Genetic Programming

Chapter 6

Source: Eiben & Smith
Edited by: Hosein Alizadeh
http://webpages.iust.ac.ir/halizadeh/
Computer Engineering Department, Iran University of Science & Technology

GP quick overview

- Developed: USA in the 1990's
- Early names: J. Koza
- Typically applied to:
  - machine learning tasks (prediction, classification…)
- Attributed features:
  - competes with neural nets and alike
  - needs **huge populations** (thousands)
  - **slow**
- Special:
  - **non-linear chromosomes**: trees, graphs
  - mutation possible but not necessary

GP technical summary tableau

Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

Example: Credit Scoring

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

ID	No of children	Salary	Marital status	OK?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
…				

Example: Credit Scoring

- A possible model:

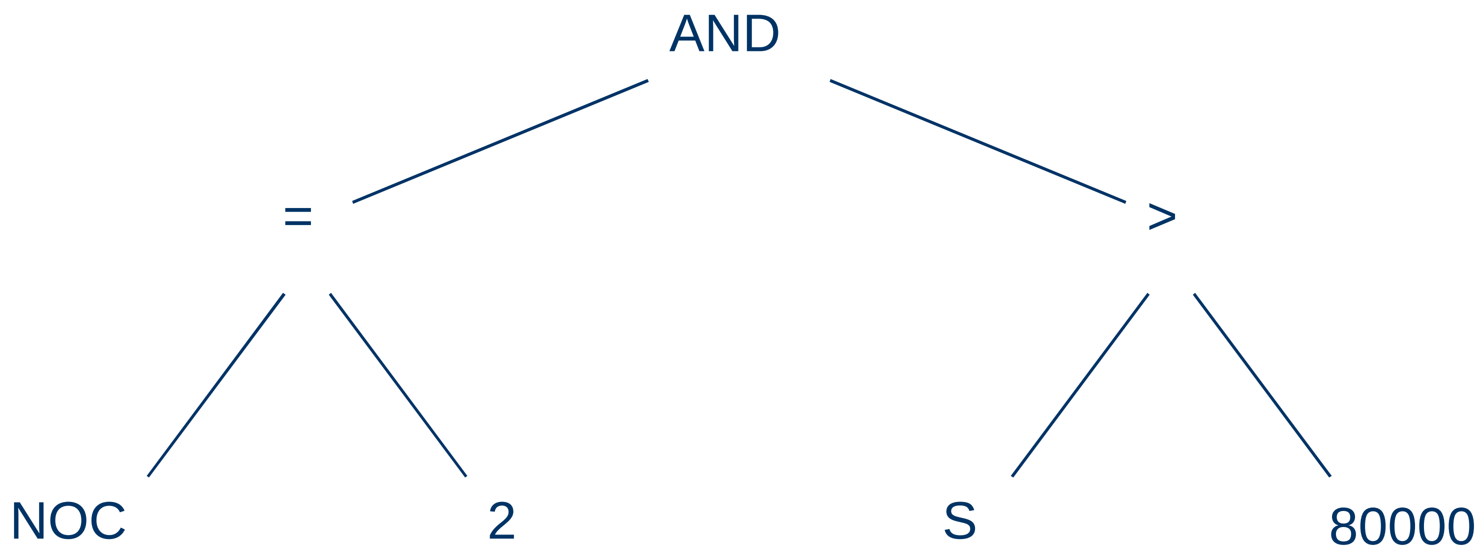
  IF (NOC = 2) AND (S > 80000) THEN good ELSE bad
- In general:

  IF formula THEN good ELSE bad
- Only unknown is the right formula, hence
- Our search space (phenotypes) is the set of formulas
- Natural fitness of a formula: percentage of well classified cases of the model it stands for
- Natural representation of formulas (genotypes) is: **trees**

Example: Credit Scoring

IF (NOC = 2) AND (S > 80000) THEN good ELSE bad

can be represented by the following tree



Tree based representation

- Trees are a universal form, e.g. consider
- Arithmetic formula

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

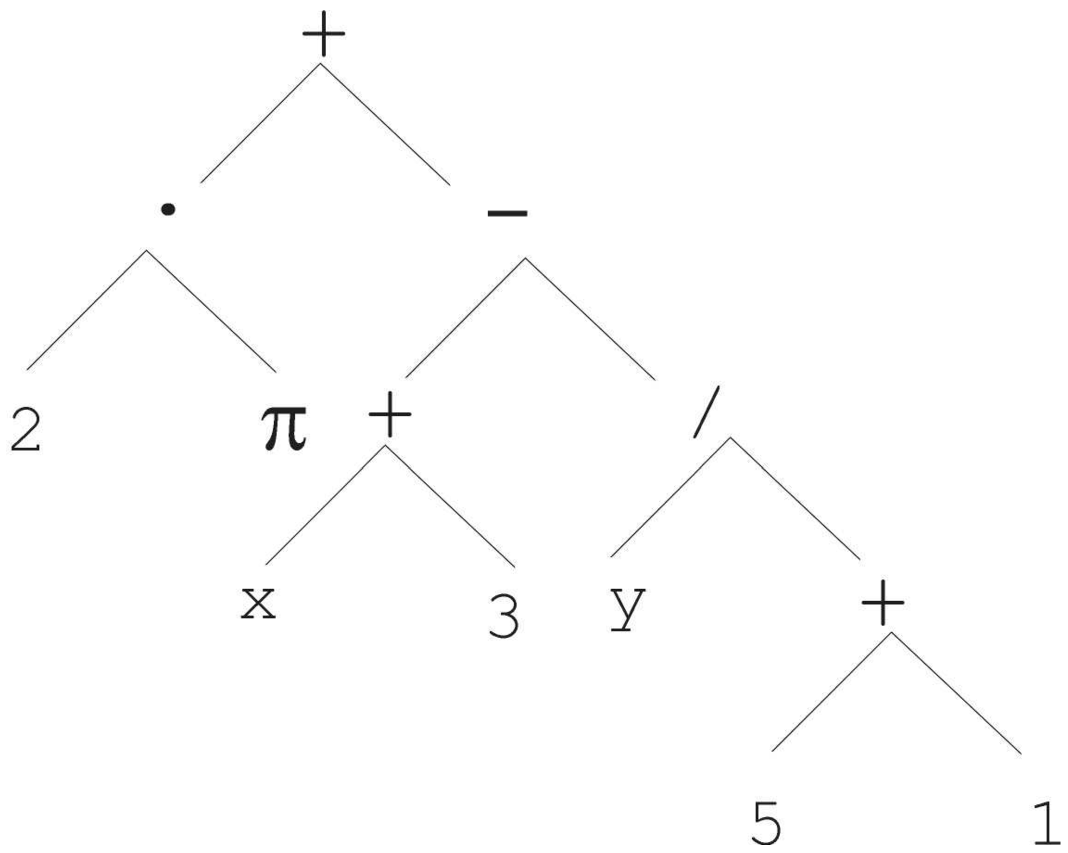
