Artificial Intelligence

Evolutionary Algorithms

Lesson 6: Genetic Programming

Vincenzo Piuri

Università degli Studi di Milano

Contents

Motivation

Genetic Programming (GP) creates programs to solve a given application problem.

It is a universal way of learning computer programs

GP is based on:

- Describing a solution by connecting certain inputs with a certain output
- Searching for a matching computer program
- Representing programs by parse trees

Genetic Programming (1)

Representation of candidate solutions by function expressions or programs

- i.e. complex chromosomes of variable length

Genetic Programming (2)

Grammar for describing a language

- F: set of function symbols and operators
- T: set of terminal symbols (constants and variables)

The sets F and T are specific for each problem

Genetic Programming (3)

Examples for symbol sets

- Learning a boolean function
  - $\mathcal{F} = \{\text{and}, \text{or}, \text{not}, \text{if} \ldots \text{then} \ldots \text{else} \ldots, \ldots\}$
  - $\mathcal{T} = \{x_1, \ldots, x_m, 1, 0\}$ or $\mathcal{T} = \{x_1, \ldots, x_m, t, f\}$
- Symbolical regression
  - finding an approximating function for given data while minimizing the sum of squared errors
  - $\mathcal{F} = \{+, -, *, /, \sqrt{\ }, \sin, \cos, \log, \exp, \ldots\}$
  - $\mathcal{T} = \{x_1, \ldots, x_m\} \cup \mathbb{R}$

Genetic Programming (4)

Completeness

- Genetic Programming can only solve problems efficiently and effective, if function and terminal symbol sets are **sufficient/complete** to ensure an appropriate program can be found
- Finding the smallest sufficient set is (often) NP-hard

Symbolic Expressions (1)

- **Chromosomes** = expressions (composition of elements from $C = F \cup T$ and brackets)
  - Symbols for constants and variables are symbolic expressions
  - If $t_1, \ldots, t_n$ are symbolic expressions, and $f \in F$ is an (n-ary) operator symbol, then $f(t_1, \ldots, t_n)$ is a symbolic expression

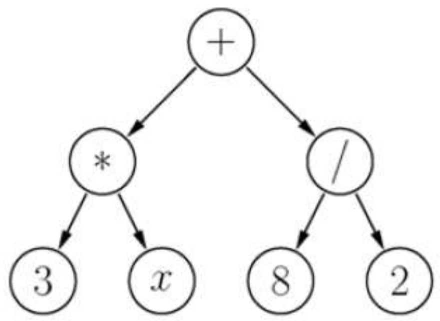
    - „$(+\ (*\ 3\ x)\ (/\ 8\ 2))$“ is a symbolic expression
      Lisp- or Scheme-like notation, means: $3 \cdot x + \frac{8}{2}$
    - „$2\ 7\ *\ (3\ /$“ is not a symbolic expression

Symbolic Expressions (2)

- Represent symbolic expressions by parse trees

symbolic expression:
$(+\ (*\ 3\ x)\ (/\ 8\ 2))$

parse tree:



Genetic Programming Algorithm

- Generate an **initialization population** of random expressions
- **Evaluate** these expressions by calculating their fitness
  - e.g., boolean functions: ratio of correct output for all inputs given to a sample
  - e.g., symbolic regression: sum of squared errors of the given measurement points
- **Selection** with one of the Evolutionary Algorithms strategies
- Application of **genetic operators**, usually only crossover

Initialization (1)

- Parameters for the creation process:
  - Maximal nesting (maximal tree height) d_{max}
  - Maximal number of tree nodes n_{max}

- Three different ways for initialization
  - Grow
  - Full
  - Ramped-half-and-half

Initialization (2)

- Grow

```
Input: node $n$, depth $d$, maximumDepth $d_{\max}$
if $d = 0$ {
    $n \leftarrow$ draw a node from $\mathcal{F}$ uniformly distributed
} else { if $d = d_{\max}$ {
    $n \leftarrow$ draw a node from $\mathcal{T}$ uniformly distributed
} else {
    $n \leftarrow$ draw a node from $\mathcal{F} \cup \mathcal{T}$ uniformly distributed
}
if $n \in \mathcal{F}$ {
    for each $c \in$ arguments of $n$ {
        initialize-grow($c, d + 1, d_{\max}$)
    }
} else {
    return
}
```

