Genetic Programming

- Developed in USA during 90's
- Patented by J. Koza
- Solves typical problems:
  - Prediction, classification, approximation, programming
- Properties
  - Competitor of neural networks
  - Need for huge populations (thousands)
  - Very slow convergence
- Specialties:
  - Non-linear chromosomes: trees, graphs
  - Mutation possible but not necessarily

Problem of automatic programming

"How can computers learn to solve problems without being explicitly programmed? In other words, how can computers be made to do what is needed to be done, without being told exactly how to do it?"

---Attributed to Arthur Samuel - about 1959

Problem of automatic programming

"Genetic programming *is* automatic programming. For the first time since the idea of automatic programming was first discussed in the late 40's and early 50's, we have a set of non-trivial, non-tailored, computer-generated programs that satisfy Samuel's exhortation: 'Tell the computer what to do, not how to do it.' "

– John Holland, University of Michigan, 1997

Introductory example: credit scoring

- Bank wants to distinguish good from bad loan applicants
- Model needed that matches historical data

ID	No children	Income	Status	OK?
ID-1	2	45000	Married	YES
ID-2	0	30000	Single	NO
ID-3	1	40000	Divorced	NO
…				

Introductory example: credit scoring

- A possible model :

IF (CHILDREN = 2) AND (I > 40000) THEN YES ELSE NO

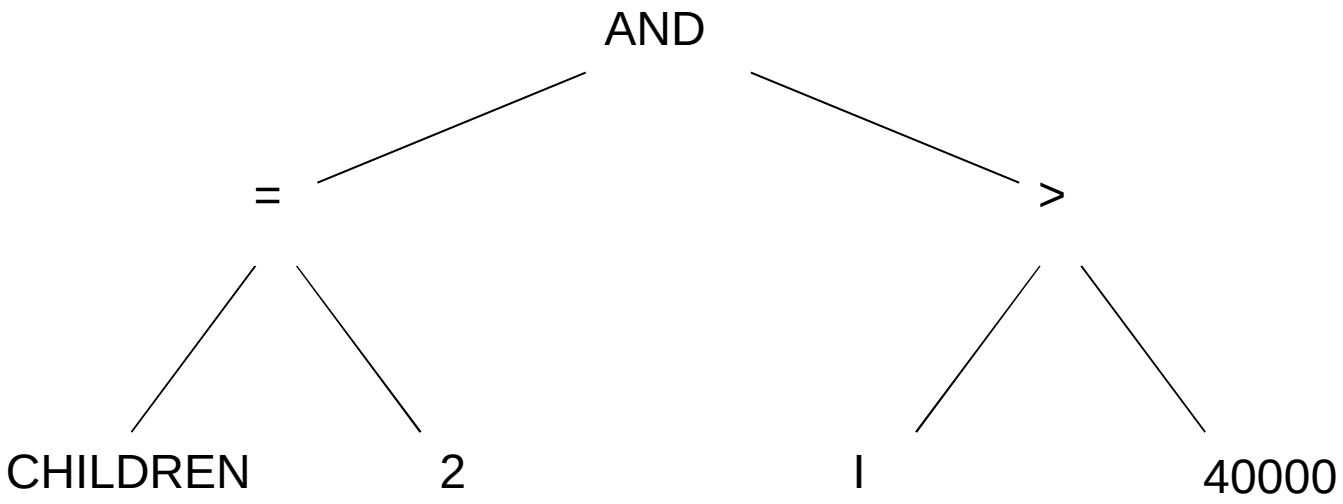
- In general :

IF logical expression THEN YES ELSE NO

- The only unknown is logical expression and therefore, the search domain (genotype) is the space of all logical expressions
- The objective function is a number of samples (data) well described by the logical statement
- Natural representation of logical expressions: tree structure

Introductory example: credit scoring

- IF (CHILDREN = 2) AND (I > 40000) THEN YES ELSE NO
- Can be described by following tree:



