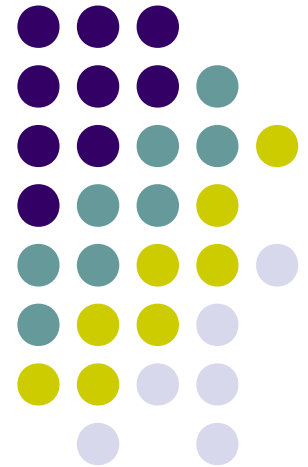
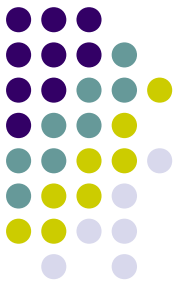


Genetic Programming

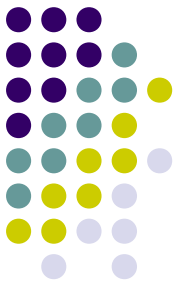




GP quick overview

- 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 (disputed!)

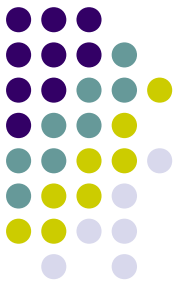
Introductory example: credit scoring



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

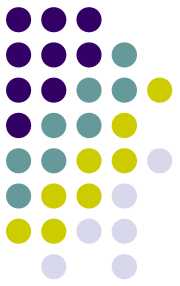
ID	No of children	Salary	Marital status	OK?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
...				

Introductory example: credit scoring

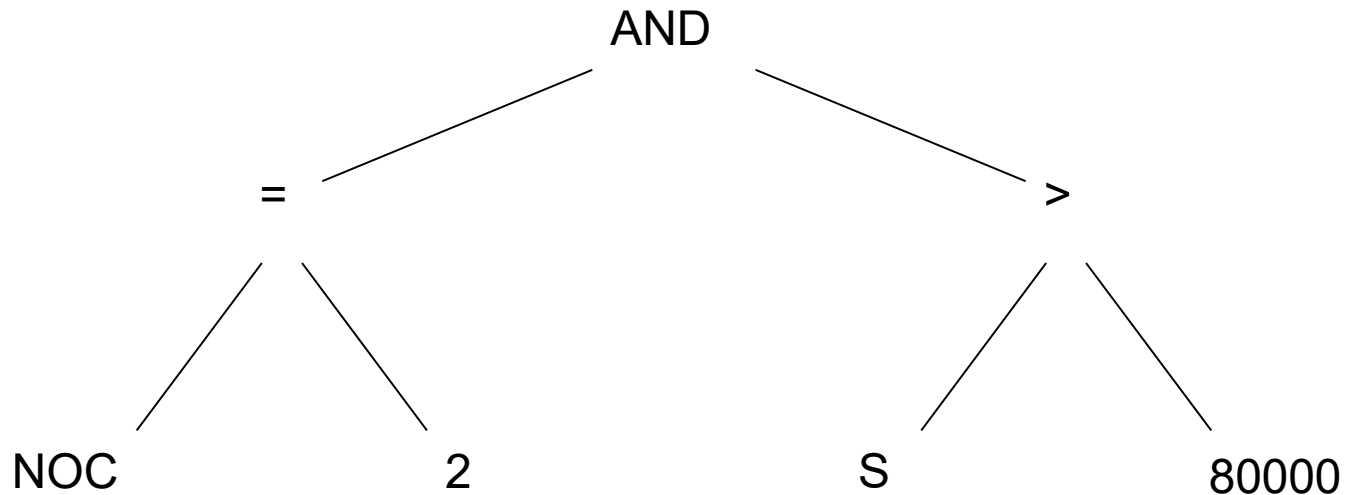


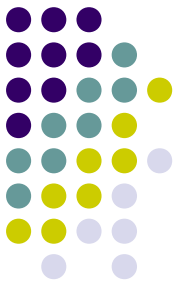
- A possible model:
IF (NOC = 2) AND (S > 80000) THEN good ELSE bad
- In general:
IF formula THEN good ELSE bad
- Only unknown is the right formula, hence
- Our search space (phenotypes) is the set of formulas
- Natural fitness of a formula: percentage of well classified cases of the model it stands for
- Natural representation of formulas (genotypes) is: parse trees