Initialization (3)

- Full

```
Input: nodes $n$, depth $d$, maximum depth $d_{\max}$
1: if $d \leq d_{\max}$ {
2:     $n \leftarrow$ draw nodes from $\mathcal{F}$ uniformly distributed
3:     for each $c \in$ arguments of $n$ {
4:         initialize-full($c, d+1, d_{\max}$)
5:     }
6: } else {
7:     $n \leftarrow$ draw nodes from $\mathcal{T}$ uniformly distributed
8: }
9: return
```

Initialization (4)

- Ramped-half-and-half
  - Combination of grow and full methods

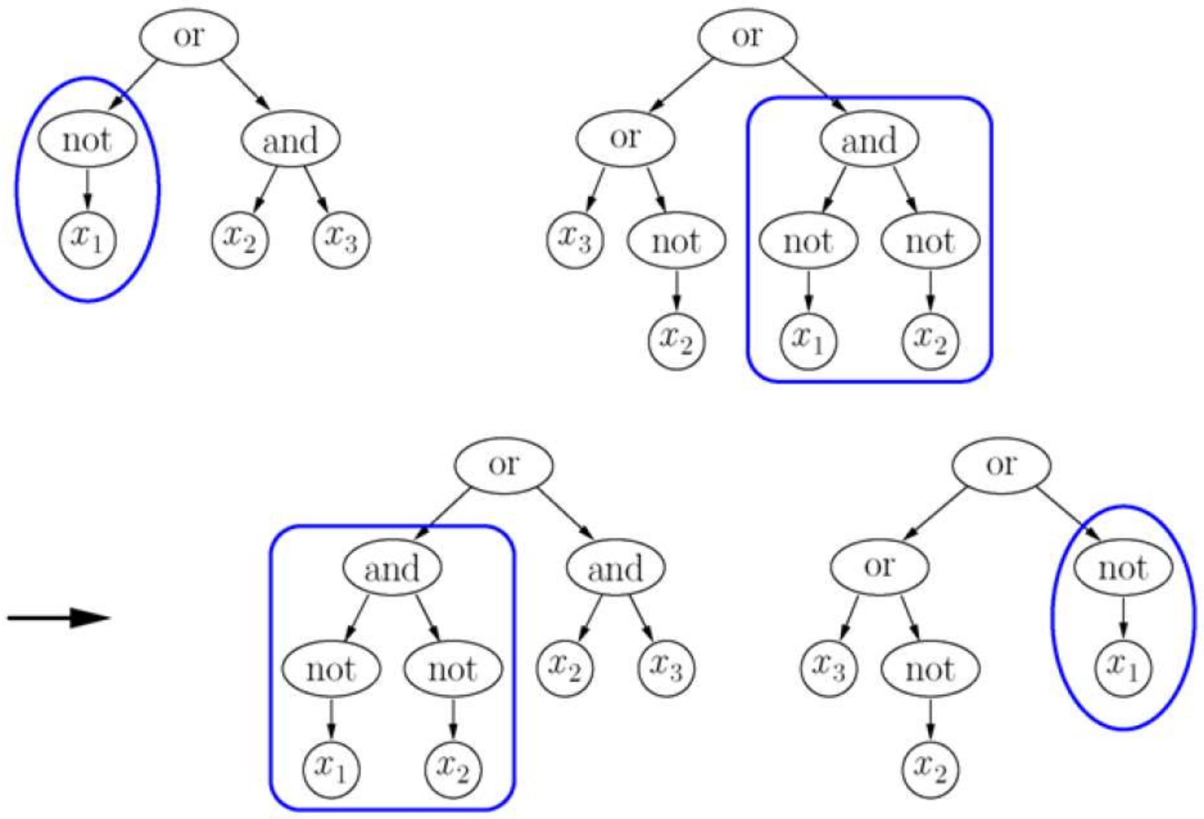
```
Input: maximum depth $d_{\max}$, population size $\mu$ (even multiple of $d_{\max}$)
1: $P \leftarrow \emptyset$
2: for $i \leftarrow 1 \ldots d_{\max}$ {
3:     for $j \leftarrow 1 \ldots \mu/(2 \cdot d_{\max})$ {
4:         $P \leftarrow P \cup$ initialize-full(root, 0, $i$)
5:         $P \leftarrow P \cup$ initialize-grow(root, 0, $i$)
6:     }
7: }
```

