Evolutionary Computing

Genetic Programming

Dr. Petr Musilek

Department of Electrical and Computer Engineering
University of Alberta

Fall 2019

Automatic Programming

- Automatic generation of computer programs
- Saying **WHAT** is wanted but not **HOW** to do it
- The programs can be of the most GENERAL form:
  - Subroutines (e.g. sorting, search)
  - Planning (e.g. sequence of moves of robotic arm)
  - Strategy in games (e.g. Pacman, Quake, Civilization ...)
  - Classification (e.g. character recognition)
  - Prediction (e.g. consumption of electricity)
  - Control (e.g. vehicle steering ...)
  - Design of electronic circuits (e.g. chip layout)

Genetic Programming (GP)

- GP is a branch of EC with a goal to generate a computer program (or just function/expression) that best fits user's requirements.
- The computer program acts as a candidate solution (like a binary string in GA).
- GP is computationally expensive and its application area was limited in 1990s.
- Recently, GP has started delivering human-competitive machine intelligence thanks to exponential growth in CPU power.

Representation of programs

- Programming languages: C, Java, Prolog, machine language, LISP
- Special languages: robots, Turing machines
- Typically, data and programs are treated the same way (lists or S-expressions):
  - `(dotimes i 3 (setq v (* i i)))`
  - `(3 4 5 (to b c))`
  - **Language = functions + terminals**

Example: Even parity

- Generate a computer program that takes 10 bits and returns whether the number of 1's is even:
  - `Even-Parity(1,0,0,0,1,0,0,1,1,0)` $\Rightarrow$ `TRUE`
- In terms of:
  - Functions: `AND, OR, NAND, NOR, NOT`
  - Terminals: `B0, B1, B2..., B9`
- Large number of input/output test cases ($2^{10} = 1024$ cases)

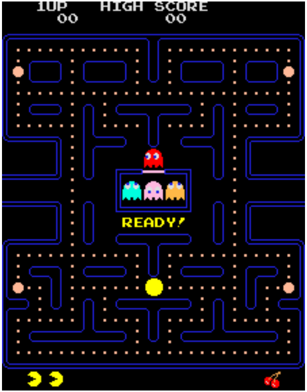
Test cases for 10-bit even parity (1024 cases)

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	OUT
0	0	0	0	0	0	0	0	0	0	TRUE
0	0	0	0	0	0	0	0	0	1	FALSE
0	0	0	0	0	0	0	0	1	1	TRUE
...										
1	1	1	1	1	1	1	1	1	1	TRUE

Example: Strategy for Pacman game

...

- Functions:
  - `if-obstacle`, `if-pill`, `if-power-pill`, `if-ghost` (they are of the form if-then-else)
  - `sequence2, 3, 4` ...
- Terminals: `advance, turn-left, -right`
  - `advance, turn-left, -right`
- **GOAL**: to eat all pills within a time limit



Example of program in Pacman game

```
(if-ghost
    (sequence3 (turn-left)
               (turn-left)
               (advance))
(if-power-pill
    (advance)
    (turn-right)))
```

GP Terminology

- Parse tree
  - A tree structure representing a program for GP
- Function set
  - Function set is the set of operators used for the program.
  - Functions take the internal points of the parse tree.
  - An example of function set is $\{+, -, *, \%, >, \mathit{IF}\}$.
- Terminal set
  - The set of terminal nodes in the parse tree.
  - A terminal might be a variable, a constant, or a function taking no argument.
  - An example of terminal set is X, Y, random-constants

Preparation Steps of GP

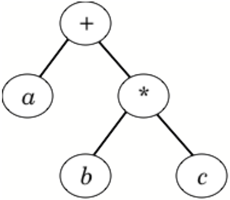
Determination of

- terminal set
- function set
- fitness measure
- parameters for GP run
- termination criterion

Program Representation

The program is generally represented by a parse tree of the function set and terminal set rather than lines of code.

E.g., a simple expression $a + b \cdot c$ would be represented as:

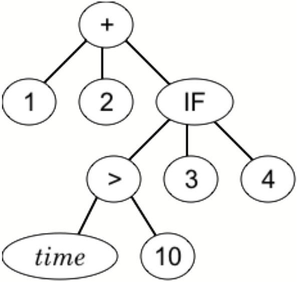


Program Representation

As another example, consider the following simple function in C language:

```
int foo (int time)
{
    int a, b;
    if(time > 10)
        a = 3;
    else
        a = 4;
    b = a + 1 + 2;
    return b;
}
```

The function can be represented by the following tree:



GP Algorithm

1. Formation of an initial random population computer of programs, using functions and terminals
2. Execution (!) of all programs and their evaluation (fitness function)
3. Selection of well-performing programs
4. Creation of a new population by applying genetic operators to selected programs
5. Return to 2 until finding a “good” program

GP as search

- Genetic Programming performs a heuristic search in the space of computer programs (a type of “best-first” search with an opened list of limited size – the population)
- The heuristic function is the *fitness function*
- The search operators are the *genetic operators* (mutation and crossover)

Generation of an initial population

- Choose a function for the root of the tree
- Check the arity of the function
- For each argument of the function, generate:
  - a terminal; or
  - a subtree
- Practically, trees deeper that certain constant should not be generated

Methods to generate individuals

- *Full:* same depth for all branches of a tree
- *Grow:* variable depth (up to a set limit)
- *Ramped half and half:*
  - Mixture of full and grow
  - 50% will be full and 50% grow
- **Objective:** to maximize diversity

Functions and Terminals

- Functions: functions or macros that take arguments
  - e.g. `(+34)`
- Terminals:
  - Functions that do not have arguments; e.g. (advance)
  - Constant: $3, a, \ldots$
  - (Ephemeral) random constant $\mathcal{R}$: for numerical problems and symbolic regression
  - Input variables: `D0,D1`, ...

