Lecture 8

Genetic programming

Genetic programming (GP)

Specific application of GA, where the chromosomes - binary vectors are substituted by the **parse trees**.

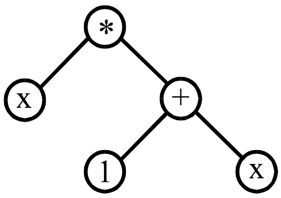
Invented by John R. Koza (1989)

Literature:

1. J.R. Koza: *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA. 1992.

2. J.R. Koza: *Genetic Programming II. Automatic Discovery of Reusable Programs*. MIT Press, Cambridge. MA. 1994.

1. **Parse tree** - graph-theoretical structure (rooted tree) that is very useful for interpretation of single algebraic expressions - programs. It represents one of basic concepts of computer science.

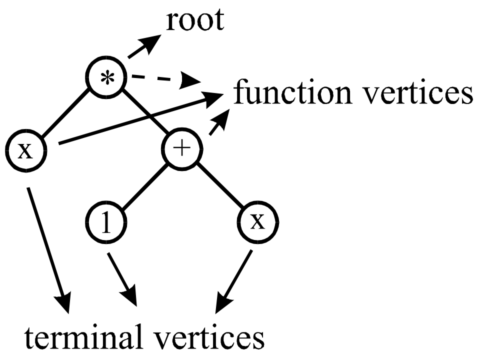


This tree t (called the **parse tree**) is interpreted as a function (expression)

$$t(x) = x*(1+x) = x + x^2$$

Vertices of the parse tree are classified as follows:

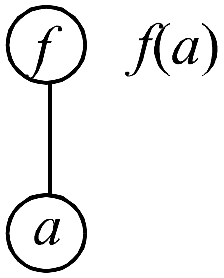
(1) Top vertex - **root**
(2) Leaf vertices - **terminal vertices**
(3) All other vertices (and top vertex) - **function vertices**



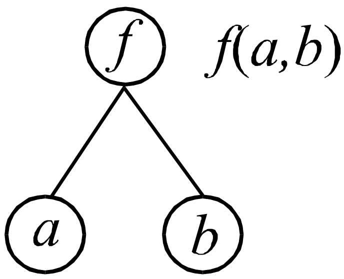
Interpretation of vertices:

(1) Terminal vertices correspond either to the **independent variable**(s) x (y,...) or to **constants** (0,1,2,...).
(2) Function vertices correspond to simple **operations** (unary, binary, ternary,....)

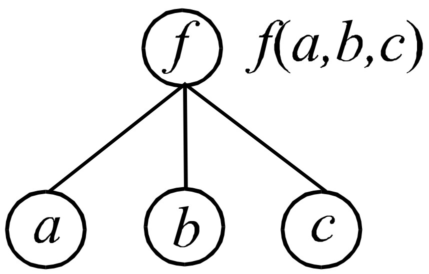
Function vertex of **unary** operation



Function vertex of **binary** operation



Function vertex of **ternary** operation



For prescribed set of function and terminal vertices we may define a **set of all possible parse trees** that are composed of the prescribed type vertices

$$T = \{t_1, t_2, \ldots\}$$

Convention: Each parse tree $t \in T$ is **identified** with its interpretation - function ($t(x)$). This means that the set T may be understood as a **set of functions**.

2. **Regression analysis**. Let us have a training set of points (regression table)

$$A_{train} = \left\{ x_i \ / \ y_i \, ; \, i = 1, 2, \ldots, n \right\}$$

The goal of **standard regression analysis** is to find optimal parameter(s) of a given function $G(x;w)$ (w is (are) adjustable parameter(s)) so that the following **objective function** is minimized

$$E(w) = \sum_{i=1}^{n} \left| G(x_i \, ; w) - y_i \right|$$

$$w_{opt} = \arg\min_{w} E(w)$$

We say that the "**adapted**" function $G(x;w_{opt})$ **models** the training set (regression table).

Symbolic regression analysis goes further, it looks for a function in a set (space) T so that the objective function (functional) is minimized

$$E(t) = \sum_{i=1}^{n} \left| t(x_i) - y_i \right|$$

$$t_{opt} = \arg \min_{t \in T} E(t)$$

The symbolic regression is the main goal of Koza's genetic programming with a restriction that the allowed functions are those ones that can be represented by parse trees (composed of prescribed function and terminal vertices).

