Genetic Algorithms

Dr. Mahmoud Nabil Mahmoud
mnmahmoud@ncat.edu

North Carolina A & T State University

September 6, 2021

Outline

Motivation

- Generally GA are search algorithms
- How can computer learn a program / algorithm to solve problem?
  - Idea of evolving programs.
  - Learning from data.
- Can computer generate a digital logic circuit given *input/output* specification.

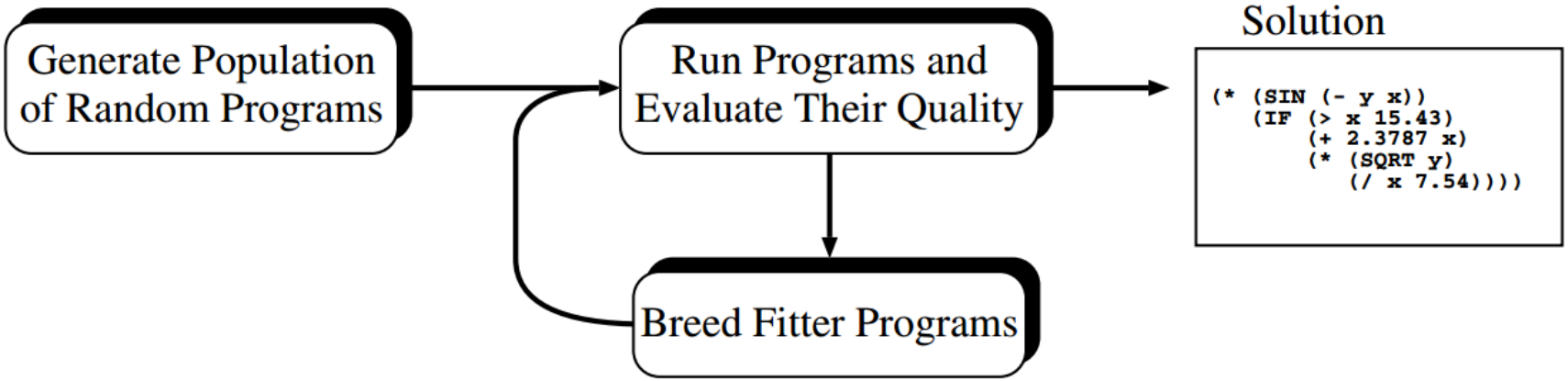
Genetic Programming

Genetic programming (GP)

Genetic programming is a domain-independent method that genetically evolve a population of computer programs to solve a problem.

- Individuals in the population are computer programs.
- Generation by generation GP *iteratively* transforms populations of programs into other populations of programs
- GP constructs new programs by applying *genetic operations*
  - Mutation of programs
  - Recombination of programs.

GP Main Loop


Generate Population of Random Programs
Run Programs and Evaluate Their Quality
Breed Fitter Programs
Solution
(* (SIN (- y x))
(IF (> x 15.43)
(+ 2.3787 x)
(* (SQRT y)
(/ x 7.54))))

Outline

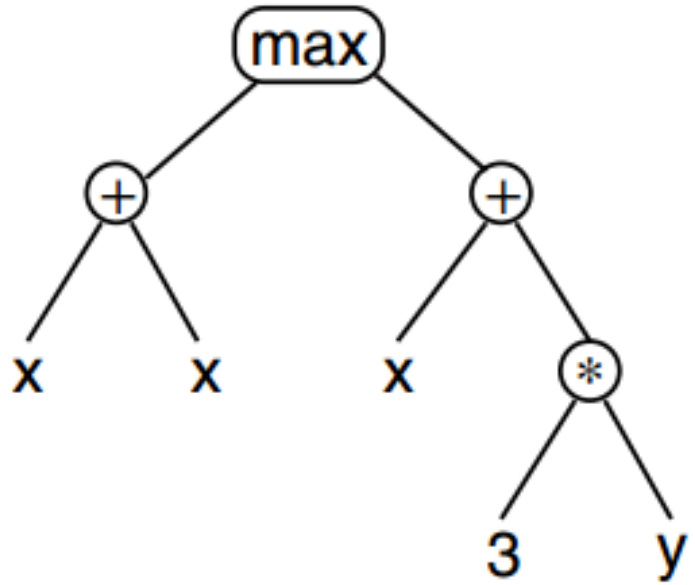
Representation

- In GP programs are usually expressed as *syntax trees* rather than as lines of code.
- Any computer program can represented as *syntax trees*.

Ex.

