VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Engineering and Computer Science

COMP 307 — Lecture 11

Evolutionary Computing 2 (ML 8)

Genetic Algorithms to Genetic Programming

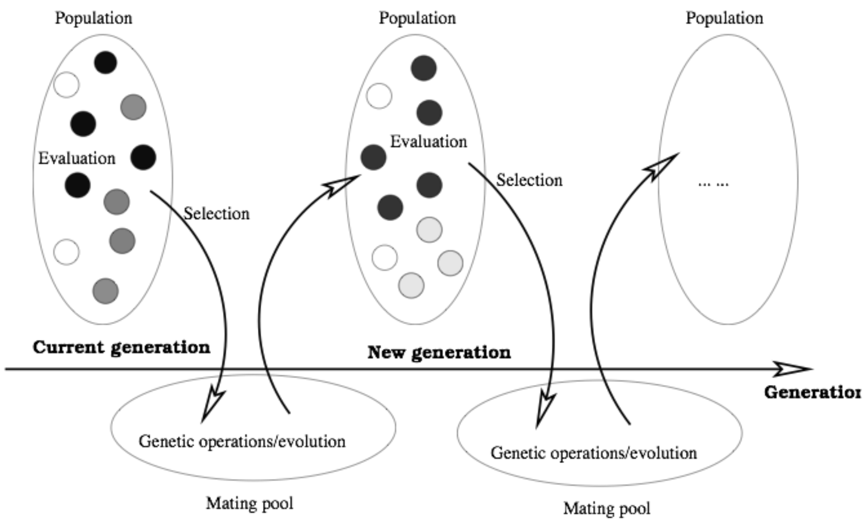
Dr Bing Xue (Prof. Mengjie Zhang)

bing.xue@ecs.vuw.ac.nz

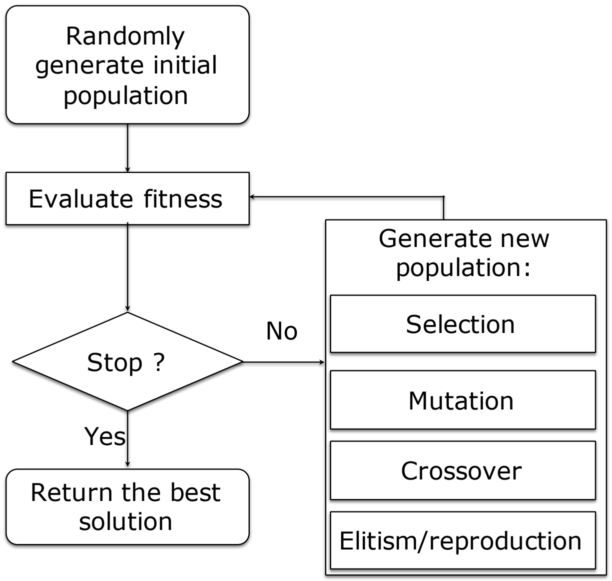
Outline

- Overview of an EC (GA) process
- A typical representation
- Terminals and functions
- Program generation
- Genetic operators
- Fitness functions
- A basic GP algorithm
- Tackling a problem with GP

Evolutionary Search

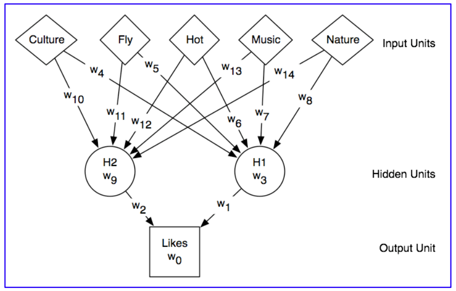


Flowchart of a Simple GA



Example: Training A Neural Network

- Use a GA to adjust the weights of the neural network
- Representation — bit strings
  - Each individual/chromosome represents one neural network
  - Each bit/dimension represents one weight or bias:
  - *14 weights, 14 bits, dimensions*
    - *(0.2, 0.3, 0.4, 0.61, …. , 0.23, 0.71)*
    - (W_0, W_1, W_2, W_3, …. , W_{13}, W_{14})
- Fitness function:
  - Classification error rate



