What is Genetic Programming?

One of the central challenges of computer science is to get a computer to do what needs to be done, without telling it how to do it. Genetic programming addresses this challenge by providing a method for automatically creating a working computer program from a high-level problem statement of the problem. Genetic programming achieves this goal of *automatic programming* (also sometimes called *program synthesis* or *program induction*) by genetically breeding a population of computer programs using the principles of Darwinian natural selection and biologically inspired operations. The operations include reproduction, crossover (sexual recombination), mutation, and architecture-altering operations patterned after gene duplication and gene deletion in nature.

Genetic programming is a domain-independent method that genetically breeds a population of computer programs to solve a problem. Specifically, genetic programming iteratively transforms a population of computer programs into a new generation of programs by applying analogs of naturally occurring genetic operations. The genetic operations include crossover (sexual recombination), mutation, reproduction, gene duplication, and gene deletion.

Preparatory Steps of Genetic Programming

The human user communicates the high-level statement of the problem to the genetic programming system by performing certain well-defined preparatory steps.

The five major **preparatory steps** for the basic version of genetic programming require the human user to specify

(1) the set of terminals (e.g., the independent variables of the problem, zero-argument functions, and random constants) for each branch of the to-be-evolved program,
(2) the set of primitive functions for each branch of the to-be-evolved program,
(3) the fitness measure (for explicitly or implicitly measuring the fitness of individuals in the population),
(4) certain parameters for controlling the run, and
(5) the termination criterion and method for designating the result of the run.

Executional Steps of Genetic Programming

Genetic programming typically starts with a population of randomly generated computer programs composed of the available programmatic ingredients. Genetic programming iteratively transforms a population of computer programs into a new generation of the population by applying analogs of naturally occurring genetic operations. These operations are applied to individual(s) selected from the population. The individuals are probabilistically selected to participate in the genetic operations based on their fitness (as measured by the fitness measure provided by the human user

in the third preparatory step). The iterative transformation of the population is executed inside the main generational loop of the run of genetic programming.

The executional steps of genetic programming (that is, the **flowchart of genetic programming**) are as follows:

(1) Randomly create an initial population (generation 0) of individual computer programs composed of the available functions and terminals.

(2) Iteratively perform the following sub-steps (called a *generation*) on the population until the termination criterion is satisfied:

  (a) Execute each program in the population and ascertain its fitness (explicitly or implicitly) using the problem's fitness measure.

  (b) Select one or two individual program(s) from the population with a probability based on fitness (with reselection allowed) to participate in the genetic operations in (c).

  (c) Create new individual program(s) for the population by applying the following genetic operations with specified probabilities:

    (i) *Reproduction:* Copy the selected individual program to the new population.

    (ii) *Crossover:* Create new offspring program(s) for the new population by recombining randomly chosen parts from two selected programs.

    (iii) *Mutation:* Create one new offspring program for the new population by randomly mutating a randomly chosen part of one selected program.

    (iv) *Architecture-altering operations:* Choose an architecture-altering operation from the available repertoire of such operations and create one new offspring program for the new population by applying the chosen architecture-altering operation to one selected program.

(3) After the termination criterion is satisfied, the single best program in the population produced during the run (the best-so-far individual) is harvested and designated as the result of the run. If the run is successful, the result may be a solution (or approximate solution) to the problem.