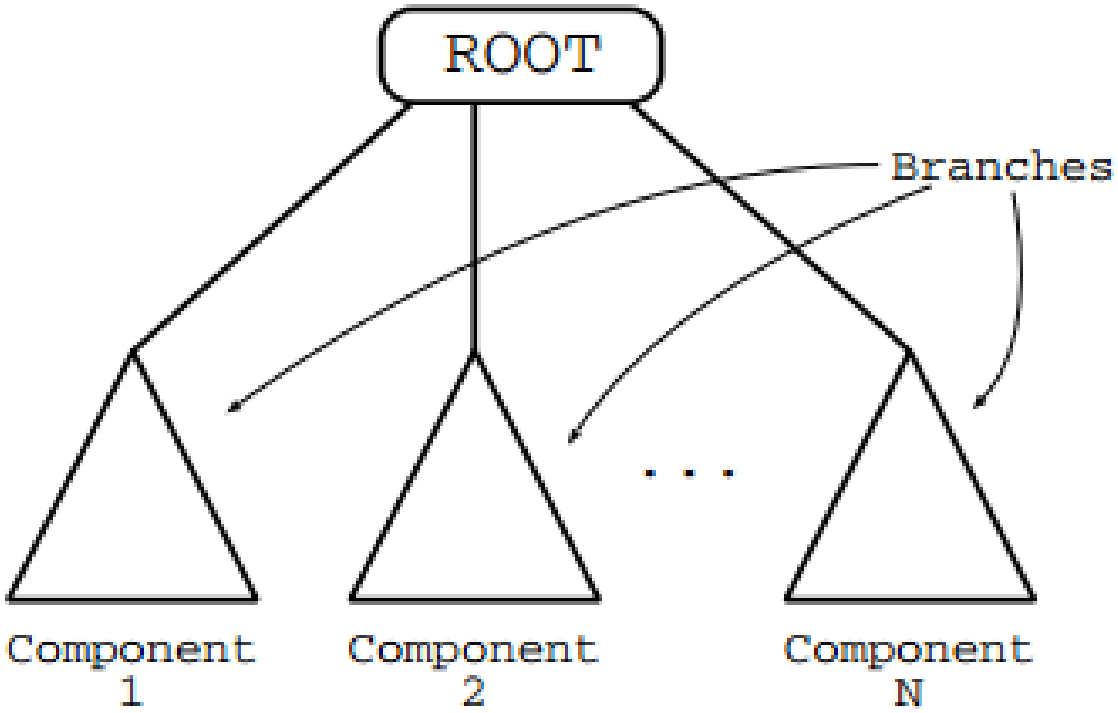
`max(x*x,x+3*y)`

- `(x, y, and 3)` called *terminals* in GP
- `(+, *, and max)` are internal nodes (typically called *functions*).



Representation

- In more advanced forms of GP, programs can be composed of multiple components (say, subroutines).
- In this case the representation used in GP is a set of trees (one for each component) grouped together under a special root node that acts as glue.

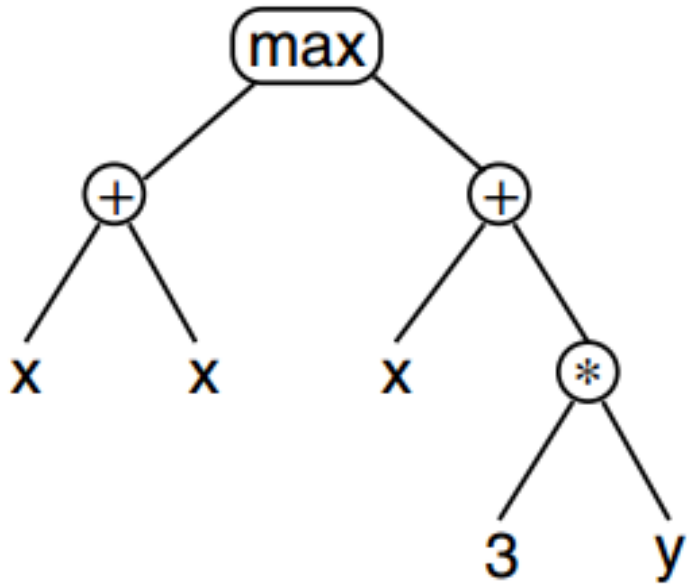


Representation

- It is common in the GP literature to represent expressions in a *prefix notation*.

`max(x*x,x+3*y)`

- prefix notation is
  `(max (* x x) (+ x (* 3 y)))`
- This notation often makes it easier to see the relationship between (sub)expressions and their corresponding (sub)trees.



Terminal Set

Terminal Set

They are the leaves of the *syntax tree.*

- programs external inputs, typically taking the form of named variables (say x, y);
- functions with no arguments
  - function such as `rand()`, `dist_to_wall()`

  that returns random
- constants, which can either be pre-specified or randomly generated.

Terminal Set	
Kind of Primitive	*Example(s)*
Variables	`x`, `y`
Constant values	`3`, `0.45`
0-arity functions	`rand`, `go_left`

Function Set

Function Set

They are the internal nodes of the *syntax tree.*

- Typically driven by the nature of the problem domain
- if the goal is to synthesis an analogue electrical circuits the function set might include components such as transistors, capacitors, resistors, and so on.

Function Set	
Kind of Primitive	*Example(s)*
Arithmetic	`+`, `*`, `/`
Mathematical	`sin`, `cos`, `exp`
Boolean	`AND`, `OR`, `NOT`
Conditional	`IF-THEN-ELSE`
Looping	`FOR`, `REPEAT`
⋮	⋮

Function Set Properties

- **Closure:**
  - Type consistency: functions can get inconsistent types during evolution.
    - How to handle an if function
    - Automatic type conversion is a solution.
  - Evaluation safety: commonly used functions can fail in various ways.
    - division, overflow, underflow, etc.
    - protected functions need to be defined
- **Sufficiency:** The primitives in the primitive set are capable of expressing the solutions to the problem.
  Ex.
  - `AND, OR, NOT, x1, x2, ..., xN`, is always sufficient for Boolean function induction problems.

