Genetic Programming

Mihai Oltean

www.cs.ubbcluj.ro/~moltean

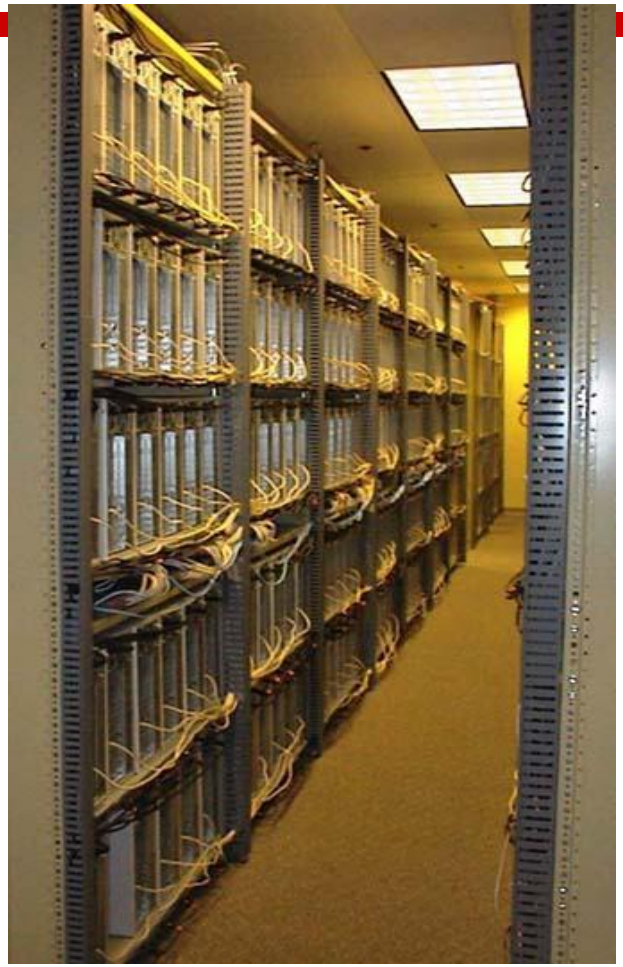
moltean@cs.ubbcluj.ro

Structure

- What is
- Why we need it
- Representation
  - Standard GP
  - Cartesian GP
  - Grammatical Evolution
  - Multi Expression Programming
  - Gene Expression Programming
- Fitness
  - Regression, classification
- Applications

John Koza (invented GP in 1988)

Genetic-Programming Inc.
1000 computers



www.genetic-programming.com

What is the purpose of GP?

- GP evolves complex computer programs.
- GA wants to evolve only solutions for particular problems (function optimization, TSP, knapsack…)

Attributes of GP

- **Starts with "What needs to be done":** It starts from a high-level statement specifying the requirements of the problem.

- **Produces a computer program:** It produces an entity that coulds run on a computer.

What GP should also do?

- Automatic determination of program size
- Code reuse
- Internal storage
- Iterations, loops, and recursions
- Problem-independent
- …

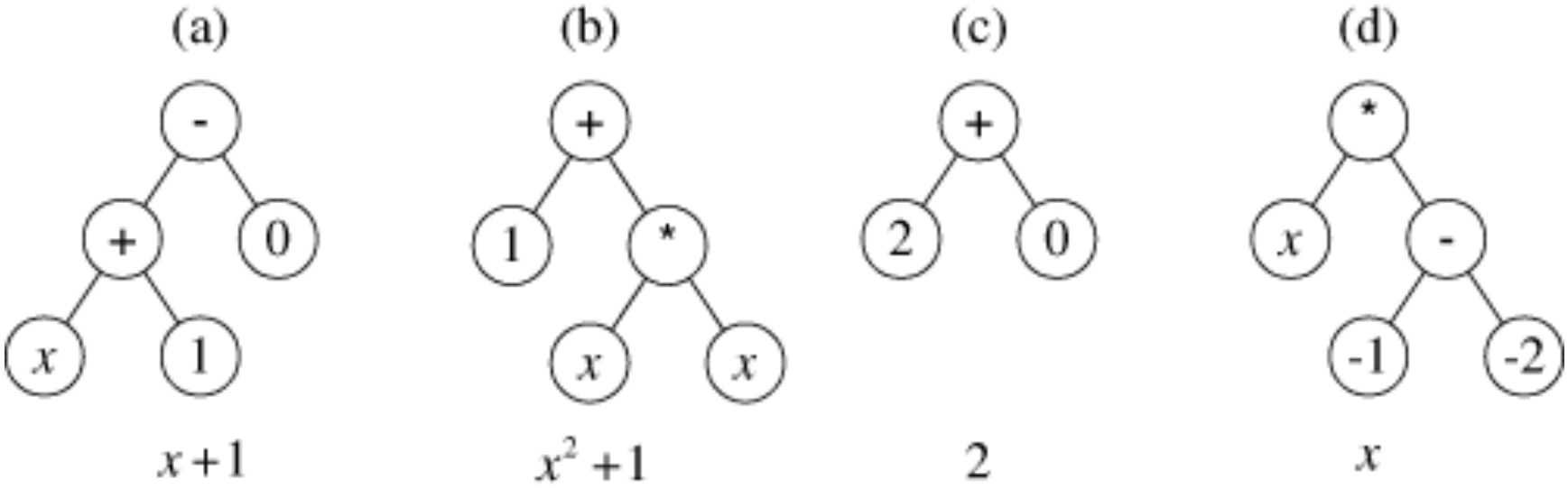
How GP represents computer programs?

- Very important and difficult task!
- At least 10 different representations have been developed so far.