Tree based representation

- Tree structure is universal, e.g.
- Arithmetical expression

$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

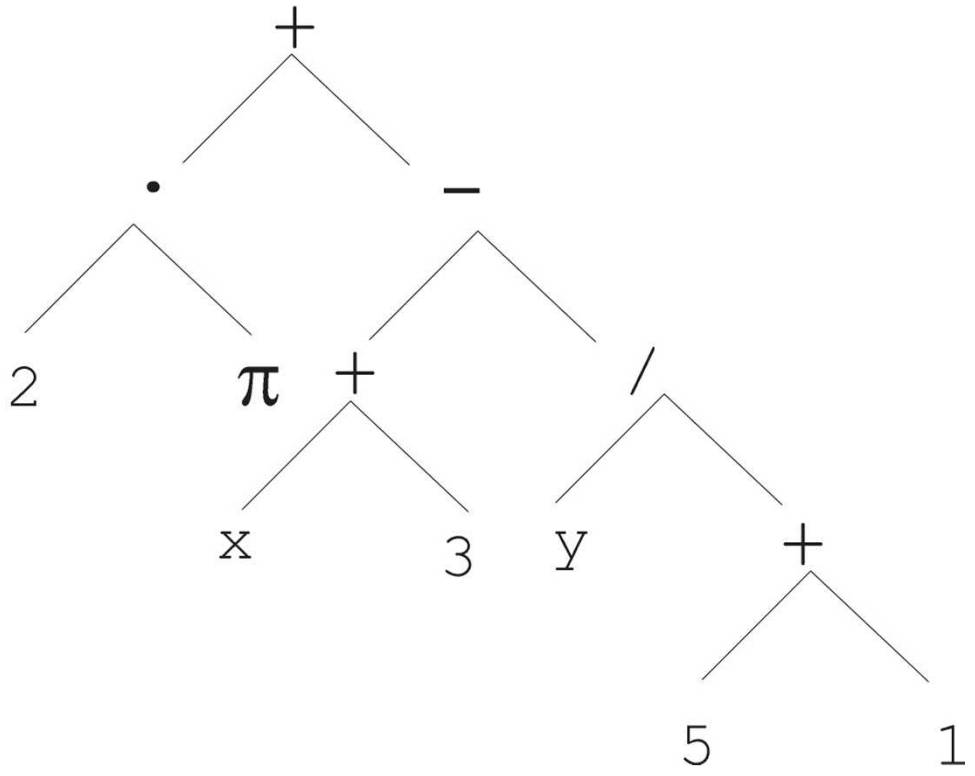
- Logical expression

$$(x \wedge \text{true}) \rightarrow ((\, x \vee y \,) \vee (z \leftrightarrow (x \wedge y)))$$

- Program

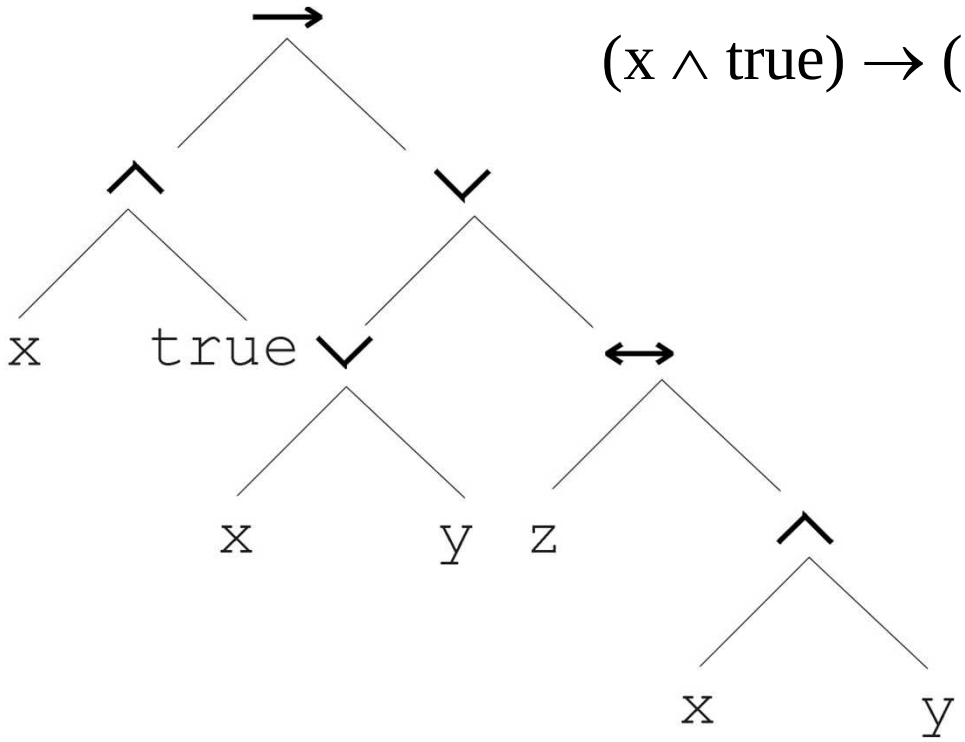
```
i =1;
while (i < 20)
{
        i = i +1
}
```