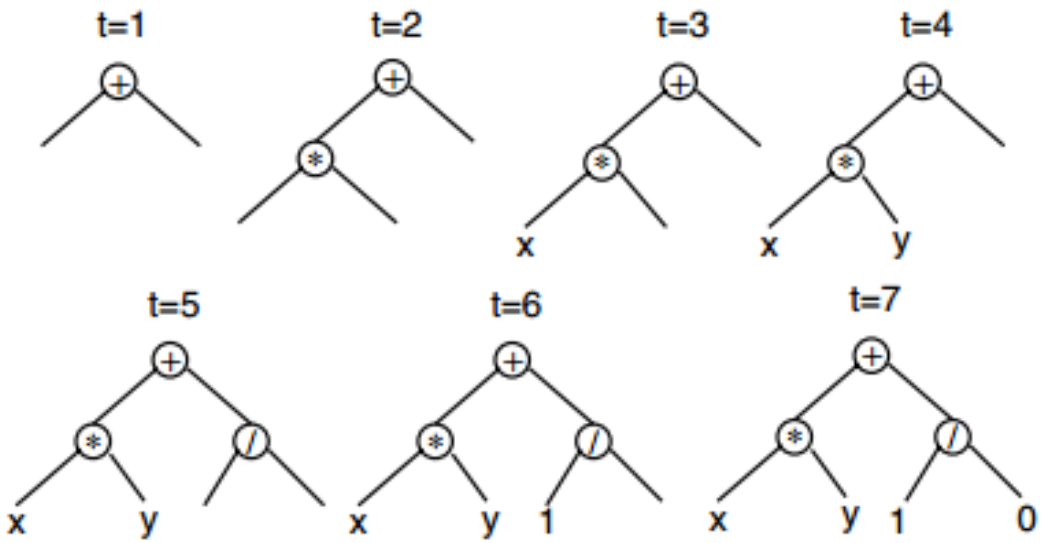
Initialization

In GP the individuals in the initial population are randomly generated. The earliest methods are

- Full method.
- Grow method.
- Ramped half-and-half.

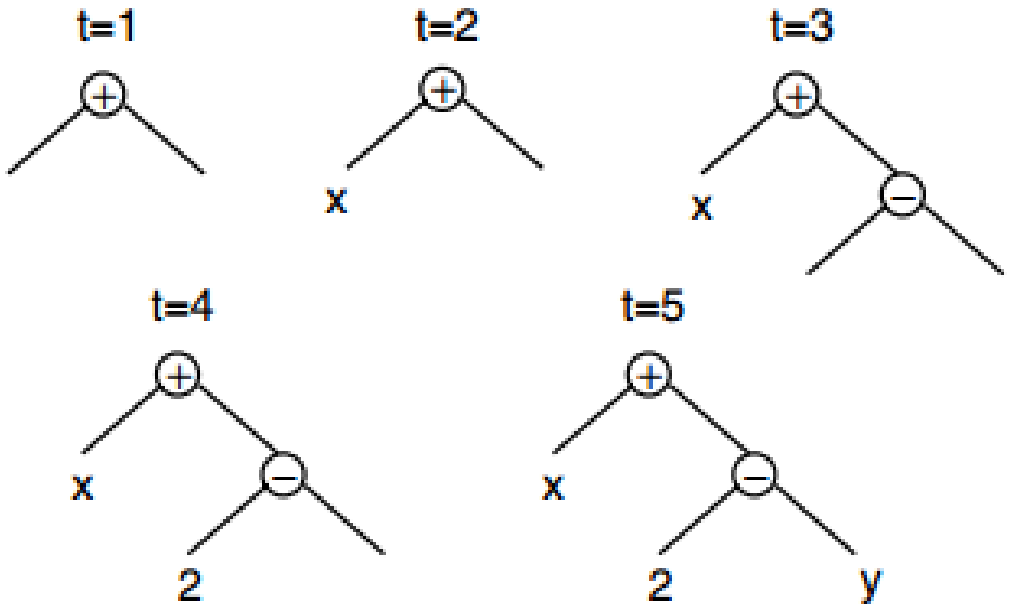
Full method

- The initial individuals are generated subject to a *pre-established maximum depth*.
- Full (COMPLETE) trees are generated.
- Nodes are taken at random from the *function set* until this maximum tree depth is reached, and beyond that depth only *terminals* can be chosen.



Grow method

- The initial individuals are generated subject to a *pre-established maximum depth*.
- Incomplete trees maybe generated.
- Nodes are taken at random from the *function set or terminal set*