3. How to modify GA for purposes of GP?

(1) The population is composed of chromosomes - parse trees.

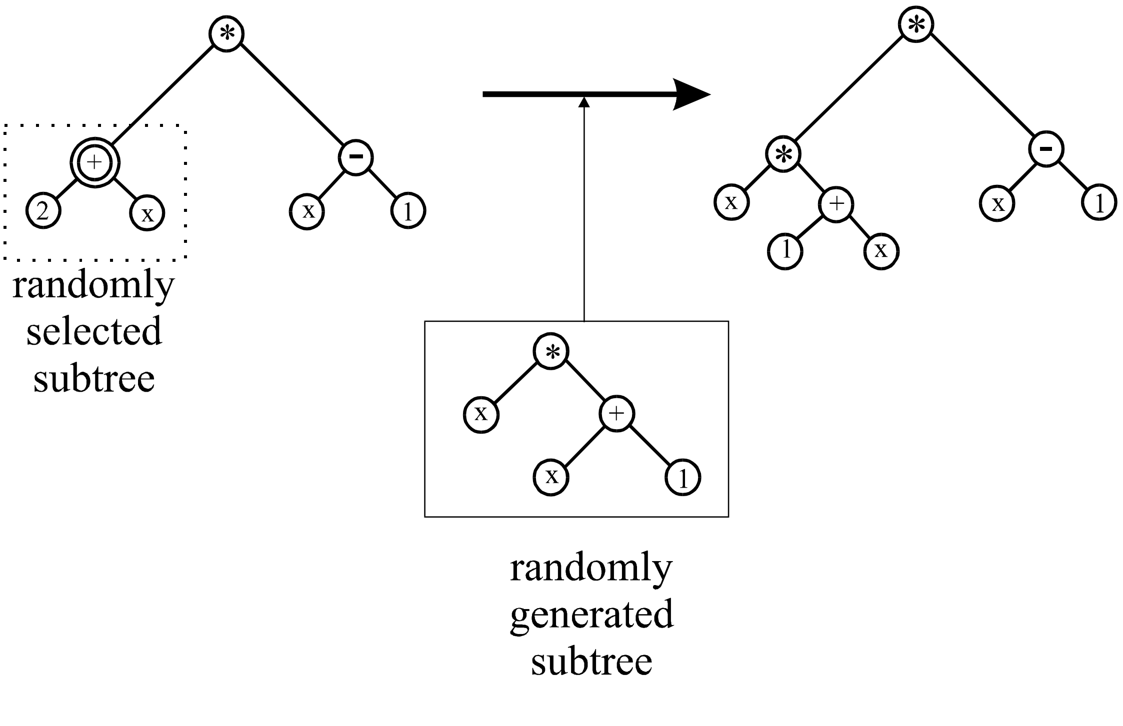
(2) Each chromosome of the population is evaluated by the fitness so that

$$E(t_1) \leq E(t_2) \Rightarrow f(t_1) \geq f(t_2)$$

(3) **Mutation**, in a parse tree t a randomly selected subtree is changed by another randomly generated subtree, we get t'

$$t' = O_{mut}(t)$$

Illustrative example of mutation



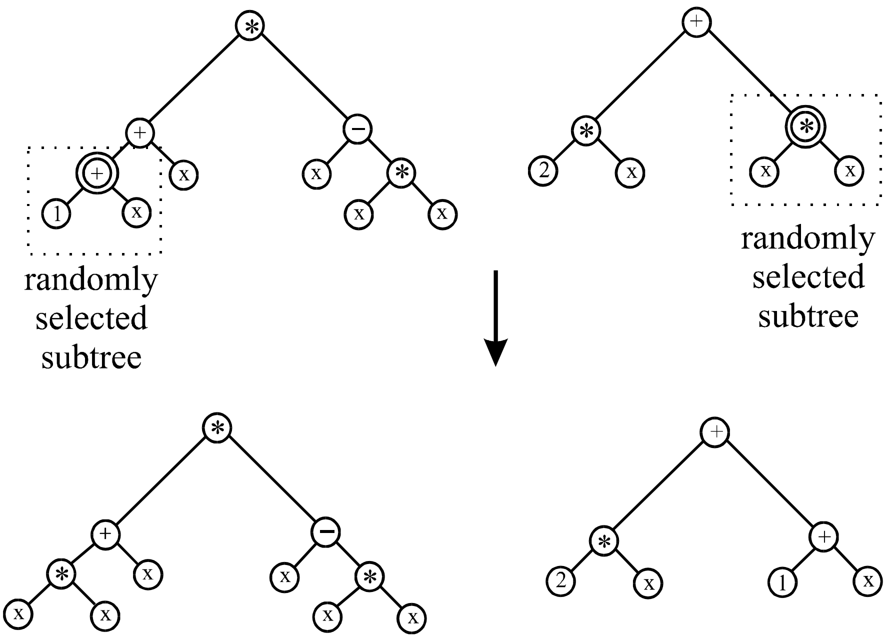
$$t(x) = (2 + x) * (x - 1)$$
$$t'(x) = \left(x * (1 + x)\right) * (x - 1)$$