Tree based representation



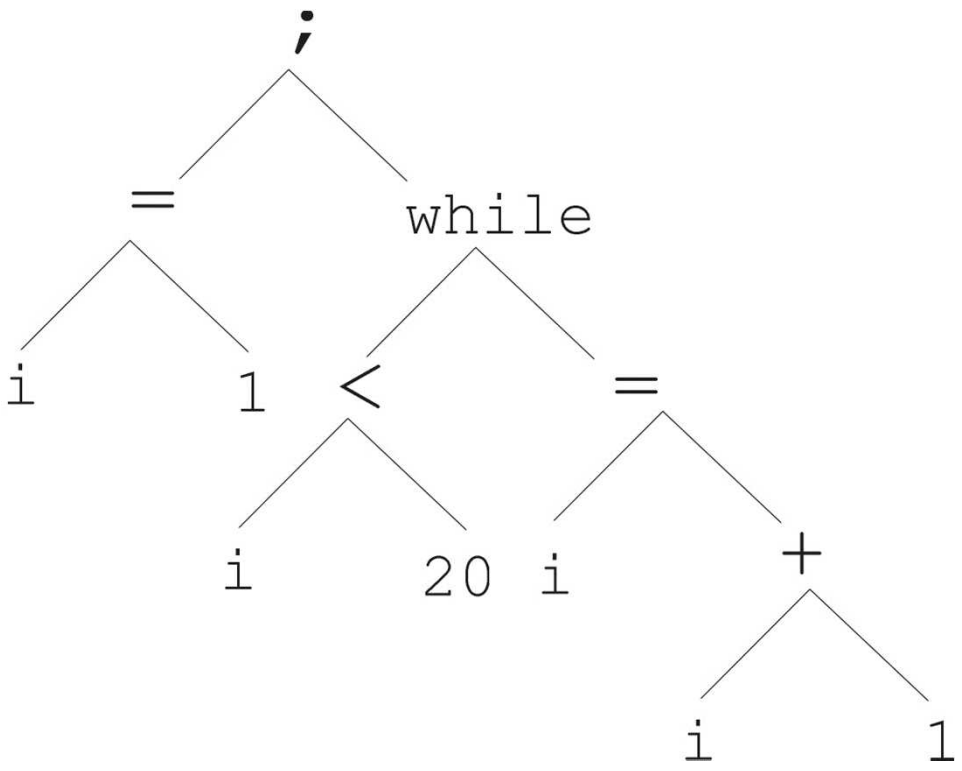
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

Tree based representation



$$(x \wedge \text{true}) \rightarrow ((\ x \vee y\ ) \vee (z \leftrightarrow (x \wedge y)))$$

Tree based representation




```
i =1;
while (i < 20)
{
        i = i +1
}
```

Tree based representation

- In GA, ES, DE chromosomes are linear structures (bit strings, integer string, real-valued vectors, permutations)
- Tree shaped chromosomes are non-linear structures
- In GA, ES, DE the size of the chromosomes is fixed
- Trees in GP may vary in depth and width

Tree based representation

- Symbolic expression can be represented by:
  - Set of terminal symbols T
  - Set of functions F (with the arities of function symbols)

- Adopting the following general recursive definition:
  1. Every $t \in T$ is a correct expression
  2. $f(e_1, \ldots, e_n)$ is a correct expression if $f \in F$, arity(f)=n and $e_1, \ldots, e_n$ are correct expressions
  3. There are no other forms of correct expressions

- In general, expressions in GP are not typed (closure property: any $f \in F$ can take any $g \in F$ as argument)

Algorithm

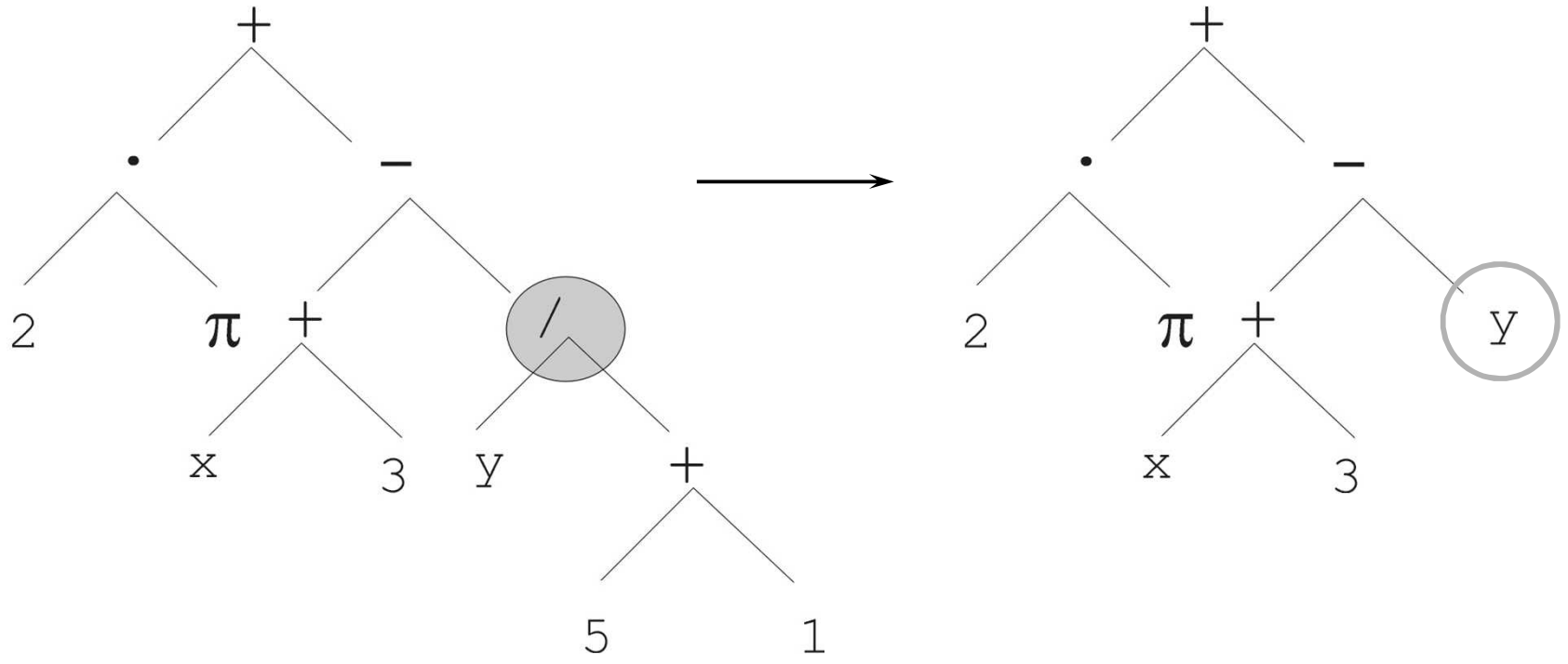
Compare

- GA scheme using crossover AND mutation sequentially (be it probabilistically)

