Cairo University
Faculty of Computers & Artificial Intelligence
Department of Computer Science
Soft Computing Course - 2023

Lab 3 - Genetic Programming

What is genetic programming?

Genetic programming offers a solution through the evolution of computer programs by methods of natural selection. It is an extension of the Genetic Algorithm. It searches the space of possible computer programs for a program that is highly fit for solving the problem at hand.

In genetic programming, each individual in the population represents a computer program and the goal is to generate a computer program that solves the problem at hand.

Genetic programming steps:

1. Generate an initial population of random compositions of functions
2. Execute each program in the population and assign it a fitness value according to how well it solves the problem.
3. Select individuals from the current population for genetic operators to be performed on. Most common selection criteria is tournament selection.
4. Create a new population of computer programs by applying **one** of these genetic operators:
    a. Create new computer programs by mutation.
    b. Create new computer programs by crossover
5. Replace the old generation with the new generation, using generational replacement
6. The best computer program that appeared in any generation, the best-so-far solution, is designated the result of genetic programming.

Just like in GAs, in genetic programming the stages are initialization, fitness evaluation, selection, crossover or mutation, and replacement.

Note:

*Computer programs in genetic programming are not only code, they may be **arithmetic operations**, **logical operations**, or **code/pseudocode**.*

Tree-based representation:

Individuals in genetic programming are represented as trees. A prominent difference between genetic algorithms and genetic programming is that genetic programming individuals have a non-linear structure (trees) and have variable size, i.e. a tree can have any number of nodes.

Since each individual represents a separate computer program, think of individuals as the parse tree of that program.

Functions and Terminal Sets:

The terminal and function sets are important components of genetic programming. The terminal and function sets are the alphabet of the programs to be made. The terminal set consists of the variables and constants of the programs. The function set consists of the functions of the program.

In an example of a computer program that performs arithmetic operations, a potential terminal set would be {X, Y, integers}, where X, Y are variables and integers are any integer. A potential function set would be {*, +, /, -} which are the set of functions (operations) that can be performed on the terminals.

In the tree that represents individuals, elements of the terminal set are allowed as leaves, while symbols from the function set are internal nodes.

Genetic Programming Example:

Let's look at an example of symbolic regression using genetic programming.

Problem Definition: Symbolic Regression is like a treasure hunt for the perfect mathematical equation to describe a dataset. Imagine having a bunch of data points, and you want to find the best way to represent them mathematically.

Objective:

Let's say we were given a bunch of data points that were generated using the quadratic polynomial function x^2+x+1 by substituting with x values between –1 to +1. We want to use genetic programming to automatically create a computer program whose output is equal to the values of that quadratic polynomial. i.e. generate a program of the function x^2+x+1.

Givens:

The function in question has one independent variable *x*.
The constant terms in the functions are in the range from –5.0 to +5.0.
A set of data points that we want to generate a function to represent mathematically.
The set = {(1,3), (2,7), (3,13), (4,21), (5,31)}

Requirements:

1. Subtree mutation
2. Subtree crossover
3. Generational replacement
4. Tournament Selection with tournament size = 2
5. Perform mutation on 50% of the selections and crossover on 50% of the selections.

Preparatory Steps:

1. Determine terminals set. Since we are generating a mathematical function of one independent variable *x*, and the constant terms range from –5.0 to +5.0, a good terminal set would be T = {*x,* R} where R denotes numerical terminals from –5.0 to +5.0).