Genetic Operators (1)

- Usually initialized population has no good fitness
- The evolutionary progress changes population via genetic operators
- The 3 most important operators are:
  - Crossover
  - Mutation
  - Clonale reproduction (duplication of an individual)

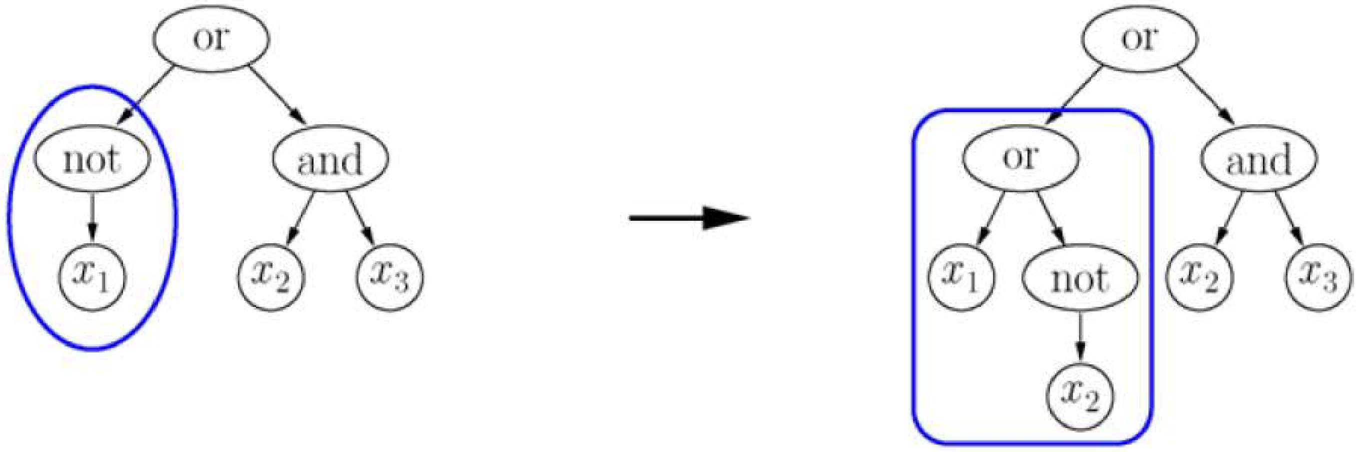
Genetic Operators (2)

- Crossover
  - Exchanging two subexpressions (subtrees)



Genetic Operators (2)

- Mutation
  - Exchanging a subexpression (subtree) by randomly generated one



Introns (1)

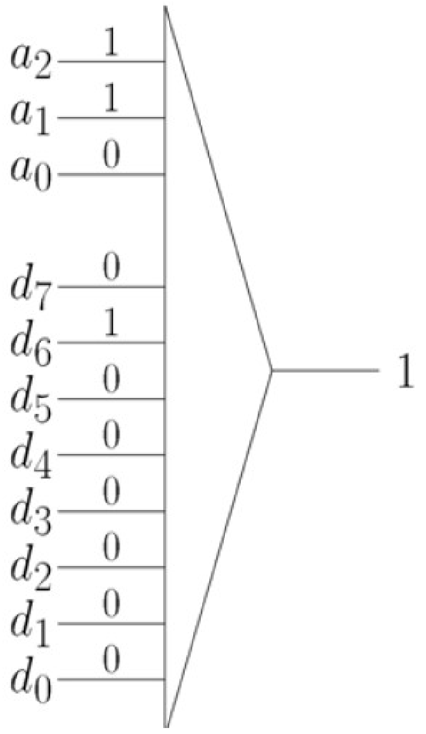
- Individuals are growing with progressing generation count
- Biology: **Introns** - some strips of DNA carry no information
  - Inactive (maybe outdated) strips within one gene that serves no function (*junk DNA*)
    - e.g. arithmetical expressions $a + (1 - 1)$ is easy to simplify

Introns (2)

