Programación genética

María del Rocío Ochoa Montiel

Contacto: ma.rocio.ochoa@gmail.com

A Field Guide to Genetic Programming

Genetic programming (GP) is an evolutionary computation (EC)[1] technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance. At the most abstract level GP is a *systematic*, *domain-independent* method for getting computers to solve problems *automatically* starting from a *high-level statement* of what needs to be done.

[1] These are also known as *evolutionary algorithms* or EAs.

R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming.* Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions by J. R. Koza). GPBiB

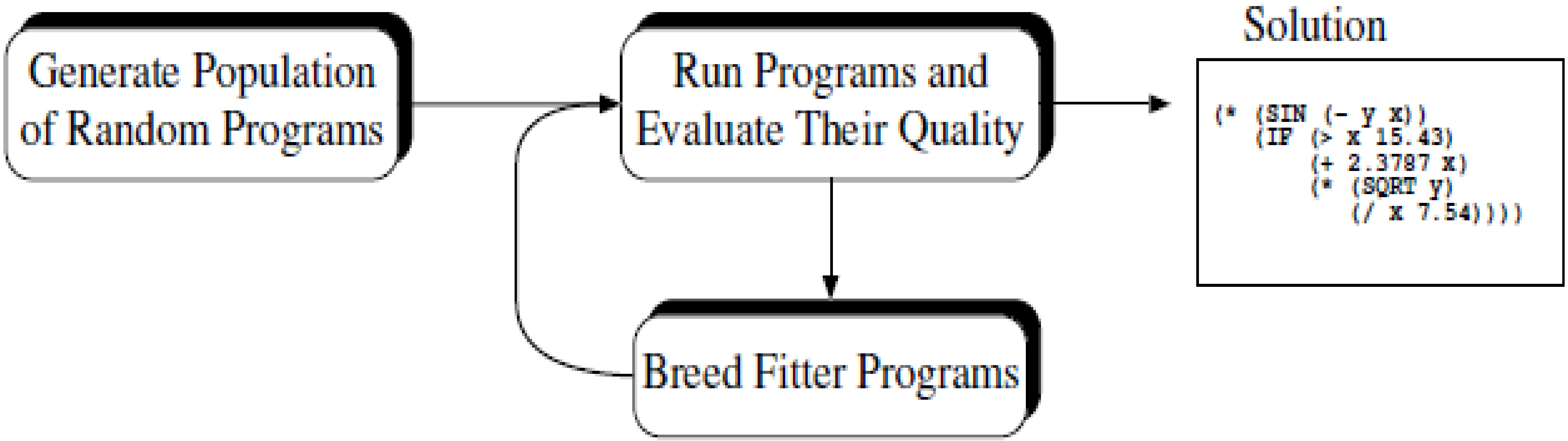
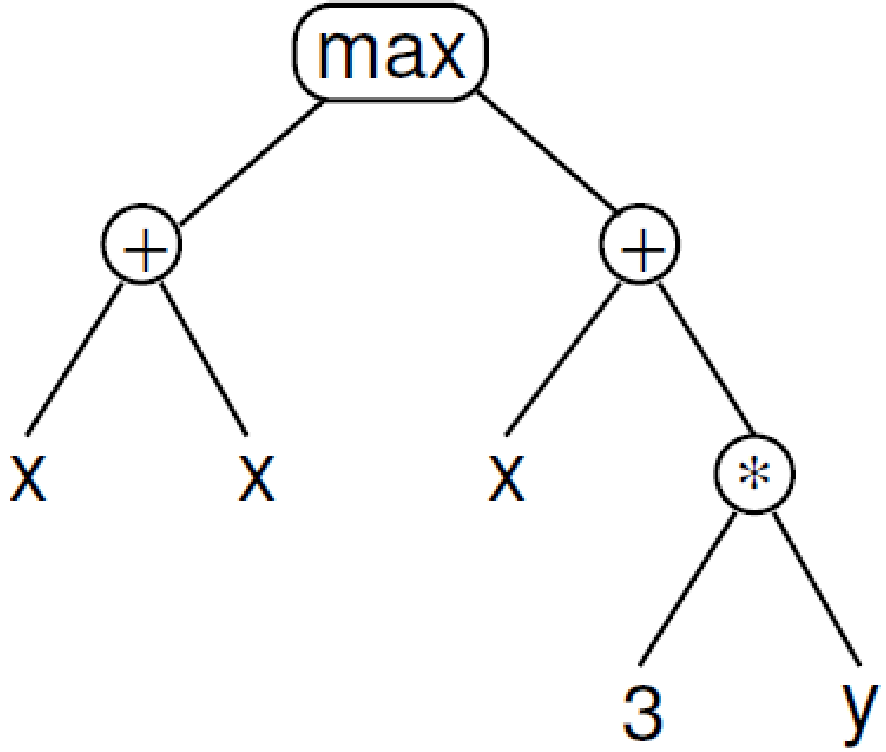


Figure 1.1: The basic control flow for genetic programming, where survival of the fittest is used to find solutions.

