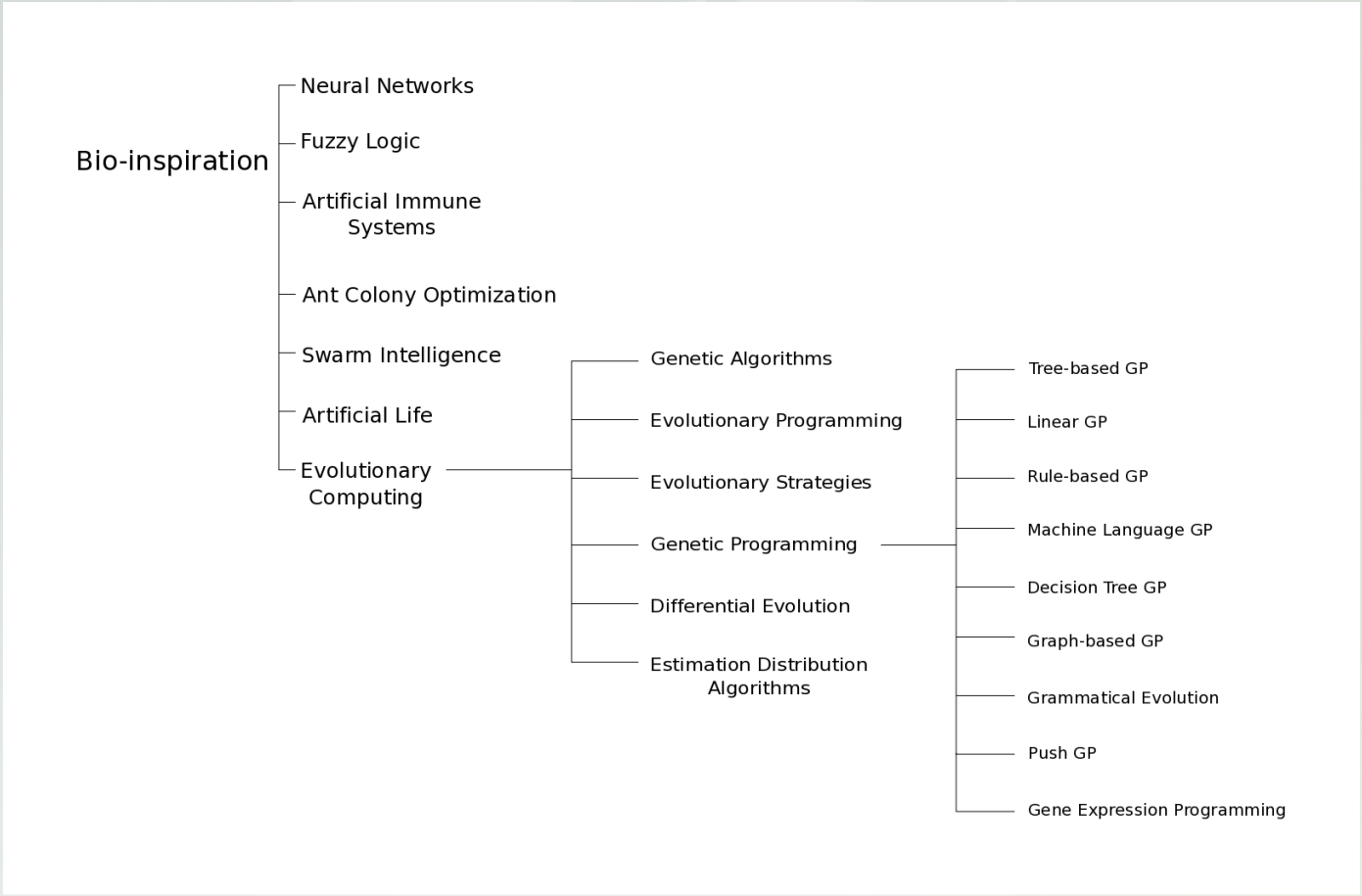
CSE/ECE 848 Introduction to Evolutionary Computation