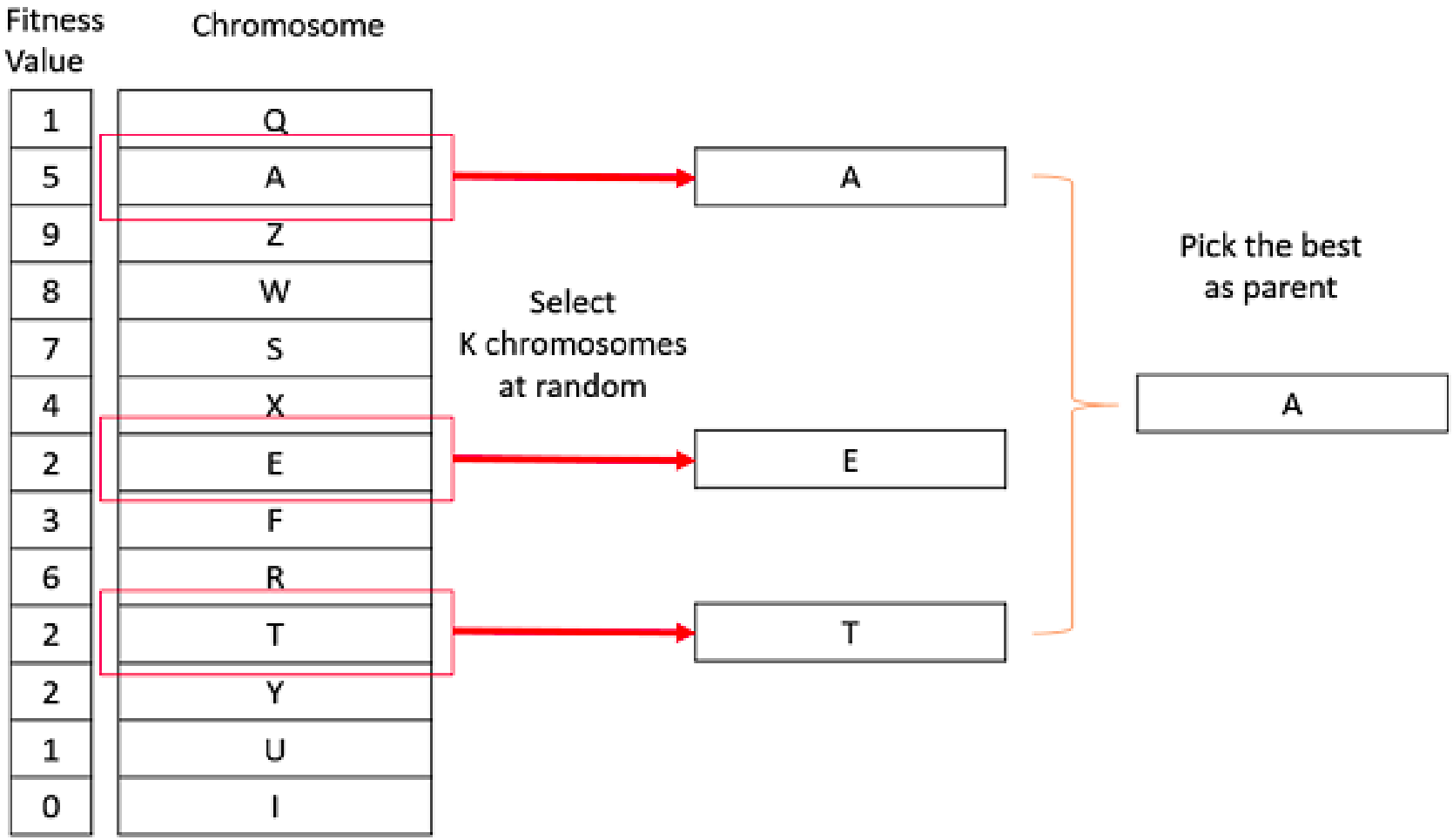


Notes

- The Full method generates trees of a specific size and shape.
- The Grow method allows for the creation of trees of varying size and shape.
- If, for example, one has significantly more terminals than functions, the Grow method will almost always generate very short trees regardless of the depth limit
- If number of functions >> number of terminals, then the Grow method will behave quite similarly to the Full method
- In ramped half-and half. Half the initial population is constructed using Full and half is constructed using Grow.

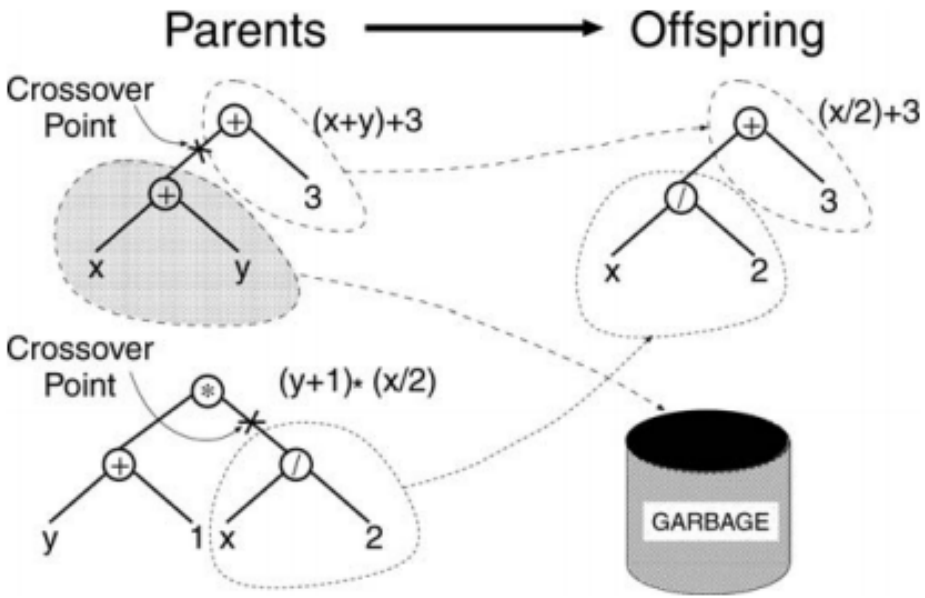
Selection

- The most commonly employed method for selecting individuals in GP is tournament selection.