1: Randomly create an *initial population* of programs from the available primitives (more on this in Section 2.2).
2: **repeat**
3: *Execute* each program and ascertain its fitness.
4: *Select* one or two program(s) from the population with a probability based on fitness to participate in genetic operations (Section 2.3).
5: Create new individual program(s) by applying *genetic operations* with specified probabilities (Section 2.4).
6: **until** an acceptable solution is found or some other stopping condition is met (e.g., a maximum number of generations is reached).
7: **return** the best-so-far individual.

Algorithm 1.1: Genetic Programming

In GP, programs are usually expressed as *syntax trees* rather than as lines of code. For example Figure 2.1 shows the tree representation of the program `max(x+x,x+3*y)`. The variables and constants in the program (`x`, `y` and `3`) are leaves of the tree. In GP they are called *terminals*, whilst the arithmetic operations (`+`, `*` and `max`) are internal nodes called *functions*. The sets of allowed functions and terminals together form the *primitive set* of a GP system.



2.1: GP syntax tree representing `max(x+x,x+3*y)`.

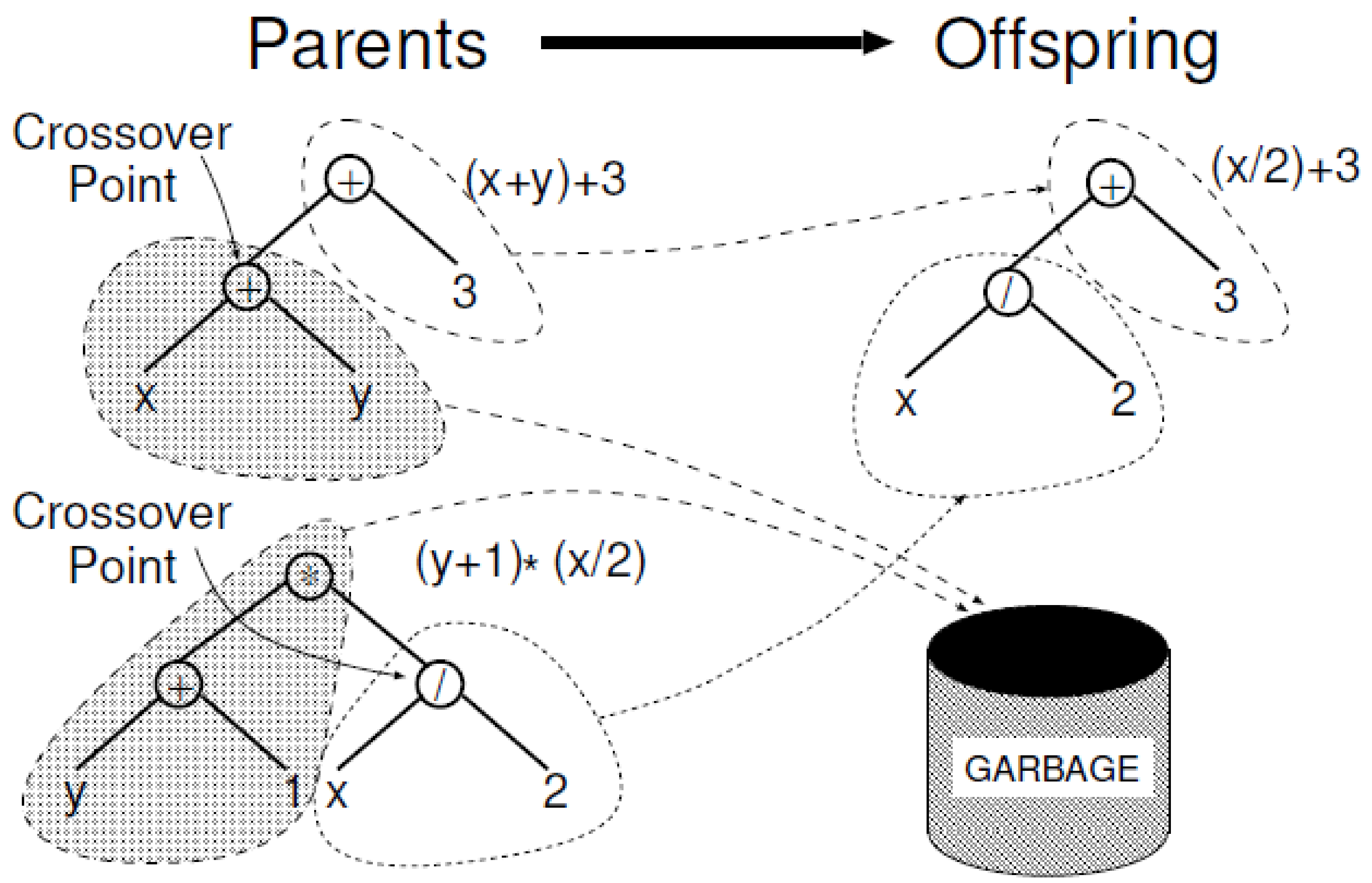


Figure 2.5: Example of subtree crossover. Note that the trees on the left are actually *copies* of the parents. So, their genetic material can freely be used without altering the original individuals.