Introductory 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

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

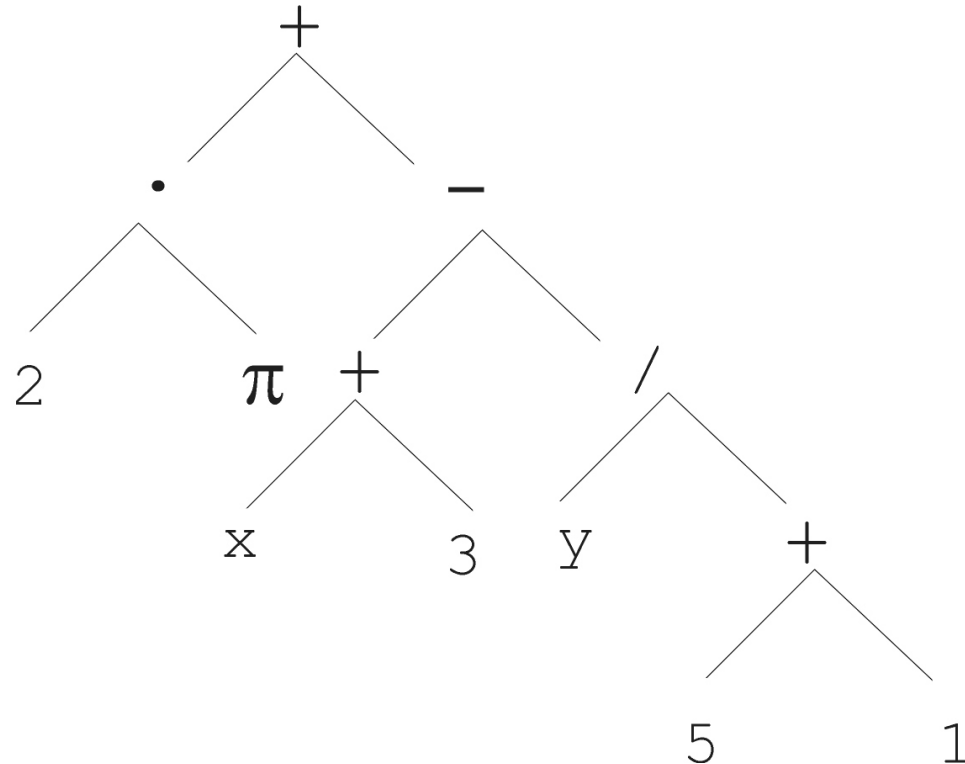
- Logical formula

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

- Program

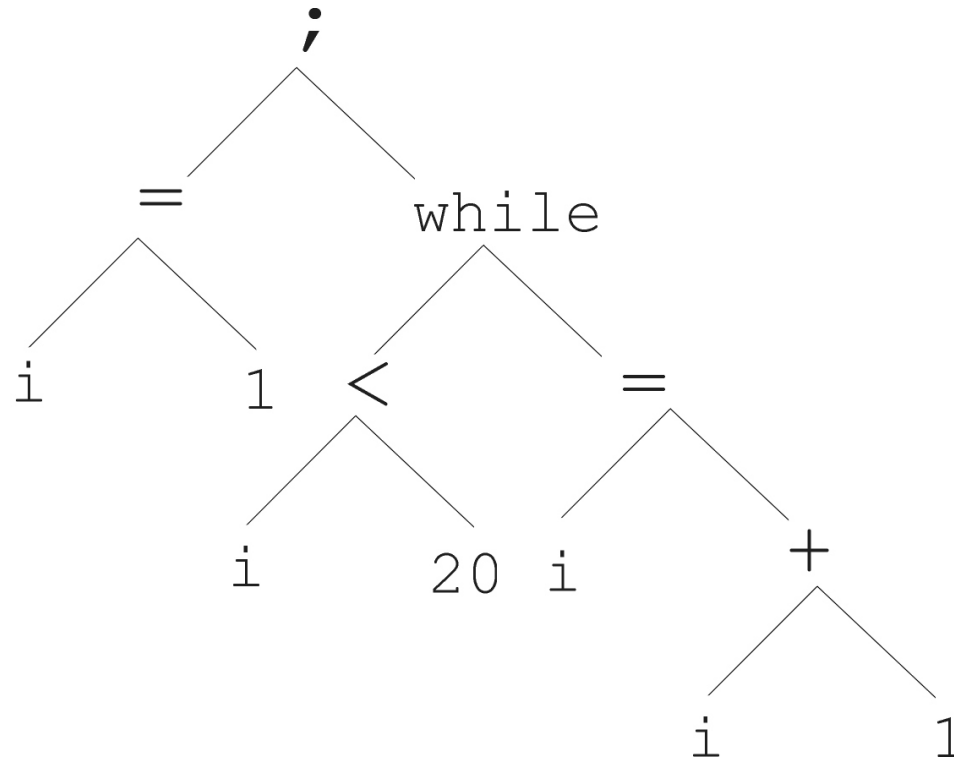
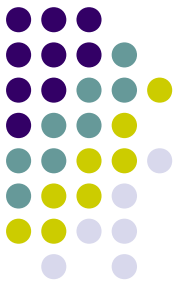
```
i = 1;
while (i < 20)
{
    i = i + 1
}
```

Tree based representation

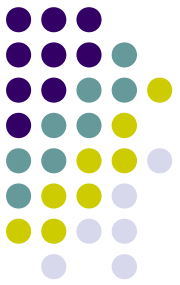


$$\pi + \left((x + 3) - \frac{y}{5 + 1} \right)$$

Tree based representation

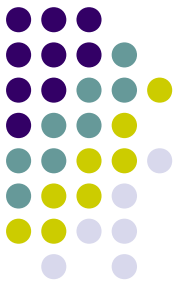


```
i = 1;
while (i < 20)
{
    i = i + 1
}
```

