Evolutionary Computation

- Dean F. Hougen
- w/ contributions from
  - Pedro Diaz-Gomez & Brent Eskridge
  - Robotics, Evolution, Adaptation, and Learning Laboratory
    - (REAL Lab)
  - School of Computer Science
  - University of Oklahoma

Outline

- Introduction to Evolutionary Computation
- Genetic Algorithms
- Genetic Programming
- “Real World” Example

Motivation for Evolutionary Computation (EC)

- Science
  - Understand evolutionary mechanisms
    - mutation
    - crossover
    - co-evolution
    - etc.
  - Understand evolved characteristics
    - behavior
    - learning
    - etc.
- Engineering
  - Create better designs

Implementation of Evolutionary Computation (EC)

- Inspired by biological evolution
- Required components:
  - Replicators (genes)
  - Replication (copying)
  - Selection mechanism (survival)
- Requirement:
  - Replication must be high fidelity
- Result:
  - Differential reproduction of replicators

Details of Biological Evolution

- Additional terminology
  - Chromosome: A collection of genes
  - Locus: A location within a chromosome
  - Allele: A possible gene at a given locus
  - Genotype: All the genes of an individual
  - Phenotype: The expression of an

Details of Biological Evolution

- At what level does selection take place?
  - Gene
  - Chromosome
  - Individual
  - Species
  - Genus
**EC Types**

- Genetic Algorithms
- Genetic Programming
- Evolution Strategies
- Evolutionary Programming
- Grammatical Evolution
- Learning Classifier Systems
- Estimation of Distribution Systems
- Etc.

**Genetic Algorithms (GAs)**

- Genes – bits, integers, floats, etc.
- Chromosome – array of genes, e.g.: 001000010111011101110010111101
  - Also called genotype or individual
  - Note lack of distinction between:
    - chromosome and genotype
    - genotype and phenotype
- Locus – position in array
- Population – collection of individuals
- Generation – population at a given time

**GA Operators 1**

- Crossover
  - Example two point, two offspring
    - Parents:
      - 00001|011011101110|0010
      - 11111|111111111111|1111
    - Offspring:
      - 11111|011011101110|1111
      - 00001|111111111111|0010

**GA Operators 2**

- Mutation
  - Example single point mutation
    - Original: 11111011011111011111
      - Mutated: 111010110111101111

**GA Procedure – Steady State**

Randomly initialize population

Repeat
  - Selection
  - Reproduction – Crossover
  - Mutation

Until solution found or resources exhausted

**GA Procedure – Generational**

Randomly initialize population

Repeat
  - Selection
  - Reproduction – Crossover
  - Mutation

Until new generation created

Until solution found or resources exhausted
GA Selection

Randomly initialize population
Repeat
  Repeat
    Selection – using fitness function
    Reproduction – Crossover
    Mutation
  Until new generation created
Until solution found or resources exhausted

GA Fitness Function

- Example: Onemax
- Chromosomes:

<table>
<thead>
<tr>
<th>label</th>
<th>string</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>00000110</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>11101110</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>00100000</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>00110100</td>
<td>3</td>
</tr>
</tbody>
</table>

GA Selection

<table>
<thead>
<tr>
<th>label</th>
<th>string</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>000001102</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>111011106</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>001000001</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>001101003</td>
<td></td>
</tr>
</tbody>
</table>

- Can do selection proportional to fitness: AARRBBBCDDDD
- Generate numbers from 1 to 12
- Select corresponding parents

GA Procedure – Generational

Randomly initialize population
Repeat
  Repeat
    Selection
    Reproduction – Crossover
      – e.g., Probability 60%
    Mutation
  Until new generation created
Until solution found or resources exhausted

GA Crossover

- Suppose one crossover
- Use selected chromosomes:
  B  11101110
  D  00110100
- Generate numbers from 1 to chromosome length (here 8), say 1 and 5, and generate offspring:
  B’ 1[110]110
  D’ 0[1101100]
GA Procedure – Generational

Randomly initialize population
Repeat
  Repeat
    Selection
    Reproduction – Crossover
    Mutation
    – e.g., Probability 0.1% per gene
  Until new generation created
Until solution found or resources exhausted

GA Mutation & Results

• Suppose no mutation, then population of next generation is:

<table>
<thead>
<tr>
<th>label</th>
<th>string</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>B'</td>
<td>10110110</td>
<td>5</td>
</tr>
<tr>
<td>D'</td>
<td>01101100</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>11101110</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>00100000</td>
<td>1</td>
</tr>
</tbody>
</table>

Results of One Generation

• Has average population fitness gone up, gone down, or stayed the same?
  • Why?
• Are we making progress?
  • Why?

GA Procedure – Generational

Randomly initialize population
Repeat
  Repeat
    Selection
    Reproduction – Crossover
    Mutation
    Until new generation created
  Until solution found or resources exhausted

GA Procedure – Generational

Randomly initialize population
Repeat
  Repeat
    Selection
    Reproduction – Crossover
    Mutation
    Until new generation created
Until solution found or resources exhausted

Genetic Programming (GP)

• Genes – typically operators and operands
• Chromosome – typically tree of genes
  • Also called genotype or individual
  • Note lack of distinction between:
    • chromosome and genotype
    • genotype and phenotype
• Locus – not well defined
• Population – collection of individuals
• Generation – population at a given time
GP Individual
- Structure

GP Individual
- Complete

$$\frac{a + (b \times (c + d)) - (e / f)}{}$$

GP Crossover

GP Mutation

Artificial Ant: Problem Definition
- Navigate along food trail (Koza, 1992)
  - Trail has
    - turns
    - gaps
    - maximum moves allowed
- Fitness – amount of uneaten food at run end

Artificial Ant: Setup

<table>
<thead>
<tr>
<th>Non-Terminals</th>
<th>Terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF-FOOD-AHEAD</td>
<td>MOVE (forward)</td>
</tr>
<tr>
<td>PROGN2</td>
<td>LEFT (turn)</td>
</tr>
<tr>
<td>PROGN3</td>
<td>RIGHT (turn)</td>
</tr>
<tr>
<td></td>
<td>All terminals modify state</td>
</tr>
</tbody>
</table>
Artificial Ant: Sante Fe Trail

Sante Fe Trail: Sample Solution

(progn
  (if-food-ahead left
    (if-food-ahead
      (progn2 move left)
        (if-food-ahead right right)
      )
    )
  (if-food-ahead
    (progn2 move left)
      (if-food-ahead right right)
    )
  (progn3
    (if-food-ahead move right)
      (progn2 move right)
      (progn2 right left)
    )
)

Artificial Ant: Los Altos Trail

Humie Example: Antenna Design

• NASA Space Technology 5 Mission:
  • Three micro-satellites exploring Earth’s magnetic fields
  • Requirements:
    • wide beam width
    • circularly-polarized wave
    • wide bandwidth
  • Competitive bid selected human engineering team (contractor)
    • team created antenna design based on best engineering practices

• NASA Space Technology 5 Mission:
  • In addition, different team used evolutionary computation methods
    • Evolvable Systems Group at NASA Ames Research Center
    • genetic algorithms
    • genetic programming

Annual "Humie" Awards
For Human-Competitive Results
Produced by Genetic and Evolutionary Computation
Held at the Annual Genetic and Evolutionary Computation Conference

www.human-competitive.org
Humie Example: Antenna Design

- NASA Space Technology 5 Mission
  - Conventional design
    - did not meet mission requirements
    - required 5 person-months to complete
  - Evolved designs
    - did meet mission requirements
    - required 3 person-months to complete

Questions?