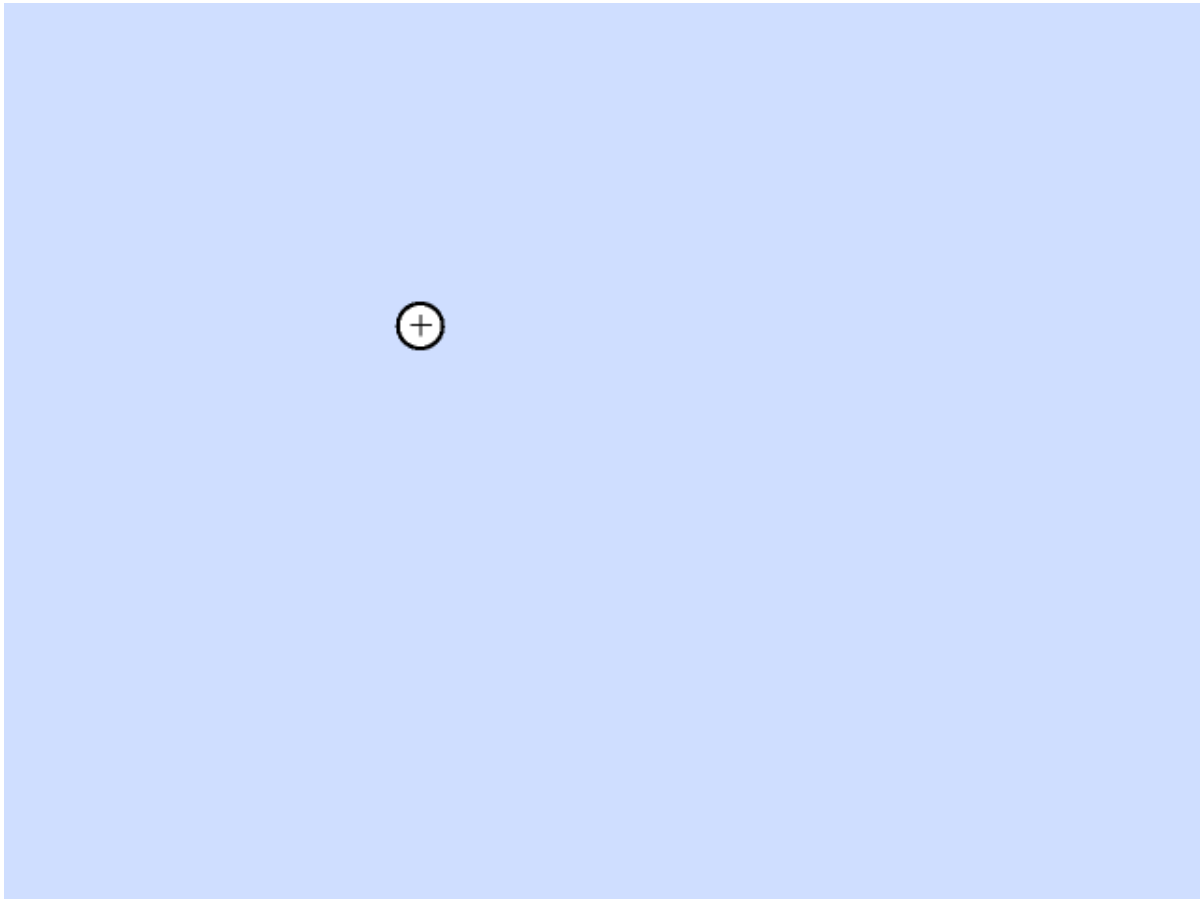
What a GP chromosome can contain?

- Terminals T = {x, y, a, b, 0.1, 7, 65}
- Functions F = {+, -, *, /, sin, cos, if}