- GP scheme using crossover OR mutation (chosen probabilistically)

Mutation

- Most common mutation: replace randomly chosen subtree by randomly generated tree



Mutation

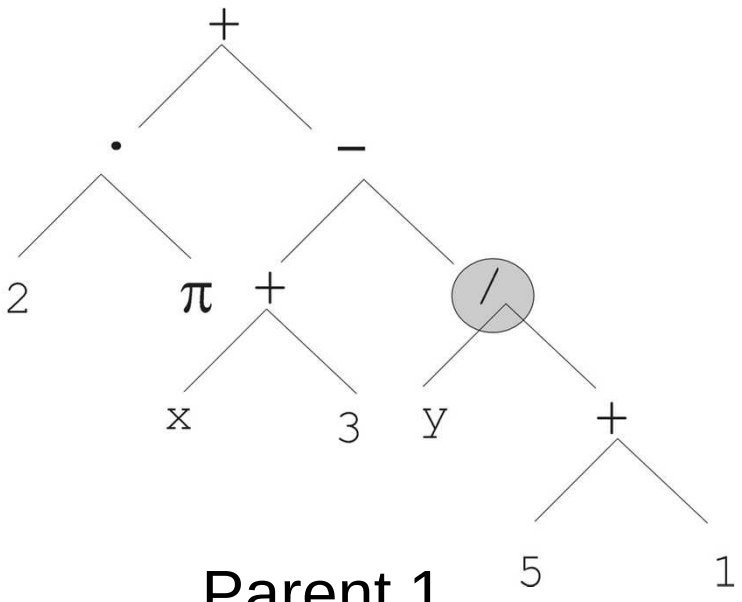
- Mutation has two parameters:
  - Probability p_m to choose mutation vs. recombination
  - Probability to chose an internal point as the root of the subtree to be replaced
  - Remarkably p_m is advised to be $p_m=0$

    [Koza, 1992] or very small, like 0.05

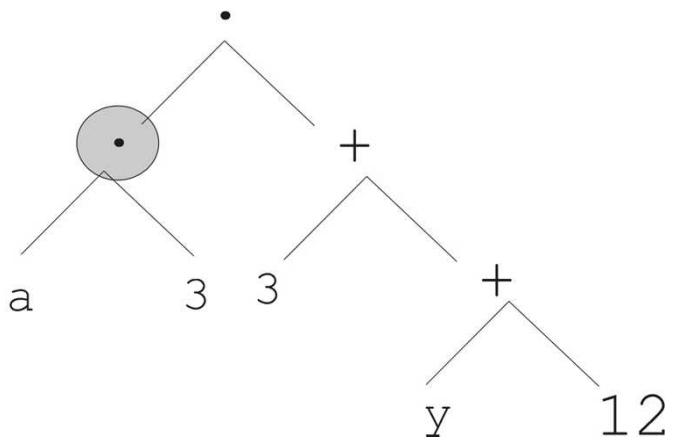
    [Banzhaf et al., 1998]
- The size of the child can exceed the size of the parent

Recombination

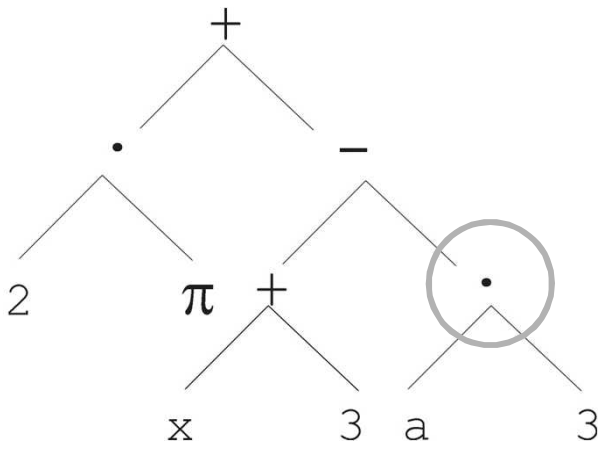
- Most common recombination: exchange two randomly chosen subtrees among the parents
- Over-selection in very large populations
  - rank population by fitness and divide it into two groups:
  - group 1: best x% of population, group 2 other (100-x)%
  - 80% of selection operations chooses from group 1, 20% from group 2
  - for pop. size = 1000, 2000, 4000, 8000 x = 32%, 16%, 8%, 4%
  - motivation: to increase efficiency, %'s come from rule of thumb