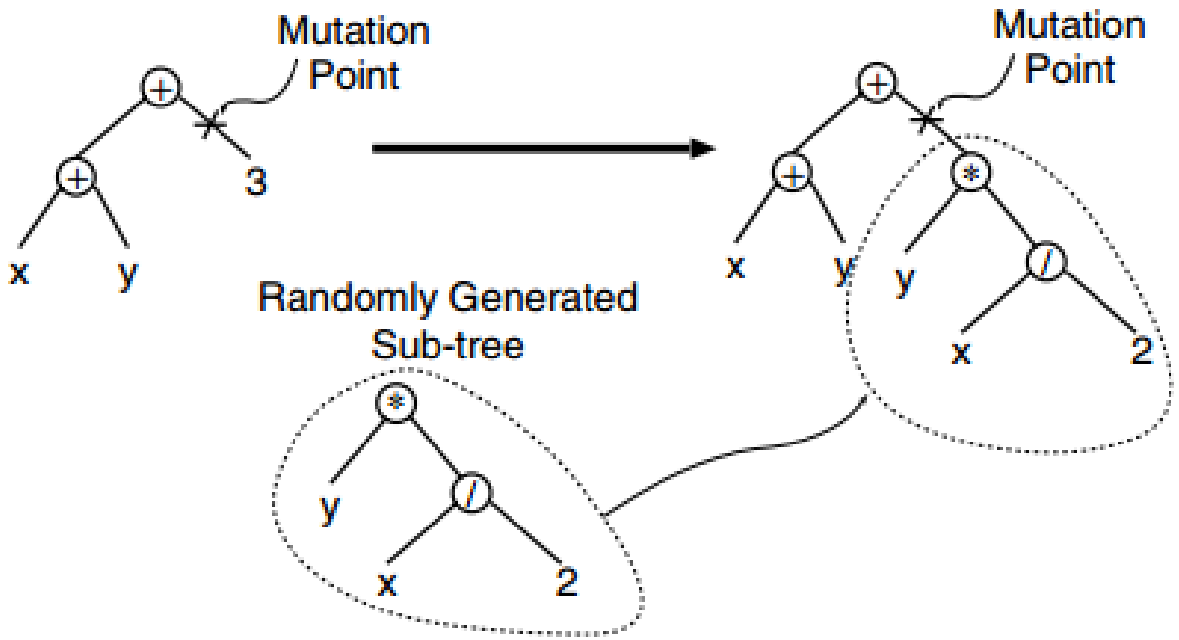
Recombination

- The most commonly used form of crossover is subtree crossover.
- Given two parents, subtree crossover randomly selects a crossover point in each parent tree. Then, it creates the offspring by replacing the sub-tree rooted at the crossover point in a copy of the first parent with a copy of the sub-tree rooted at the crossover point in the second parent



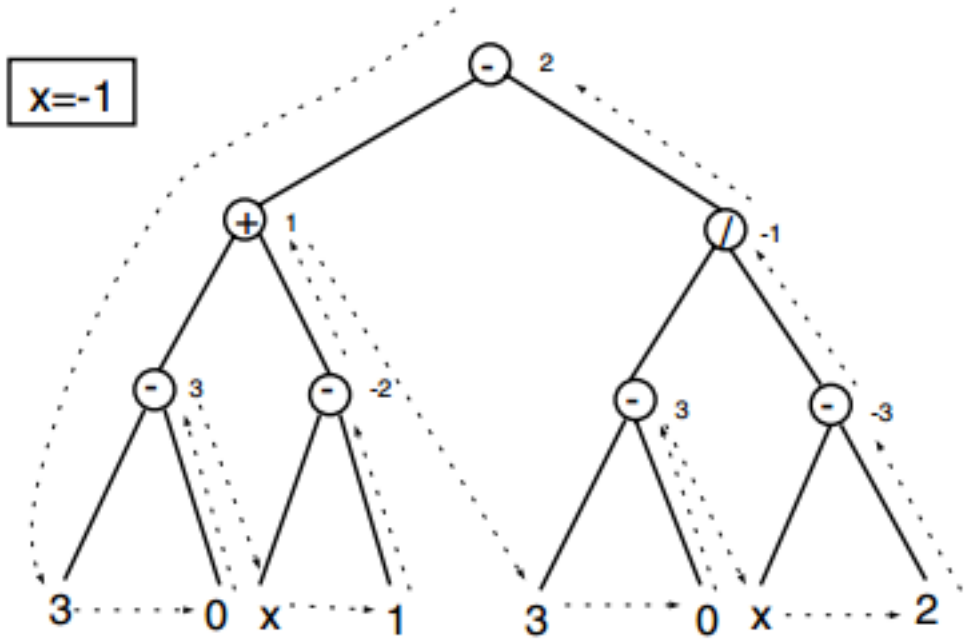
Mutation

The most commonly used form of mutation in GP (which we will call subtree mutation) randomly selects a mutation point in a tree and substitutes the sub-tree rooted there with a randomly generated sub-tree.



Fitness function

- Fitness evaluation normally requires executing all the programs in the population, typically multiple times.
  - Compilation is time consuming.
  - Usually needs an interpreter.



