Genetic Programming

Nature Inspired Computing

Advantages of Genetic Programming

- Allows the automatic creation of programs
- Evolves programs that generate solutions instead of the solutions themselves
- More generic
- More powerful

Disadvantages of Genetic Programming

- Needs more CPU time for training
  - Evaluating programs typically takes longer than evaluating solutions
  - Programs need to be tested in as many situations as possible
- It is often hard to understand the programs that were generated
- Sometimes, it is necessary to use simplifiers in order to be able to understand the result

Representations

Trees A tree represents a program where branches represent non-terminals and leaves represent terminals. The number of sub-trees of a branch is the arity of the function

Linear Similar to machine code. Each instruction uses registers for its arguments and storing the result

Graphs The program is represented by a graph. Data is usually passed using the stack. The execution flow uses the edges of the graph

Advantages of Tree Representation

- Intuitive structure for representing programs
- Very similar to Lisp or Syntax Trees
- Very easy to manipulate by genetic operators
- Easy to visualize

Terminals

- Functions of zero arity
- They represent:
  1. The program inputs
  2. The constants (Random Ephemeral Constants) that are numbers that are generated when GP is initialized and used by all the individuals
  3. Functions with no parameters but that produce side effects

Functions

Booleans	And, Or, Not, Xor
Arithmetic	$+$, -, $\times$, $/$
Transcending	$\sin x$, $\cos x$, $\log x$, e^x
Attribution	(Assign a 1), (Read a)
Conditionals	If Then Else
Composing functions	(prog2 fun1 fun2)
Others	read sensor, turn left, turn right, move ahead

Initialization

Full Only non-terminals are used until the maximum depth of the tree is reached. At that time, only terminals are used.

Grow When generating a node, both terminals and non-terminals are used. This generates irregular trees

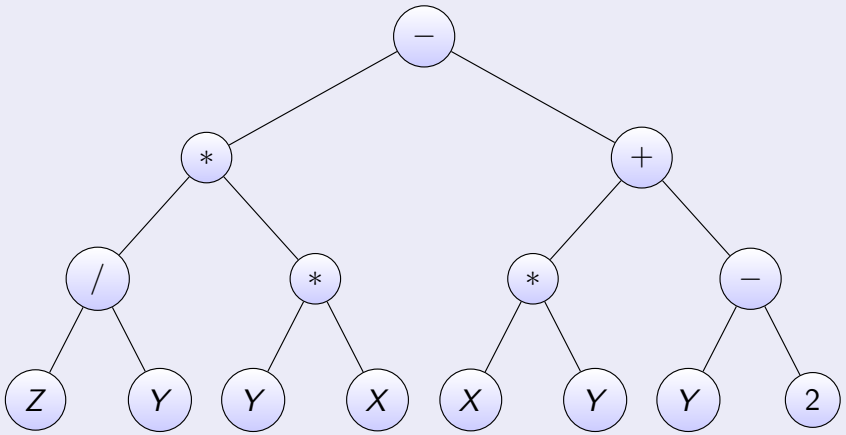
Ramped half and half Method that combines **Full** and **Grow**

Ramped half and half

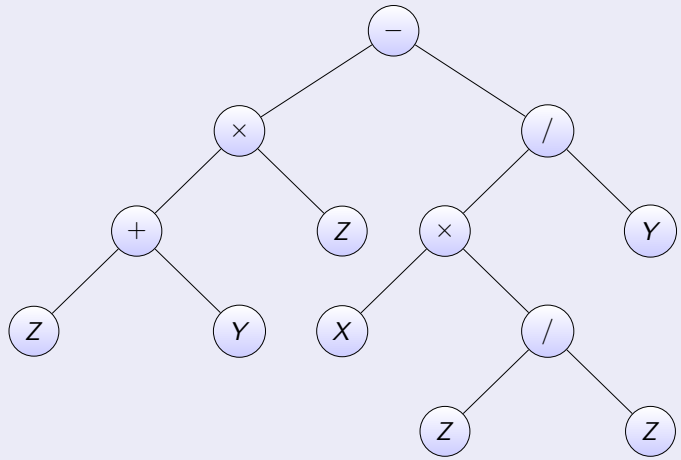
1. The same number of trees is generated for all depths ranging from 2 until the maximum depth
2. For each depth, half of the trees are generated using the *Full* technque while the other half uses the *Grow* technique
3. The same number of trees is generated for all depths ranging from 2 until the maximum depth

Initialization types

Full



Grow

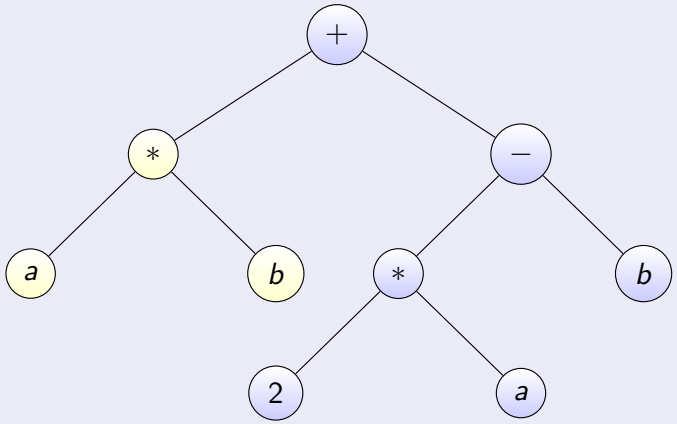


Crossover

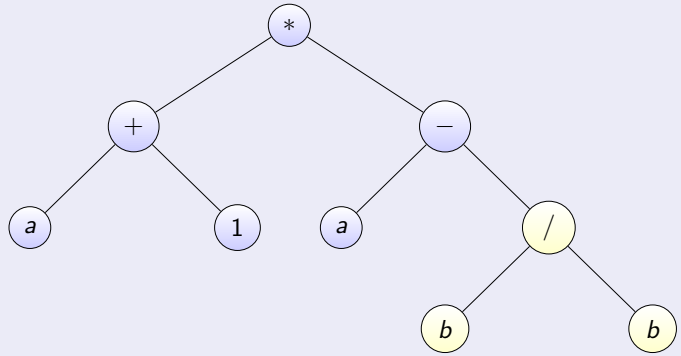
- Two parents are selected for exchanging genetic material
- A subtree is selected at random in each parent
- The subtrees are exchanged between parents
- The exchange is only performed if it doesn't violate the maximum depth constraint
- One of the advantages of crossover between trees is that it can generate different offspring even when both parents have the same information

Crossover

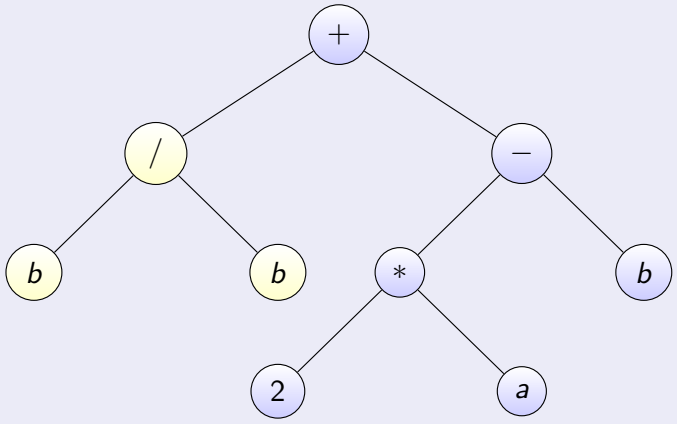
Parent 1



Parent 2



Child 1



Child 2