Koza claims that the **mutation may be omitted** in GP !?!

(4) **Crossover**, for a pair of parse trees t_1 and t_2 two randomly selected subtrees are mutually exchanged, we get t_1' and t_2'

$$\left(t_1', t_2'\right) = O_{cross}\left(t_1, t_2\right)$$

Illustrative example of crossover



$$t_1(x) = (1 + 2x) * (x - x^2)$$
$$t_2(x) = 2x + x^2$$
$$t_1'(x) = (x^2 + x) * (x - x^2)$$
$$t_2'(x) = 2x + (1 + x)$$

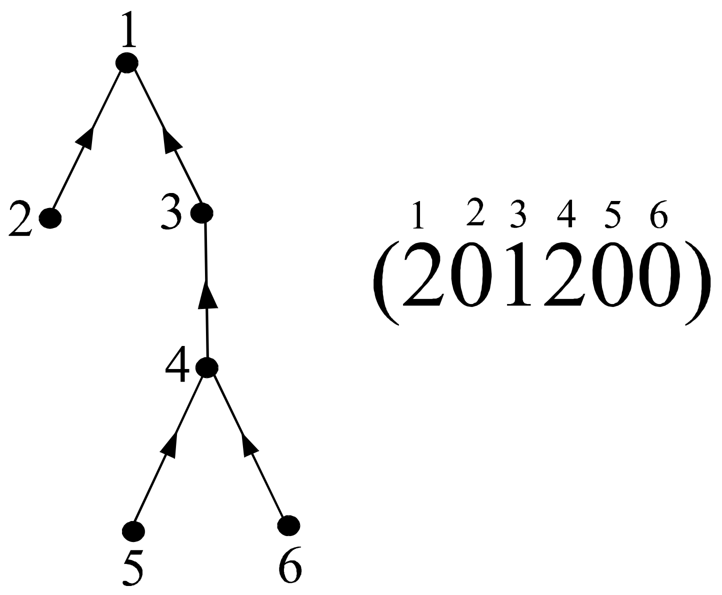
4. How to code parse trees

Read's code - rooted trees composed of n vertices are coded by sequences of n non-negative integers

$$code(t) = (\tau_1, \tau_2, \ldots, \tau_n) \in \{0,1,2,\ldots\}^n$$

Illustrative example:

Interpretation of the *code*(*t*)=(201200)



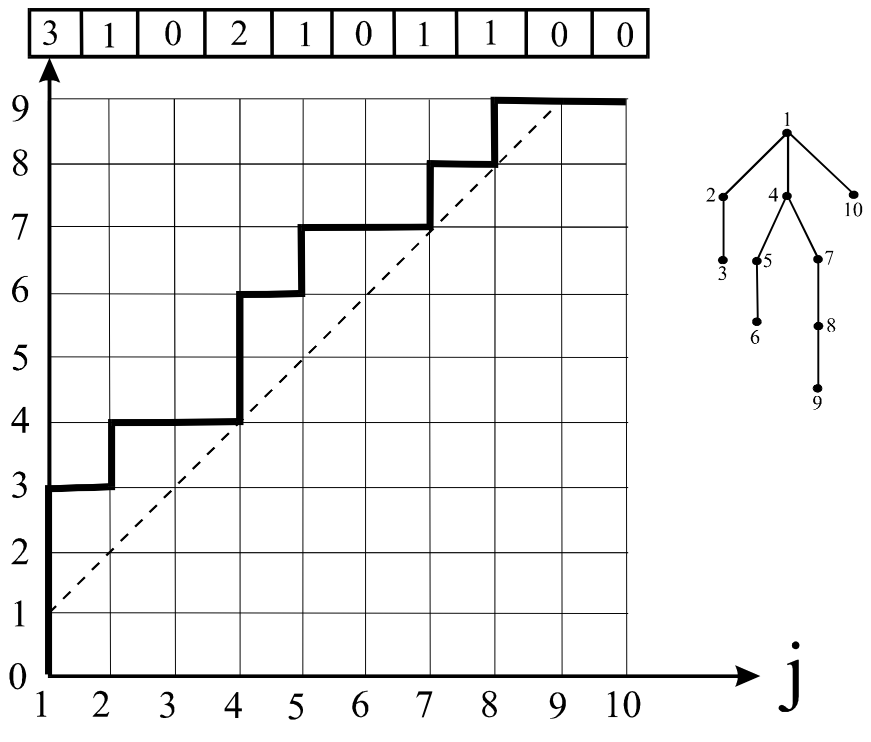
Theorem. Necessary and sufficient conditions that a sequence of nonnegative integers