- Preventing introns
  - **Breeding recombination** generates many children using different parameters, with only the best one going on into the next generation
  - **Intelligent recombination** chooses crossover points selectively
  - **Continuous slight changes** of the evaluation function can change constraints thus inactive subprograms (introns) might become active again

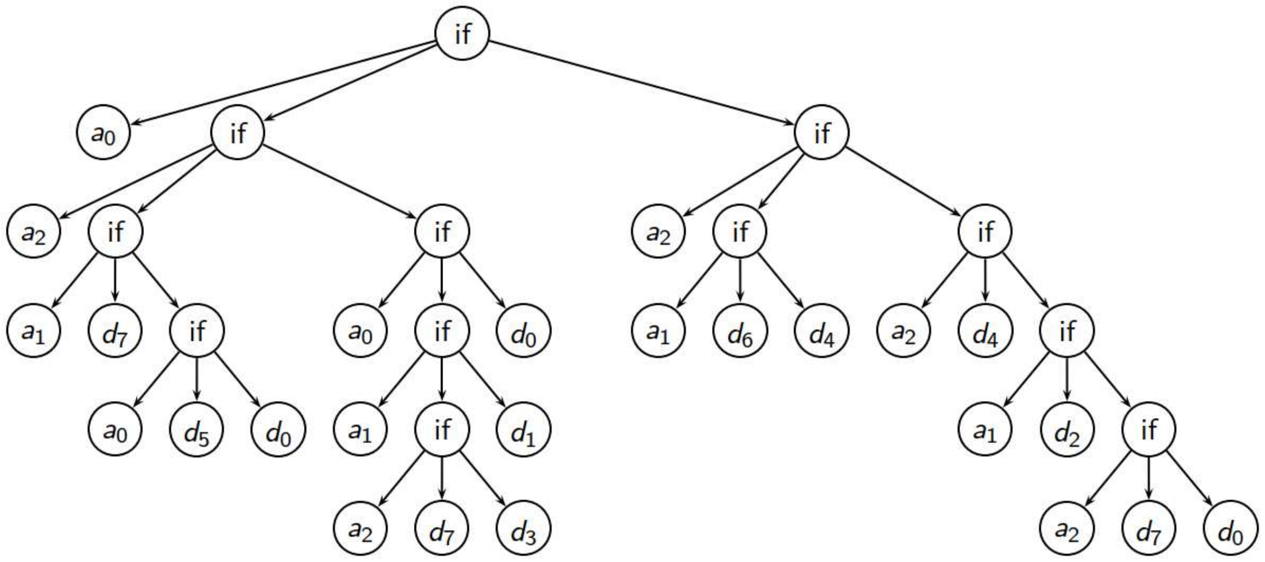
Examples (1)

- 11-Multiplexer
  - Multiplexer with 8 data and 3 address lines
  - $2^{11} = 2048$ possible inputs with 1 corresponding address each
  - Fitness function:
    $f(s) = 2048 - \sum_{i=1}^{2048} e_i$
    with e_i being the error for the i-th input

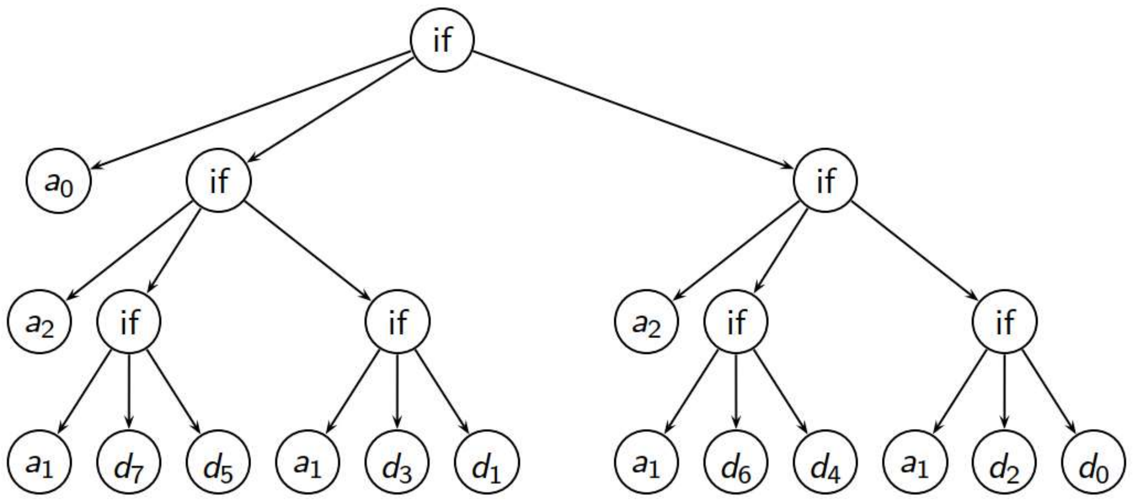


Examples (2)