Module 3 - Lecture 11 - Part 1

Genetic Programming - Introduction

Wolfgang Banzhaf, CSE
John R. Koza Chair in Genetic Programming

The "Tree" of Bio-inspired Computing



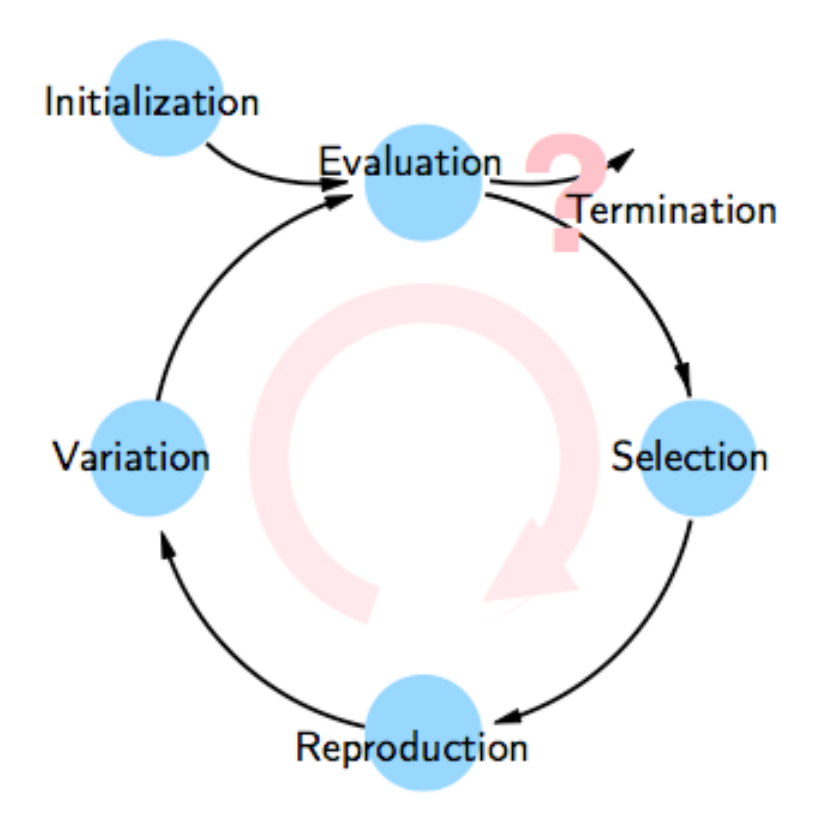
Similarities of Genetic Programming with other EA Methods

Genetic Programming
Similarities with other EAs

Central Idea

1. Multitude of computer programs ("population")
2. Computer varies behavior ("mutation, recombination")
3. Programmer determines selection criteria for better variants of behavior ("fitness")
4. Cumulative selection biases future variants toward preferred behavior ("selection")

