Genetic Algorithms

22c: 145, Chapter 4.1

What are Genetic Algorithms?
A technique borrowed from the theory of biological evolution that is used to create optimization procedures or methodologies, usually implemented on computers, that are used to solve problems.

Classes of Search Techniques

- Evolutionary strategies
- Centralized
- Steady-state
- Generational
- Distributed
- Parallel

- Genetic algorithms
- Evolutionary algorithms
- Simulated annealing
- Guided random search techniques
- Dynamic programming
- Enumerative techniques

The Argument
Evolution has optimized biological processes; therefore, adoption of the evolutionary paradigm to computation and other problems can help us find optimal solutions.

Natural Selection
- Limited number of resources
- Competition results in struggle for existence
- Success depends on fitness --
  - fitness of an individual: how well-adapted an individual is to their environment. This is determined by their genes (blueprints for their physical and other characteristics).
- Successful individuals are able to reproduce and pass on their genes

When changes occur ...  
- Previously “fit” (well-adapted) individuals will no longer be best-suited for their environment
- Some members of the population will have genes that confer different characteristics than “the norm”. Some of these characteristics can make them more “fit” in the changing environment.
Genetic Change in Individuals

- Mutation in genes
  - may be due to various sources (e.g. UV rays, chemicals, etc.)
  
  Start:
  1001001001001001001001

  After Mutation:
  1001000001001001001001

Recombination (Crossover)

- occurs during reproduction -- sections of genetic material exchanged between two chromosomes

The Nature of Computational Problems

- Require search through many possibilities to find a solution
  - (e.g. search through sets of rules for one set that best predicts the ups and downs of the financial markets)
  - Search space too big -- search won't return within our lifetimes

- Require algorithm to be adaptive or to construct original solution
  - (e.g. interfaces that must adapt to idiosyncrasies of different users)

Why Evolution Proves to be a Good Model for Solving these Types of Problems

- Evolution is a method of searching for an (almost) optimal solution
  - Possibilities -- all individuals
  - Best solution -- the most "fit" or well-adapted individual

- Evolution is a parallel process
  - Testing and changing of numerous species and individuals occur at the same time (or, in parallel)

- Evolution can be seen as a method that designs new (original) solutions to a changing environment

The Metaphor

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

- Bit strings  
  (0101 ... 1100)
- Real numbers  
  (43.2 -33.1 ... 0.0 89.2)
- Permutations of element  
  (E11 E3 E7 ... E1 E15)
- Lists of rules  
  (R1 R2 R3 ... R22 R23)
- Program elements  
  (genetic programming)
- ... any data structure ...

Encoding the Problem

- Example: Looking for a new site which is closest to several nearby cities.
- Express the problem in terms of a bit string  
  \[ z = (10010101011100) \]

  where the first 8 bits of the string represent the X-coordinate and the second 8 bits represent the Y-coordinate.

Genetic Algorithms

- Closely follows a biological approach to problem solving
- A simulated population of randomly selected individuals is generated then allowed to evolve

Genetic Algorithms

- Closely follows a biological approach to problem solving
- A simulated population of randomly selected individuals is generated then allowed to evolve

Basic Genetic Algorithm

- Step 1. Generate a random population of \( n \) individuals
- Step 2. Assign a fitness value to each individual
- Step 3. Repeat until \( n \) children have been produced
  - Choose 2 parents based on fitness proportional selection
  - Apply genetic operators to copies of the parents
  - Produce new individuals
- Step 4. If the stop criterion meets, stop; else go to Step 2.

Notes:

- GAs fall into the category of “generate and test” algorithms
- They are stochastic, population-based algorithms
- Variation operators (recombination and mutation) create the necessary diversity and thereby facilitate novelty
- Selection reduces diversity and acts as a force pushing quality
Fitness Function

- For each individual in the population, evaluate its relative fitness
- For a problem with \( m \) parameters, the fitness can be plotted in an \( m+1 \) dimensional space

Sample Search Space

- A randomly generated population of individuals will be randomly distributed throughout the search space

An Abstract Example

- Distribution of Individuals in Generation 0
- Distribution of Individuals in Generation \( N \)