$$\left(\tau_1, \tau_2, \ldots \tau_n\right) \in \{0,1,2,\ldots\}^n$$

corresponds to a rooted tree are

$$\sum_{i=1}^{j} \tau_i \geq j \qquad \left(j = 1,2,\ldots,n-1\right)$$

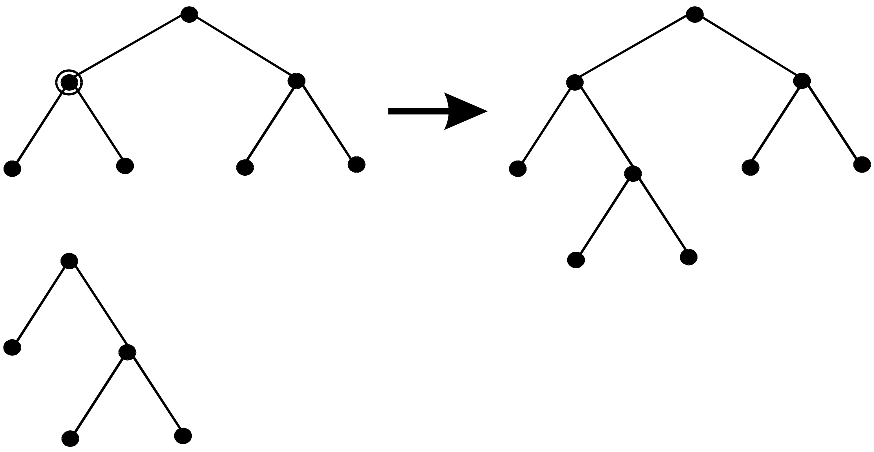
$$\sum_{i=1}^{n} \tau_i = n-1$$



Mutation

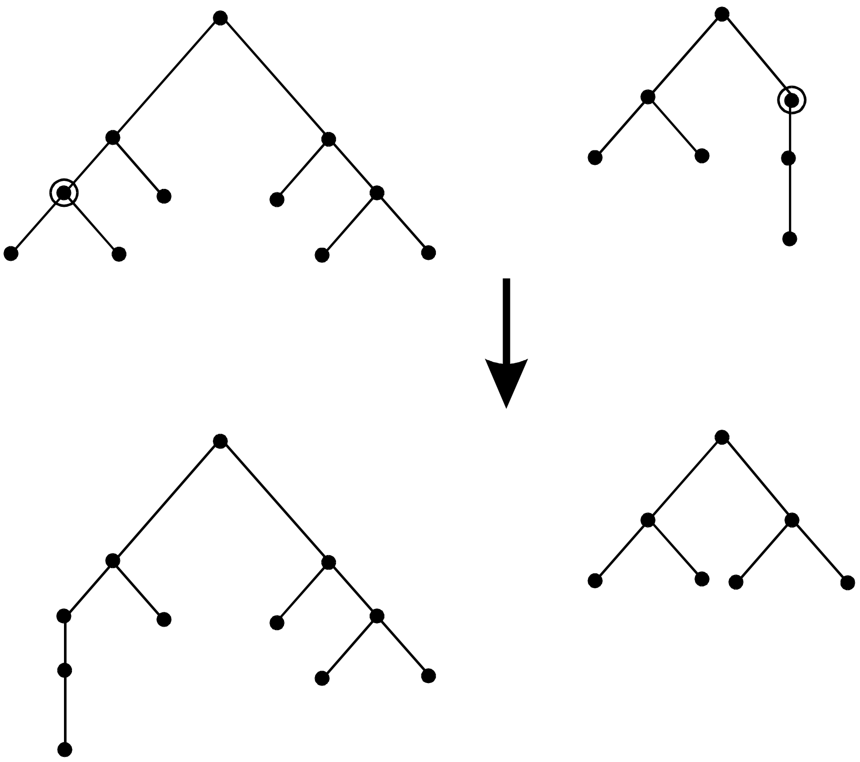
$$(2(200)200) \longrightarrow (2(20200)200)$$