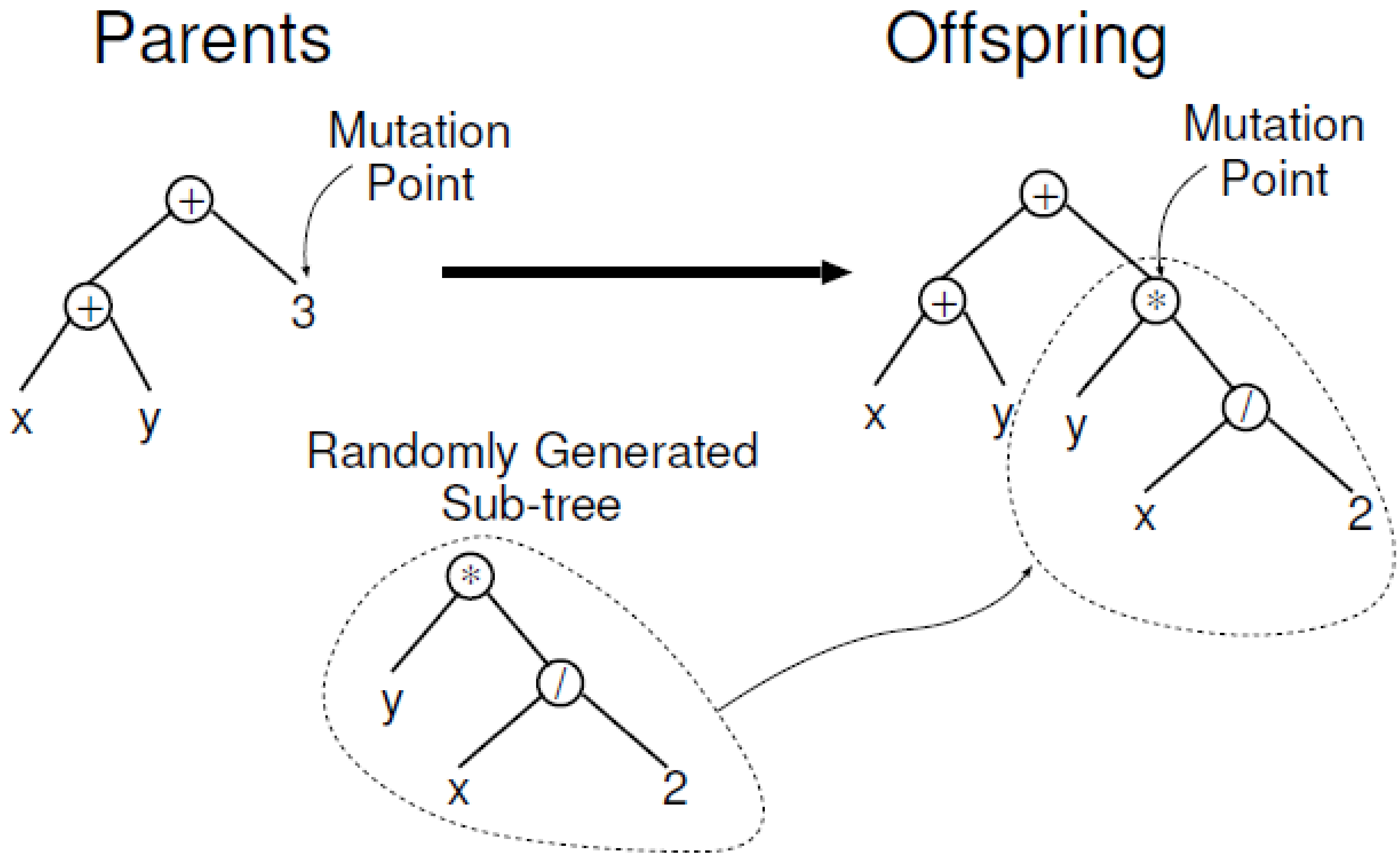


Figure 2.6: Example of subtree mutation.

To apply a GP system to a problem, several decisions need to be made; these are often termed the *preparatory steps*. The key choices are:

1. What it the *terminal set*?
2. What is the *function set*?
3. What is the *fitness measure*?
4. What *parameters* will be used for controlling the run?
5. What will be the *termination* criterion, and what will be designated the *result of the run*?

Examples of primitives in GP function and terminal sets.

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

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

Referencias

1. Gestal M., Rivero D., Rabuñal J. R., Dorado J y Pazos A. Introducción a los Algoritmos genéticos y la Programación Genética. Universidad de Coruña. ISBN: 978-84-9749-422-9. Madrid, 2010.
2. Coello Coello C.A. Introducción a la Computación Evolutiva (Notas de Curso). CINVESTAV-IPN. México, D.F., mayo 2015.
3. L. Haupt Randy and Ellen Haupt Sue. Practical Genetic Algorithms. John Wiley & Sons, Inc. USA.1998.
4. Koza, J.R.: Genetic Programming. On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge, MA (1992).
5. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming - An Introduction: On The Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers, San Francisco, CA (1998).