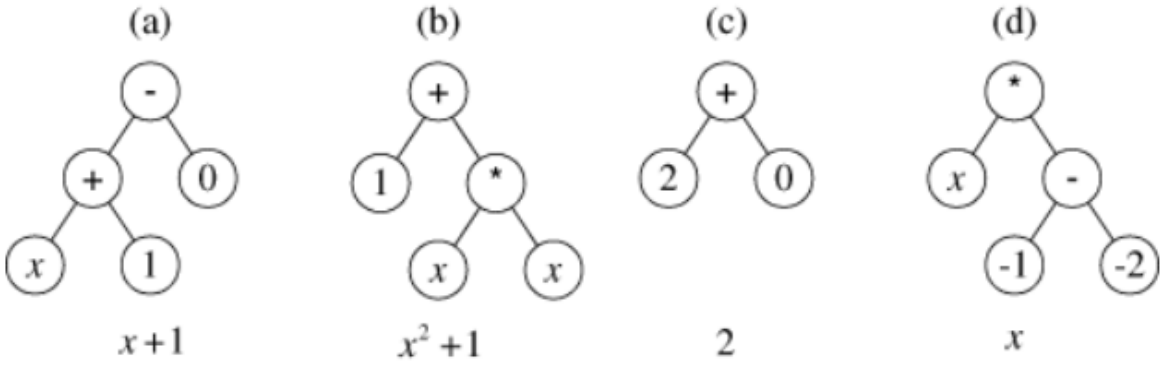
2. Determine the functions set. Since we are generating a mathematical function, one possible choice for the function set consists of the four ordinary arithmetic functions of addition, subtraction, multiplication, and protected division. This choice is reasonable because mathematical expressions typically include these functions. Thus, the function set, F, for this problem is F = {+, -, *, /}.

3. Determine the fitness function. In that problem, since the objective is to generate the best function to represent a given set of data points mathematically, a good measure of fitness is running the generated program and comparing the output to the data points by measuring the error between them.

Genetic Programming Steps:

1. Initialize the population.

Assume a population size of 4, create a population of four individual computer programs.



2. Fitness Evaluation.

We can evaluate the fitness by substituting with the values of x from the given data points in each generated program and comparing the output with the values of y. Measure the error using the sum of squared errors.

{(1,3), (2,7), (3,13), (4,21), (5,31)}

Tree	Output (Marked in red)	Fitness
(a) x+1	(1,2)(2,3)(3,4)(4,5),(5,6)	$(2-3)^2+(3-7)^2+(4-13)^2+(5-21)^2+(6-31)^2 = 979$
(b) x^2+1	(1,2)(2,5)(3,10)(4,17),(5,26)	$(2-3)^2+(5-7)^2+(10-13)^2+(17-21)^2+(26-31)^2 = 664$
(c) *2*	(1,2)(2,2)(3,2)(4,2),(5,2)	$(2-3)^2+(2-7)^2+(2-13)^2+(2-21)^2+(2-31)^2 = 1349$
(d) x	(1,1)(2,2)(3,3)(4,4),(5,5)	$(1-3)^2+(2-7)^2+(3-13)^2+(4-21)^2+(5-31)^2 = 1094$

3. Selection

Perform tournament selection to select 4 individuals to perform genetic operations on.

Choose two individuals (tournament size = 2) from the population at random, and the individual with the better fitness is selected, until 4 individuals are selected.

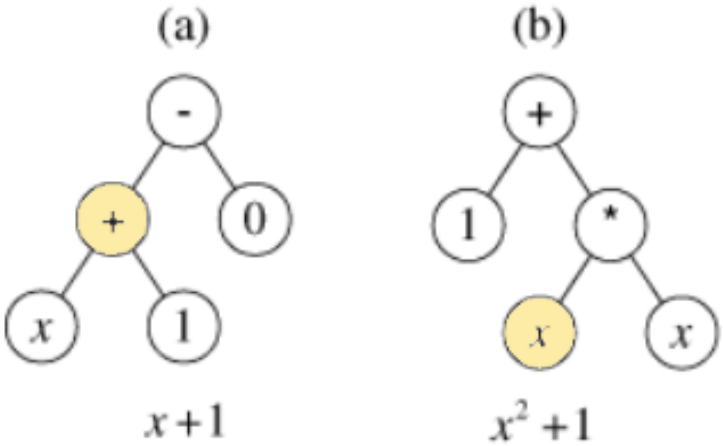
- First tournament between (a) and (c): (a) won, therefore it is selected.
- Second tournament between (b) and (d): (b) won, therefore it is selected.
- Third tournament between (c) and (d): (d) won, therefore it is selected.
- Fourth tournament between (a) and (d): (a) won, therefore it is selected.

