Genetic Algorithms: A Tutorial

“Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime.”

- Salvatore Mangano
  Computer Design, May 1995
The Genetic Algorithm

- Directed search algorithms based on the mechanics of biological evolution
- Developed by John Holland, University of Michigan (1970’s)
  - To understand the adaptive processes of natural systems
  - To design artificial systems software that retains the robustness of natural systems
The Genetic Algorithm (cont.)

- Provide efficient, effective techniques for optimization and machine learning applications
- Widely-used today in business, scientific and engineering circles
Classes of Search Techniques
Components of a GA

A problem to solve, and ...

- Encoding technique \((\text{gene, chromosome})\)
- Initialization procedure \((\text{creation})\)
- Evaluation function \((\text{environment})\)
- Selection of parents \((\text{reproduction})\)
- Genetic operators \((\text{mutation, recombination})\)
- Parameter settings \((\text{practice and art})\)
Simple Genetic Algorithm

{ 
initialize population;
evaluate population;
while TerminationCriteriaNotSatisfied
{
select parents for reproduction;
perform recombination and mutation;
evaluate population;
}
}
The GA Cycle of Reproduction

- Reproduction
  - Parents
  - Children
- Population
  - Evaluated Children
- Evaluation
  - Modified Children
- Modification
  - Children
- Discard
  - Deleted Members
Population

Chromosomes could be:

♦ 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 ...
Reproduction

Parents are selected at random with selection chances biased in relation to chromosome evaluations.
Chromosome Modification

- Modifications are stochastically triggered
- Operator types are:
  - Mutation
  - Crossover (recombination)
**Mutation: Local Modification**

Before: \((1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0)\)

After: \((0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0)\)

Before: \((1.38 \ -69.4 \ 326.44 \ 0.1)\)

After: \((1.38 \ -67.5 \ 326.44 \ 0.1)\)

- Causes movement in the search space (local or global)
- Restores lost information to the population
Crossover: Recombination

\[
P_1 \quad (0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0) \quad \rightarrow \quad (0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0) \quad C_1
\]
\[
P_2 \quad (1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0) \quad \rightarrow \quad (1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0) \quad C_2
\]

Crossover is a critical feature of genetic algorithms:

- It greatly accelerates search early in evolution of a population
- It leads to effective combination of schemata (subsolutions on different chromosomes)
Evaluation

- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving
Deletion

- Generational GA: entire populations replaced with each iteration
- Steady-state GA: a few members replaced each generation
An Abstract Example

Distribution of Individuals in Generation 0

Distribution of Individuals in Generation N
"The Gene is by far the most sophisticated program around."

- Bill Gates, Business Week, June 27, 1994
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) Dunedin      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: (2 5 7 2 1 6 3 4)

This operator is called the *Order1* crossover.
Mutation

Mutation involves reordering of the list:

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

After: $(5 \ 8 \ 6 \ 2 \ 1 \ 7 \ 3 \ 4)$
TSP Example: 30 Cities
Solution \_i (Distance = 941)
Solution \( j \) (Distance = 800)

TSP30 (Performance = 800)
Solution $k$ (Distance = 652)
Best Solution (Distance = 420)

TSP30 Solution (Performance = 420)
Overview of Performance

TSP30 - Overview of Performance

- Best
- Worst
- Average
Considering the GA Technology

“Almost eight years ago ... people at Microsoft wrote a program [that] uses some genetic things for finding short code sequences. Windows 2.0 and 3.2, NT, and almost all Microsoft applications products have shipped with pieces of code created by that system.”

- Nathan Myhrvold, Microsoft Advanced Technology Group, Wired, September 1995
Issues for GA Practitioners

- Choosing basic implementation issues:
  - representation
  - population size, mutation rate, ...
  - selection, deletion policies
  - crossover, mutation operators

- Termination Criteria

- Performance, scalability

- Solution is only as good as the evaluation function (often hardest part)
Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Supports multi-objective optimization
- Good for “noisy” environments
- Always an answer; answer gets better with time
- Inherently parallel; easily distributed
Benefits of Genetic Algorithms (cont.)

- Many ways to speed up and improve a GA-based application as knowledge about problem domain is gained
- Easy to exploit previous or alternate solutions
- Flexible building blocks for hybrid applications
- Substantial history and range of use
When to Use a GA

- Alternate solutions are too slow or overly complicated
- Need an exploratory tool to examine new approaches
- Problem is similar to one that has already been successfully solved by using a GA
- Want to hybridize with an existing solution
- Benefits of the GA technology meet key problem requirements
## Some GA Application Types

<table>
<thead>
<tr>
<th>Domain</th>
<th>Application Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>gas pipeline, pole balancing, missile evasion, pursuit</td>
</tr>
<tr>
<td>Design</td>
<td>semiconductor layout, aircraft design, keyboard configuration, communication networks</td>
</tr>
<tr>
<td>Scheduling</td>
<td>manufacturing, facility scheduling, resource allocation</td>
</tr>
<tr>
<td>Robotics</td>
<td>trajectory planning</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>designing neural networks, improving classification algorithms, classifier systems</td>
</tr>
<tr>
<td>Signal Processing</td>
<td>filter design</td>
</tr>
<tr>
<td>Game Playing</td>
<td>poker, checkers, prisoner’s dilemma</td>
</tr>
<tr>
<td>Combinatorial Optimization</td>
<td>set covering, travelling salesman, routing, bin packing, graph colouring and partitioning</td>
</tr>
</tbody>
</table>
Conclusions

Question: ‘If GAs are so smart, why ain’t they rich?’

Answer: ‘Genetic algorithms are rich - rich in application across a large and growing number of disciplines.’

- David E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning