Artificial Intelligence
Chapter 4.
Machine Evolution

Overview

- Introduction to Evolutionary Computation
  - Biological Background
  - Evolutionary Computation
- Genetic Algorithm
- Genetic Programming
- Summary
  - Applications of EC
  - Advantage & disadvantage of EC
- Further Information

Biological Basis

- Biological systems adapt themselves to a new environment by evolution.
  - Generations of descendants are produced that perform better than do their ancestors.
- Biological evolution
  - Production of descendants changed from their parents
  - Selective survival of some of these descendants to produce more descendants

Evolutionary Computation

- What is the Evolutionary Computation?
  - Stochastic search (or problem solving) techniques that mimic the metaphor of natural biological evolution.
- Metaphor

<table>
<thead>
<tr>
<th>EVOLUTION</th>
<th>PROBLEM SOLVING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual Fitness</td>
<td>Candidate Solution</td>
</tr>
<tr>
<td>Environment</td>
<td>Quality</td>
</tr>
<tr>
<td></td>
<td>Problem</td>
</tr>
</tbody>
</table>
Paradigms in EC

- Evolutionary Programming (EP)
  - [L. Fogel et al., 1966]
  - FSMs, mutation only, tournament selection
- Evolution Strategy (ES)
  - [I. Rechenberg, 1973]
  - Real values, mainly mutation, ranking selection
- Genetic Algorithm (GA)
  - [J. Holland, 1975]
  - Bitstrings, mainly crossover, proportionate selection
- Genetic Programming (GP)
  - [J. Koza, 1992]
  - Trees, mainly crossover, proportionate selection

(Simple) Genetic Algorithm (1)

- Genetic Representation
  - **Chromosome**
    - A solution of the problem to be solved is normally represented as a chromosome which is also called an individual.
    - This is represented as a bit string:
      - ![Example Chromosome](image)
      - This string may encode integers, real numbers, sets, or whatever.
  - **Population**
    - GA uses a number of chromosomes at a time called a population.
    - The population evolves over a number of generations towards a better solution.
Genetic Algorithm (2)

- **Fitness Function**
  - The GA search is guided by a *fitness function* which returns a single numeric value indicating the *fitness* of a chromosome.
  - The fitness is maximized or minimized depending on the problems.
  - Example: The number of 1’s in the chromosome
  - Numerical functions

Genetic Algorithm (3)

- **Selection**
  - Selecting individuals to be parents
  - Chromosomes with a higher fitness value will have a higher probability of contributing one or more offspring in the next generation
  - Variation of Selection
    - Proportional (Roulette wheel) selection
    - Tournament selection
    - Ranking-based selection

Genetic Algorithm (4)

- **Genetic Operators**
  - **Crossover (1-point)**
    - A crossover point is selected at random and parts of the two parent chromosomes are swapped to create two offspring with a probability which is called crossover rate.
    - Crossover Point
      - Parents
      - Offspring
    - This mixing of genetic material provides a very efficient and robust search method.
    - Several different forms of crossover such as k-points, uniform

Genetic Algorithm (5)

- **Mutation**
  - Mutation changes a bit from 0 to 1 or 1 to 0 with a probability which is called mutation rate.
  - The mutation rate is usually very small (e.g., 0.001).
  - It may result in a random search, rather than the guided search produced by crossover.

- **Reproduction**
  - Parent(s) is (are) copied into next generation without crossover and mutation.
Example of Genetic Algorithm

The Problem is to Maximize $f(x) = x^2$

<table>
<thead>
<tr>
<th>Number</th>
<th>String</th>
<th>Fitness</th>
<th>% of the Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01101</td>
<td>169</td>
<td>18.4</td>
</tr>
<tr>
<td>2</td>
<td>11000</td>
<td>576</td>
<td>49.2</td>
</tr>
<tr>
<td>3</td>
<td>01000</td>
<td>64</td>
<td>5.5</td>
</tr>
<tr>
<td>4</td>
<td>10011</td>
<td>361</td>
<td>30.9</td>
</tr>
<tr>
<td>(cse)</td>
<td></td>
<td>1170</td>
<td>100.0</td>
</tr>
</tbody>
</table>

1. **Roulette Wheel Selection**

   $P_k = \frac{f(x_k)}{\sum_{i=1}^{n} f(x_i)}$

2. **One-Point Crossover**

   $P_m \in [0.6 \ldots 1.0]$

3. **Mutation**

   $P_m \in [0.001 \ldots 0.1]$

   Mutate the first node

   **Fitness**

Genetic Programming

- Genetic programming uses variable-size tree-representations rather than fixed-length strings of binary values.
- Program tree
  - S-expression
  - LISP parse tree
- Tree = Functions (Nonterminals) + Terminals

GP Tree: An Example

- Function set: internal nodes
  - Functions, predicates, or actions which take one or more arguments
- Terminal set: leaf nodes
  - Program constants, actions, or functions which take no arguments