From EC to GP

- Genetic programming (**GP**) inherits properties from EC techniques (e.g. GAs) and automatic programming
- GP uses a similar evolutionary process to the general evolutionary algorithms (e.g. GAs)
  - GA uses bit strings to represent solutions, GP uses tree-like structures that can represent computer programs such LISP programs
  - GA bit strings use a fixed length representation, GP trees can vary in length
  - The term comes from the notion that computer programs can be represented by a tree-structured genome.
- Automatically learning a set of computer programs for a particular task is a dream of computer scientists
- GP is such a technique that can help us achieve this goal

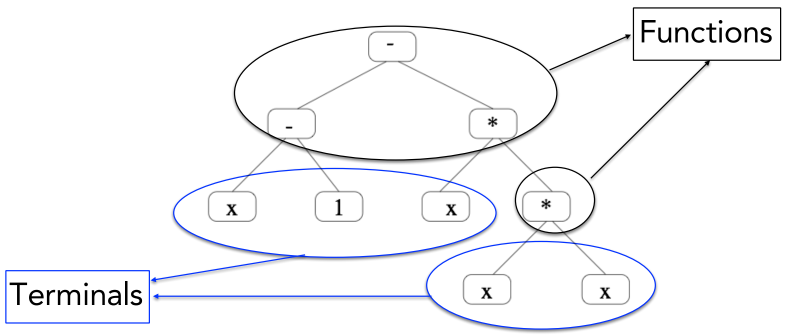
LISP S-Expressions

- Form of a LISP function (FUNCTION-NAME ARG1 ARG2 ARG3) The arguments are evaluated, the function is applied to the arguments and the value returned.
- (+ 1 2 3) evaluates to 6
- (+ (- 3 2) (* 2 4)) evaluates to (+ 1 8) which is 9
- (IF (> TIME 10) 3 4) evaluates to 3 if TIME is 11 or more and to 4 if time is 10 or less
- If TIME is 20, what is the value of (+ 1 2 (IF (> TIME 10) 3 4))
- Programs in GP have not yet extended to the kinds of programs we are accustomed to writing
- Most work is done with S expressions

Programs as Tree Structures

- Programs are constructed from a *terminal* set and a *function* set.
- Terminals and functions are also called primitives.

$$(x-1)-x^3$$



Terminal Set

- A terminal set consists of a set of terminals
  - attributes/features
  - constants
- Terminals have no arguments and form the leaves of the tree.
- Terminals represent the *inputs* of a GP program, form input from the environment (a specific task)

- Attributes or features of a problem domain are usually used as terminals.
- Random numbers are also usually used as terminals.

Function Set

- A function set consists of a set of functions or operators
- Functions form the root and the internal nodes of the tree representation of a program.
- Two kinds of functions: general functions and domain specific functions.

- General functions:
  - Arithmetic functions: +, -, *, %.
  - Protected division (%): returns 0 if denominator is 0
  - Other standard functions: sin,cos,exp,rlog,abs, ...

- Domain Specific functions: e.g. image processing operators

Sufficiency and Closure

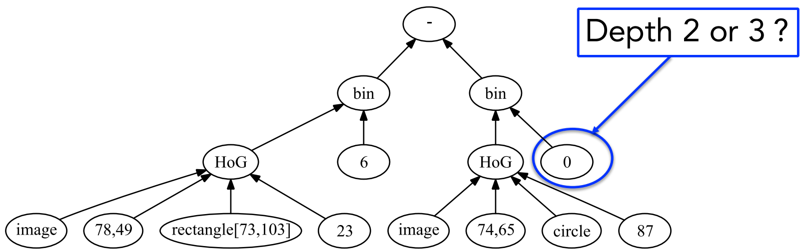
- Selection of the functions and terminals is critical to success.
- The terminal set and the function set should be selected so as to satisfy the requirements of closure and sufficiency.

- Sufficiency: There must be some combination of terminals and function symbols that can solve the problem
- Closure: Any function can accept any input value returned by any function (and any terminal).

- A bad selection could result in very slow convergence or even not being able to find a solution at all.

Program Generation

- For initialising a population or mutation.
- Maximum program size: the maximum size permitted for a program, which is the maximum depth of a tree.
- Depth: The depth of a node is the minimal number of nodes that must be traversed to get from the root node of the tree to the selected node.



Program Generation

- There are several ways of generating programs: full, grow and ramped half-and-half
- Full method:
  - Functions are selected as the nodes of the program tree until a given depth is reached.
  - Then terminals are selected to form the leaf nodes.
  - This ensures that full, entirely balanced trees are constructed.

