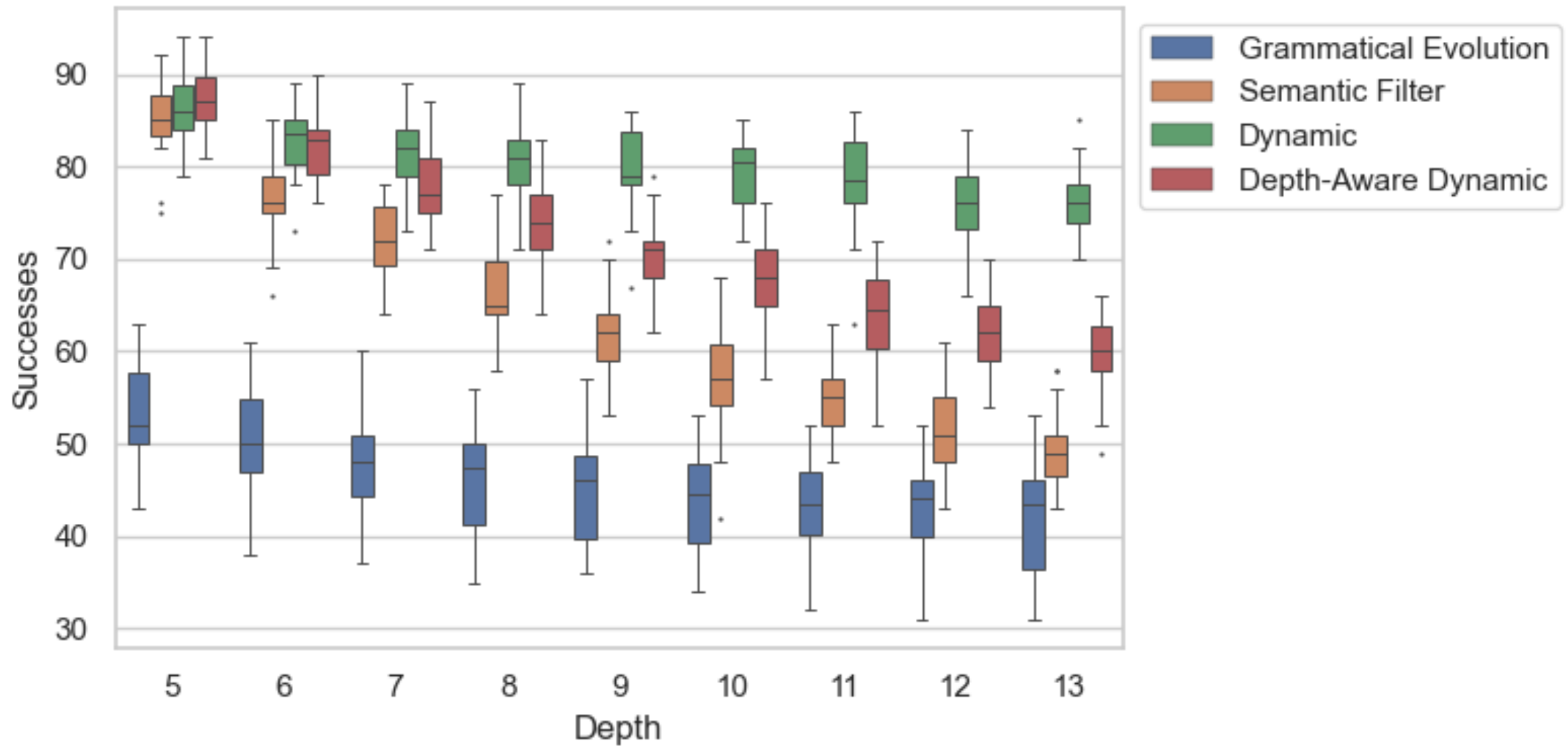
That is the same as:

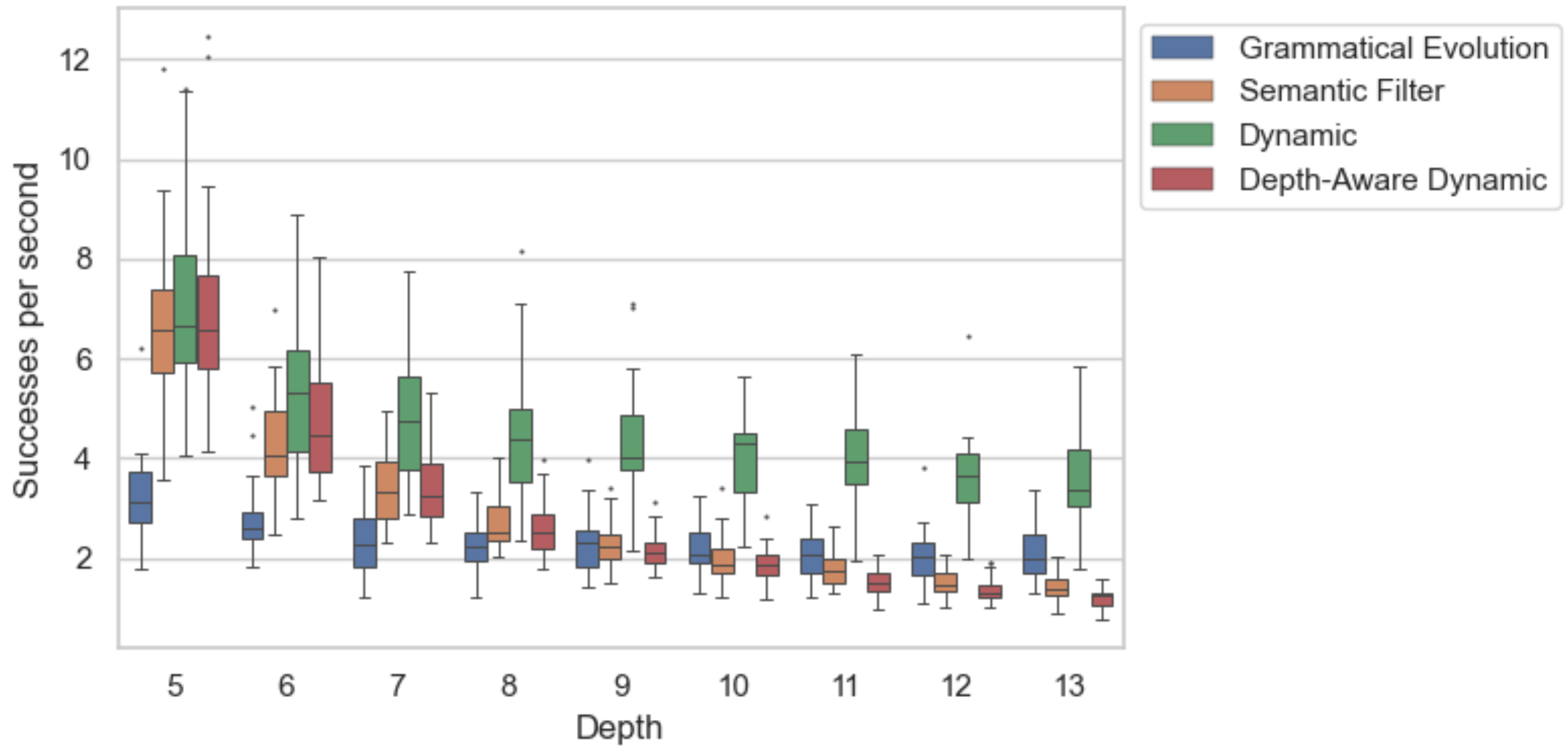
$\{y:\text{Int} \mid 10 * y == 10000 \}$