Genetic Operators

- Cross-over
- Mutation

Production of New Chromosomes

- 2 parents give rise to 2 children

Generations

- As each new generation of \( n \) individuals is generated, they replace their parent generation
- To achieve the desired results, typically 500 to 5000 generations are required
The Evolutionary Cycle

- Selection → Parents
- Recombination → Offspring
- Population
- Mutation
- Replacement → Offspring

Ultimate Goal
- Each subsequent generation will evolve toward the global maximum
- After sufficient generations a near optimal solution will be present in the population of chromosomes

Example: Find the max value of \( f(x_1, \ldots, x_{100}) \).
- Population: real vectors of length 100.
- Mutation: randomly replace a value in a vector.
- Combination: Take the average of two vectors.

Discrete Recombination
- Similar to crossover of genetic algorithms
- Equal probability of receiving each parameter from each parent
  - \( (8, 12, 31, \ldots, 5) \) \( (2, 5, 23, \ldots, 14) \)
  - \( (2, 12, 31, \ldots, 14) \)

Intermediate Recombination
- Often used to adapt the strategy parameters
- Each child parameter is the mean value of the corresponding parent parameters
  - \( (8, 12, 31, \ldots, 5) \) \( (2, 5, 23, \ldots, 14) \)
  - \( (5, 8.5, 27, \ldots, 9.5) \)

Dynamic Evolution
- Genetic algorithms can adapt to a dynamically changing search space
- Seek out the moving maximum via a parasitic fitness function
  - as the chromosomes adapt to the search space, so does the fitness function
A Simple Example

The Traveling Salesman Problem:

Find a tour of a given set of cities so that
- each city is visited only once
- the total distance traveled is minimized

Representation

Representation is an ordered list of city numbers known as an order-based GA.

1) London    3) Iowa City    5) Beijing    7) Tokyo
2) Venice    4) Singapore    6) Phoenix    8) Victoria

CityList1    (3 5 7 2 1 6 4 8)
CityList2    (2 5 7 6 8 1 3 4)

Crossover

Crossover combines inversion and recombination:

Parent1    (3 5 7 2 1 6 4 8)
Parent2    (2 5 7 6 8 1 3 4)
Child      (5 8 7 2 1 6 3 4)

- Copy a randomly selected portion of Parent1 to Child
- Fill the blanks in Child with those numbers in Parent2 from left to right, as long as there are no duplication in Child.
  This operator is called the Order1 crossover.

Mutation

Mutation involves swapping two numbers of the list:

Before:    (5 8 7 2 1 6 3 4)
After:     (5 8 6 2 1 7 3 4)

Different Recombination (Crossover) for TSP

TSP: The Traveling Salesman Problem

- Near Neighbor Crossover
- Greedy Edge Crossover

Nearest Neighbor Crossover (NNX)

- Input: Two TSP tours T1, T2
- Output: one new TSP tour T3.
- Step 1: Create a graph of n nodes and 2n edges from T1 and T2
- Step 2: Randomly choose a node and let T3 contains only this node
- Step 3: Repeat until all n nodes are in T3:
  - Pick the nearest node x (not in T3) to the last node in T3 and add x into T3.
Greedy Edge Crossover (GEX)

- **Input:** Two TSP tours T1, T2
- **Output:** one new TSP tour T3.
- **Step 1:** Sort the edges from T1 and T2
- **Step 2:** For each edge e in the sorted list, if adding e into T3 creates neither a node with three neighbors, nor a cycle of less than n node, add e into T3.

**TSP Example: 30 Cities**

**Solution i (Distance = 941)**

**Solution j (Distance = 800)**
Overview of Performance

Typical run: progression of fitness

Are long runs beneficial?

Is it worth expending effort on smart initialisation?