Functions

- Funcions/terminals sufficient to express the solution (e.g. for boolean functions, `AND`, `OR` and `NOT` are sufficient)
- It may be useful to include powerful functions (so that the system does not have to rediscover them); e.g. `sin(x)`
- Differentiate between functions (evaluate arguments; e.g. `sum`) and macros (do not evaluate: `if-then-else`);
- Functions have to execute with any argument and without producing errors (closure); e.g. protection against division by zero
- In standard GP there are no data types. All functions must be prepared to receive any type and value of argument

Evaluation of programs (fitness)

- *Raw*:
  - e.g. number of correctly guessed cases or hits (even-parity)
  - e.g. function, such as [$0.7 \times$ points $+ \, 0.3 \times$ time] (Pacman)
- *Standard*: (to be minimized)
  - Standard Fitness = Maximum - raw
- *Adjusted*: $1/(1 +$ standard); it exaggerates fitness near 0
- *Normalized* (or relative):
  - ajusted / (sum of adjusted fitnesses in population)

GP Operations

- Reproduction (selection)
- Mutation
- Crossover
- Architecture-altering operations

Selection

Selection of capable (well-performing) individuals

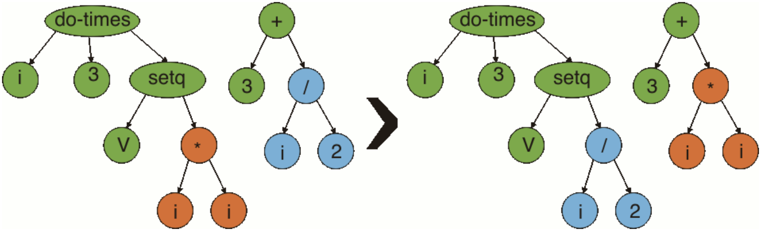
- Fitness proportional (probabilistic): uses normalized fitness
  - Problem: super-individuals
- Ranked
- Tournament
  - Match several individuals to compete among themselves: the best one reproduces
- Greedy Overselection (of the fitter individuals): create a high fitness group, H, and a low fitness group, L
  - 80% of the time select the parent from group H
  - 20% of the time select the parent from group L

Mutation Operation

- Given an individual, randomly pick a (internal or terminal) point in the parse tree.
- Delete subtree at the picked point.
- Grow new subtree at the picked point in the same way as used to generate initial individuals.
- Repair the individual if the associated program is invalid.

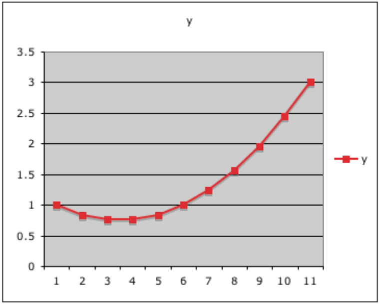
Crossover

- Given two individuals, pick a point in each parse tree independently.
- Swap the subtrees at the picked points.
- Repair the parse tree if the associated program is invalid.



Example I: Nonlinear Function Identification

Find a nonlinear function that satisfies the following relationship:



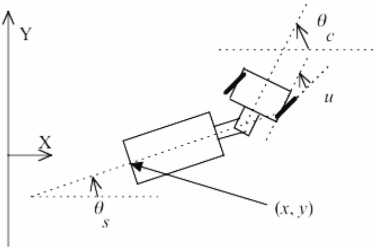
x	y
-1.00	1.00
-0.80	0.84
-0.60	0.76
-0.40	0.76
-0.20	0.84
0.00	1.00
0.20	1.24
0.40	1.56
0.60	1.96
0.80	2.44
1.00	3.00

Example I: Nonlinear Function Identification

Terminal set	$T = \{x, \text{Random-constants}\}$
Function set	$F = \{+, -, *, \%\}$
Fitness	Sum of errors between the candidate program's outputs and y's for all values of x.
Parameter	Population size: 4
Termination	When the sum of errors is less than, e.g. 0.1

Example II: Truck Backer Upper Control Law Design

Objective: Develop a control law $u = (x, y, \theta_S, \theta_C)$ that leads the trailer tail to $(x, y, \theta_S) = (0, 0, 0)$ when the backward linear velocity of the wheels is fixed.



Example II: Truck Backer Upper Control Law Design

There are four input variables of the controller:

- Horizontal position, x
- Vertical position, y
- Trailer angle, θ_S
- Cap angle, θ_C

Thus, *terminals* $T = \{x, y, \theta_S, \theta_C, \text{Random-constants}\}$

Function set: arithmetic and trigonometric functions

To *evaluate* a solution candidate during evolution process, we need to conduct simulation runs over many initial conditions of the trailer for hundreds of time steps.