$\{z:\text{Int} \mid 2+z == 6 \}$

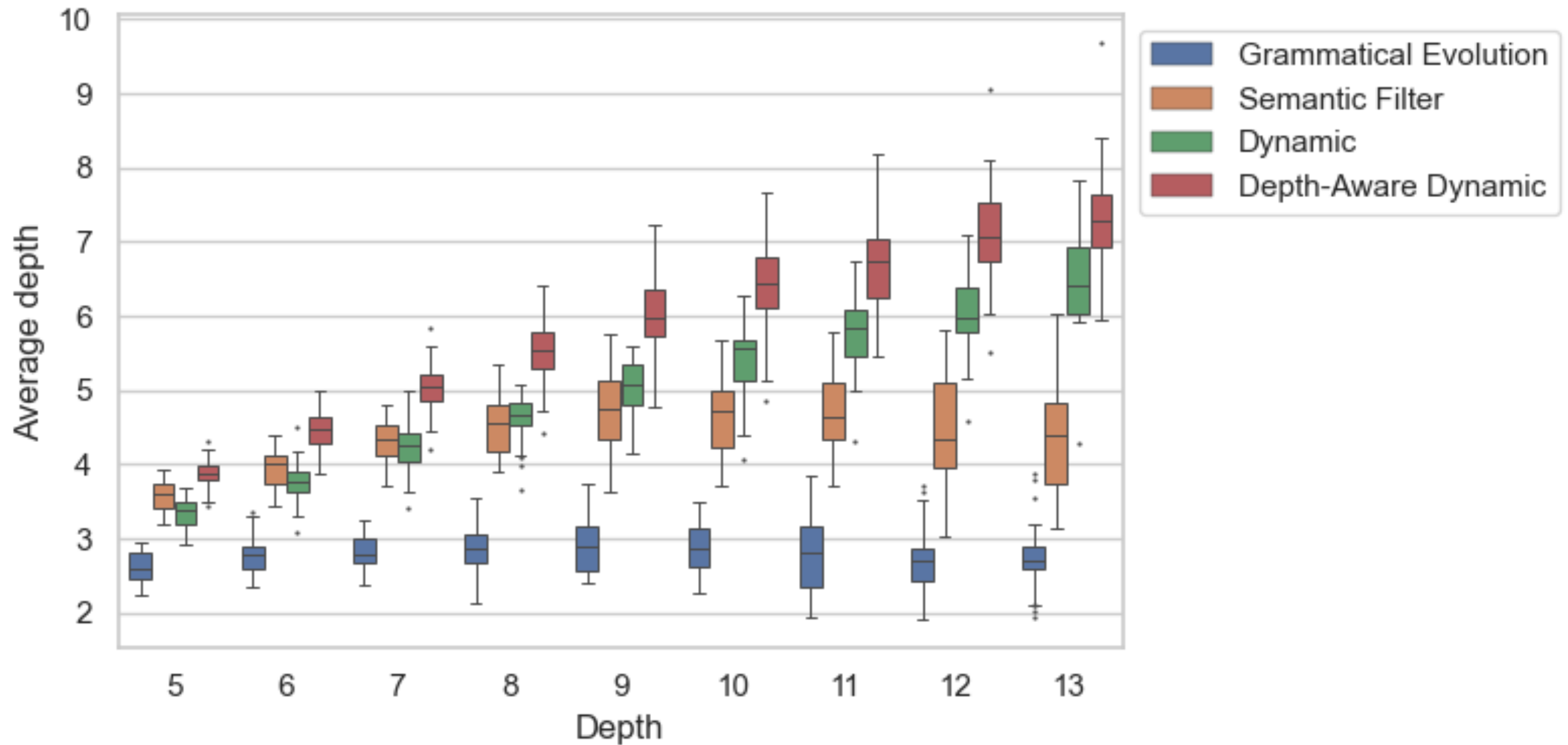
$\vdash \{x:\text{Int} \mid (x \geq 21 \ \&\& \ x < y) \parallel x < (2*10) \parallel x == (80/z)\}$

