

Automating Algorithm Design through Genetic Programming Hyper-Heuristics

Elsa Browning

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

April 15, 2017
Morris, MN

What does the title mean?

- Reducing the human component in algorithm design
- More work at the beginning, more possibilities
- Genetic programming hyper-heuristics as a method to the madness



<https://scratch.mit.edu/discuss/m/topic/200574/>

Outline

- 1 Background
- 2 Hyper-heuristics
- 3 Genetic Programming Variants
- 4 Autoconstruction
- 5 Summary

Outline

- 1 **Background**
 - Evolutionary Computation
 - Genetic Programming
- 2 Hyper-heuristics
- 3 Genetic Programming Variants
- 4 Autoconstruction
- 5 Summary

Outline

- 1 Background
- 2 Hyper-heuristics**
 - What they are
 - What they aren't
 - How they work
- 3 Genetic Programming Variants
- 4 Autoconstruction
- 5 Summary

Outline

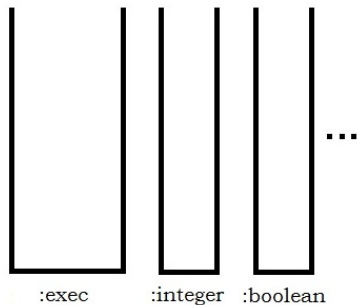
- 1 Background
- 2 Hyper-heuristics
- 3 Genetic Programming Variants**
 - Why they matter
 - Stack-based genetic programming
- 4 Autoconstruction
- 5 Summary

Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```

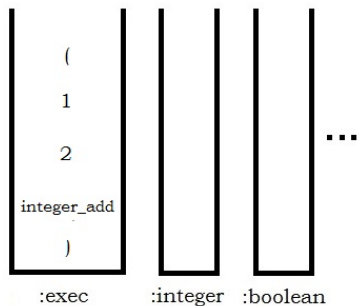


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```

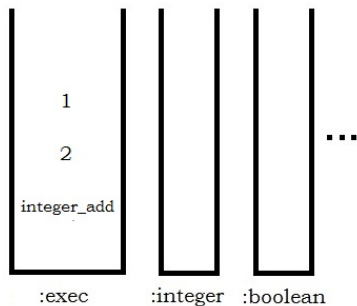


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```

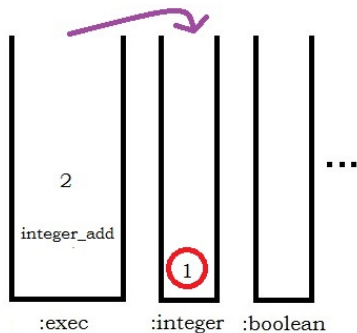


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```

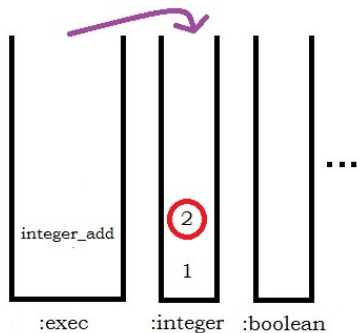


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```

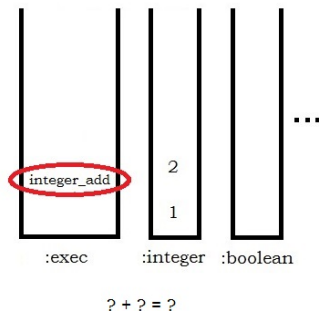


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```

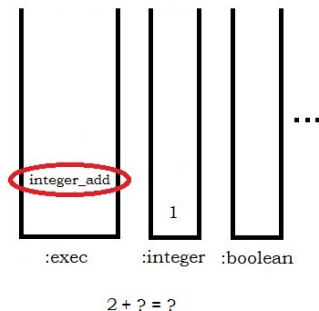


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```

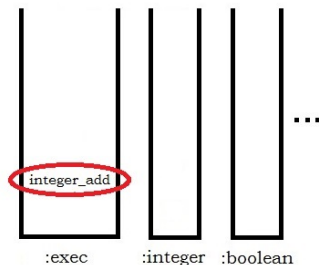


Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```



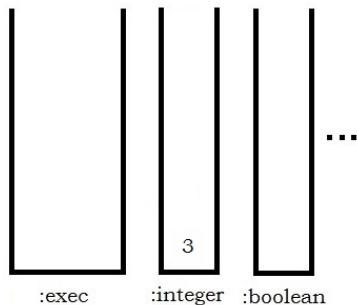
$2 + 1 = 3$

Stack-based genetic programming

Data-stacks are used for managing input and output of operations.

Programs are represented as linear sequences of literals and instructions. Below is an example of a simple Push program:

```
(1 2 integer_add)
```



Outline

- 1 Background
- 2 Hyper-heuristics
- 3 Genetic Programming Variants
- 4 Autoconstruction**
 - What is it?
 - AutoDoG
 - Evolution is evolving!
 - Results
- 5 Summary

Outline

- 1 Background
- 2 Hyper-heuristics
- 3 Genetic Programming Variants
- 4 Autoconstruction
- 5 Summary**