Cumulative Selection
Similarities

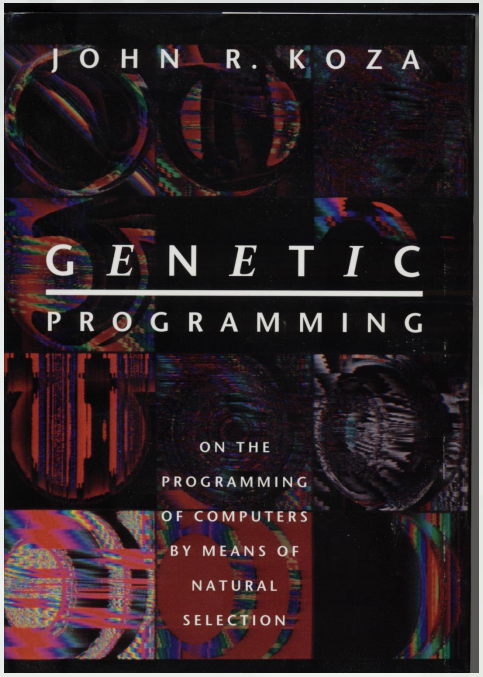


Predecessors of GP

- 1978: Holland & Reitman: If-then rules under evolution
- 1980: Smith: Variable-length if-then rule individuals
- 1981: Forsyth: Logical rule system with training cases
- 1985: Cramer: Trees and linear sequences for program evolution; subtree crossover
- 1986/87: Hicklin, Fujiki & Dickinson: LISP & PROLOG
- 1987 Dickmanns et al, Assembler
- 1989: Koza: Parse trees in LISP, subtree crossover

The Breakthrough

1992: John Koza publishes

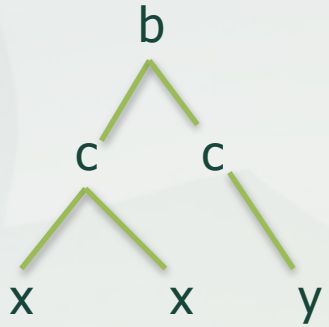


Differences to other EA Methods

Difference 1: Variable length

1	2	3	4	5	6

- GAs typically represent solutions as fixed length binary strings, with each feature on the string being a part of the overall solution
- A GP is a variable length solution, e.g., in the form of a tree



Variable length - contd.

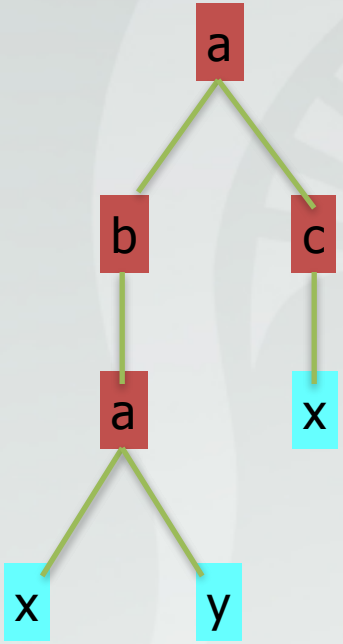
Variable length is a major feature of GP

- Previous work had been done on variable-length GA but its advantages were not clear
- Variable-length strings affect a number of important aspects of GP, in particular schema theory and related aspects
- Plus it is a pain to code!

Difference 2: Uses of the "answer"

- In a GA, the individual elements of the string are portions of the solution to the problem
- In a GP, the tree (typically) represents a program that can be used to solve a problem (it is not a solution, but a solver) or a structure that solves the problem
- Two kinds of nodes in a GP tree (program):
  - Functions (internal nodes) of some number of arguments (number of children is number of arguments)
  - Terminals (leaf/external nodes)

Functions and Terminals



functions: {a, b, c}

terminals: {x, y}

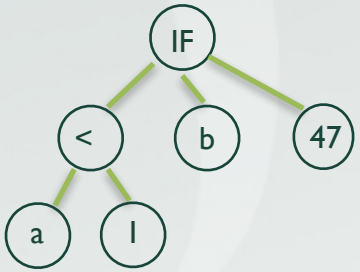
Search space of Programs

GA : $\{0, 1\}^m$ (0 1 1 1 1 0)

ES : R^m (2.3 ; 3.4 ; 0.2)

GP : Programming language constructs

Example 1: Tree, functional language



(IF (< a l) b 47)

Example 2: Linear, register-based language

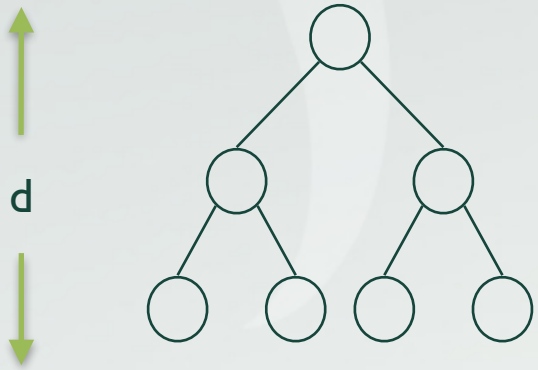
$R_0 := R_1 + R_0$

$R_2 := R_1 * R_0$

$R_0 := R_1 << R_2$

Search space

- Typical GA-Search space $\approx 10^{100}$
- Typical GP-Search space $\approx 10^{100,000}$