$$(20200)$$



Crossover

parents:		offspring:
(22(200)020200)	→	(22(110)020200)
(2200(110))		(2200(200))



Advantages of Read's code in GP

(1) Graph-theoretical structures - rooted trees are **substituted** by simple algebraic structure - Read's code.

(2) **Simple algebraic manipulations** with Read's code, e.g. looking for subtrees is equivalent to looking for subcodes.

(3) **Simple implementation** in procedural programming languages (C++, Pascal).

5. Illustrative example

Regression table

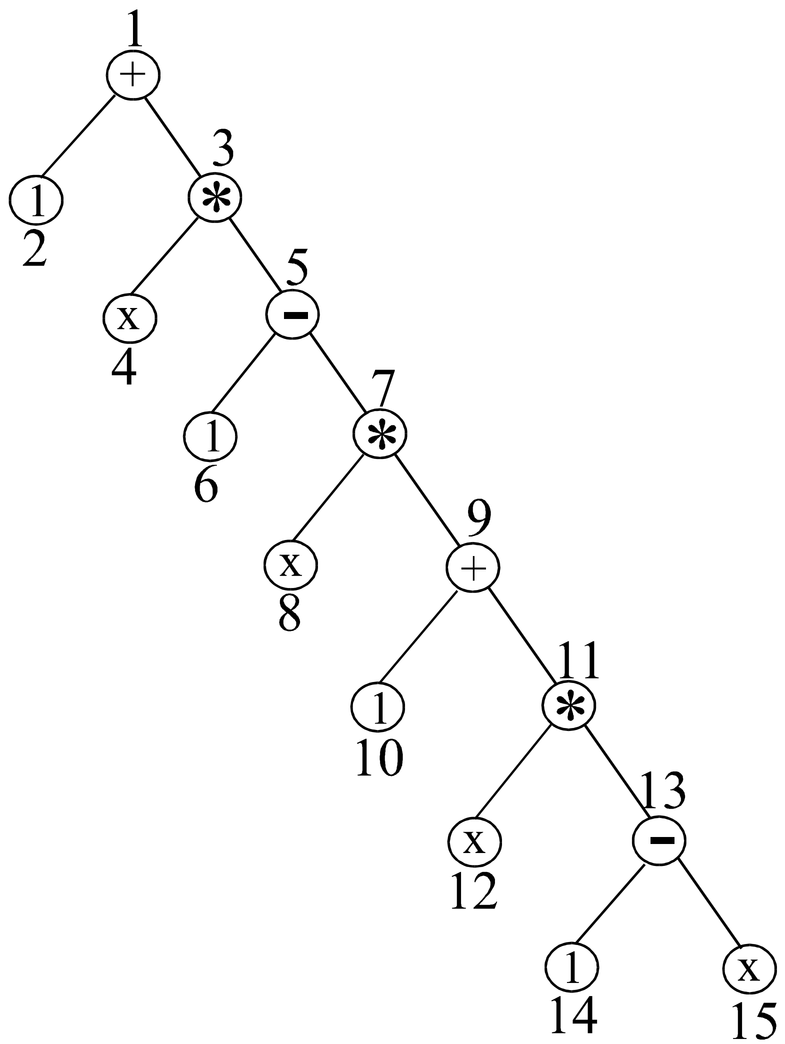
No.	x	y
1	-10	10891
2	-8	4357
3	-5	1471
4	-4	301
5	-2	19
6	0	1
7	2	7
8	4	181
9	6	1051
10	7	3529

Type of vertices:

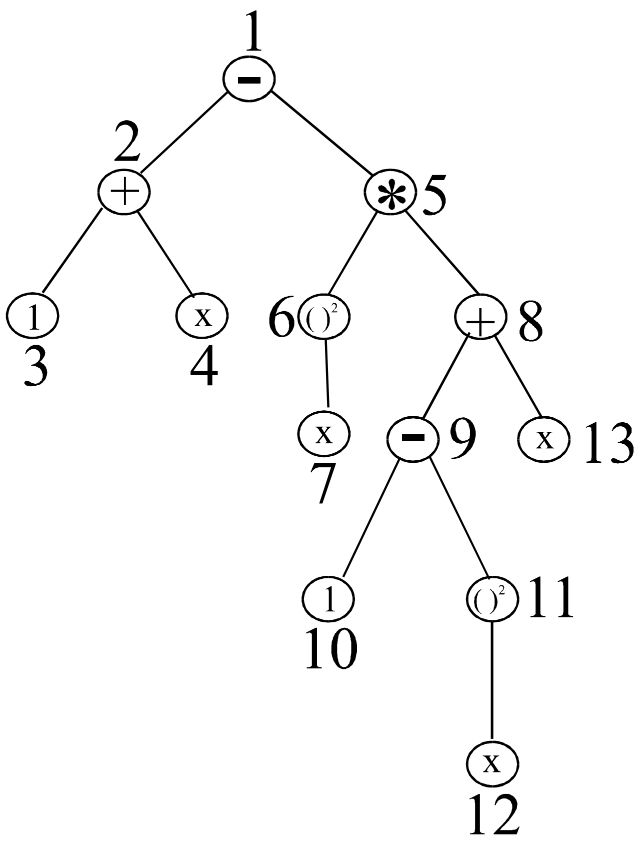
(1) terminal vertices: {x,1,2,3,...}
(2) unary vertices: $\{+/-, (\ )^2\}$
(3) binary vertices: $\{+,-,*\}$

Parse tree constructed by Horner's scheme

$$\begin{aligned} f(x) &= 1 + x - x^2 - x^3 + x^4 \\ &= 1 + x\left(1 - x - x^2 + x^3\right) \\ &= 1 + x\left(1 - x\left(1 + x - x^2\right)\right) \\ &= 1 + x\left(1 - x\left(1 + x(1 - x)\right)\right) \end{aligned}$$



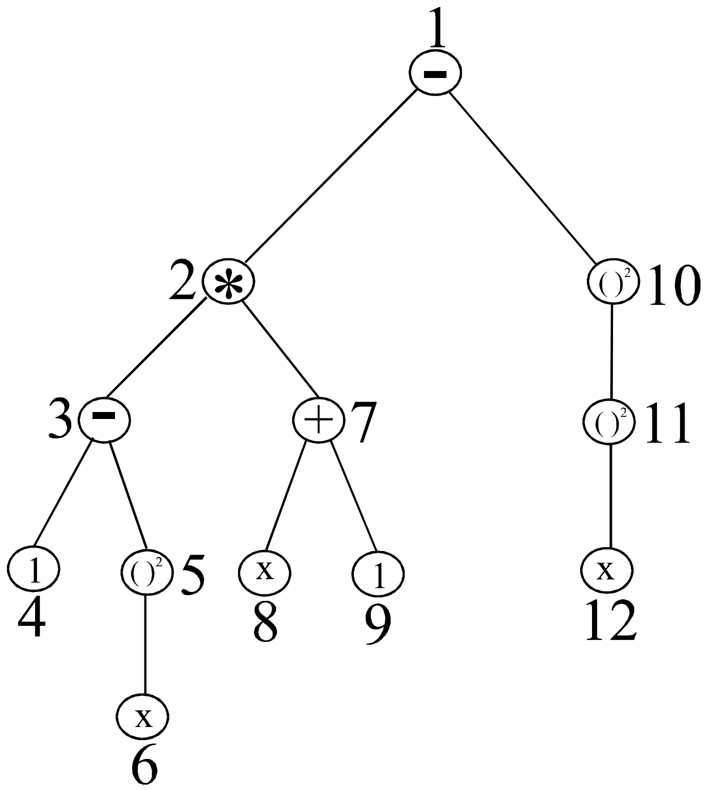
One of possible solutions recorded by GP



$$x_{13} = x$$
$$x_{12} = x$$
$$x_{11} = x^2$$
$$x_{10} = 1$$
$$x_9 = 1 - x^2$$
$$x_8 = 1 - x^2 + x$$
$$x_7 = x$$

$$x_6 = x^2$$
$$x_5 = x^2 * \left(1 - x^2 + x\right)$$
$$x_4 = x$$
$$x_3 = 1$$
$$x_2 = 1 + x$$
$$x_1 = (1 + x) - x^2 * \left(1 - x^2 + x\right)$$
$$= 1 + x - x^2 - x^3 + x^4$$

