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.: 00100001011011101110001011110l
  - 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

    - Off-spring:

      11111|011011101110|1111

      ↕ ↕

      00001|111111111111|0010

GA Operators 2

- Mutation
  - Example single point mutation
    - Original: 11111011011101110111 1
      ↓
    - Mutated: 111*0*1011011101110111 1

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
  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:

label	string	fitness
A	00000110	2
B	11101110	6
C	00100000	1
D	00110100	3

GA Selection

label string fitness
A 000001102
B 111011106
C 001000001
D 001101003

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

GA Selection

label string fitness
A 000001102
B 111011106
C 001000001
D 001101003

- Can do selection proportional to fitness:
  AABBBBBBCDDD
- Generate numbers from 1 to 12 (6, 10, 9, 6)
- Select corresponding parents (B, D, C, B)

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|0110|110
  D' 0|1101|100

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:

label	string	fitness
B'	10110110	5
D'	01101100	4
B	11101110	6
C	00100000	1

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*

  – need a criterion,

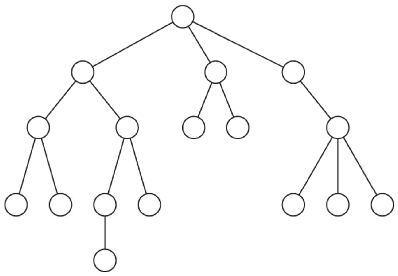
  e.g., an individual has all ones

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

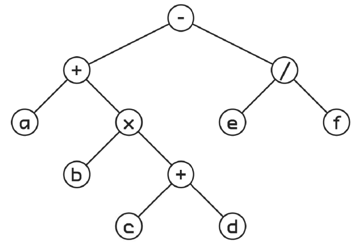
GP Individual

- Structure



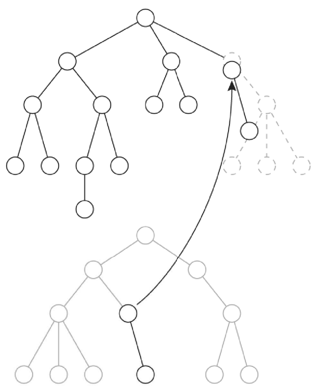
GP Individual

- Complete

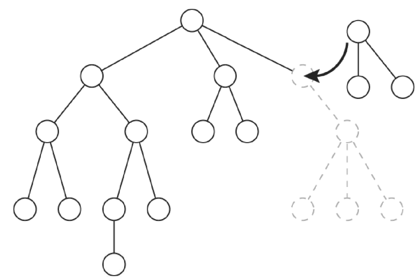


(a + (b x (c + d))) - (e / f)

GP Crossover

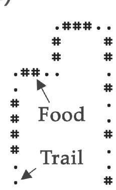


GP Mutation



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

Non-Terminals

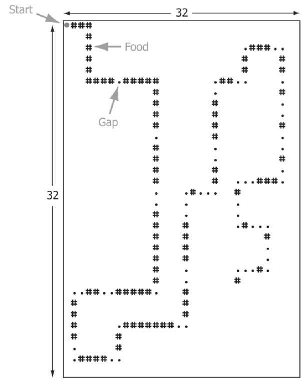
- IF-FOOD-AHEAD
- PROGN2
- PROGN3

Terminals

- MOVE (forward)
- LEFT (turn)
- RIGHT (turn)

All terminals modify state

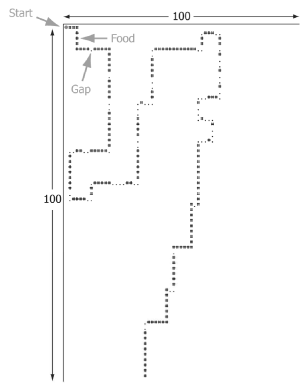
Artificial Ant: Sante Fe Trail

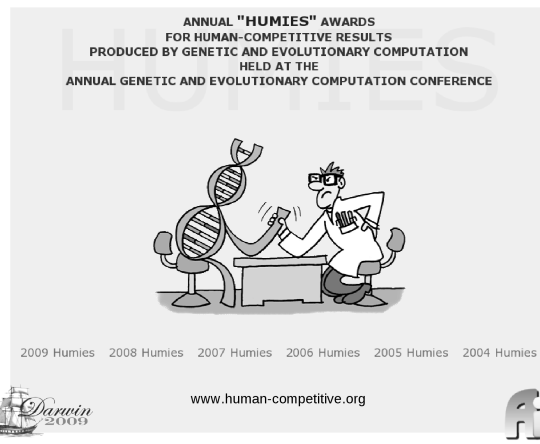


Sante Fe Trail: Sample Solution

```
(progn3
  (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

Humie Example: Antenna Design

- 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

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?