- After 9 generations: solution with fitness of 2048
  - Complicated for humans to understand
  - Can be simplified by editing

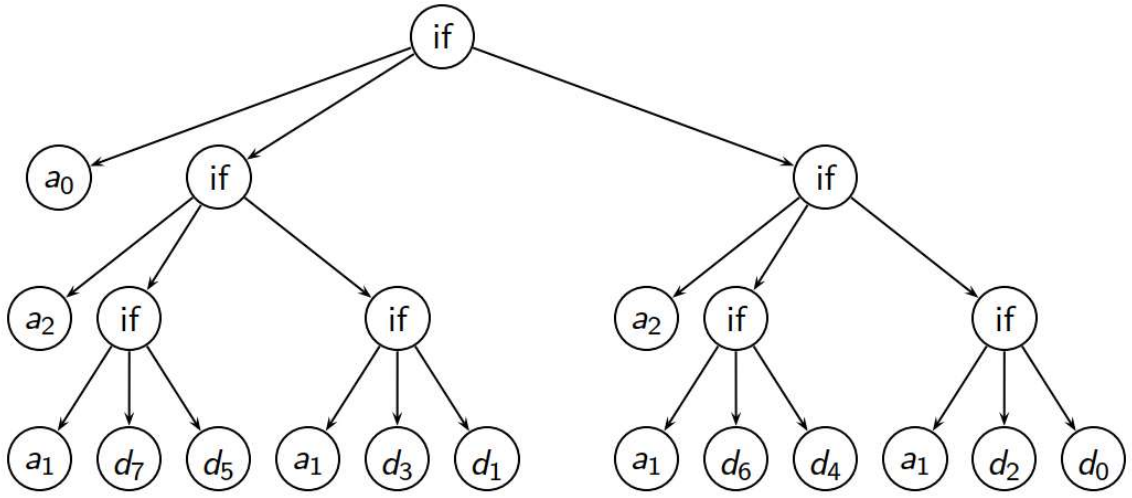


Examples (3)



- Editing
  - Asexual operations on one individual
  - Serves simplification through general and specific rules
  - If function without side effects on constant arguments occurs within the tree, then evaluate this function and replace the subtree with the result

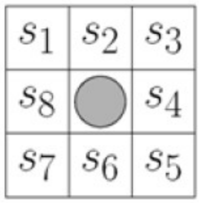
Examples (4)

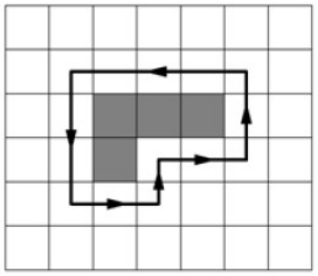


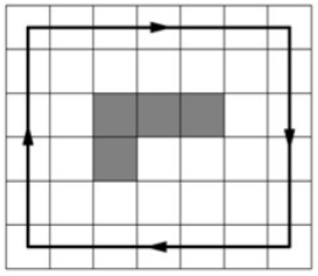
- Best solution truncated by editing

Examples (5)

- Learning a Robot Control System
  - 8 sensors $s_1, \dots, s_8$ yield state of the neighboring fields
  - 4 actions: go east, go north, go west, go south
  - direct deduction of the actions from $s_1, \dots, s_8$, no memory
  - task: circulate an object within a room, or follow the walls of the room