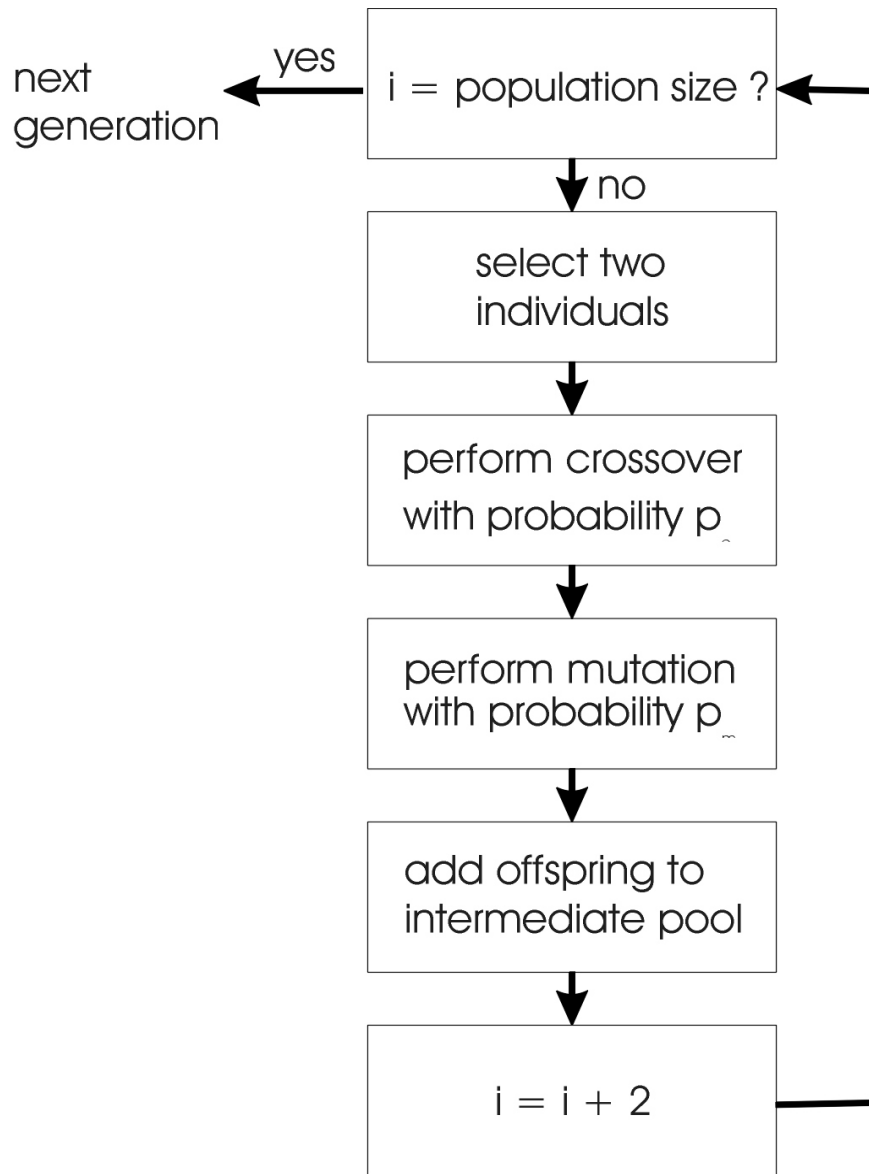
Tree based representation

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

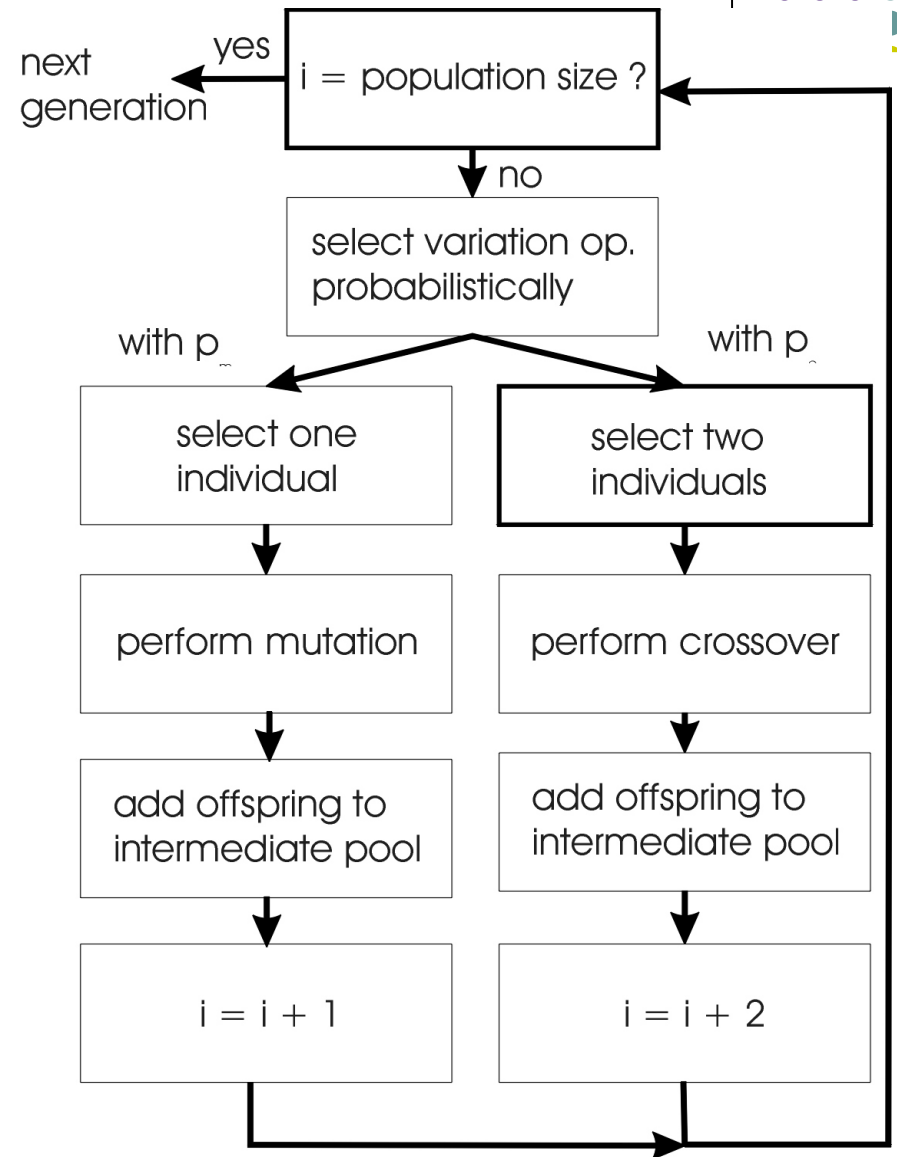


Tree based representation

- 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. $f(e_1, \dots, e_n)$ is a correct