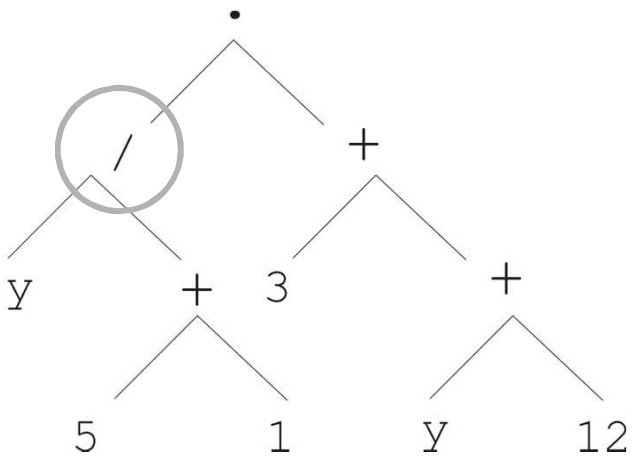
Parent 1



Parent 2



Child 1



Child 2

Initialisation

- Maximum initial depth of trees D_{max} is set

- Full method (each branch has depth = D_{max}):
  - nodes at depth $d < D_{max}$ randomly chosen from function set F
  - nodes at depth $d = D_{max}$ randomly chosen from terminal set T

- Grow method (each branch has depth $\leq D_{max}$):
  - nodes at depth $d < D_{max}$ randomly chosen from $F \cup T$
  - nodes at depth $d = D_{max}$ randomly chosen from T

- Common GP initialisation: ramped half-and-half, where grow & full method each deliver half of initial population

Bloat

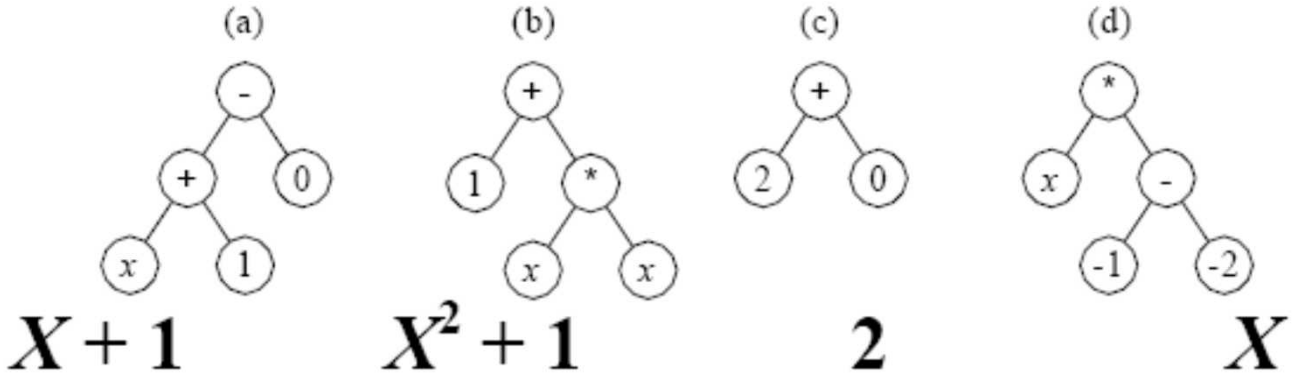
- Bloat = “survival of the fattest”, i.e., the tree sizes in the population are increasing over time
- Ongoing research and debate about the reasons
- Needs countermeasures, e.g.
  - Prohibiting variation operators that would deliver “too big” children
  - Parsimony pressure: penalty for being oversized
  - Multi-objective optimization

Example: symbolic regression

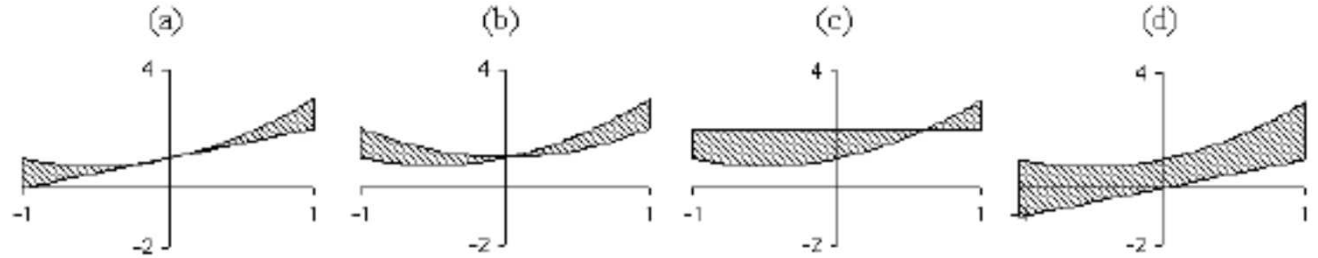
- Given some points in R^2, $(x_1, y_1), \ldots, (x_n, y_n)$
- Find function f(x) s.t. $\forall i = 1, \ldots, n : f(x_i) = y_i$
- Possible GP solution:
  - Representation by F = {+, -, /, sin, cos}, T = R $\cup$ {x}
  - Fitness is the error $err(f) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$
  - All operators standard
  - pop.size = 1000, ramped half-half initialisation
  - Termination: n "hits" or 50000 fitness evaluations reached (where "hit" is if $| f(x_i) - y_i | < 0.0001$)