Example interpretation of a syntax tree (the terminal x is a variable has a value of 1)

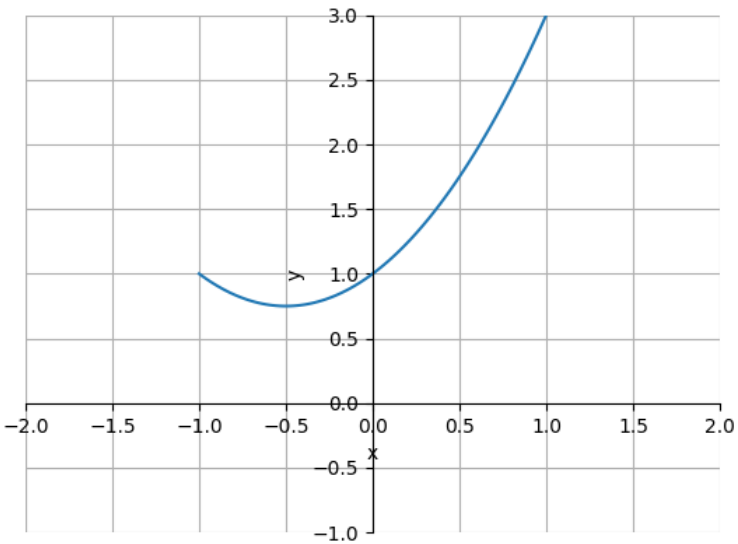
Parameters and Termination

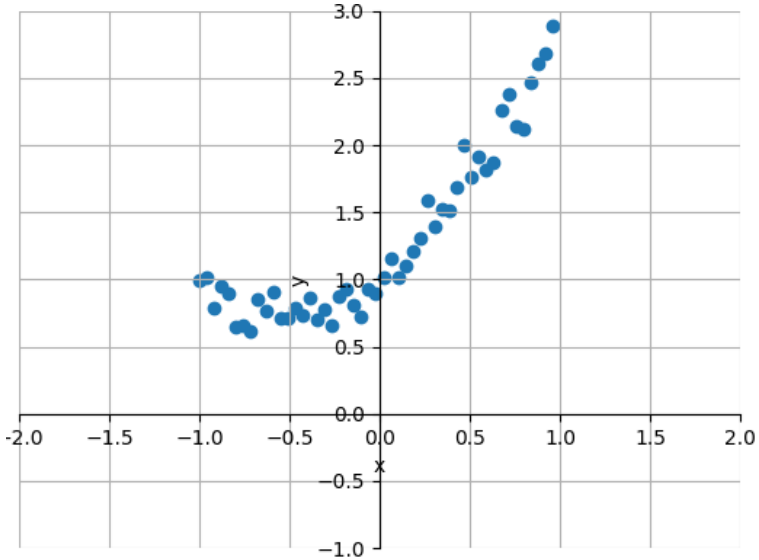
- The termination criterion may include a maximum number of generations to be run as well as a problem specific success predicate.
- Parameter such as population size is set depending on the problem.

Outline

Example

- Suppose we would like to evolve an program expression whose values match those of the quadratic polynomial $x^2 + x + 1$ in the in the range $[1, +1]$
- Known as `Symbolic Regression` or `system identification`





Terminal and Functions

- **Terminal Set.**
  - One independent variable x.
  - Random constants, drawn from some reasonable range, say from 5.0 to +5.0

$$T = \{x, \Re\}$$

  - **Function Set.** In this example, we will restrict ourselves to the simple function set

$$F = \{+, -, *, \%\}$$

Fitness measure

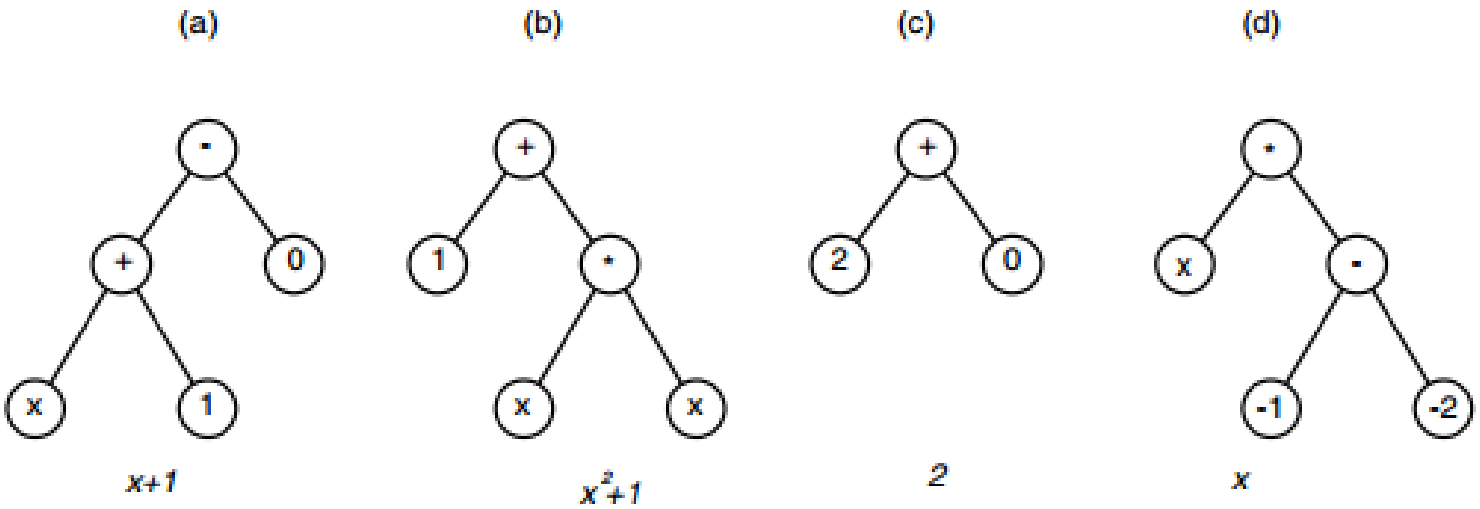
- The sum of square error difference between the target symbolic expression and the predict value by our program.
- We will measure the error at specific values we will measure the errors for x = 1.0, 0.9, , 0.9, 1.0

