

Genetic Programming \subsetneq Genetic Algorithms

Genetic Programming

Authors: Edwin Camilo Cubides Garzón, Ph.D.(c)
eccubidesg@unal.edu.co

Jonatan Gómez Perdomo, Ph.D.
jgomezpe@unal.edu.co

Grupo de investigación en vida artificial – Research Group on Artificial Life – (Alife)
Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia

2nd Semester 2019



Outline

- 1 Contextualization
- 2 Evolutionary Algorithm
- 3 Genetic Programming
 - Representation of programs
 - Generation of initial populations
 - Selection strategy
 - Generation of offspring
 - Replacement strategy
 - Advantages and disadvantages



Soft Computing



¬Hard computing \Rightarrow imprecision, uncertainty,
low accuracy, partial truths



Outline

1 Contextualization

2 Evolutionary Algorithm

3 Genetic Programming

- Representation of programs
- Generation of initial populations
- Selection strategy
- Generation of offspring
- Replacement strategy
- Advantages and disadvantages



General Algorithm to Evolve Individuals

Algorithm 1 EVOLUTIONARY_ALGORITHM(f, μ , terminationCondition)

 $t = 0$ $P_0 = \text{INITPOPULATION}(\mu)$ evaluate(P_0, f)**while** (terminationCondition(t, P_t, f) = false) **do** $\text{newIndividuals} = \text{GENERATE}(P_t, f, \text{SELECT_FUNCTION})$ $P_{t+1} = \text{offspring} = \text{REPLACEMENT}(P_t, \text{newIndividuals}, f)$ evaluate(P_{t+1}, f) $t = t + 1$ **end while****return** P_t

The parameter SELECT_FUNCTION is a function used for selecting the parents to generate new individuals using the genetic operators.



Evolutionary Computation

$$\begin{array}{ccccccc}
 \text{EC} & = & \text{GA} & + & \text{ES} & + & \text{EP} \\
 \text{Evolutionary} & & \text{Genetic} & & \text{Evolutionary} & & \text{Evolutionary} \\
 \text{Computation} & & \text{Algorithms} & & \text{Strategies} & & \text{Programming} \\
 & & (\text{Holland, 75}) & & (\text{Rechenberger, 73}) & & (\text{Fogel, Owens,} \\
 & & & & & & \text{Walsh, 66})
 \end{array}$$


GP
 Genetic
 Programming
 (Jhon Koza, 1989)



Outline

- 1 Contextualization
- 2 Evolutionary Algorithm
- 3 Genetic Programming
 - Representation of programs
 - Generation of initial populations
 - Selection strategy
 - Generation of offspring
 - Replacement strategy
 - Advantages and disadvantages



Definition I

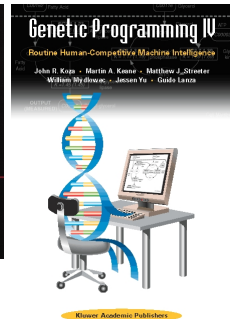
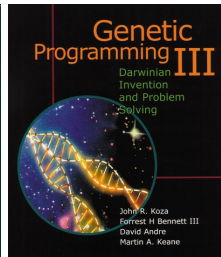
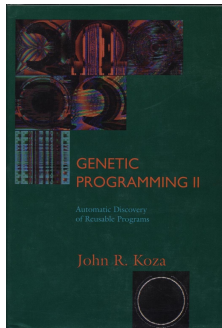
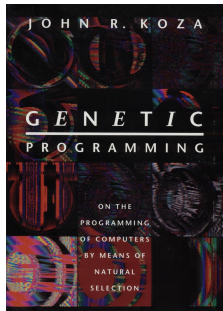
Definition

Genetic Programming (GP) is a branch of Genetic Algorithms, in which individuals (chromosomes) of population are computer programs (Koza, 92, Koza, 94, Koza, 99, Koza, 03) [1, 2, 3, 4].