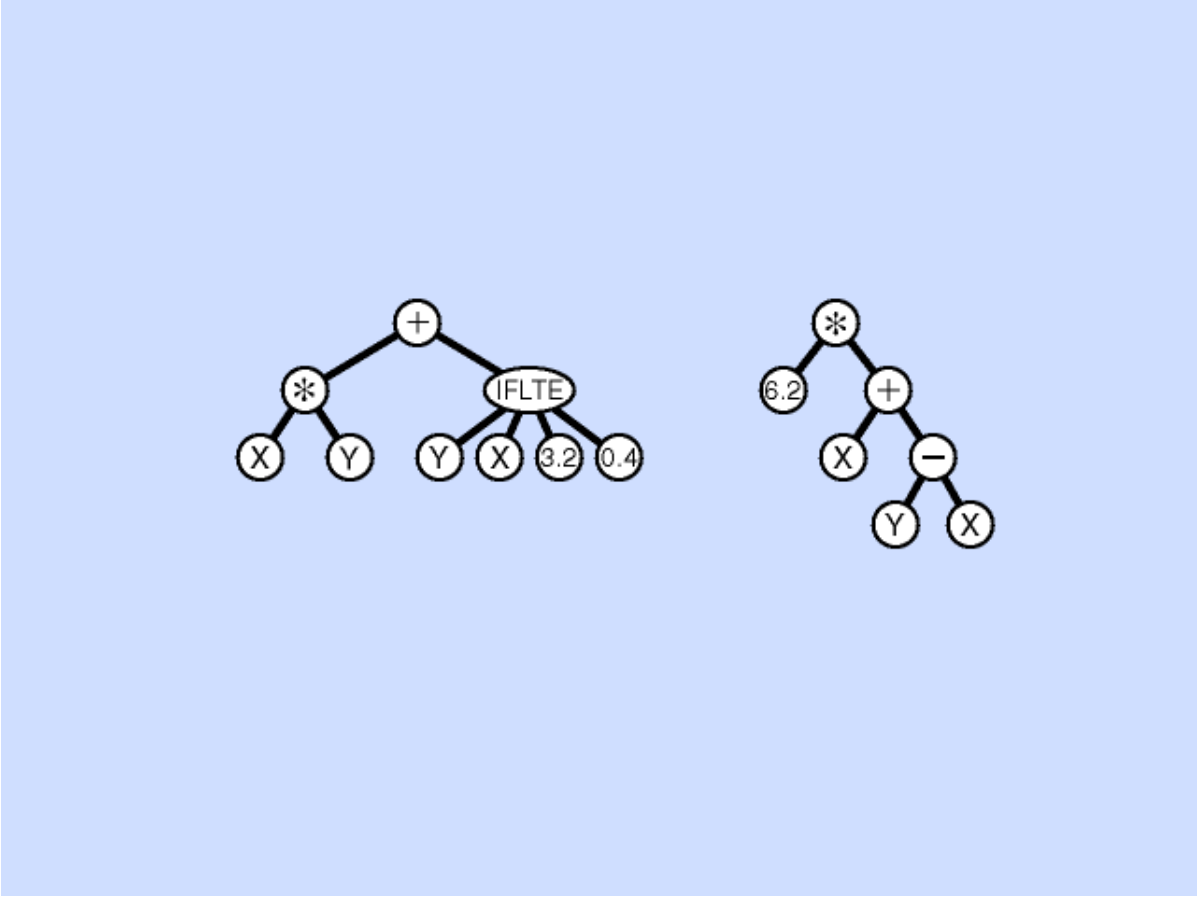
STANDARD GP REPRESENTATION



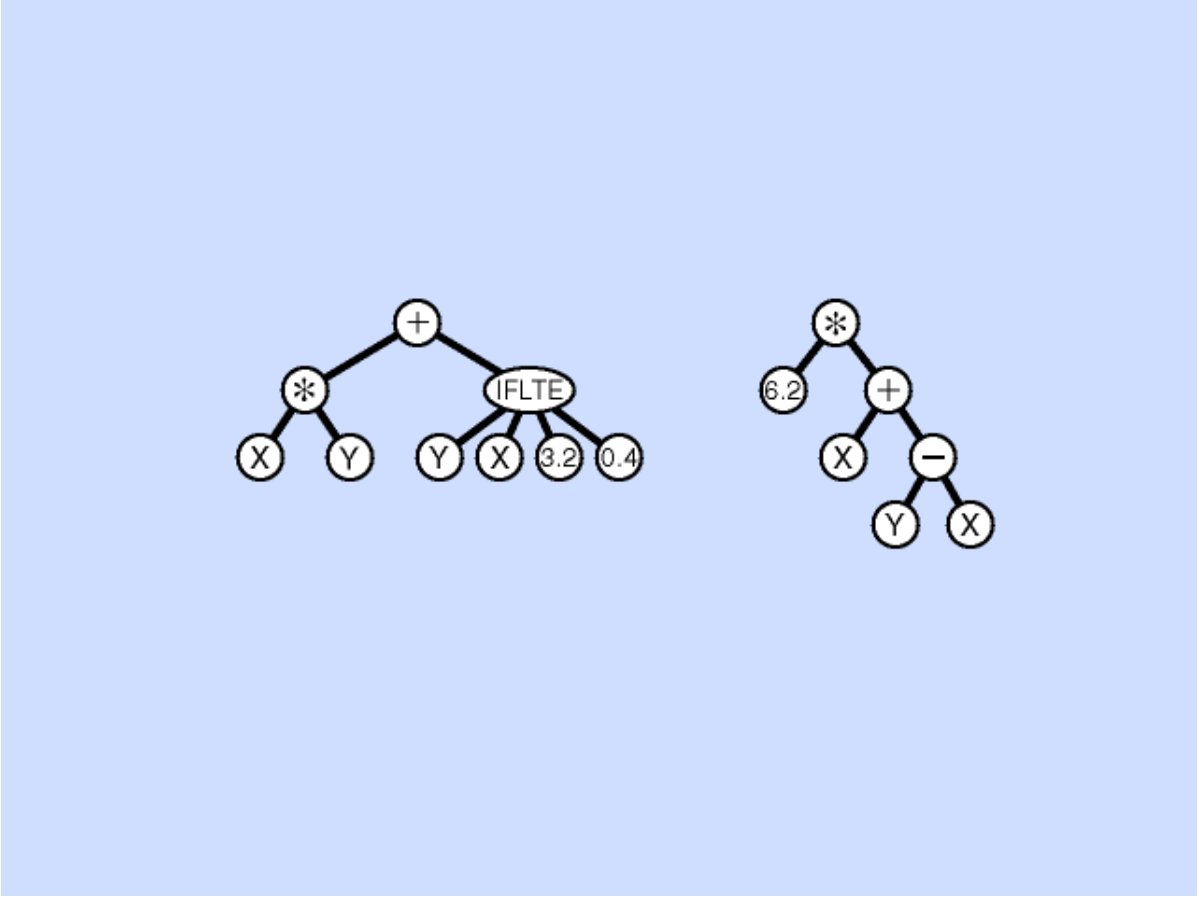
Initialization



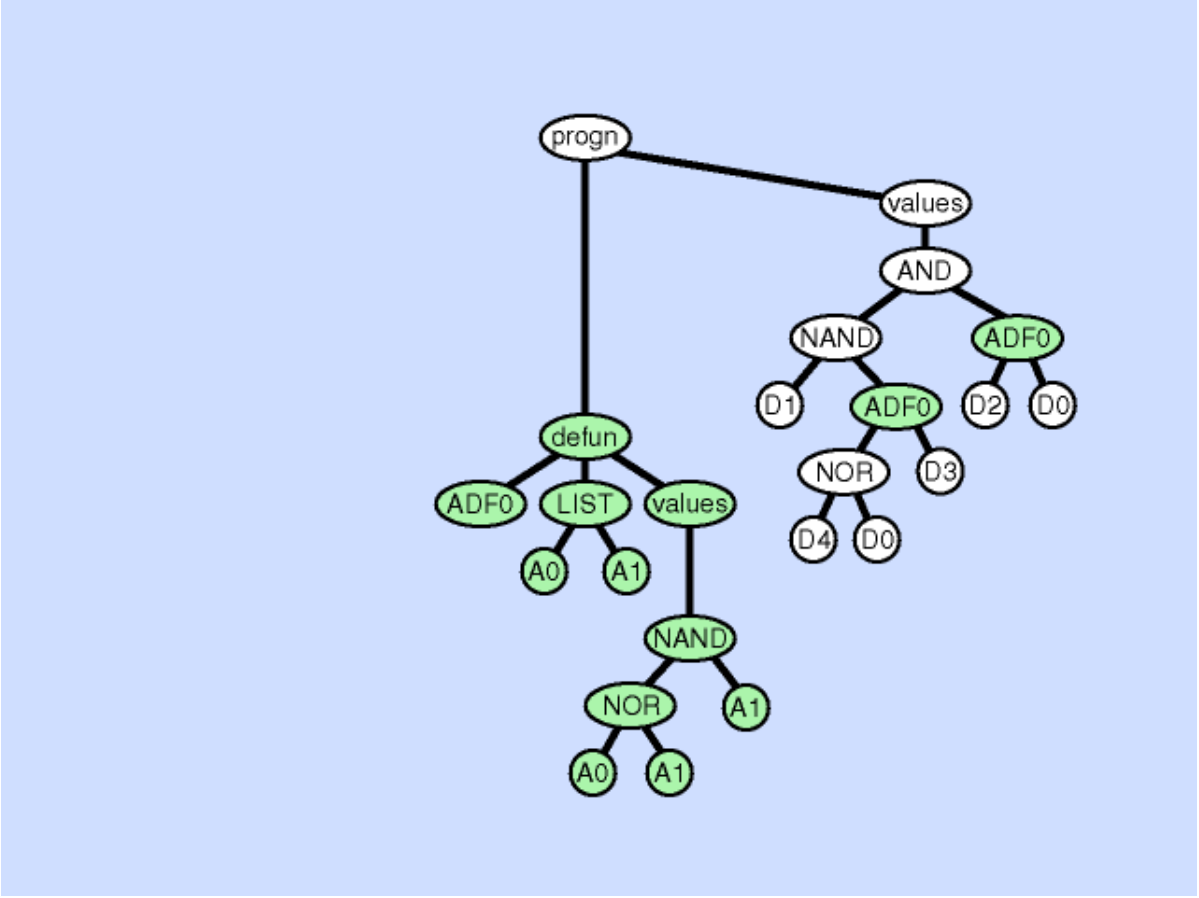
CROSSOVER



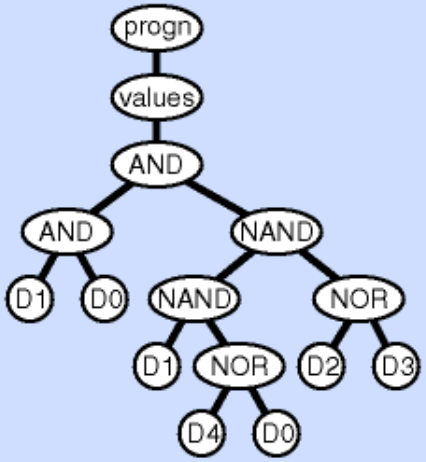
Mutation



BRANCH DUPLICATION



BRANCH DUPLICATION



Fitness of a GP chromosome

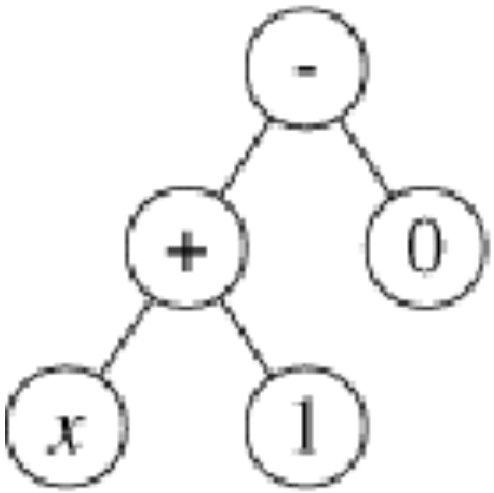
- You need a training set.
- Apply the GP chromosome to each training data.
- Compute the difference between what you want to obtain and what you've actually obtained.
- Fitness should be minimized.