expression if $f \in F$, $\text{arity}(f)=n$ and e_1, \dots, e_n are correct expressions
 3. There are no other forms of correct expressions

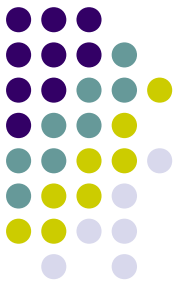


GA flowchart

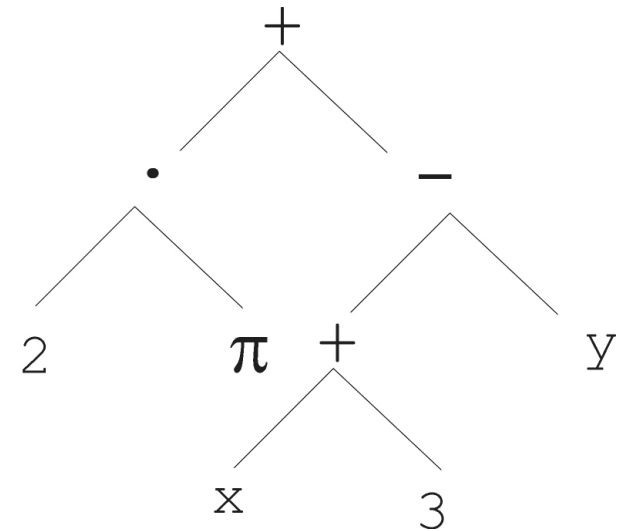
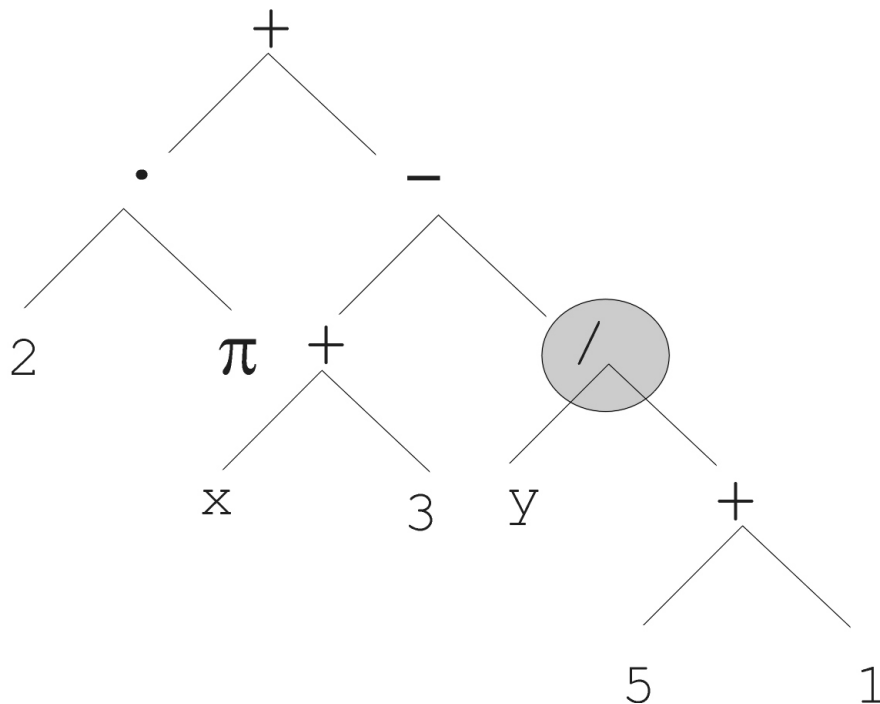


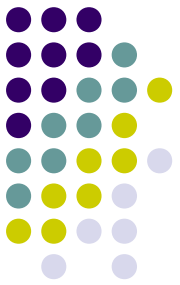
GP flowchart

Mutation



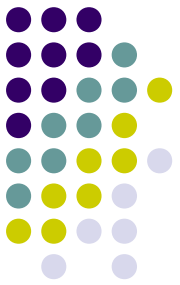
- Most common mutation: replace randomly chosen subtree by randomly generated tree





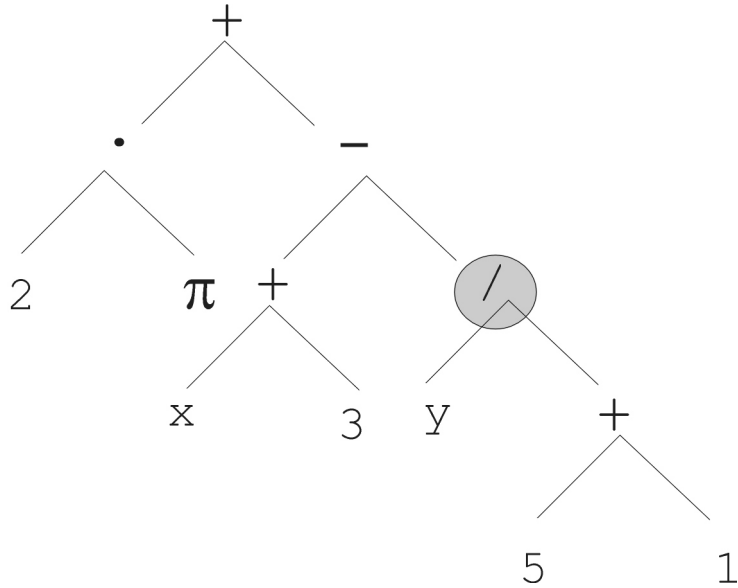
Mutation cont'd

- Mutation has two parameters:
 - Probability p_m to choose mutation vs. recombination
 - Probability to choose 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

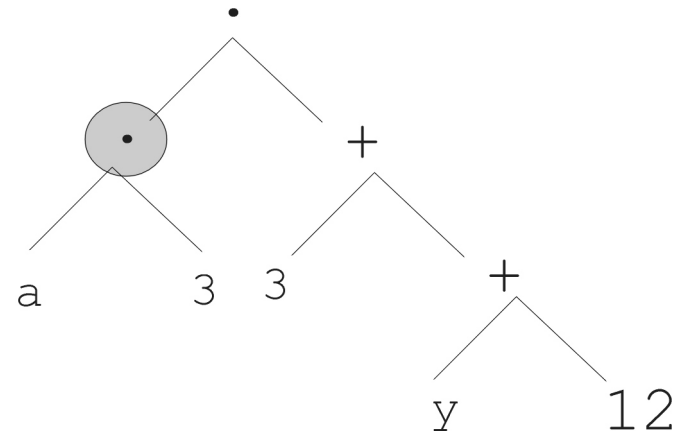


Recombination

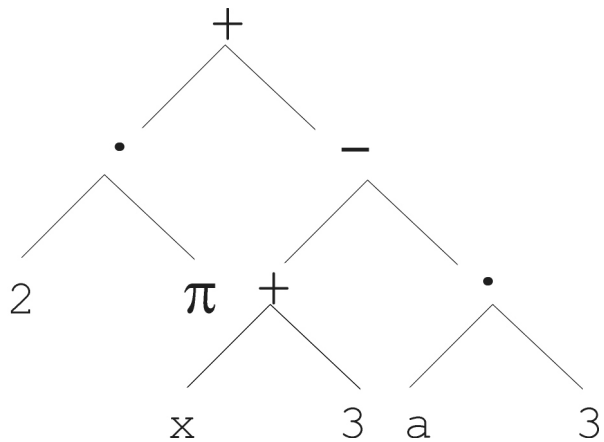
- Most common recombination: exchange two randomly chosen subtrees among the parents
- Recombination has two parameters:
 - Probability p_c to choose recombination vs. mutation
 - Probability to choose an internal point within each parent as crossover point
- The size of offspring can exceed that of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



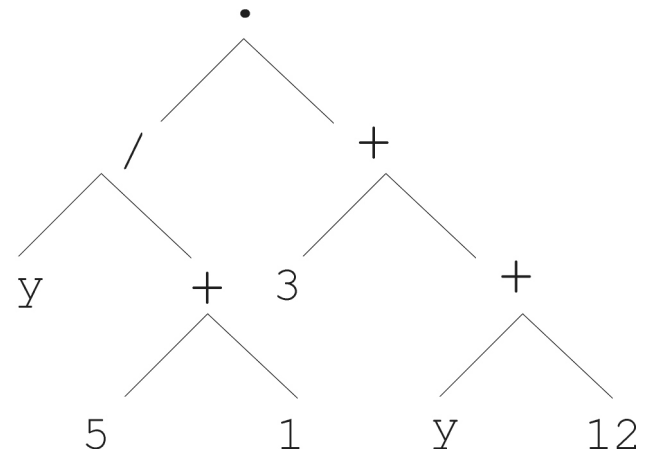
Parent 1



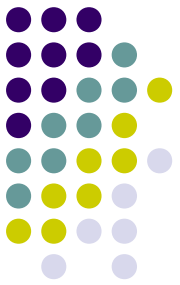
Parent 2



Child 1



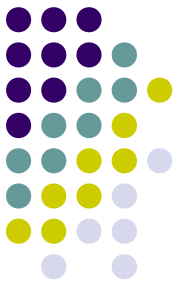
Child 2



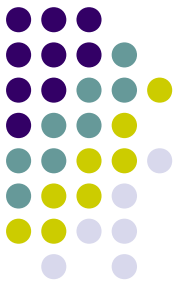
Selection

- Parent selection 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 (thus none)
 - Recently steady-state is becoming popular for its elitism

Initialisation



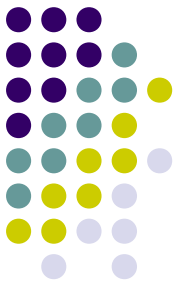
- Maximum initial depth of trees D_{\max} is set
- Full method (each branch has depth = D_{\max}):
 - nodes at depth $d < D_{\max}$ randomly chosen from function set F
 - nodes at depth $d = D_{\max}$ randomly chosen from terminal set T
- Grow method (each branch has depth $\leq D_{\max}$):
 - nodes 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

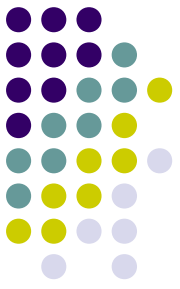
- Bloat = “survival of the fattest”, i.e., the tree sizes in the population are increasing over time
- Ongoing research and debate about the reasons
- Needs countermeasures, e.g.
 - Prohibiting variation operators that would deliver “too big” children
 - Parsimony pressure: penalty for being oversized

Problems involving “physical” environments

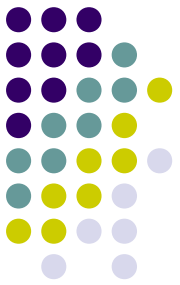


- Trees for data fitting vs. trees (programs) that are “really” executable
- Execution can change the environment → the calculation of fitness
- Example: robot controller
- Fitness calculations mostly by simulation, ranging from expensive to extremely expensive (in time)
- But evolved controllers are often to very good

Example application: symbolic regression



- Given some points in \mathbf{R}^2 , $(x_1, y_1), \dots, (x_n, y_n)$
- Find function $f(x)$ s.t. $\forall i = 1, \dots, n : f(x_i) = y_i$
- Possible GP solution:
 - Representation by $F = \{+, -, /, \sin, \cos\}$, $T = \mathbf{R} \cup \{x\}$
 - Fitness is the error
 - All operators standard $err(f) = \sum_{i=1}^n (f(x_i) - y_i)^2$
 - 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

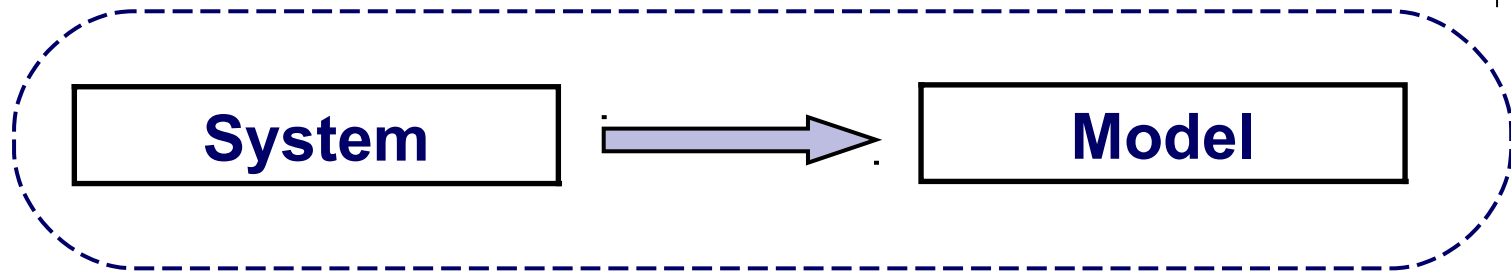
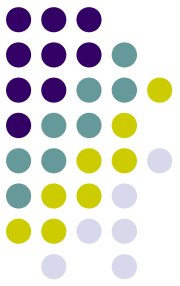
Is GP:

The art of evolving computer programs ?