- Logical formula

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

- Program

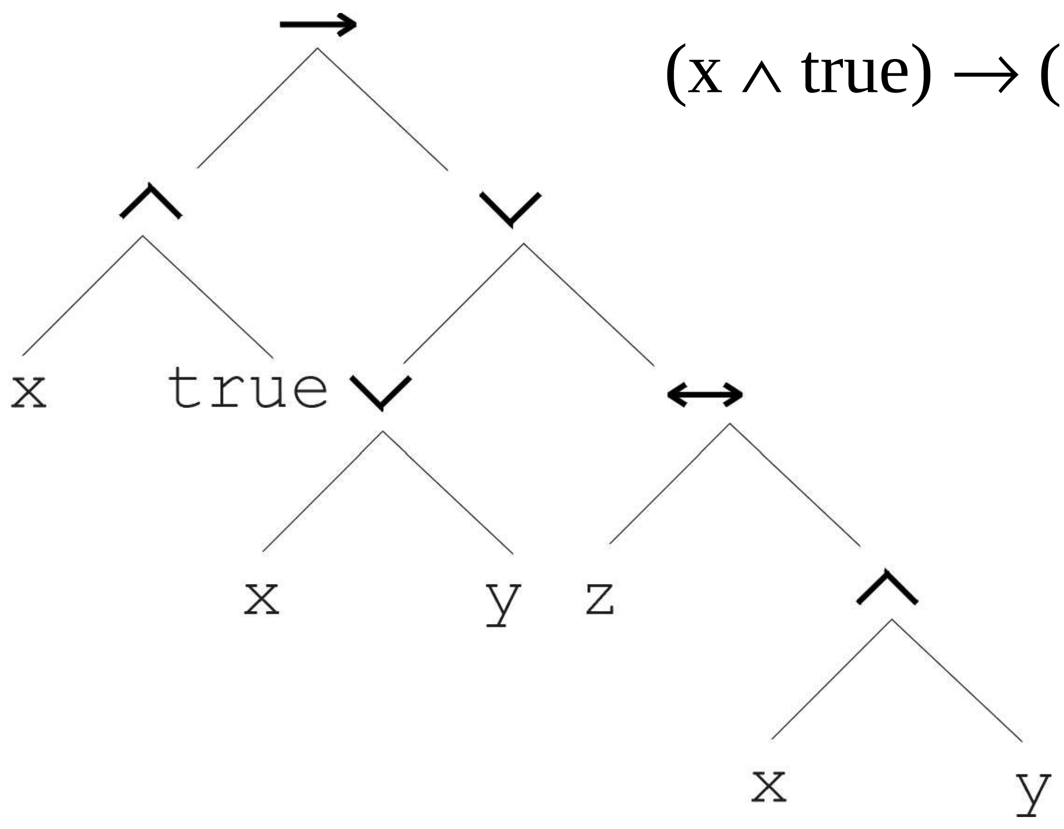
```
i =1;
while (i < 20)
{
        i = i +1
}
```

Tree based representation



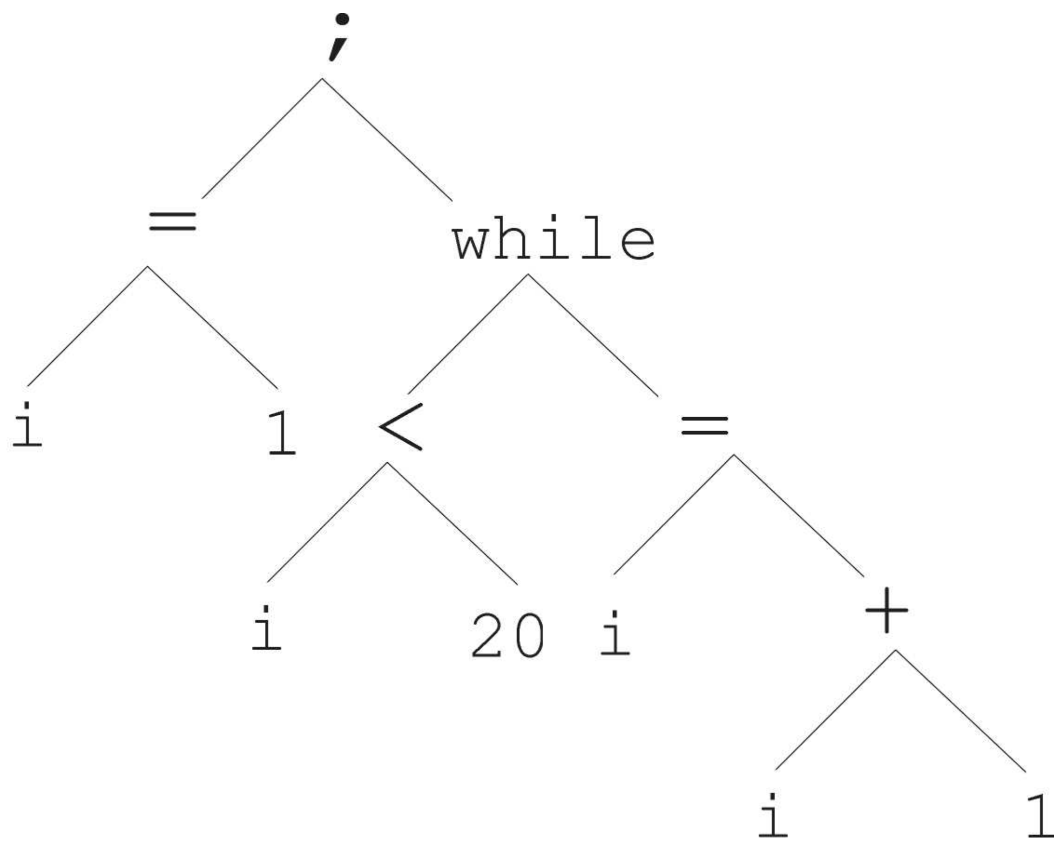
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

