Evolutionary computing produces high-quality partial solutions to problems through natural selection and survival of the fittest. Compare to natural biological systems that adapt and learn over time.

Genetic Algorithms
CSCI-2300 Introduction to Algorithms

David Goldschmidt, Ph.D.
Rensselaer Polytechnic Institute
April 29, 2013

Genetic Algorithm Example

- Find the maximum value of function
  \[ f(x) = -x^2 + 15x \]
- Represent problem using chromosomes built from four genes:

<table>
<thead>
<tr>
<th>Integer</th>
<th>String</th>
<th>Decoded Integer</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0001</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>7</td>
<td>44</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>9</td>
<td>56</td>
</tr>
<tr>
<td>6</td>
<td>1000</td>
<td>10</td>
<td>248</td>
</tr>
</tbody>
</table>

Fitness function here is simply the original function \( f(x) = -x^2 + 15x \)

Genetic Algorithm Example

- Determine chromosome fitness for each chromosome:

<table>
<thead>
<tr>
<th>Chromosome label</th>
<th>Chromosome string</th>
<th>Decoded integer</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>1100</td>
<td>12</td>
<td>16.5</td>
</tr>
<tr>
<td>X2</td>
<td>0100</td>
<td>4</td>
<td>20.2</td>
</tr>
<tr>
<td>X3</td>
<td>0001</td>
<td>1</td>
<td>6.4</td>
</tr>
<tr>
<td>X4</td>
<td>1110</td>
<td>14</td>
<td>6.4</td>
</tr>
<tr>
<td>X5</td>
<td>0111</td>
<td>7</td>
<td>25.7</td>
</tr>
<tr>
<td>X6</td>
<td>1001</td>
<td>9</td>
<td>24.8</td>
</tr>
</tbody>
</table>

Genetic Algorithm Example

- Use fitness ratios to determine which chromosomes are selected for crossover and mutation operations:

| X1 | 16.5% |
| X2 | 20.2% |
| X3 | 6.4%  |
| X4 | 6.4%  |
| X5 | 25.3% |
| X6 | 24.8% |
The fitness function determines which chromosomes are selected for reproduction and which are discarded. Order the population based on fitness values, and calculate the fitness of each individual chromosome using \( f(x) \).
Genetic Algorithms – Step 5

- Using $p_c$, select pairs of chromosomes for crossover
- Using $p_m$, select chromosomes for mutation
- Chromosomes are selected based on their fitness values using a roulette wheel approach:

Genetic Algorithms – Step 6

- Create a pair of offspring chromosomes by applying a crossover operation:

Genetic Algorithms – Step 6

- Mutate an offspring chromosome by applying a mutation operation:

Genetic Algorithms – Steps 7 & 8

- Step 7:
  - Place all generated offspring chromosomes in a new population
- Step 8:
  - Go back to Step 5 until the size of the new population is equal to the size of the initial population, $N$

Genetic Algorithms – Steps 9 & 10

- Step 9:
  - Replace the initial population with the new population
- Step 10:
  - Go back to Step 4 and repeat the process until termination criteria are satisfied
  - Typically repeat this process for 50-5000+ generations

Iteration
Crossword Puzzle Construction

- **Given:**
  - Dictionary of valid words and phrases
  - Empty crossword grid
- **Problem:**
  - Fill the crossword grid such that all words both across and down are valid
    - (assign clues later)

Termination Criteria

- **When do we stop?**
  - Pause a genetic algorithm after a given number of generations, then check the fittest chromosomes
    - If the fittest chromosomes are fit beyond a given threshold, terminate the genetic algorithm
  - Also consider stopping when the highest fitness value does not change for a large number of generations

Crossword Puzzle Construction

- **Genetic Algorithm (GA)**
  - Evolve a solution by crossovers and mutations through many generations
  - **Initial population** of crossword grids:
    - Random letters?
    - Random letters based on Scrabble® frequencies?
    - Random words from dictionary?
  - **Fitness** of each grid is number of valid words

Termination Criteria

- **When do we stop?**
  - Pause a genetic algorithm after a given number of generations, then check the fittest chromosomes
    - If the fittest chromosomes are fit beyond a given threshold, terminate the genetic algorithm
  - Also consider stopping when the highest fitness value does not change for a large number of generations

Computational Complexity

- **How long does it take for an algorithm to produce a solution?**
  - Depends on the size of the input and the complexity of the algorithm
  - The size of the input is \( n \)
  - The complexity of the algorithm is classified based on its expected run time

Crossword Puzzle Construction

- **Genetic Algorithms**
  - Genetic algorithms are often well-suited to producing reasonable solutions to intractable problems
    - Intractable problems are problems with excessive computational complexity
      - i.e. in the Nondeterministic Polynomial (NP) class of problems
    - A reasonable solution is a partial or inexact solution that adequately solves the problem in polynomial time

Computational Complexity

- **Big O notation** measures the expected run time of an algorithm (i.e. its computational complexity)
  - **Constant time**: \( O(1) \)
  - **Logarithmic time**: \( O(\log n) \)
  - **Linear time**: \( O(n) \)
  - **Linearithmic time**: \( O(n \log n) \)
  - **Quadratic time**: \( O(n^2) \)
  - **Exponential time**: \( O(c^n) \)
  - **Factorial time**: \( O(n!) \)
Genetic Algorithms Example

- Consider the Traveling Salesman Problem (TSP) in which a salesman aims to visit \( n \) cities exactly once covering the least distance
  
  - Starting at any given node, choose from \( n-1 \) remaining nodes, then choose from \( n-2 \) remaining nodes, etc.
  
  - Testing every possible route takes \((n-1)!\) steps

http://mathworld.wolfram.com/TravelingSalesmanProblem.html
http://www.tsp.gatech.edu/games/index.html

- Use a genetic algorithm to evolve a near-optimal solution to the TSP
  
  - Label cities A, B, C, D, E, F, etc.
  
  - Example circuits: ABCDEF, BDAFCE, FBECAD
  
  - How do we perform crossover operations?
    
    - Basic crossovers might result in invalid members of the population
    
    - e.g. combining ABCDEF and BDAFCE may result in ABFCE

Genetic Algorithms Example

- Key challenge of developing a genetic algorithm is often the representation of the problem
  
  - For TSP, consider a standard ordering ABCDEF, assigning the code 123456
  
  - All other sequences encoded based on the removal of letters
  
  - Basic crossover works...

Genetic Algorithms Example

- All other sequences encoded based on the removal of letters from standard ordering
  
  - Sequence BDAFCE has code 231311

  - Basic crossover works...

Genetic Algorithms Example

- Another approach:
  
  http://www.dna-evolutions.com/dnaappletsample.html

Genetic Algorithms Example

- Combining ACEDB with ABCED...

  ...yields ACBED

From A. K. Dewdney’s The (New) Turing Omnibus, Computer Science Press, New York, 1993
Genetic Algorithms

- Advantages of genetic algorithms:
  - Often outperform “brute force” approaches by randomly jumping around the search space.
  - Ideal for problem domains in which near-optimal (as opposed to exact) solutions are adequate.

- Disadvantages of genetic algorithms:
  - Might not find any satisfactory partial solutions.
  - Tuning can be a challenge.