Evolutionary Algorithms

1. Introduction

Evolutionary algorithms are optimization techniques based on Darwin's theory of evolution. Information representing possible solutions to a problem are encoded in chromosomes. Through the process of natural selection and application of crossover and mutation operators, a randomly created initial population is iteratively refined. Crossover involves exchanging material between two chromosomes. Evolutionary algorithms are search processes involving the following tasks:

- Encoding potential solutions as chromosomes - Chromosomes are composed of genes representing the characteristics of the individual. The genotype specifies the genetic composition of an individual and the phenotype the behaviour of the individual.
- A fitness function to evaluate the **fitness** of each chromosome - The fitness function calculates a measure of how close an individual is to a solution. The fitness measure is then used by selection methods to choose the parents of the next generation.
- Initialization of the initial population - Before applying selection and reproduction operators an initial population is created. Each chromosome is formed by randomly choosing genes.
- Selection operators - Selection operators mimic natural selection and are used to choose parents which the reproduction operators are applied to. These methods choose the fitter individuals of the population. The two most popular selection methods are fitness proportionate selection and tournament selection:

  Fitness proportionate selection - Involves calculating a probability that each individual will be copied into the next generation. Based on this probability a mating pool is created. Each parent is randomly selected from the mating pool.

  Tournament selection - In tournament selection a number of elements of the population are randomly selected. The number of elements is called the tournament. The element in the tournament with the best fitness is returned as the winner.

  In some cases elitism is used, i.e. only the fittest individuals are selected as parents.

- Genetic operators - Genetic operators are used to create the offspring of each generation. The simplest operator is reproduction which makes a copy of the parent. The crossover operator swaps components in copies of two parents to create two offspring. The mutation operator creates an offspring by making random changes to a copy of the chosen parent.

A typical evolutionary algorithm is depicted in Figure 1:

An evolutionary algorithm usually finds a solution by converging to an area of the search space that contains a solution. In some cases the algorithm may converge quickly, but to an area of the search space that does not contain a solution. In this case we say the algorithm has converged prematurely to a local optimum (as opposed to a global optimum). A selection method that is too elitist can often lead to premature convergence. Evolutionary algorithms include the following categories:

- Genetic algorithms
- Genetic programming
- Evolutionary programming

- Coevolution

```
Create an initial population
Repeat
 Evaluate the population
 Select parents for the next generation
 Apply genetic operators to chosen parents
Until termination criteria are met
```

Figure 1: An example of an evolutionary algorithm

2. Genetic Algorithms (GAs)

Genetic algorithms were introduced by John Holland of Michigan University. Initially, each element of the population, i.e. each chromosome was represented as a fixed length binary string, e.g. 1001100. This often required preprocessing and postprocessing of individuals. Initially, fitness proportionate selection was used for choosing parents and only the crossover operator was applied to parents to produce offspring. However, since its inception there other representations, selection methods and genetic operators such as mutation have been used. Three types of crossover typically used in genetic algorithms are uniform crossover, one-point crossover and two-point crossover.

3. Genetic Programming (GP)

The concept of genetic programming was coined by John Koza, once a doctoral student of John Holland. While genetic algorithms search a space of potential solutions to a problem, genetic programming searches a space of algorithms or programs which when executed will provide a solution. Thus, the representation commonly used in genetic programming is a parse tree which represents the corresponding program. An example of a parse tree is illustrated in Figure 2.

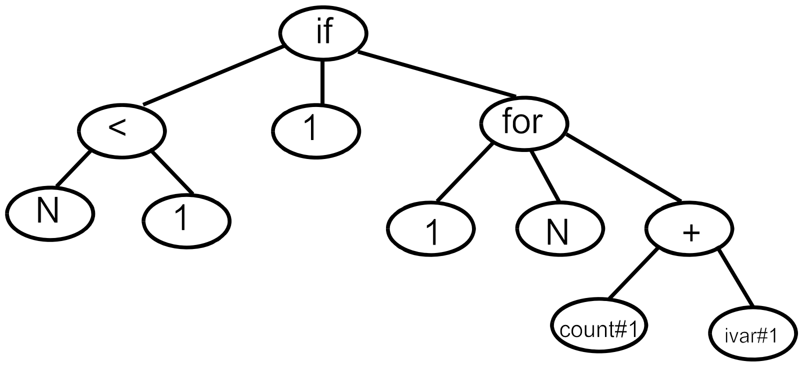

Figure 2: Parse tree representing a program

The trees do not have a fixed size and are generated up until a maximum depth. The most effective initial population has proven to be one containing a mixture of fixed sized and variable sized trees. Mutation generally involves replacing a subtree in a parse tree with a newly created subtree. Crossover involves swapping two subtrees randomly chosen in each of the parents.

Tournament selection is found to be most effective in genetic programming systems. John Koza has also attempted to implement the concept of modularization in programming by using automatically defined functions (ADFs). ADFs represent subprograms and evolve together with a main program.

4. Evolutionary Programming

In evolutionary programming emphasis is on the behaviour, i.e. the phenotype, of the individuals rather than the structure, i.e. genotype. A set of optimal behaviours is obtained from a space of observable behaviours. A fitness function is used to assess the optimality of each behaviour, i.e. the behaviour error. Mutation is the only genetic operator used. Two applications that EP has been applied to is the evolution of finite automata and function optimization.

5. Coevolution

A GA or a GP that employs coevolution maintains more than one independently-acting populations. Evolution is not population-specific and the evolution of any one population is also dependant on changes in the other populations. One of the main applications of coevolution is two-agent games requiring the evolution of a game strategy to beat an opponent. One of three types of fitness is used in coevolution, namely, relative fitness, fitness sampling and hall of fame. A relative fitness function expresses the performance of individuals in one population in comparison with individuals in another population. Types of fitness sharing include simple fitness, fitness sharing and competitive fitness sharing. Fitness sampling evaluates the performance of individuals against a sample of individuals from a competing population. Fitness sampling methods include all versus all sampling, random sampling, tournament sampling, all versus best sampling and shared sampling. Hall of fame is an elitist approach which ensures that the best individual of each generation is carried through to the next generation.

Exercises

1. Implement a genetic algorithm to generate a given word of the English language. The program must take the word as input. It must generate an initial population of words which are randomly created. This initial population must be evolved by means of
evaluation, tournament selection, crossover and mutation until the given word is generated.

1.1 Define the terminal set.
1.2 Define the function set.
1.3 What would a chromosome look like?
1.4 Define the fitness function.
1.5 Define mutation and crossover operators?

2. Suppose that you are required to implement a GP system to evolve an algorithm for the factorial of a positive number n.

2.1 Define the terminal set.
2.2 Define the function set.
2.3 Define the fitness function.
2.4 What test cases would you use?
2.5 What would a solution tree look like?
2.6 Define mutation and crossover operators?