$F=\{AND, OR, NOR, NAND\}$

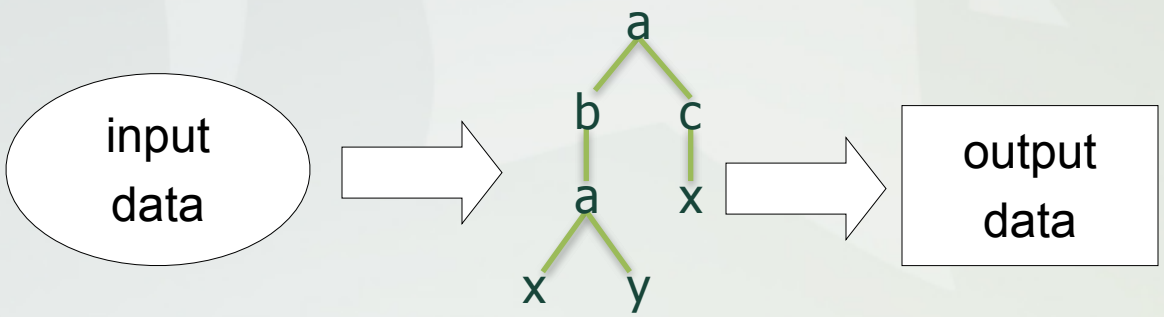
$T=\{x_1, x_2, x_3\}$

$N(6) \approx 10^{69}$

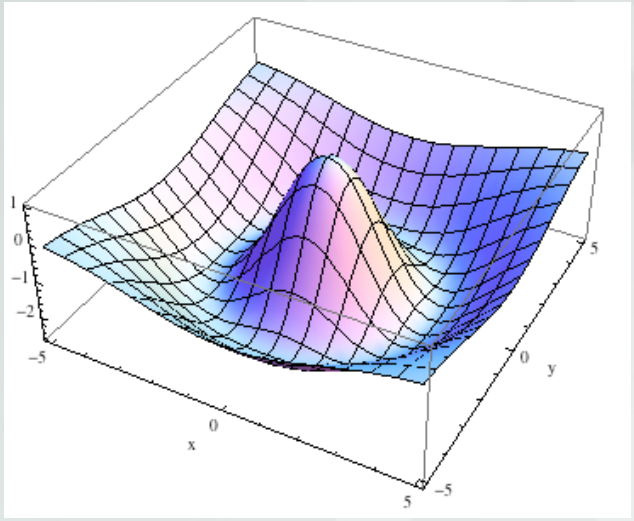
$N(17) \approx 10^{143,735}$

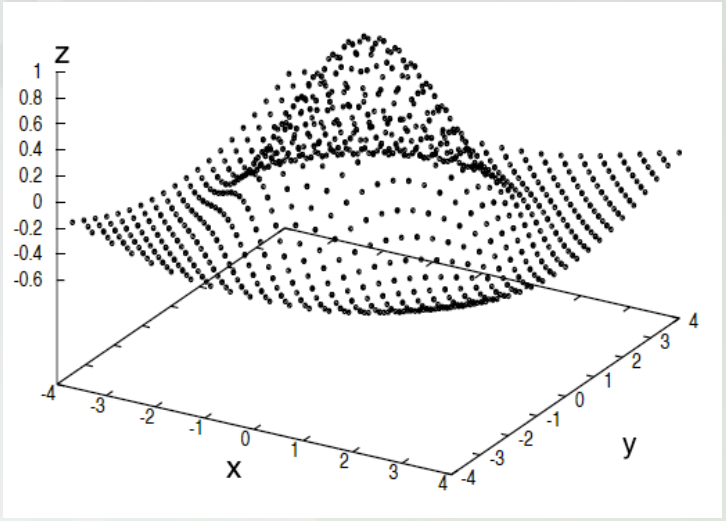
Difference 3: Evaluation

- In a GA, a solution X can (typically) be evaluated directly by the function f(X)
- Since a GP solution is a program, it must be run as a program on a set of input data and the set of outputs must be evaluated to determine the overall performance
- It is a model of the data