Example: symbolic regression x^2+x+1

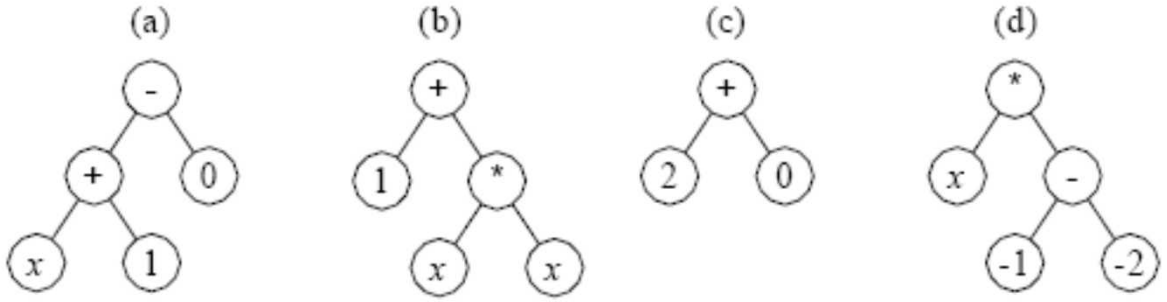
Generation 0



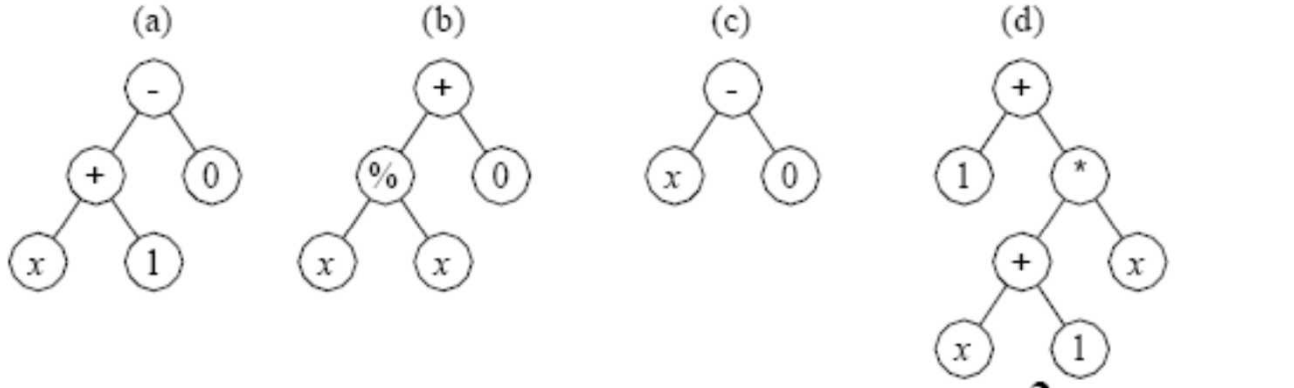
FITNESS



Example: symbolic regression x^2+x+1



Generation 1



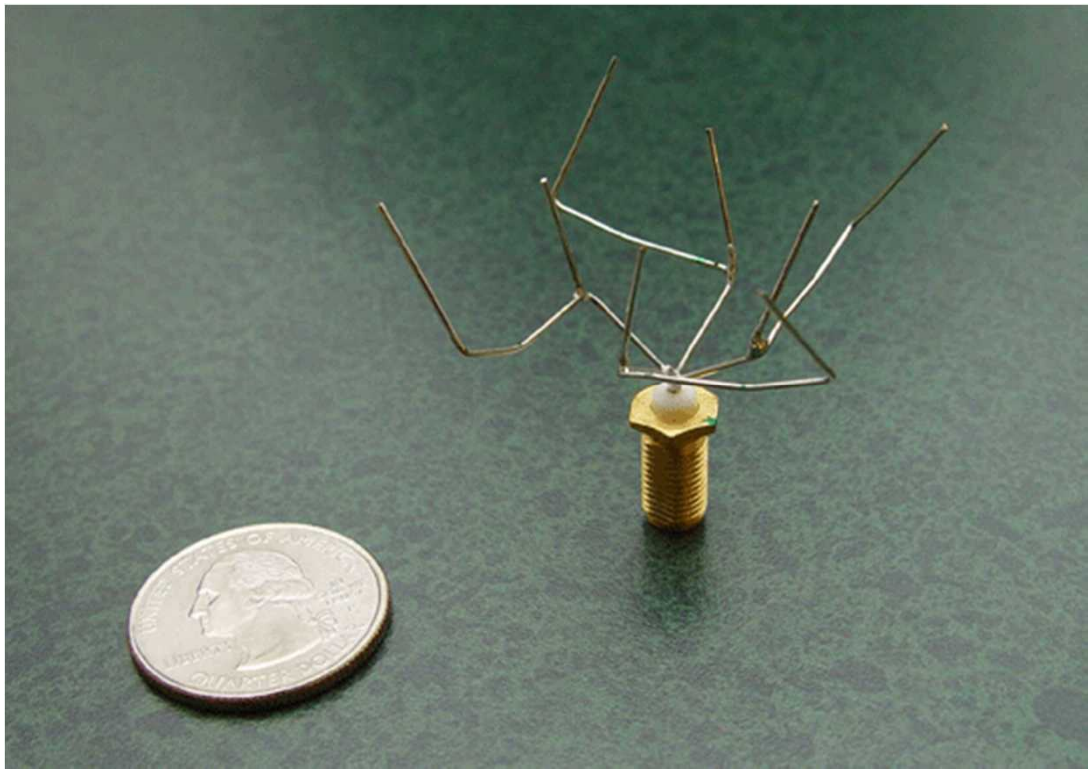
$x+1$ 1 X x^2+x+1

Copy from (a) Mutant from (c) Children form (a) and (b)