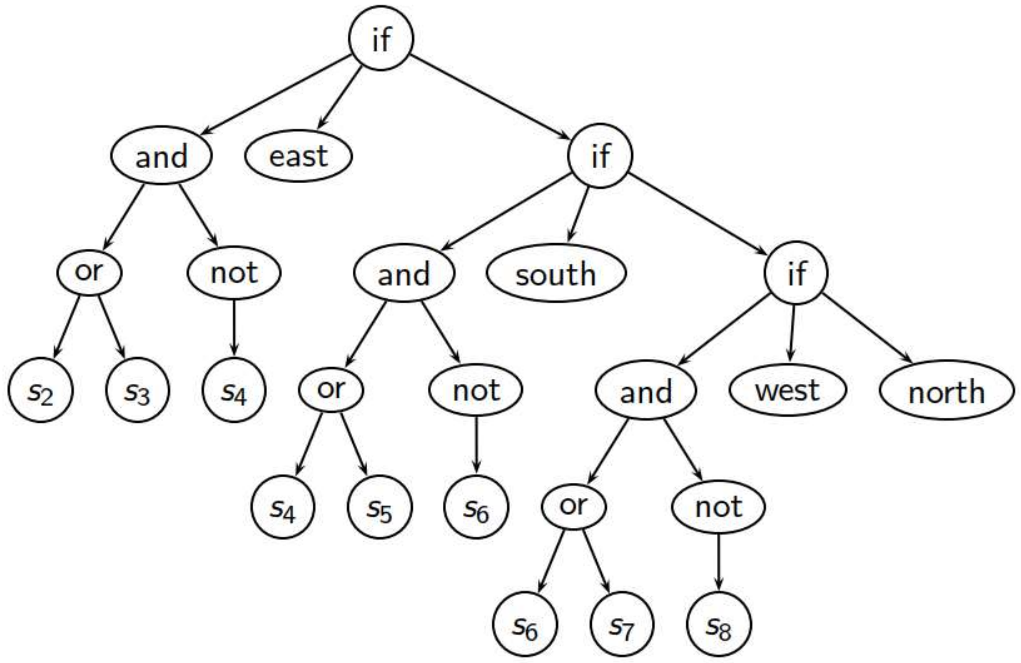






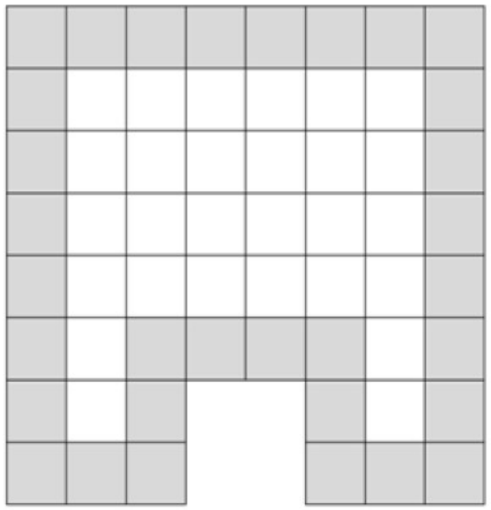
Examples (6)

- Optimal solution created by hand
  - Very unlikely to get exactly this solution



Examples (7)

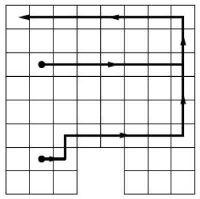
- Choose solution candidates by using a test space
  - Perfectly operating control unit would make the agent pass the grey labelled fields
  - Initial field is chosen randomly
  - If the action cannot be performed, the execution is quit
  - Total number of visited gray fields is fitness



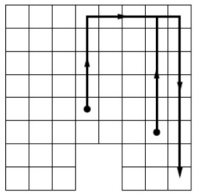
Examples (8)

- Generations

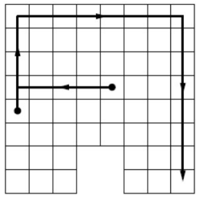
0



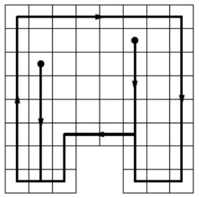
2



6



10



Generation 10

```
(if  (if  (if  s5 0 s3)
          (or  s5 east)
          (if  (or  (and s4 0)
                    s7)
               (or  s7 0)
               (and (not (not (and s6 s5)))
                    s5)))
     (if  s8
          (or north
              (not (not s6)))
          west)
     (not (not (not (and (if  (not south)
                              s5
                              s8)
                         (not s2))))))
```