  S-expression: $(+ 3 (/ (\times 5 4) 7))$
  Terminals = $\{3, 4, 7\}$
  Functions = $\{+, \times, /\}$

Setting Up for a GP Run

- The set of terminals
- The set of functions
- The fitness measure
- The algorithm parameters
  - population size, maximum number of generations
  - crossover rate and mutation rate
  - maximum depth of GP trees etc.
- The method for designating a result and the criterion for terminating a run.
Example: Wall-Following Robot

- Program Representation in GP
  - Functions
    - \( \text{AND} \ (x, y) = 0 \) if \( x = 0 \); else \( y \)
    - \( \text{OR} \ (x, y) = 1 \) if \( x = 1 \); else \( y \)
    - \( \text{NOT} \ (x) = 0 \) if \( x = 1 \); else \( 1 \)
    - \( \text{IF} \ (x, y, z) = y \) if \( x = 1 \); else \( z \)
  - Terminals
    - Actions: move the robot one cell to each direction
      \{north, east, south, west\}
    - Sensory input: its value is \( 0 \) whenever the corresponding cell is free for the robot to occupy; otherwise, \( 1 \).
      \{n, ne, e, se, s, sw, w, nw\}
Evolving a Wall-Following Robot

- Experimental Setup
  - Population size: 5,000
  - Fitness measure: the number of cells next to the wall that are visited during 60 steps
    - Perfect score (320)
      - One Run (32) x 10 randomly chosen starting points
  - Termination condition: found perfect solution
  - Selection: tournament selection

- Creating Next Generation
  - 500 programs (10%) are copied directly into next generation.
    - Tournament selection
      - 7 programs are randomly selected from the population, 5,000.
      - The most fit of these 7 programs is chosen.
  - 4,500 programs (90%) are generated by crossover.
    - A mother and a father are each chosen by tournament selection.
    - A randomly chosen subtree from the father replaces a randomly selected subtree from the mother.
  - In this example, mutation was not used.

Two Parents Programs and Their Child

- Generation 0
  - The most fit program (fitness = 92)
    - Starting in any cell, this program moves east until it reaches a cell next to the wall; then it moves north until it can move east again or it moves west and gets trapped in the upper-left cell.
**Result (2)**

- Generation 2
  - The most fit program (fitness = 117)
    - Smaller than the best one of generation 0, but it does get stuck in the lower-right corner.

![Diagram](image1)

(C) SNU CSE Artificial Intelligence Lab (SCAI) 25

**Result (3)**

- Generation 6
  - The most fit program (fitness = 163)
    - Following the wall perfectly but still gets stuck in the bottom-right corner.

```
(IF (AND (NOT (a)))
  (IF (o) (o) (o))
  (OR (IF (w) (o) (o))
    (IF (north) (east) (west))
    (IF (north) (west) (north))
    (AND (a) (IF (w))
      (IF (o) (south) (east))
      (north))
    (AND (w) (NOT (IF (w) (o) (o))))
    (OR (OR (AND (o) (o))
      (west))
      (south))
    (sw)))
```

**Result (4)**

- Generation 10
  - The most fit program (fitness = 320)
    - Following the wall around clockwise and moves south to the wall if it doesn’t start next to it.

```
(IF (IF (IF (o) (o) (o))
       (GR (o) (east))
       (IF (GR AND (o) (o))
         (north))
       (GR (north))
       (AND (NOT (NOT (AND (a) (o)))))
       (south))
    (IF (w)
       (GR (north)
         (NOT (NOT (a))))
       (west))
    (NOT CHOT CHOT (AND (IF (NOT (south))
                         (o))
                   (w))
    (NOT (a)))))))))))
```

**Result (5)**

- Fitness Curve
  - Fitness as a function of generation number
    - The progressive (but often small) improvement from generation to generation

![Fitness Curve](image2)
Applications of EC

- Numerical, Combinatorial Optimization
- System Modeling and Identification
- Planning and Control
- Engineering Design
- Data Mining
- Machine Learning
- Artificial Life

Advantages of EC

- No presumptions w.r.t. problem space
- Widely applicable
- Low development & application costs
- Easy to incorporate other methods
- Solutions are interpretable (unlike NN)
- Can be run interactively, accommodate user proposed solutions
- Provide many alternative solutions

Disadvantages of EC

- No guarantee for optimal solution within finite time
- Weak theoretical basis
- May need parameter tuning
- Often computationally expensive, i.e. slow

Further Information on EC

- Conferences
  - IEEE Congress on Evolutionary Computation (CEC)
  - Genetic and Evolutionary Computation Conference (GECCO)
  - Parallel Problem Solving from Nature (PPSN)
  - Int Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA)
  - Int Conf. on Simulated Evolution and Learning (SEAL)
- Journals
  - IEEE Transactions on Evolutionary Computation
  - Evolutionary Computation
  - Genetic Programming and Evolvable Machines
  - Evolutionary Optimization