Tree based representation



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

Tree based representation




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

Tree based representation in GP

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

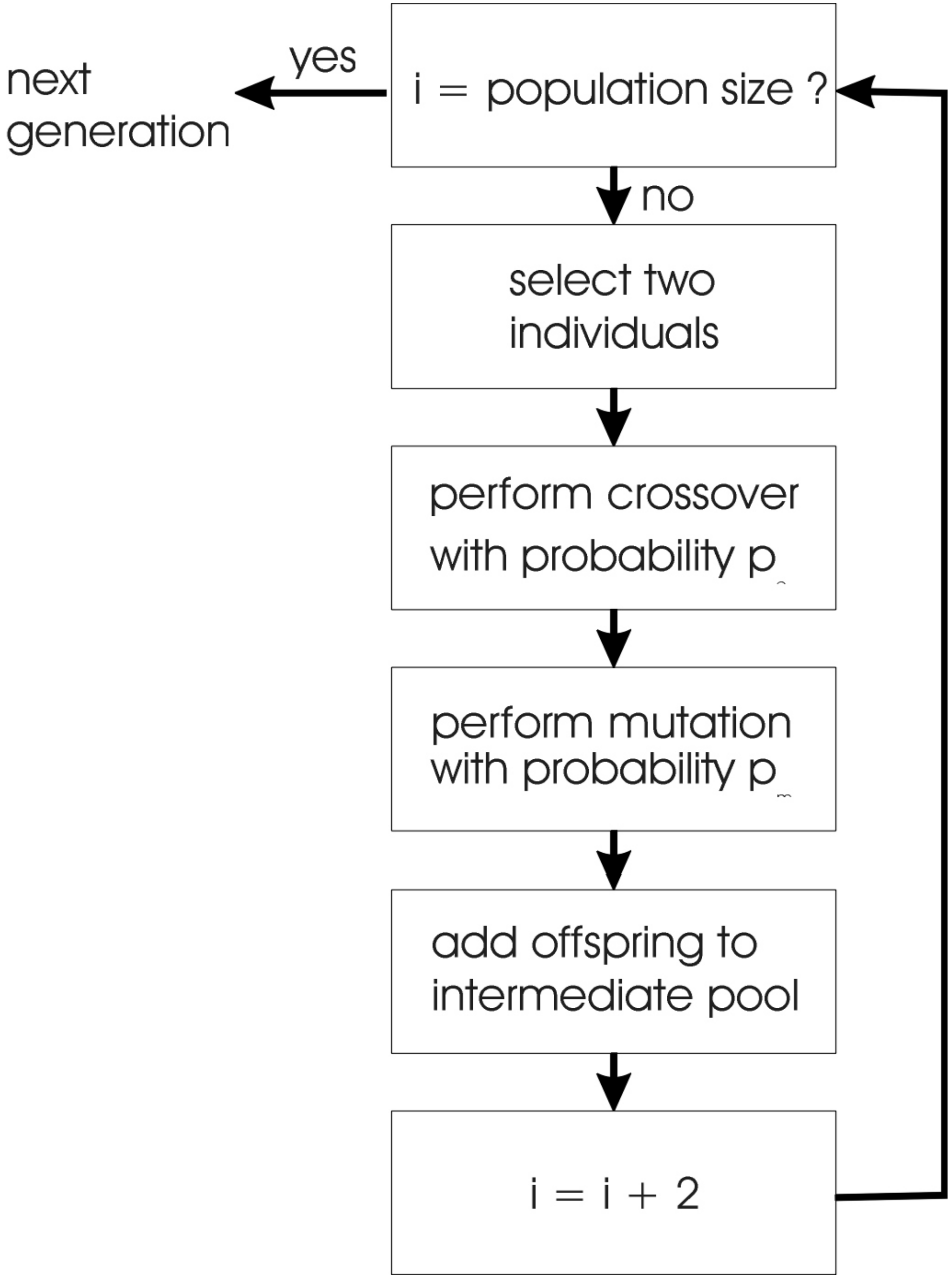
Tree based representation

- Symbolic expressions can be defined by
  - Terminal set T
  - Function set 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