GA vs. GP
Differences

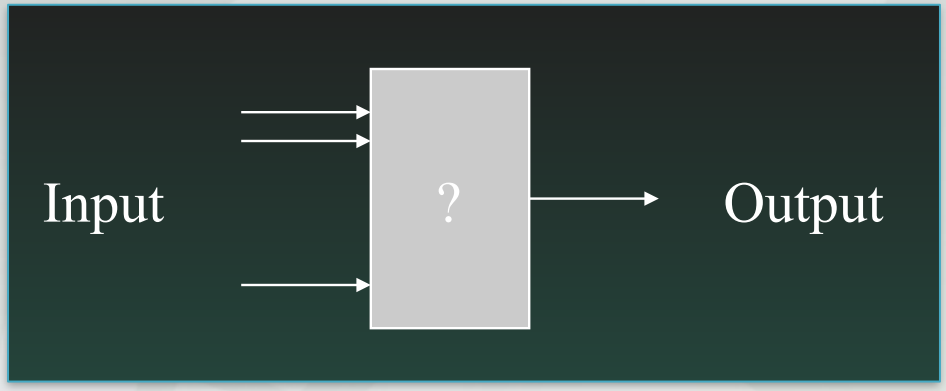




Testing vs Training

- This also means that to judge the program, to show that it truly learned, as opposed to memorize the answer, you need to apply both training and testing
- Like other ML approaches
  - Train on typical example data
  - Test on similar data that the system wasn't trained on.
  - Generalization is key!
  - Often used: Three data subsets: Training, Validation & Test

Fitness Function in GP

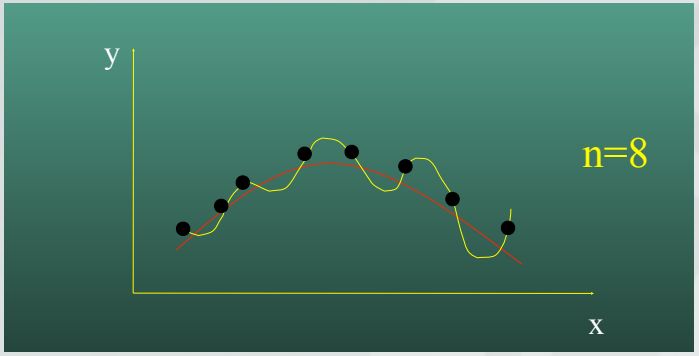


- Comparison of behaviour of program with target behaviour
- Many test (fitness) cases, fitness f is some measure of errors or loss function, e.g. SSE (sum of square error), RMSE (root mean square error)
- $f : Prg \rightarrow R$
- For comparison: GA $f : B^n \rightarrow R$

Fitness Function Example: Regression

Given $x_i, y_i \in R \quad i \in N$

Fitness cases



Int: Prg × R → R

x y

$$f(\text{Prg}) = \sum_{i=1}^{n} \left(Int(\text{Prg}, x_i) - y_i \right)^2$$

Quadratic Error

$$f(\text{Prg}) = \sum_{i=1}^{n} \left| Int(\text{Prg}, x_i) - y_i \right|$$

Absolute Error

Note: Fitness is independent of representation

Solution Quality

Validation set (x_i', y_i') $i=1,\ldots,k$

$$f_v(\text{Prg}) = \sum_{i=1}^{k} \left(Int(\text{Prg}, x_i') - y_i' \right)^2$$

Validation function usually = Fitness function

How good is the solution on unseen data?