Symbolic regression : problem of real numbers

- Real constants as terminals without any change during optimization process
- In case of linear operators, automatic linear regression can be used
- Multi-modal optimization

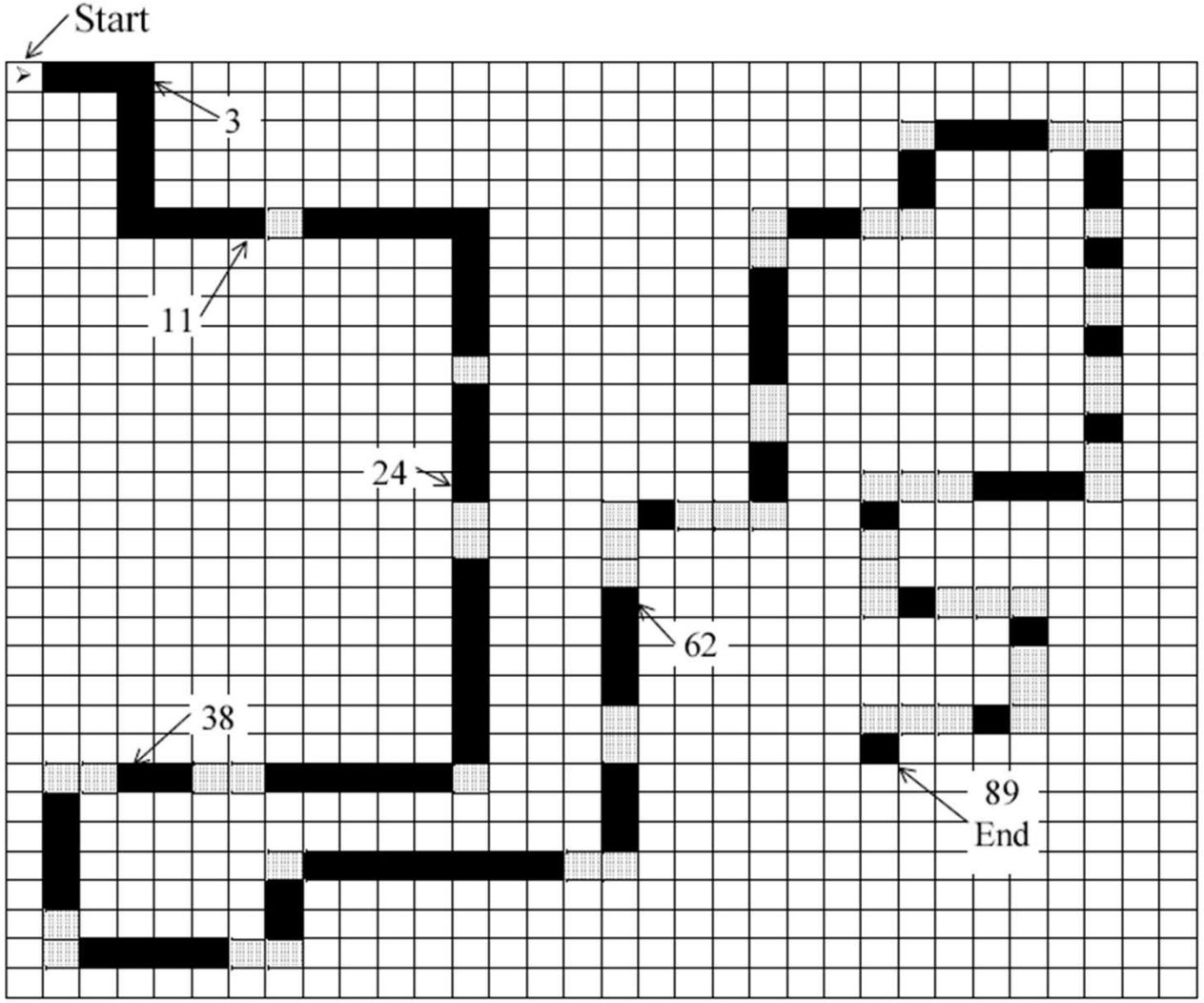
Real-life example:
aka „Human competitive results“



Jason D. Lohn

Evolvable Systems Group
Computational Sciences Division
NASA Ames Research Center
Mountain View, CA USA