Example – regression problem

- $x = 0, f(x) = 2$
- $x = 1, f(x) = 3$
- $x = 2, f(x) = 6$
- $x = 3, f(x) = 11$

$$f(x) = ?$$

QUALITY OF TREES



$E = x + 1$

X = 0 ☐ value of *E* = 1

X = 1 ☐ value of *E* = 2

X = 2 ☐ value of *E* = 3

X = 3 ☐ value of *E* = 4

Fitness = |1-2| + |2-3| + |3-6| + |4-11| = 12.

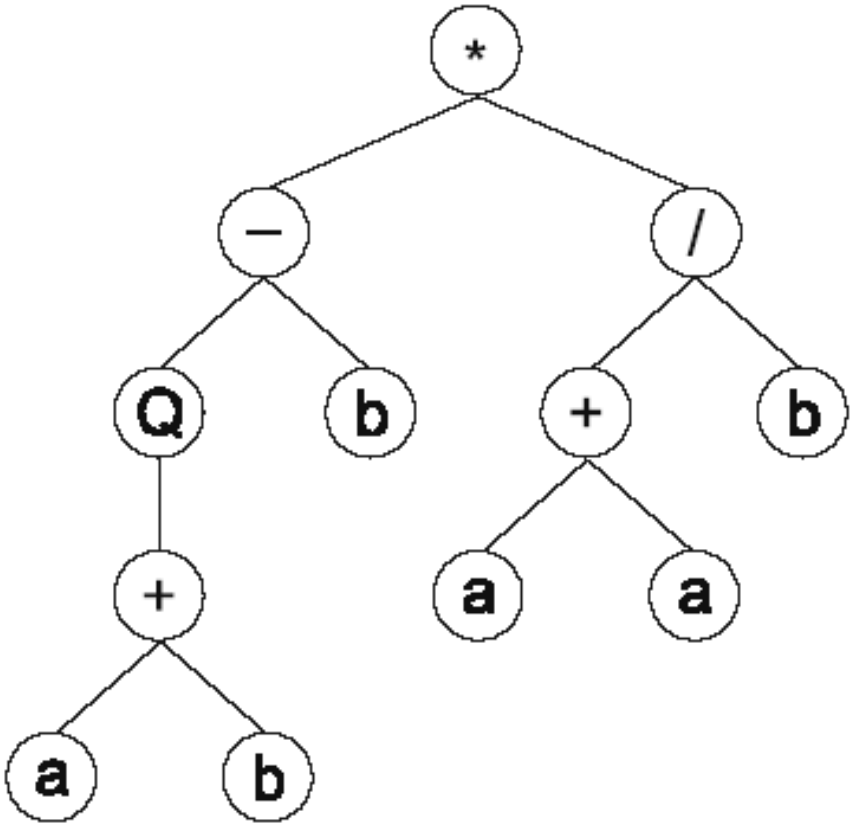
Related GP techniques

- Linear Genetic Programming
- Gene Expression Programming
- Cartesian Genetic Programming
- Grammatical Evolution
- Multi Expression Programming
- Traceless Genetic Programming