John Koza

John R. Koza is a computer scientist and a former consulting professor at Stanford University, most notable for his work in pioneering the use of **Genetic Programming** for the optimization of complex problems.



Differences between GP and GA

Differences

The main difference between GP and traditional GA is the representation of the solution. GP creates computer programs in some particular programming language for representing a solution, whereas genetic algorithms create a string of numbers that represent the solution

Example

GA: 1001101100111000

GP: $\div \left(+ \left(\times (-1, b), \sqrt{ - \left(\times (b, b), \times (\times (4, a), c) \right) } \right), \times (2, a) \right)$



Goal and Fitness of GP

Goal

The goal of the GP is to find a program that solves a problem such that its analytic solution is very complicated to find.

Fitness

The fitness of a valid individual is generally obtained by the performance and the behavior of it over a training data sets.



Functions and Terminals in GP

Functions and Terminals

The set of terminals and functions is the most important component of the GP. The set of terminals and functions is the alphabet of the programs that are build over the language programming. The variables and constants of the programs belong to set of terminals.

Example

Functions: $+$, $-$, \times , \div , $\sqrt{\quad}$,

Terminals: a , b , c , ..., 2 , -1 , 4 ,



Representation of GP I

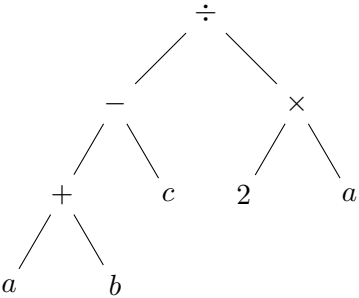
Representation

As any program can be represented as a tree or a set of trees (genotype), the programs (chromosomes) usually are represented as data structures (trees), these trees are obtained doing parsing over sentences (strings) of a program (phenotype), from where GP is applied over a particular domain (programming language).



Representation of GP II

Phenotype and Genotype

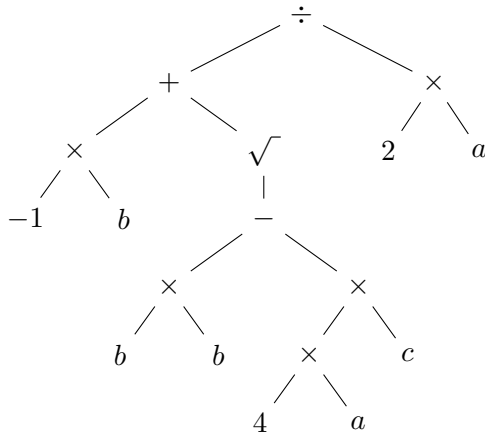
| Phenotype | Genotype |
|--------------------------------|--|
| $\frac{a + b - c}{2 \times a}$ |  |



Representation of GP III

Phenotype and Genotype

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a} = \div \left(+ \left(\times (-1, b), \sqrt{ - \left(\times (b, b), \times (\times (4, a), c) \right) } \right), \times (2, a) \right)$$



Initial population I

Methods

As the mutation is almost always lethal, it is very restricted, then, the diversity in the generations is obtained only into the initial population.

The main methods are:

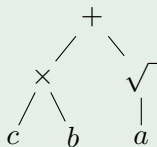
- Full
- Grow
- Ramped (half-and-half)



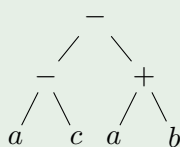
Initial population II

Full: Length of all branches equal to $l > 0$.

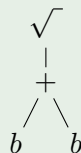
Example (Depth $l = 2$)



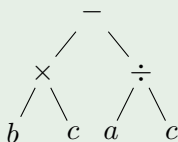
$$(c \times b) + \sqrt{a}$$



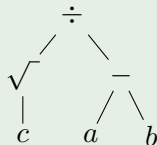
$$(a - c) - (a + b)$$



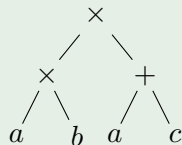
$$\sqrt{b + b}$$



$$(b \times c) - (a \div c)$$



$$\sqrt{c} \div (a - b)$$



$$(a \times b) \times (a + c)$$

Initial population III

Grow: Length of all branches of depth less than or equal to $l > 0$.