The four individuals selected are:
(a), (b), (d), (a)

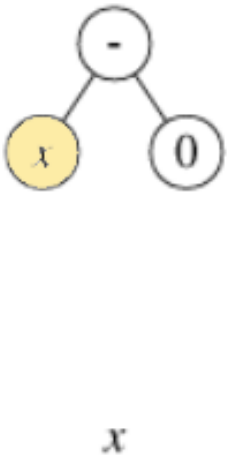
4. Crossover

Given that crossover is performed on 50% of the selections, we will perform mutation on (a) and (b).

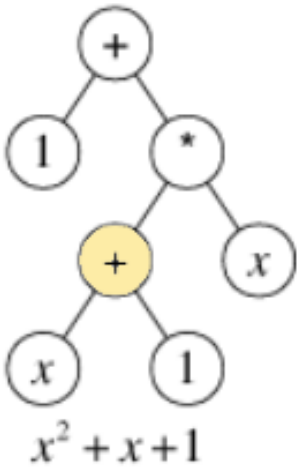
The shaded nodes were chosen randomly to perform crossover on.



Crossover is performed to produce the following two trees:



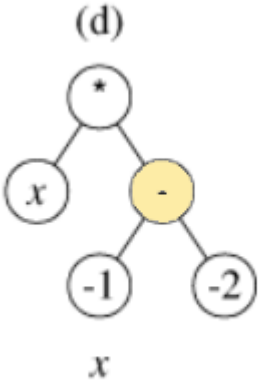
x



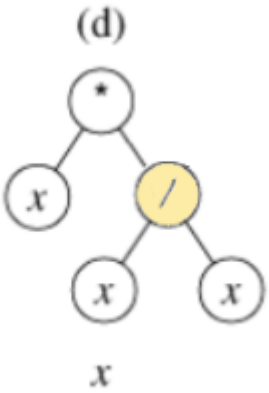
x^2+x+1

5. Mutation

Mutation is performed on (d) to produce the following:

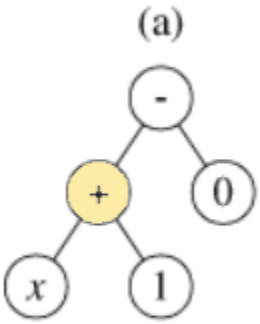


x

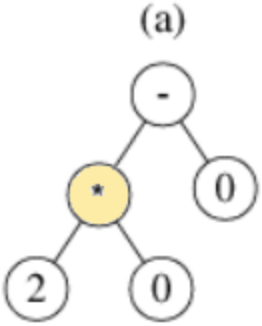


x

Mutation performed on (a) will produce the following:



$x+1$

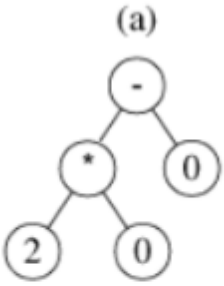


0

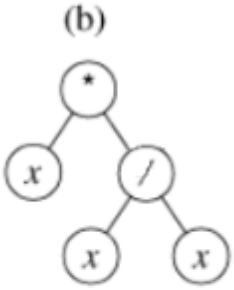
6. Replacement

Use generational replacement to replace the individuals produced from crossover and mutation to replace the older generation.

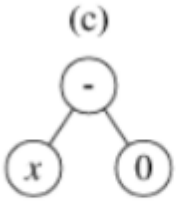
The new generation consists of:



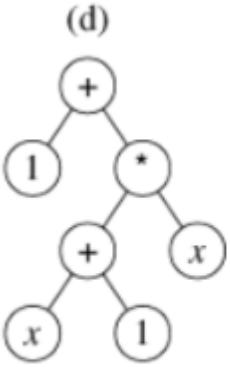
0



x



x



x^2+x+1

Coincidentally, we found the solution to the problem on the first iteration, which is individual (d)!

References:

https://www.genetic-programming.com/gpflowchart.html

https://www.genetic-programming.com/gppreparatory.html

https://www.genetic-programming.com/gpquadraticexample.html