Answer:
- It depends how much you want the last bit of progress
- It may be better to do more shorter runs
Many Variants of GA
- Different kinds of selection (not roulette)
  - Tournament
  - Elitism, etc.
- Different recombination
  - Multi-point crossover
  - 3 way crossover etc.
- Different kinds of encoding other than bit-string
  - Numeric values
  - Trees of symbols
- Different kinds of mutation

Forming the Next Generation
- Number of individuals selected to be parents (p)
  - too many: lots of persistent bad traits
  - too few: stagnant gene pool
- Total number of children produced (c)
  - limited by computer resources
  - more children ⇒ faster evolution

Evolution Process
- p parents produce c children in each generation
- Four types of processes:
  - p, c
  - p/r, c
  - p+c
  - p/r+c

p, c
- p parents produce c children using mutation only (no recombination)
- The fittest p children become the parents for the next generation
- Parents are not part of the next generation
- c ≥ p
- p/r, c is the above with recombination

D+C
- p parents produce c children using mutation only (no recombination)
- The fittest p individuals (parents or children) become the parents of the next generation
- p/r+c is the above with recombination

Tuning a GA
- “Typical” tuning parameters for a small problem
  
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>50 - 100</td>
</tr>
<tr>
<td>Children per generation</td>
<td>= population size</td>
</tr>
<tr>
<td>Crossovers</td>
<td>0 - 3</td>
</tr>
<tr>
<td>Mutations</td>
<td>&lt; 5%</td>
</tr>
<tr>
<td>Generations</td>
<td>20 - 20,000</td>
</tr>
</tbody>
</table>

- Other concerns
  - population diversity
  - ranking policies
  - removal policies
  - role of random bias
Domains of Application
- Numerical, Combinatorial Optimization
- System Modeling and Identification
- Planning and Control
- Engineering Design
- Data Mining
- Machine Learning
- Artificial Life

Drawbacks of GA
- Difficult to find an encoding for a problem
- Difficult to define a valid fitness function
- May not return the global maximum

Why use a GA?
- Requires little insight into the problem
- The problem has a very large solution space
- The problem is non-convex
- Does not require derivatives
- Objective function need not be smooth
- Variables do not need to be scaled
- Fitness function can be noisy (e.g., process data)
- When the goal is a good solution

When NOT to use a GA?
- If global optimality is required
- If problem insight can:
  - Significantly impact algorithm performance
  - Simplify problem representation
- If the problem is highly constrained
- If the problem is smooth and convex
  - Use a gradient-based optimizer
- If the search space is very small
  - Use enumeration

Taxonomy

Application to Stock Trading
- There are many technical indicators of stock movements
- A model using these indicators can be represented by a tree
- Use the past market data to define the fitness function
- Use Genetic Programming to create trees
Technical Indicators of SP500

- Move Average of past n days:
  - E.g., MA5, MA20, MA50, MA200
- Weighted Move Average
  - Let $p_1, p_2, \ldots, p_n$ be the prices of the stock in the past n days.
  - $WMA_M = \frac{n p_n + (n-1)p_{n-1} + \cdots + 2p_{n-m+2} + p_{n-m+1}}{n + (n-1) + \cdots + 2 + 1}$

- Exponential Move Average
  - $EMA_{today} = \frac{p_1 + (1-\alpha)p_2 + (1-\alpha)^2p_3 + (1-\alpha)^3p_4 + \cdots}{1 + (1-\alpha) + (1-\alpha)^2 + (1-\alpha)^3 + \cdots}$
  - $EMA_{today} = \alpha \times (p_1 + (1-\alpha)p_2 + (1-\alpha)^2p_3 + (1-\alpha)^3p_4 + \cdots)$

- Moving Average Convergence/Divergence (MACD)
  - $MACD(12) = EMA(12) - EMA(26)$
  - $MACD9: 9$ day EMA of $MACD(12)$

- McClennan Oscillator (MCCL):
  - $MCCL = 19$-days $EMA(Diff) - 39$-days $EMA(Diff)$
  - where $Diff = \#$ of advancing stocks - $\#$ of declining stocks.

A Function on Stock Market

- The terminal set: all the technical indicators, the real numbers in $[-1, 1]$ and $SP500 = (SP500_{today} - MA(200))/SP500_{deviation}$
- The function set: $+, -, *, IF(x, y, z)$, $IFGT(x, w, y, z)$:
  - $IF(x, y, z) = \text{if } (x > 0) \text{ then } y \text{ else } z$
  - $IFGT(x, w, y, z) = \text{if } (x > w) \text{ then } y \text{ else } z$
A Function on Stock Market