Example (Depth $0 < l \leq 2$)

| | | |
|--|--|--|
| $ \begin{array}{c} + \\ \swarrow \quad \searrow \\ c \quad \div \\ \quad \swarrow \quad \searrow \\ \quad a \quad 2 \end{array} $ $c + (a \div 2)$ | $ \begin{array}{c} - \\ \swarrow \quad \searrow \\ a \quad c \end{array} $ $a - c$ | $ \begin{array}{c} \sqrt{} \\ \\ \sqrt{} \\ \\ c \end{array} $ $\sqrt{\sqrt{c}}$ |
| $ \begin{array}{c} \sqrt{} \\ \\ a \end{array} $ \sqrt{a} | $ \begin{array}{c} \div \\ \swarrow \quad \searrow \\ a \quad a \end{array} $ $a \div a$ | $ \begin{array}{c} \times \\ \swarrow \quad \searrow \\ \times \quad + \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ a \quad b \quad a \quad c \end{array} $ $(a \times b) \times (a + c)$ |

Initial population II

Ramped (half-and-half): if μ is the cardinal of the population, then $\lfloor \frac{\mu}{l} \rfloor$ trees of depth 1, 2, 3, ..., l are created with the full or grow methods (the root has depth equal to 0). Thus, $\lfloor \frac{\mu}{l} \rfloor$ trees are generated, but as

$$\begin{aligned}\left\lfloor \frac{\mu}{l} \right\rfloor &\leq \frac{\mu}{l} \\ \left\lfloor \frac{\mu}{l} \right\rfloor l &\leq \frac{\mu}{l} l \\ \left\lfloor \frac{\mu}{l} \right\rfloor l &\leq \mu\end{aligned}$$

then $\mu - \lfloor \frac{\mu}{l} \rfloor l \geq 0$. If $\mu - \lfloor \frac{\mu}{l} \rfloor l > 0$ then $\mu - \lfloor \frac{\mu}{l} \rfloor l$ trees of depth 1 or 2 or ... or l are generated to complete the population.



Initial population IV

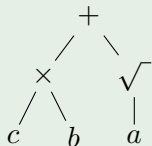
Example (For $\mu = 6$ and depth $l = 4$, $\lfloor \mu/l \rfloor = 1$, $\mu - \lfloor \mu/l \rfloor l = 2$)

$l_1 = 1$



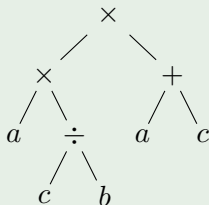
$a - c$

$l_2 = 2$



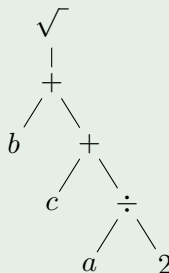
$(c \times b) + \sqrt{a}$

$l_3 = 3$



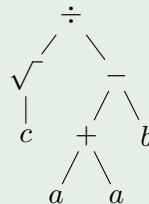
$(a \times (c \div b)) \times (a + c)$

$l_4 = 4$



$\sqrt{b + (c + (a \div 2))}$

$l_5 = 3$



$\sqrt{c} \div ((a + a) - b)$

$l_6 = 1$



\sqrt{c}

Selection

Selection: is similar to the selection into genetic algorithms, normally the selection is done over a sorted set.