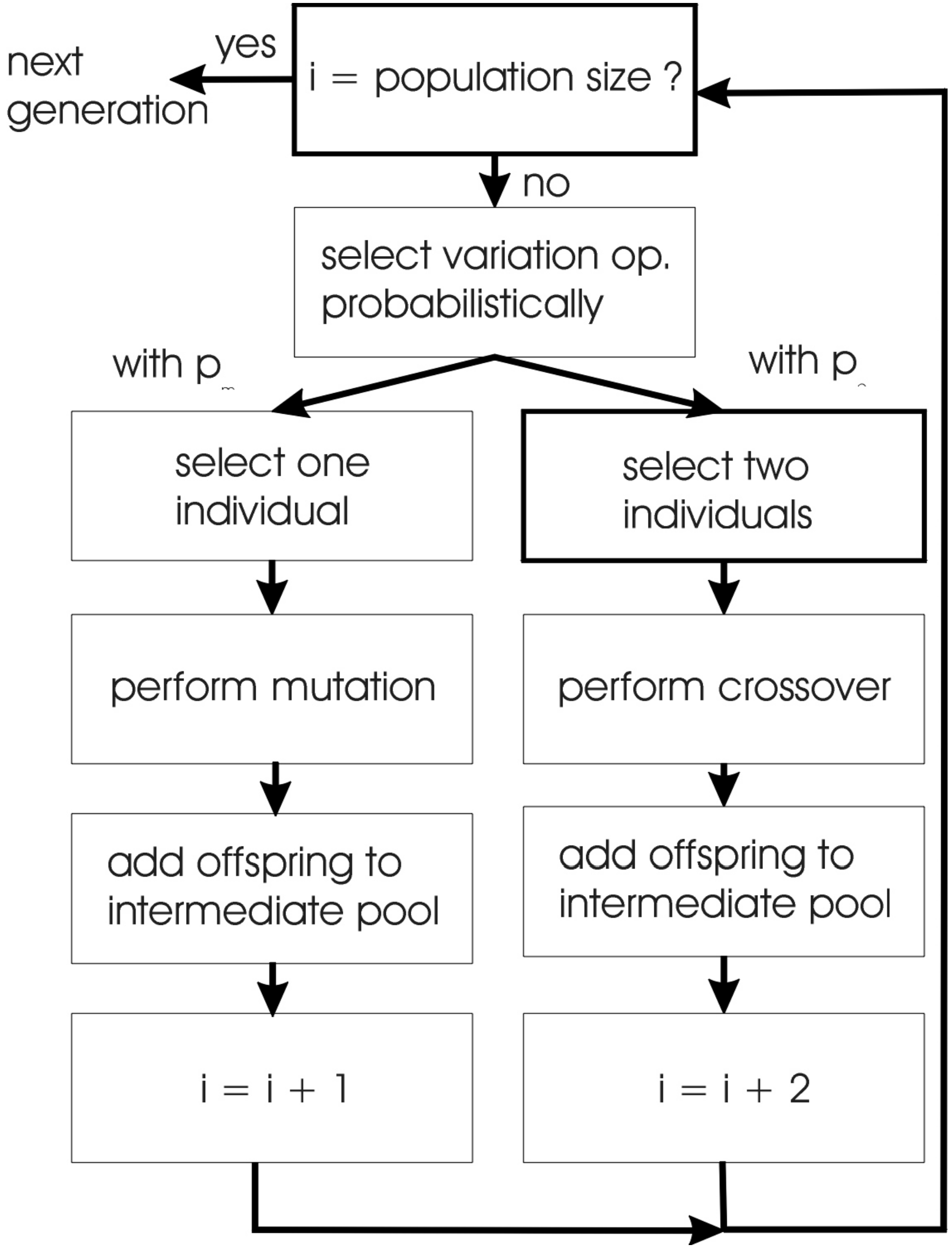
Offspring creation scheme

Compare

- GA uses crossover AND mutation sequentially (be it probabilistically)
- GP scheme uses crossover OR mutation (chosen probabilistically)