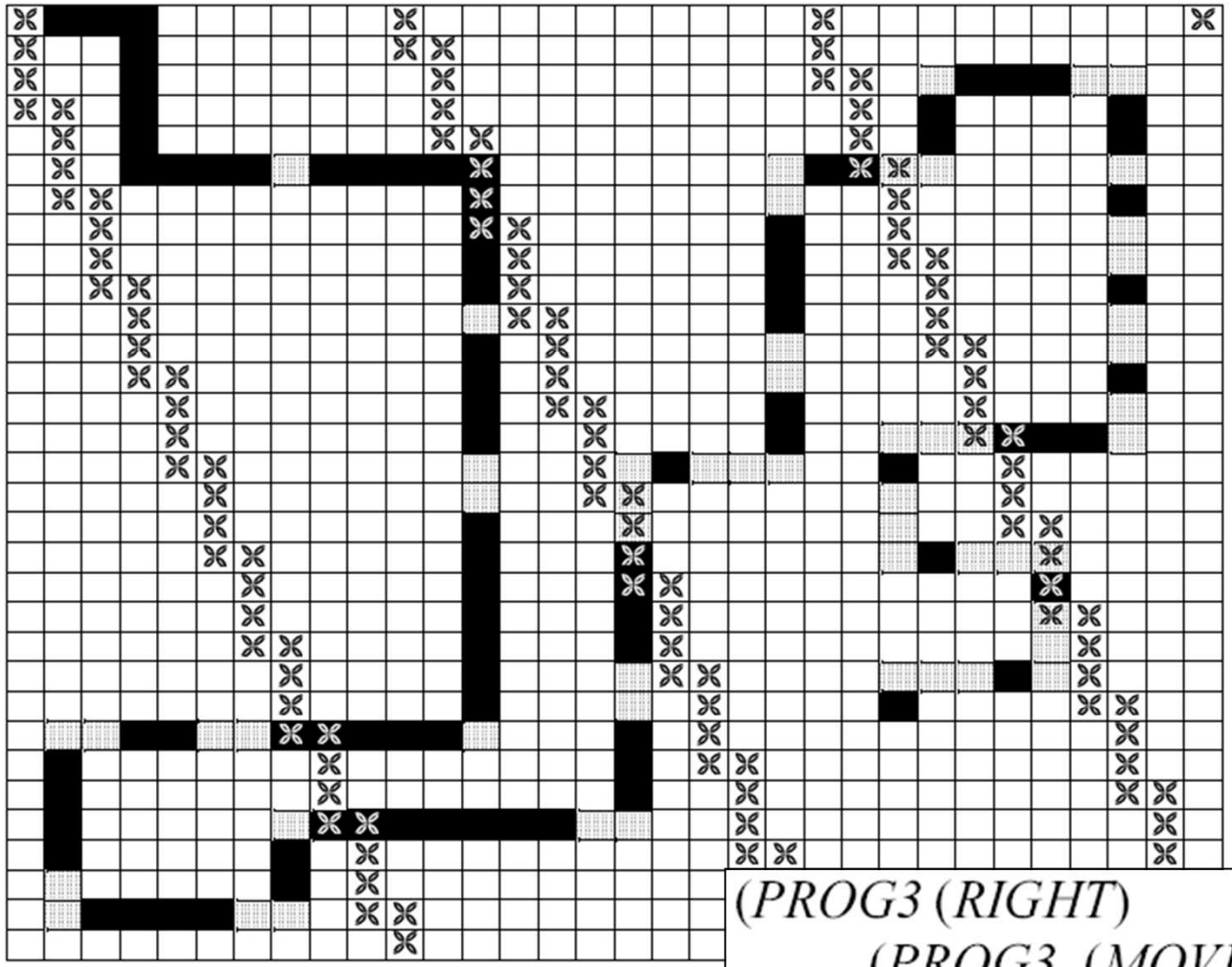
Example – Santa Fe trail



Example – Santa Fe trail

- Set of terminals:
  - T = {MOVE,LEFT, RIGHT}
- Set of functions:
  - F = {IF-FOOD-AHEAD,PROG2,PROG3} with aritas {2,2,3}
- Fitness:
  - Number of gathered food in 400 steps

Typical solution: „stitcher“ (F=12)



```
(PROG3 (RIGHT)
       (PROG3  (MOVE) (MOVE) (MOVE))
       (PROG2  (LEFT) (MOVE)))
```

References

[1] Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.

[2] Koza, J. R. (1994). Genetic Programming II: Automatic Discovery of Reusable Programs.

[3] Koza, J. R. (1999). Genetic Programming III: Darwinian Invention and Problem Solving.

[4] Koza, J. R. (2003). Genetic Programming IV: Routine Human-Competitive Machine Intelligence.

[5] www.genetic-programming.com

References

[6] Vladislavleva, E. (2008). Model-based Problem Solving through Symbolic Regression via Pareto Genetic Programming. PhD thesis, Tilburg University, Tilburg, The Netherlands.

[7] GPLab: A Genetic Programming Toolbox for MATLAB http://gplab.sourceforge.net/

[8] GPdotNET https://github.com/bhrnjica/gpdotnet

A humble plea. Please feel free to e-mail any suggestions, errors and typos to **matej.leps@fsv.cvut.cz**.

Date of the last version: 18.1.2018
Version: 001