Linear Genetic Programming

```
void LGP(double r[8])
{
R[0] = a;
R[1] = b;
…
    r[0] = r[5] + 73;
    r[7] = r[3] – 59;
    r[2] = r[5] + r[4];
    r[6] = r[7] * 25;
    r[1] = r[4] – 4;
    r[7] = r[6] * 2;
}
```

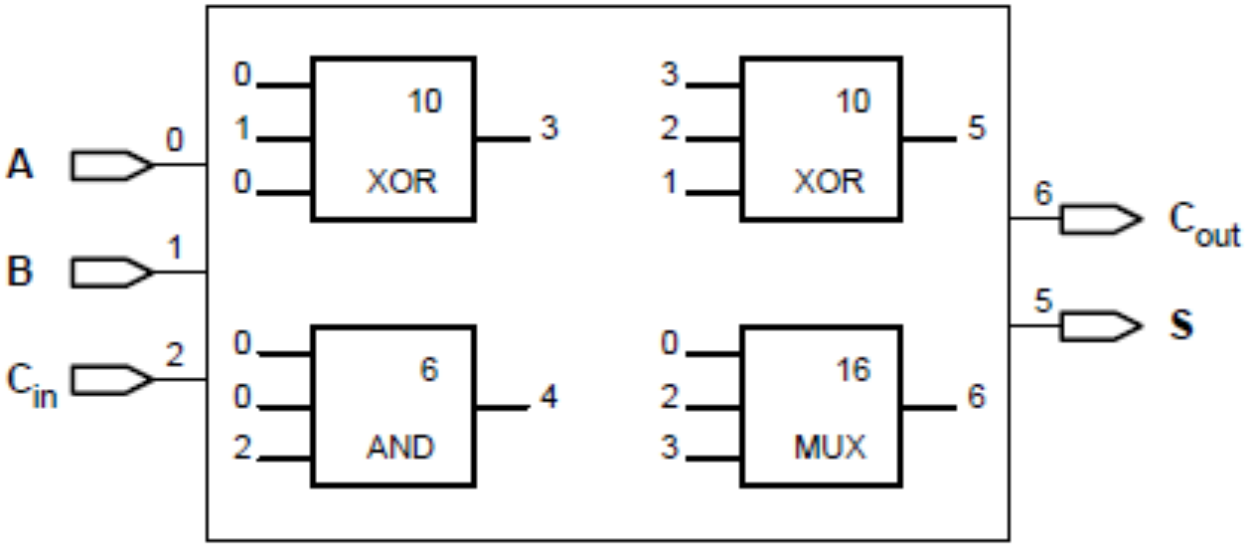
Gene Expression Programming



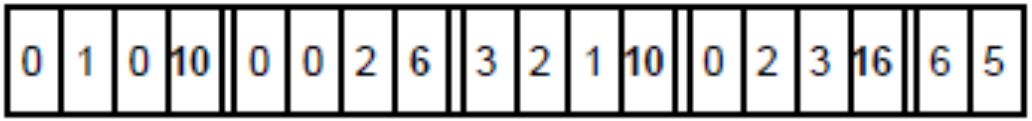
C=*-/Qb+b+aaab**abaab**

E=*-/Qb+b+aaab

CARTESIAN GP

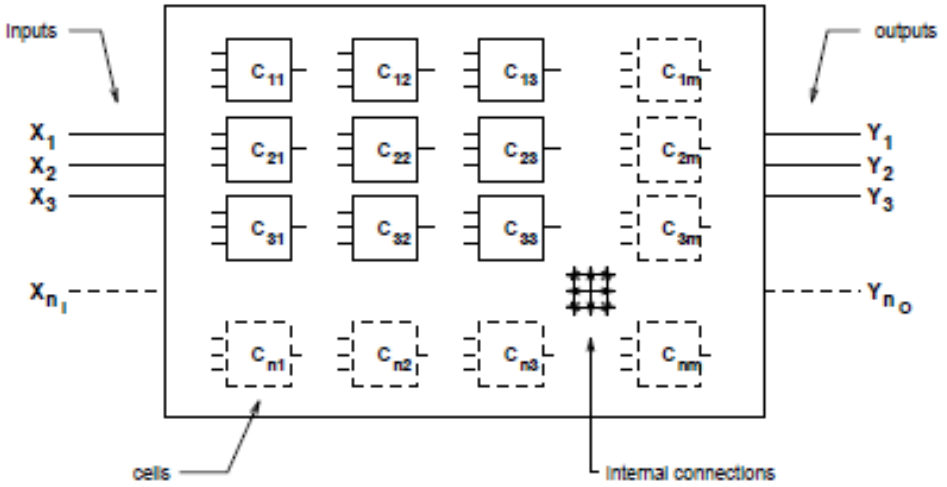


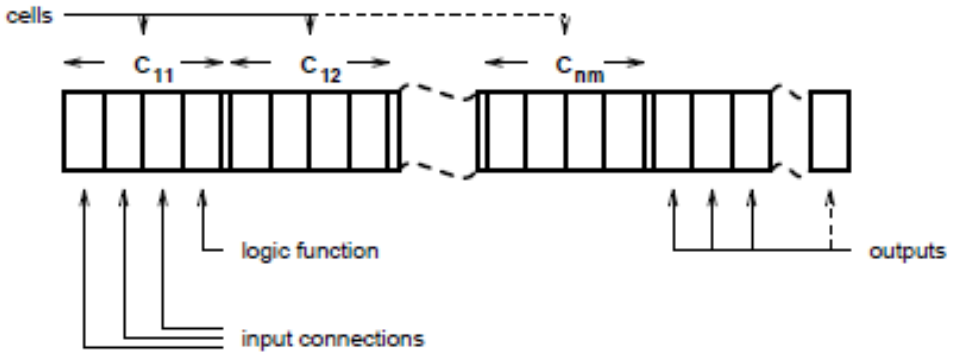
(a)



(b)

CARTESIAN GP





Multi Expression Programming

- Linear representation of chromosomes
- Similar to code with 3 addresses
- Similar to Linear GP and Cartesian GP
- *Encodes multiple solutions within a chromosome*

Example of MEP chromosome *C*

A set of operators: $F = \{+, *\}$, and a set of terminals: $T = \{a, b, c, d\}$.