Program Generation

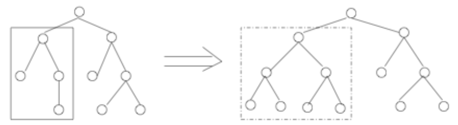
- Grow method:
  - Nodes are selected from either functions or terminals.
  - If a terminal is selected, the branch with this terminal is terminated and the generation process moves on to the next non-terminal branch in the tree.
- Ramped half-and-half method:
  - Both the full and grow methods are combined.
  - Half of the population generated for each depth value are created by using the grow method and the other half using the full method.
- Ramped half-and-half has been widely used in many GP systems

Genetic Operators in GP

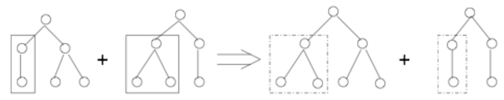
- Evolution proceeds by updating the initial population by the use of genetic operators.
  - An initial population usually has very bad fitness.
  - Three fundamental genetic operators in GP: reproduction, crossover and mutation.
- Reproduction:
  - Simply copy a selected program from the current generation to the new generation.
  - Allow good programs to survive.
  - *Elitism*
- Mutation:
  - Operate on a single selected program.
  - Remove a random subtree of the program,
  - Put a new subtree in the same place.
    - Use a program generation method to generate the new subtree

Genetic Operators in GP

- Crossover:
  - Goal: attempt to take advantage of different selected programs within a population, and integrate the useful information from them.
  - Combine the genetic material of the two selected parent programs.
  - Swap a subtree of one parent with a subtree of the other
  - Put the two newly formed programs into next generation.



(Mutation)



(Crossover)

Selection

- Selection determines which evolved program will be used for the genetic operators
- The proportional selection — (roulette wheel selection in GAs):
  - Specifies probabilities for individuals to be given a chance to pass offspring into the next generation.
  - Program with a better fitness will get more chance.
- The *tournament selection*
  - Based on competition within only a subset of the population against each other, rather than the whole population.
  - A number of programs are selected randomly according to the tournament size.
  - The genetic operators are applied to the winner(s)
  - In the smallest possible tournament, two individuals can compete.

Fitness Cases and Fitness Function

- Fitness Cases: patterns or examples in other learning paradigms
- Two different sets of fitness cases: training cases for learning and test cases for performance evaluation.
- The fitness of a program generated by the evolutionary process is evaluated according to the fitness function.
- The fitness function should be designed to give graded and continuous feedback about how well a program performs on the training set.
- The fitness function plays a very important role in the evolutionary process and varies with the problem domains.

Fitness Function Examples

- Image matching: the number of matched pixels
- Robot learning obstacle avoidance: the number of wall hits for a robot
- Classification task: the number of correctly classified examples, error rate, or classification accuracy
- Prediction application: the deviation between prediction and reality
- GP-controlled agent in a betting game: the amount of money won
- Artificial life application: the amount of food found and eaten.

Basic GP Algorithm

This GP algorithm is based on the proportional selection model — (check Slide 4)

1. Initialise the population
2. Evaluate the fitness of each individual program in the current population.
3. Until the new population is fully created, repeat the following:
   - Select programs in the current generation.
   - Perform genetic operators on the selected programs.
   - Insert the result of the genetic operations into the new generation.
4. If the termination criterion is not fulfilled, repeat steps 2-4 with the new generation.
5. Present the best individual in the population as the output.

Tackling a Problem with GP

- What is the set of terminals used in the program trees?
- What kind of functions can be used to form the function set to represent the program tree?
- What is the fitness measure?
- What values can be given for the parameters and variables for controlling the evolutionary process, for example, population size and number of generations?
- When to terminate a run?
- How do we know the result is good enough?
- What genetic operators, at what frequencies, are going to be applied?

Summary

- Overview of EC (GA) process
- GP basics: representation, genetic/evolved programs, primitives, terminals, functions, fitness, genetic operators, selection
- GAs vs GP
- Basic GP algorithm
- Suggested reading:
  - http://www.genetic-programming.com/
  - www.cs.bham.ac.uk/˜wbl/biblio/
  - www.cs.bham.ac.uk/˜wbl/biblio/gp-html/index.htm
  - http://www.cs.ucl.ac.uk/research/genprog/gp2faq/gp2faq.html
- Next lecture: GP examples, for regression and classification