SUCCESSES

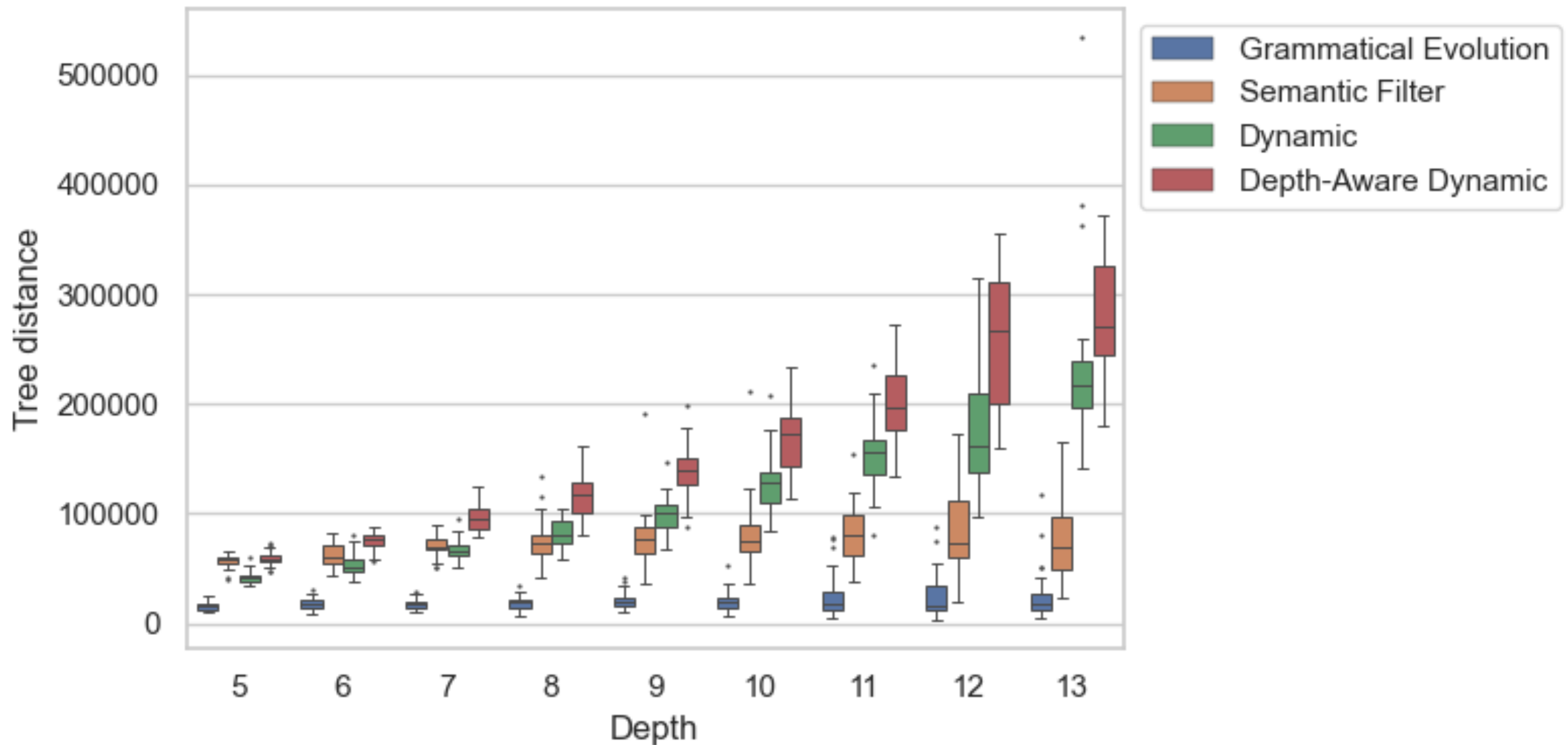


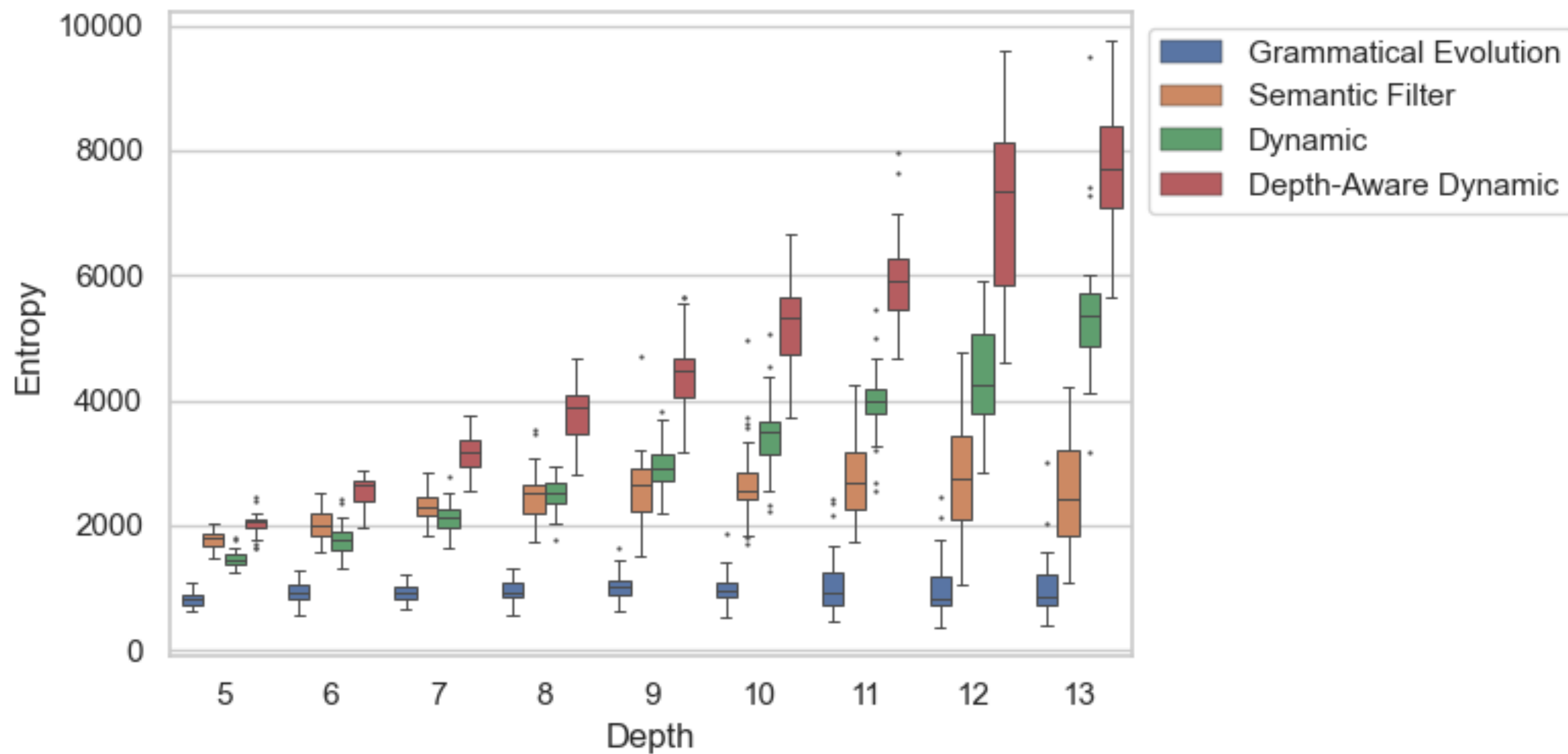


AVERAGE DEPTH



TREE DISTANCE





- Traditional population initialization using Liquid Types is not efficient.
- Adapting the probabilities of grammar rules can improve the initialization.
- Probabilities per-depth do not improve performance, but improve diversity and increase average depth.
- Bias in the initial population must be evaluated after the evaluation!

Grammatical Evolution Mapping for Semantically-Constrained Genetic Programming

Alcides Fonseca, Paulo Santos, Guilherme Espada and Sara Silva

alcides@fc.ul.pt

sara@fc.ul.pt