1: *a*
2: *b*
3: + 1, 2
4: *c*
5: *d*
6: + 4, 5
7: * 3, 5

Decoding MEP chromosomes

#	Gene	Expression
1	*a*	$E_1 = a$
2	*b*	$E_2 = b$
3	+ 1, 2	$E_3 = a + b$
4	*c*	$E_4 = c$
5	*d*	$E_5 = d$
6	+ 4, 5	$E_6 = c + d$
7	* 3, 5	$E_7 = (a + b) * d$

Fitness assignment

- The fitness of an expression

$$f(Ei) = \text{sum } |oki - ek|,\ k = 1,\ n$$

where oki is the obtained value, ek is the expected value and n is the number of training data.

- The fitness of the entire chromosome

$$f(C) = \min f(Ei)$$

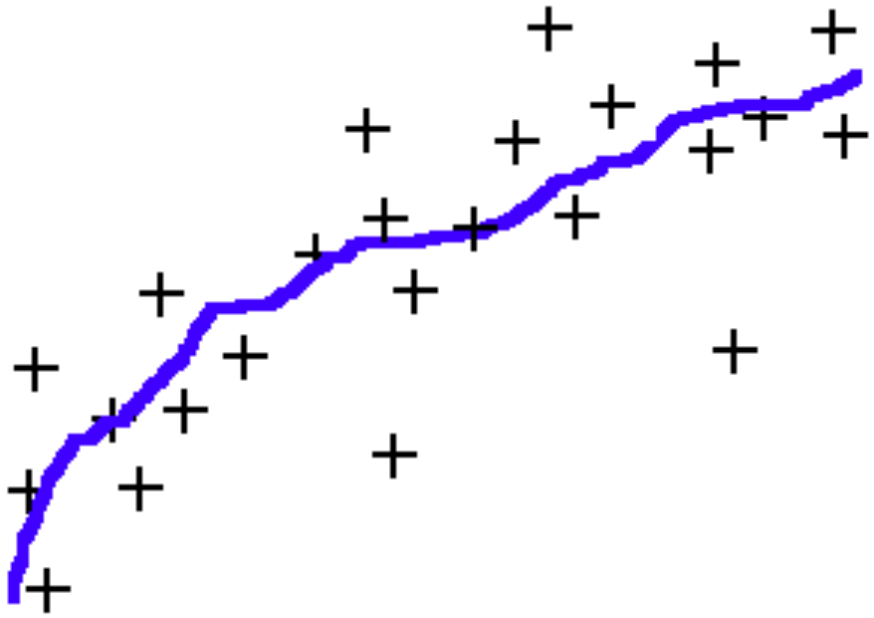
Applications of GP

- Any kind of problem where we have some inputs and we want some outputs!
- There must be a relationship between inputs and outputs!
  - You cannot predict the weather for tomorrow taking into account the score of a soccer game.

Basic applications

- Regression
- Classification
- Computing primitives for a given function.

Regression

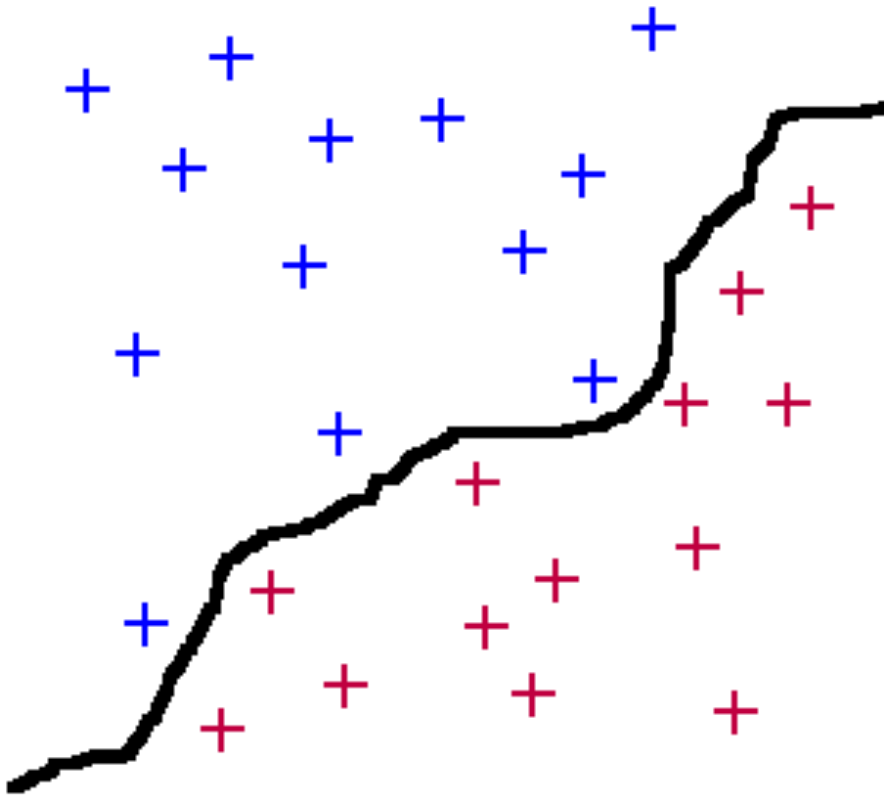


REGRESSION

- m training data
  - each training data is an array of $n+1$ values
- $x_{11}, x_{12}, x_{13}, \ldots, x_{1n}, f_1$
- ….
- $x_{m1}, x_{m2}, x_{m3}, \ldots, x_{mn}, f_m$
  - $f_1, f_2, \ldots, f_m$ are real values

- Fitness = $\square |f_k - o_k| \quad 1 \le k \le m$
  - o_k is the obtained output for the k^{th} training data.