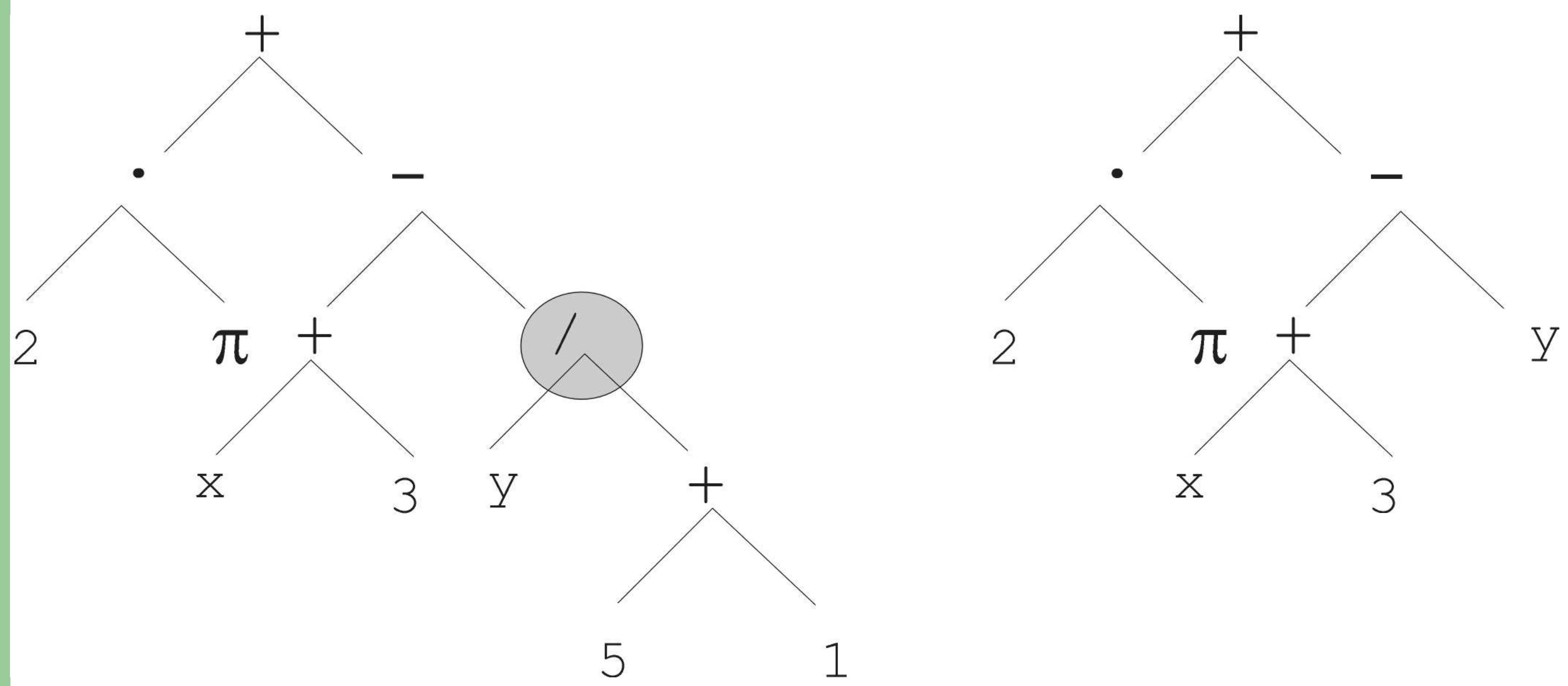
GA flowchart



GP flowchart

Mutation

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

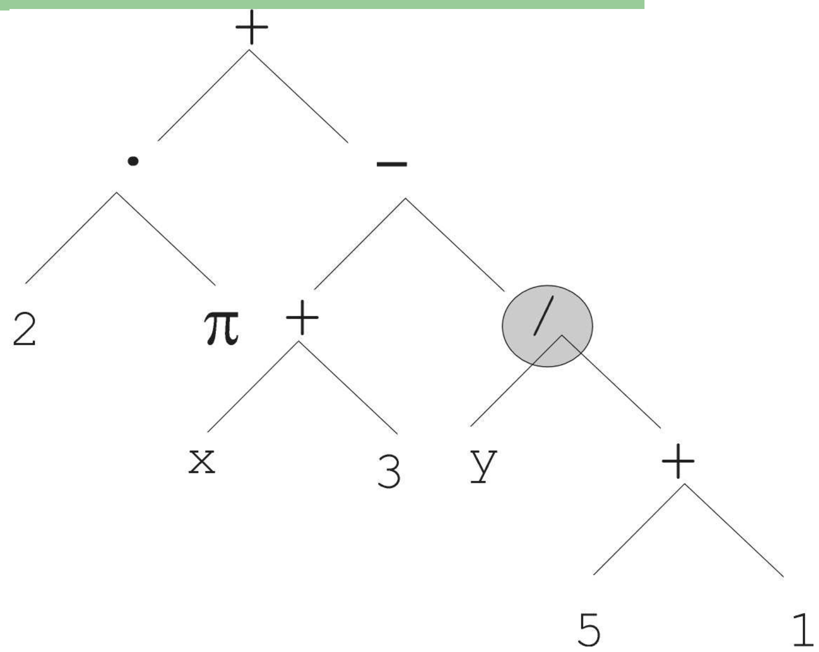


Mutation cont'd

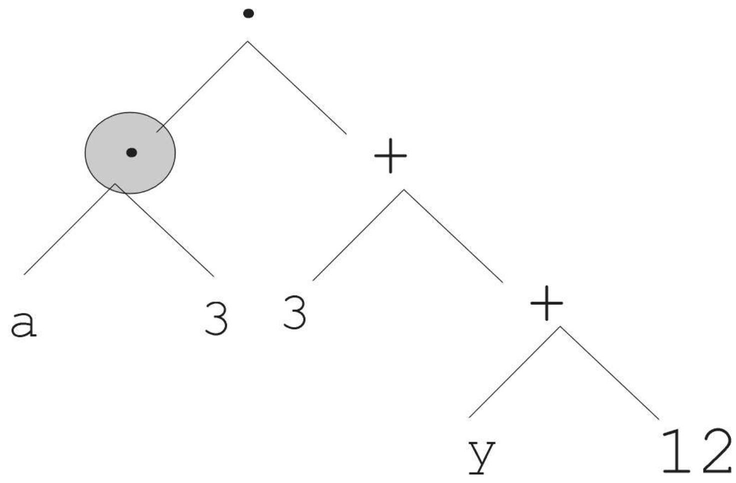
- Mutation has two parameters:
  - Probability p_m to choose mutation vs. recombination
  - Probability to chose an internal point as the root of the subtree to be replaced
- Remarkably p_m is advised to be 0 (Koza'92) or very small, like 0.05 (Banzhaf et al. '98)
- The size of the child can exceed the size of the parent

