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!)
## GP technical summary tableau

<table>
<thead>
<tr>
<th>Representation</th>
<th>Tree structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recombination</td>
<td>Exchange of subtrees</td>
</tr>
<tr>
<td>Mutation</td>
<td>Random change in trees</td>
</tr>
<tr>
<td>Parent selection</td>
<td>Fitness proportional</td>
</tr>
<tr>
<td>Survivor selection</td>
<td>Generational replacement</td>
</tr>
</tbody>
</table>
Introductory example: credit scoring

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

<table>
<thead>
<tr>
<th>ID</th>
<th>No of children</th>
<th>Salary</th>
<th>Marital status</th>
<th>OK?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID-1</td>
<td>2</td>
<td>45000</td>
<td>Married</td>
<td>0</td>
</tr>
<tr>
<td>ID-2</td>
<td>0</td>
<td>30000</td>
<td>Single</td>
<td>1</td>
</tr>
<tr>
<td>ID-3</td>
<td>1</td>
<td>40000</td>
<td>Divorced</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Introductory example: credit scoring

• A possible model:
  \[
  \text{IF (NOC} = 2) \text{ AND (S} > 80000) \text{ THEN good ELSE bad}
  \]

• In general:
  \[
  \text{IF formula THEN good ELSE bad}
  \]

• 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

```
AND
   /
  =
 /   \
NOC   2
   \
   >
   /
   \
S   80000
```
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
  \[(x \land \text{true}) \rightarrow ((x \lor y) \lor (z \leftrightarrow (x \land y)))\]
- 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

\[(x \land \text{true}) \rightarrow ((x \lor y) \lor (z \leftrightarrow (x \land y)))\]
Tree based representation

```plaintext
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. \( f(e_1, \ldots, e_n) \) is a correct expression if \( f \in F \), \( \text{arity}(f)=n \) 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)
Offspring creation scheme

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

```
          +
         /|
        . /|
       + 3  |
      /  5  |
     2 π  1
    / 
   x  3
```

```
          +
         /|
        . /|
       +  y  |
      /  1  |
     2 π  3
    / 
   x  3
```
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
A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing
Genetic Programming

Parent 1

Child 1

Parent 2

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 $D_{\text{max}}$ is set
- Full method (each branch has depth $= D_{\text{max}}$):
  - nodes at depth $d < D_{\text{max}}$ randomly chosen from function set $F$
  - nodes at depth $d = D_{\text{max}}$ randomly chosen from terminal set $T$
- Grow method (each branch has depth $\leq D_{\text{max}}$):
  - nodes at depth $d < D_{\text{max}}$ randomly chosen from $F \cup T$
  - nodes at depth $d = D_{\text{max}}$ 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 $\mathbb{R}^2$, $(x_1, y_1), \ldots, (x_n, y_n)$
• Find function $f(x)$ s.t. $\forall i = 1, \ldots, n : f(x_i) = y_i$
• Possible GP solution:
  – Representation by $F = \{+, -, /, \sin, \cos\}$, $T = \mathbb{R} \cup \{x\}$
  – Fitness is the error
  – 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?