CLASSIFICATION



CLASSIFICATION

- *m* training data
- each training data is an array of *n*+1 values
- $x_{11}, x_{12}, x_{13}, \ldots, x_{1n}, f_1$
- ….
- $x_{m1}, x_{m2}, x_{m3}, \ldots, x_{mn}, f_m$
- $f_1, f_2, \ldots, f_m$ are integer values representing the class (1^{st} class, 2^{nd} class, etc)

FITNESS FOR CLASSIFICATION

- 2 classes
  - The result of the expression is < 0 then that item (training data) belongs to class 1
  - Otherwise it belongs to class 2

- *p* classes
  - each data is classified to the nearest class (ex. if the output is 2.7 and we have three classes (1, 2 and 3), that data is considered as belonging to class 3.

- Fitness = the number of incorrectly classified training data

Practical problems – data analysis (taken from PROBEN 1)

- Building
  - Prediction of energy consumption in a building Try to predict the hourly consumption of electrical energy hot water and cold water based on the date time of day outside temperature outside air humidity solar radiation and wind speed.

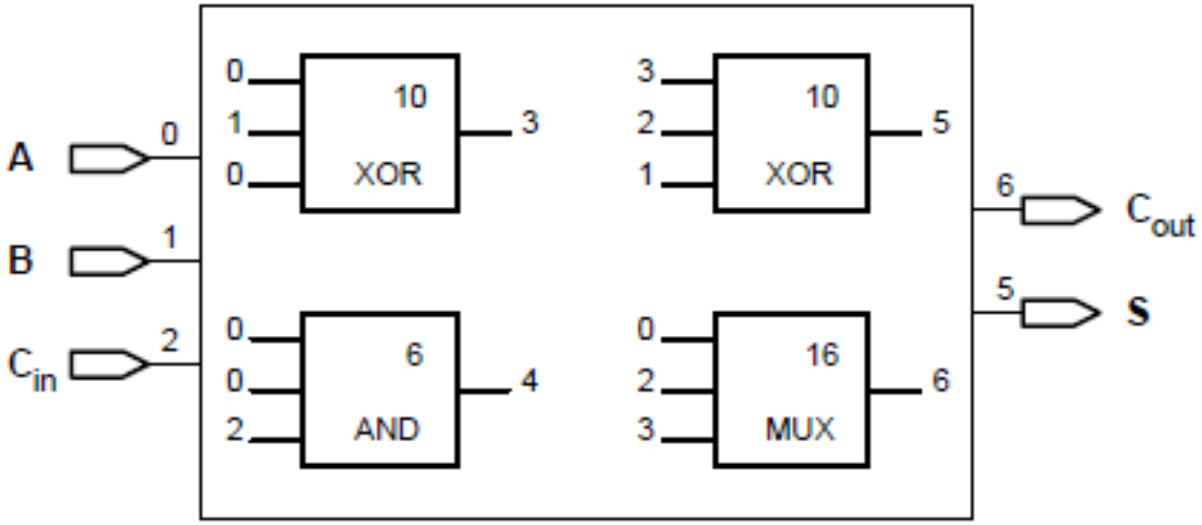
- Heart
  - Predict heart disease Decide whether at least one of four major vessels is reduced in diameter by more than 50%. The binary decision is made based on personal data such as age sex smoking habits subjective patient pain descriptions and results of various medical examinations such as blood pressure and electro cardiogram results.