Means to automated programming of
computers?

GA with another representation?

Introduction to GP based identification



Model structure

Black-box models

Model-selection

Black-box modeling

**A modellező választ
modell struktúrát**

Model parameters

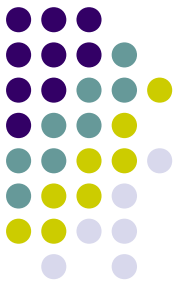
Known parameters

Based on prior knowledge

Identification

**From measured data
with some optimization**

Linear in parameters model



The original GP generates models that are nonlinear in their parameters

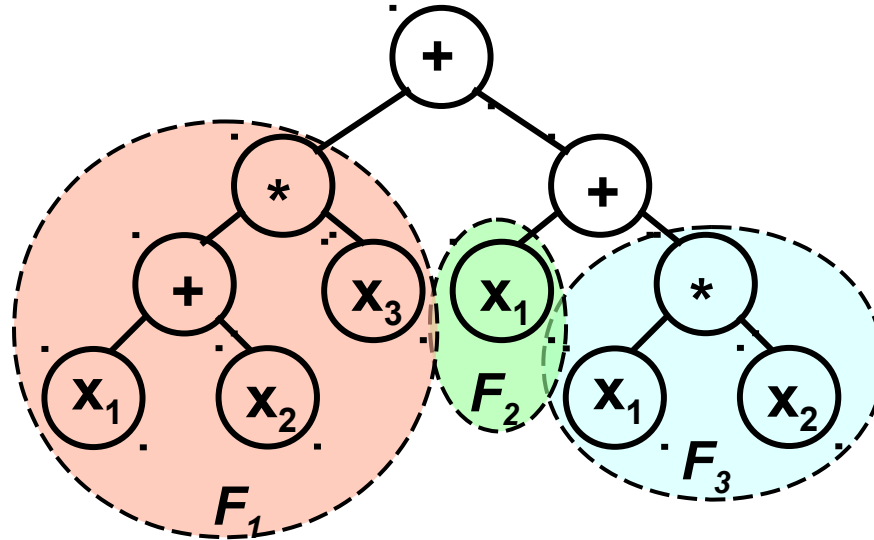
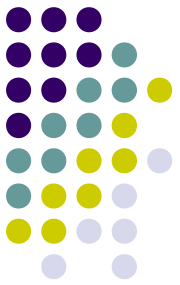


Linear in parameters model

$$\hat{y}(k) = a_0 + \sum_{i=1}^m a_i F_i(x_1(k), \dots, x_n(k))$$

The parameters can be estimated by LS

Model representation



$$\hat{y}(k) = a_1 F_1 + a_2 F_2 + a_3 F_3 + a_0$$

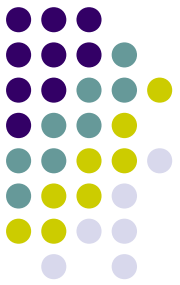
$$F_1 = (x_1(k) + x_2(k)) \cdot x_3(k)$$