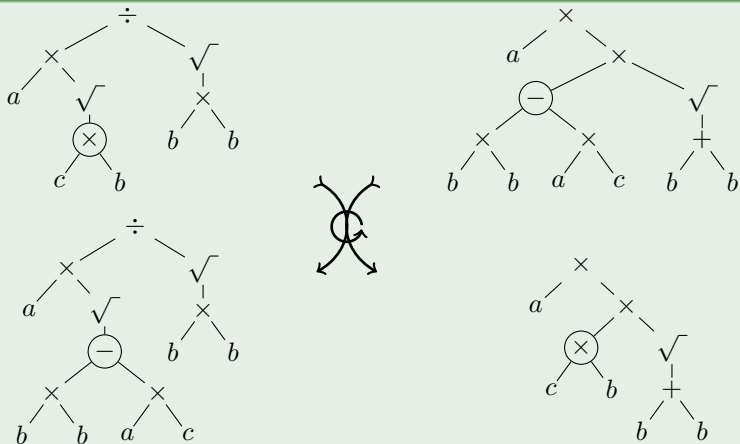
- Elitism selection
- Uniform selection
- Tournament selection
- Roulette-wheel selection
- Rank selection
- Stochastic selection



Operators I

Crossover: given a couple of programs, a subtree of each program is randomly selected and these subtrees are swapped.

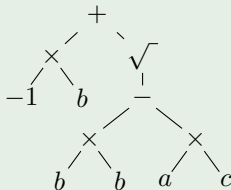
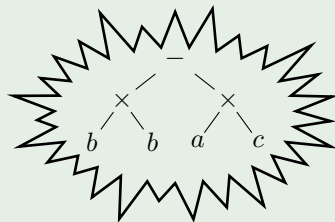
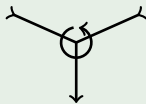
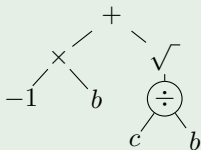
Example



Operators II

Mutation: given a program, a subtree of this program is randomly selected and it is replaced by a new randomly generated tree.

Example

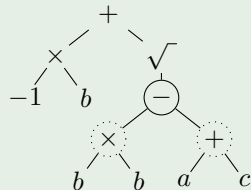
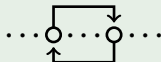
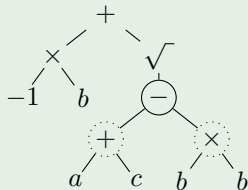


Operators II

Swap: given a function, a subfunction f is randomly selected, if this subfunction has arity $n \geq 2$, then two different parameters a_i, a_j ($i \neq j$) are randomly selected and these are swapping, thus

$$f(a_1, \dots, \underline{a_i}, \dots, \underline{a_j}, \dots, a_n) \longrightarrow f(a_1, \dots, \underline{a_j}, \dots, \underline{a_i}, \dots, a_n)$$

Example



Replacement strategy

Replacement: is similar to the replacement into genetic algorithms.

- Generational replacement
- Steady state replacement
 - Replace worst
 - Replace best
 - Replace parent
 - Replace random
 - Replace most similar (crowding)
 - Replace oldest



Advantages

- It is not necessary background knowledge.
- It is possible work with several solutions.
- Those are easily parallelizable.
- Those can use probabilistic operators.
- It is more natural represent the solution with a program instead of a real number or a binary string.
- The solution has implicit the algorithm.
- The chromosomes do not have a fixed length.



Disadvantages

- The syntax is very restricted.
- The mutation is restricted to particular places into the program.
- Generally, local optimum are obtained.
- Those can converge prematurely.



References I



John R. Koza, *Genetic programming: On the programming of computers by means of natural selection*, Cambridge, MA: MIT Press, 1992.



———, *Genetic programming ii: Automatic discovery of reusable programs*, MIT Press, 1994.



John R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, *Genetic programming iii: Darwinian invention and problem solving*, Morgan Kaufmann, 1999.



John R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza, *Genetic programming iv: Routine human-competitive machine intelligence*, Kluwer Academic Publishers, 2003.



Thank you!

Questions?