$$\text{fitness} = \sum_{i=1}^{i=n} (y - y') * *2$$

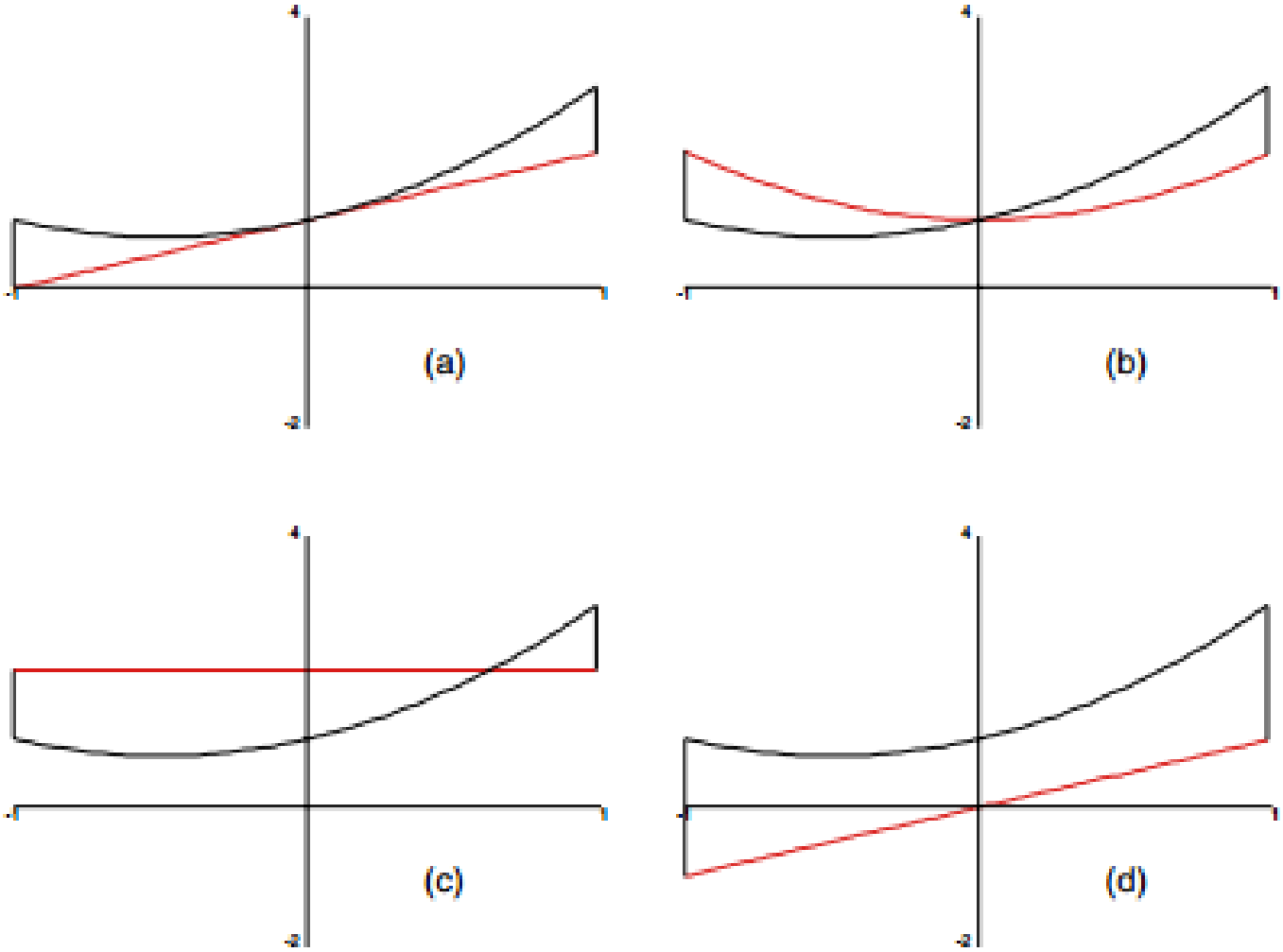
- y : it is the true or measured value by our symbolic expression
- y' : predicted or output value by our genetic program.
- n : number of measured values

Step-by-Step Sample Run

- Operators such as crossover, mutation, and selection should be define as described before.
- Initialization: We can assume the following population



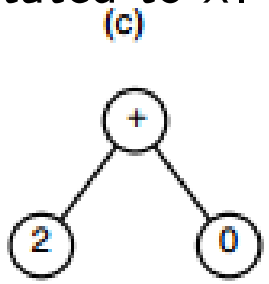
Fitness Evaluation



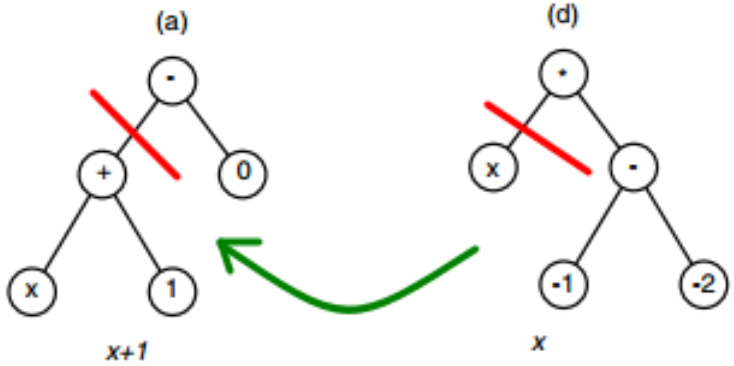
The square errors are 7.7, 11.0, 17.98 and 28.7 for individuals (a) through (d), respectively.