The best solution recorded by GP



$$
\begin{aligned}
x_{12} &= x \\
x_{11} &= x^2 \\
x_{10} &= x^4 \\
x_9 &= 1 \\
x_8 &= x \\
x_7 &= x+1
\end{aligned}
$$

$$
\begin{aligned}
x_6 &= x \\
x_5 &= x^2 \\
x_4 &= 1 \\
x_3 &= 1-x^2 \\
x_2 &= \left(1-x^2\right)*\left(x+1\right) \\
x_1 &= \left(1-x^2\right)*\left(1+x\right)+x^4 \\
&= 1+x-x^2-x^3+x^4
\end{aligned}
$$

Other illustrative example

Regression of Bolean functions

Even-parity problem of 3-bit vectors, this problem is determined by the following regression table

No.	x_1	x_2	x_3	y
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	1
8	1	1	1	0

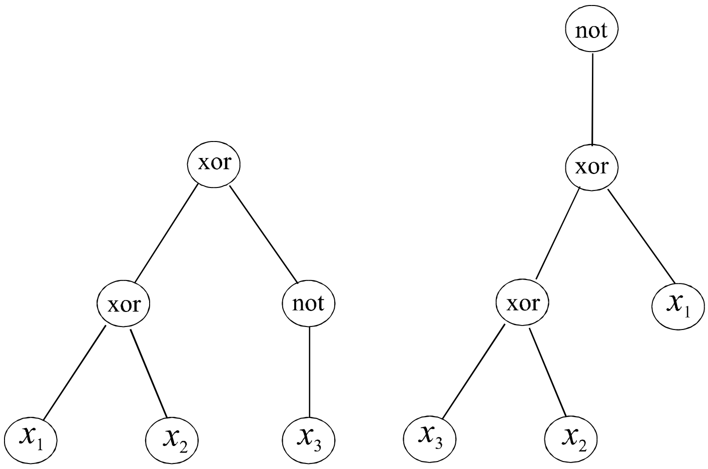
We look for a Boolean function

$$y = F(x_1, x_2, x_3)$$

Type of vertices:

(1) terminal vertices: $\{x_1, x_2, x_3\}$
(2) unary vertices: {not}
(3) binary vertices: {and, or, xor}

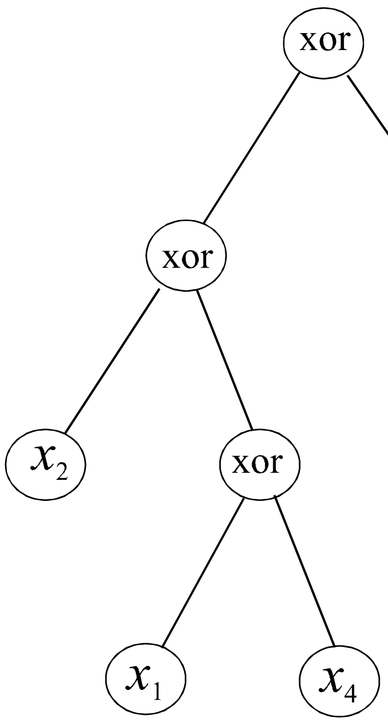
Final correct solution looks as follows e.g.



A B

$$F_A = (x_1 \text{ xor } x_2) \text{ xor } (\text{not } x_3)$$
$$F_B = \text{not}\left((x_1) \text{ xor } (x_2 \text{ xor } x_3)\right)$$

4-bit even parity problem: GP offers e.g.



$$F = \left(x_2 \text{ xor } \left(x_1 \text{ xor } x_4\right)\right) \text{ xor } \left(\text{not } x_3\right)$$

Conclusions

1. GP is **simple extension** of GA, where bit strings are substituted by parse trees.

2. **Read's code** of rooted trees allows to use simple procedural languages (like C++ or Pascal) in programming GP

3. GP may be understood as a **symbolic regression**, where the model function is not suggested by the user but constructed by the algorithm.