Computing primitives

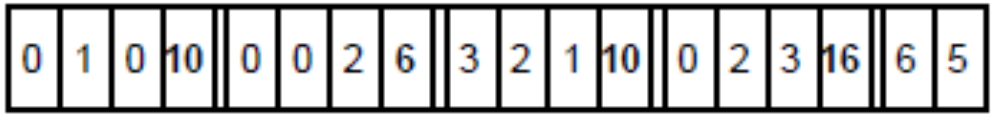
- Given f
- Find F, such that $F' = f$

- Solution:
  - Evolve expressions …
  - Fitness …

1-bit Adder with carry

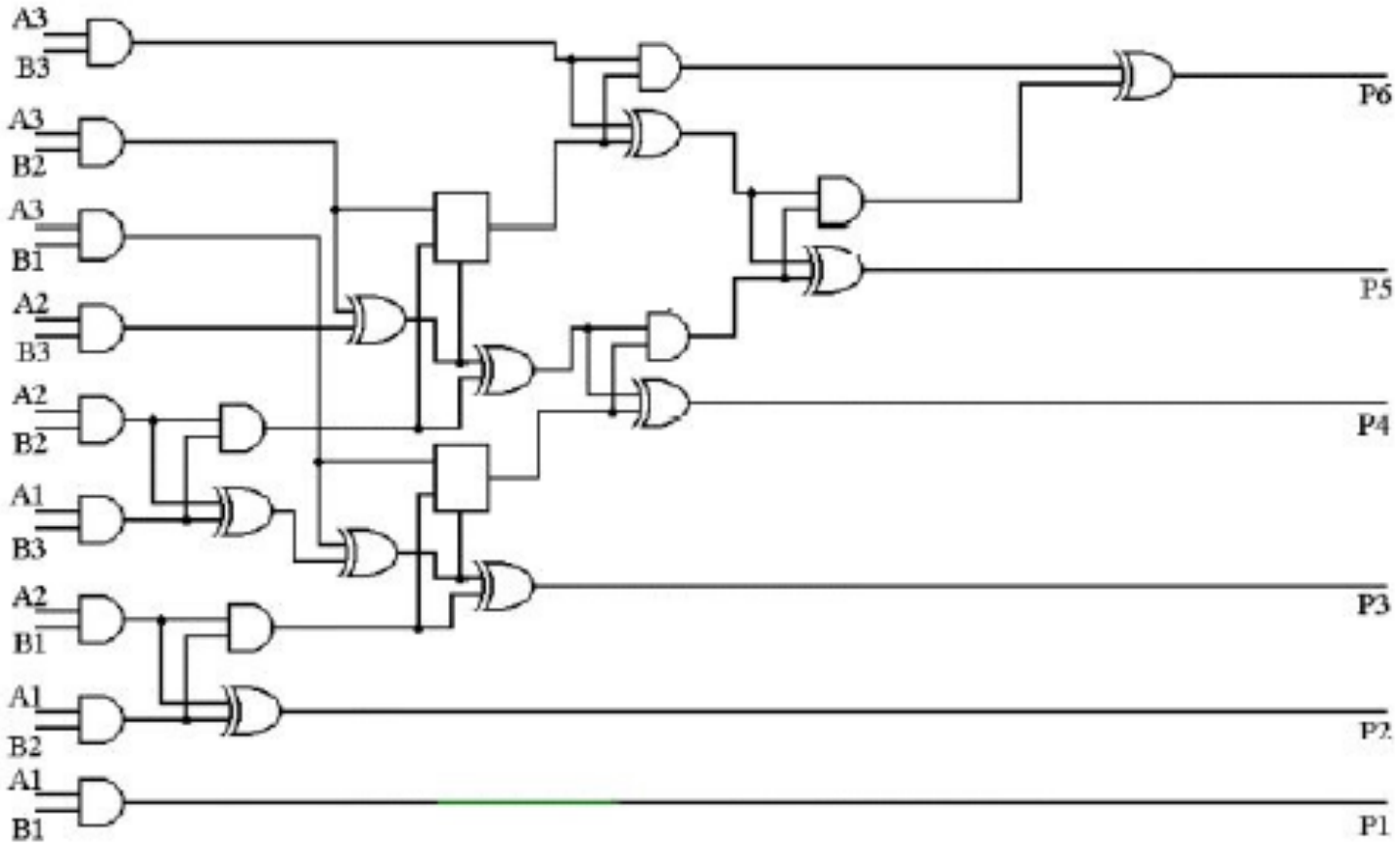


(a)



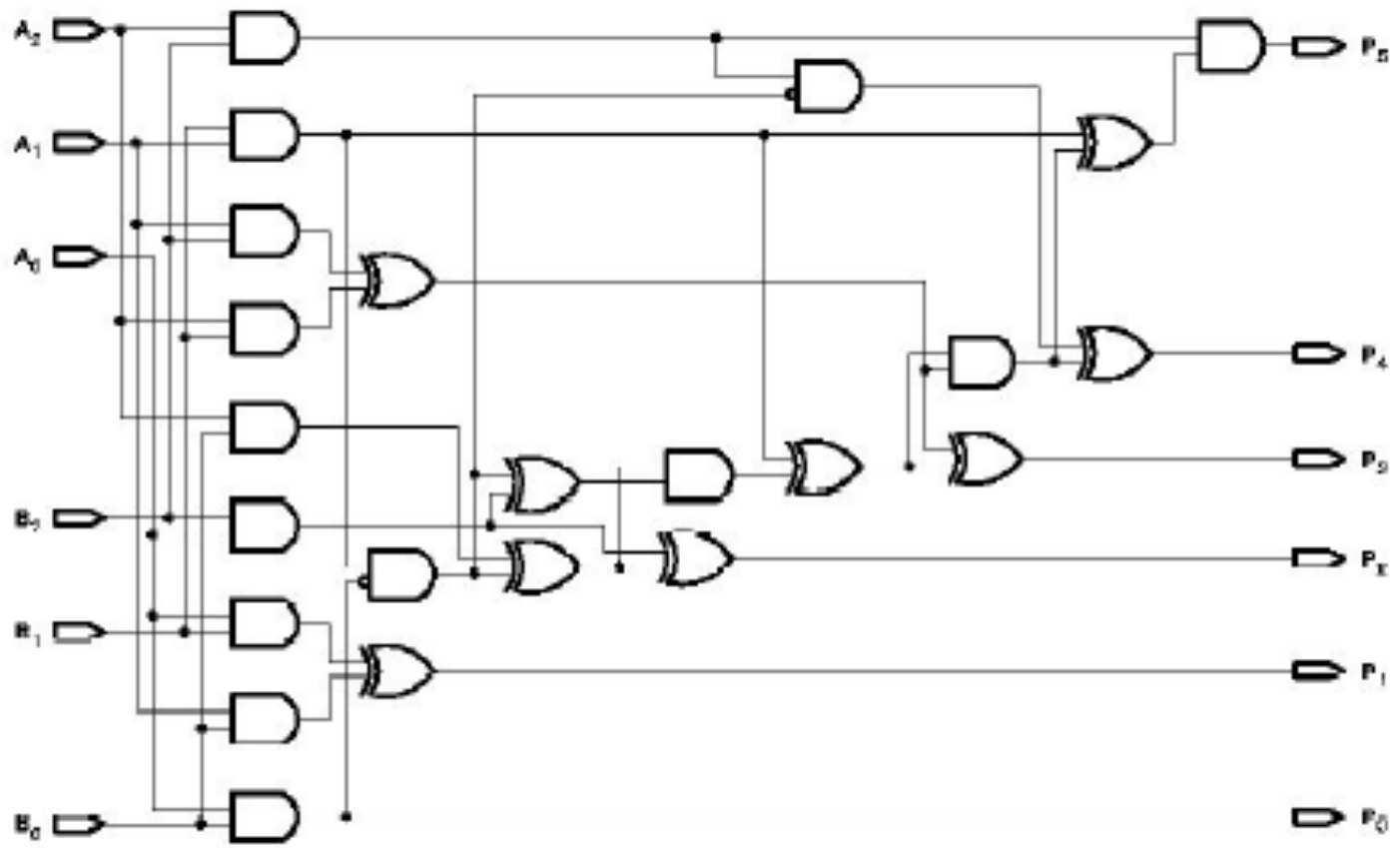
(b)

Digital Circuits – three bits multipliers



Conventional circuit (human designed) has 30 gates.

Digital Circuits – three bits multipliers



Best evolved circuit has 23 gates.