Recombination

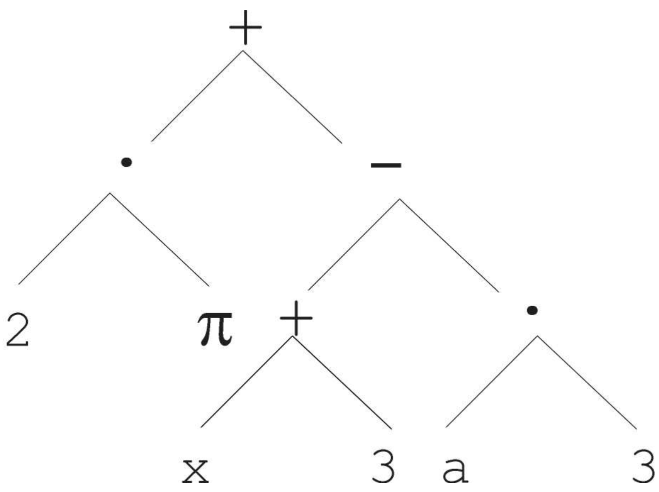
- Most common recombination: exchange two randomly chosen subtrees among the parents
- Recombination has two parameters:
  - Probability p_c to choose recombination vs. mutation
  - Probability to chose an internal point within each parent as crossover point
- The size of offspring can exceed that of the parents



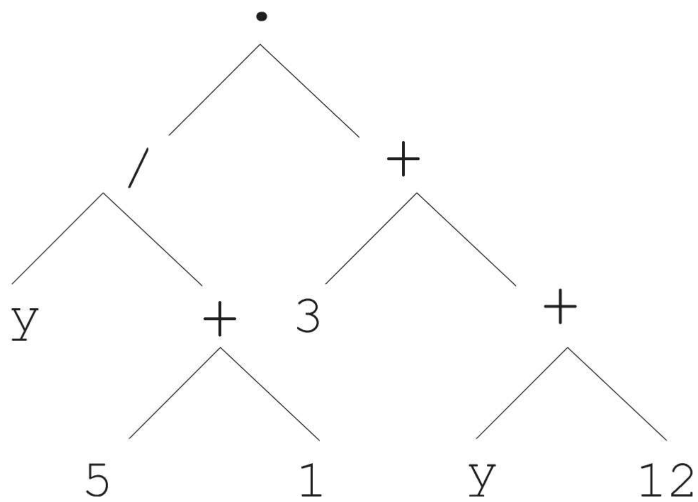
Parent 1



Parent 2



Child 1



Child 2

Selection

- Parent selection is typically fitness proportionate
- 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
- Survivor selection:
  - Typical: generational scheme

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

- Ramped Half-Half Initialisation

Bloat Problem

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

Example application: symbolic regression

- Given some points in $\mathbf{R}^2$, $(x_1, y_1), \ldots , (x_n, y_n)$
- Find function f(x) s.t. $\forall i = 1, \ldots, n : f(x_i) = y_i$
- Possible GP solution:
  - Representation by $F = \{+, -, /, \sin, \cos\}$, $T = \mathbf{R} \cup \{x\}$
  - 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 $| f(x_i) – y_i | < 0.0001$)

Discussion

Is GP:

The art of evolving computer programs ?

Means to automated programming of computers?

GA with another representation?