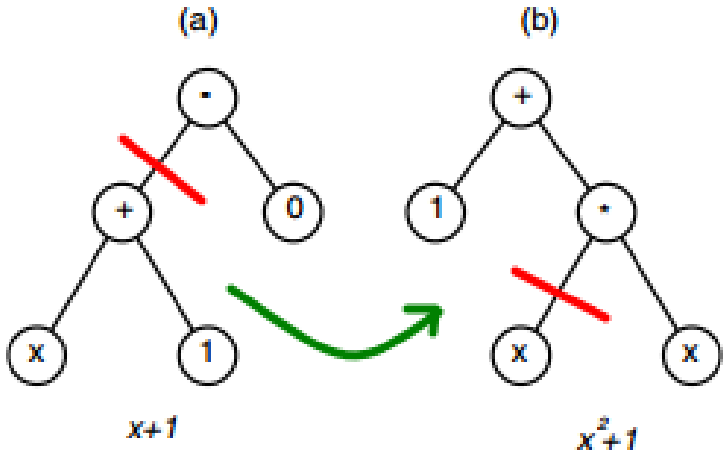
Selection, Crossover and Mutation

- Let us suppose (a) "best fit", is selected for reproduction to the next generation without alteration.
- The third best (c) individual in the population is also selected with mutation.
  - the constant terminal 2 is mutated to x.

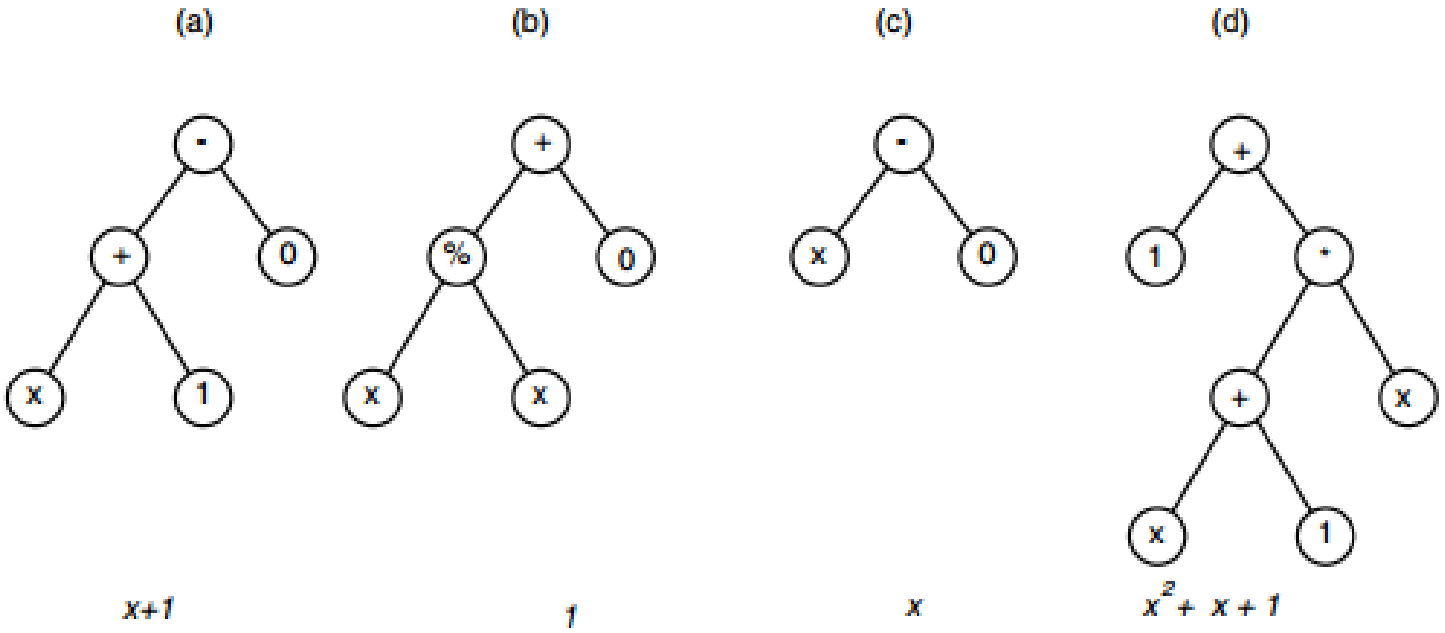


- Crossover operation is used to generate our final two individual.





Second Generation



- Individual (d) is the best solution with square error equal zero
- GP termination criterion for the run is satisfied and the run is automatically terminated.
- Note that, this example is designed specifically dor demonstration.

Applications

Koza (GP Pioneer) proposed to shift the attention from the notion of intelligence to the notion of human competitiveness.

Since 2004, a competition is held annually at ACMs Genetic and Evolutionary Computation Conference (termed the Human-Competitive awards

- Game players for Backgammon, Robocode and Chess endgame.
- Financial Trading, Time Series Prediction and Economic Modeling
- Medicine, Biology and Bioinformatics
- Artistic
- Image and Signal Processing

References

- Goldenberg, D.E., 1989. Genetic algorithms in search, optimization and machine learning.
- Michalewicz, Z., 2013. Genetic algorithms + data structures= evolution programs. Springer Science & Business Media

Questions