$$F_2 = x_1(k)$$

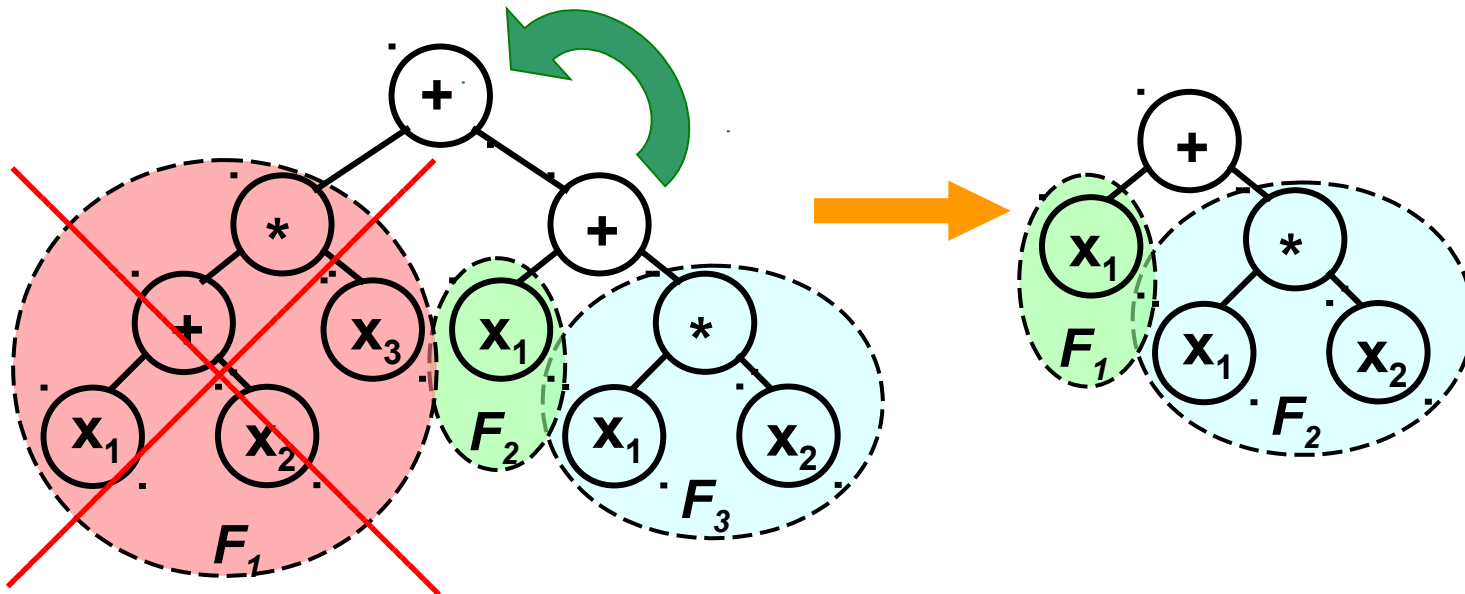
$$F_3 = x_1(k) x_2(k)$$

OLS

$$\hat{y}(k) = a_0 + \sum_{i=1}^m a_i F_i(x_1(k), \dots, x_n(k))$$



1. With orthogonal LS we can estimate the contribution
2. The useless branches will be deleted



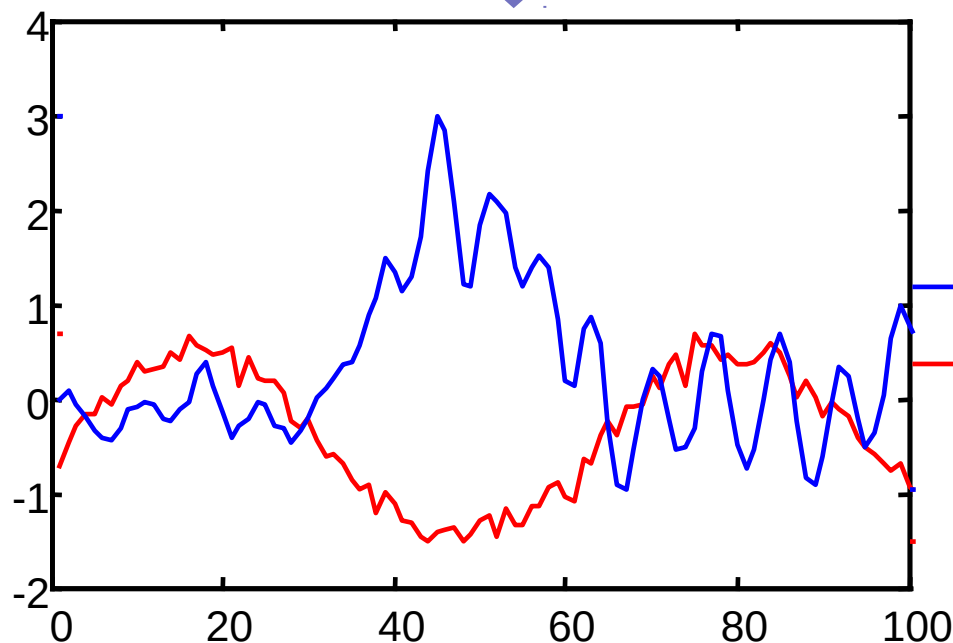
Example: system-identification



$$y(k) = 0.8u^2(k-1) + 1.2y(k-1) - 0.9y(k-2) - 0.2$$



simulation



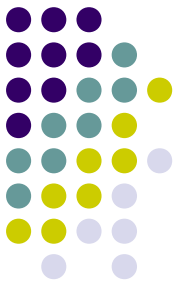
(y) + 4% noise
(u)



Result of GP

$$y(k) = 0.816u(k-1)u(k-1) + 1.175y(k-1) - 0.890y(k-2) - 0.205$$

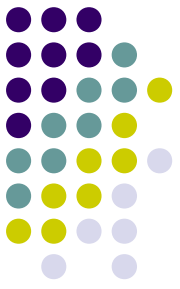
Improvement of the OLS



10-10 runs with max. 1000 evaluations

Method	1	2	3
Not exact solution from 10	10	4	3
exact solution from 10	0	6	7
Avarage runtime to find a solution	1000	880	565

1. : No penaly on the size of the tree
2. : Penaly on the size of the tree
3. : penaly on the size of the tree + OLS



Conclusions

- **Model structure identification is a not trivial task**
- **GP can be used for this purpose**
- **Linear in parameters models are ideal for GP based structure exploration**
- **It is useful to apply OLS to regularize the tree**