Introduction To Genetic Algorithms

CSE352
Professor Anita Wasilewska
Overview

- Introduction To Genetic Algorithms (GA)
- GA Operators and Parameters
- Genetic Algorithms To Solve The Traveling Salesman Problem (TSP)
- Summary
References

References

WEBSITES

- www.iitk.ac.in/kangal
- www.math.princeton.edu
- www.genetic-programming.com
- www.garage.cse.msu.edu
- www.aic.nre.navy.mie/galist
History Of Genetic Algorithms

- “Evolutionary Computing” was introduced in the 1960s by I. Rechenberg

- John Holland wrote the first book on Genetic Algorithms ‘Adaptation in Natural and Artificial Systems’ in 1975

- In 1992 John Koza used genetic algorithm to evolve programs to perform certain tasks

- He called his method “Genetic Programming”
What Are Genetic Algorithms?

Genetic Algorithms are search and optimization techniques based on Darwin’s Principle of Natural Selection.
Darwin’s Principle Of Natural Selection

- IF there are organisms that reproduce, and
- IF offsprings inherit traits from their progenitors, and
- IF there is variability of traits, and
- IF the environment cannot support all members of a growing population,

THEN those members of the population with less-adaptive traits (determined by the environment) will die out, and

THEN those members with more-adaptive traits (determined by the environment) will thrive

The result is the evolution of species
Basic Idea Of Principle Of Natural Selection

“Select The Best, Discard The Rest”
An Example of Natural Selection

- Giraffes have long necks

Giraffes with slightly longer necks could feed on leaves of higher branches when all lower ones had been eaten off
- They had a better chance of survival.
- Favorable characteristic propagated through generations of giraffes.
- Now, evolved species has long necks.

NOTE: Longer necks may have been a deviant characteristic (mutation) initially but since it was favorable, was propagated over generations. Now an established trait.

So, some mutations are beneficial
Evolution Through Natural Selection

Initial Population Of Animals

Struggle For Existence - Survival Of the Fittest

Surviving Individuals Reproduce, Propagate Favorable Characteristics

Millions Of Years

Evolved Species

(Favorable Characteristic Now A Trait Of Species)
Genetic Algorithms implement Optimization Strategies by simulating evolution of species through natural selection
Working Mechanism Of GA

Begn

Initialize population

Evaluate Solutions

T = 0

Optimum Solution?

Y

T = T + 1

Stop

N

Selection

Crossover

Mutation
Simple Genetic Algorithm

Simple_Genetic_Algorithm()
{
    Initialize the Population;
    Calculate Fitness Function;

    While(Fitness Value != Optimal Value)
    {
        Selection; // Natural Selection, Survival Of Fittest
        Crossover; // Reproduction, Propagate favorable characteristics
        Mutation; // Mutation
        Calculate Fitness Function;
    }
}
## Nature to Computer Mapping

<table>
<thead>
<tr>
<th>Nature</th>
<th>Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population</td>
<td>Set of solutions.</td>
</tr>
<tr>
<td>Individual</td>
<td>Solution to a problem.</td>
</tr>
<tr>
<td>Fitness</td>
<td>Quality of a solution.</td>
</tr>
<tr>
<td>Chromosome</td>
<td>Encoding for a Solution.</td>
</tr>
<tr>
<td>Gene</td>
<td>Part of the encoding of a solution.</td>
</tr>
<tr>
<td>Reproduction</td>
<td>Crossover</td>
</tr>
</tbody>
</table>
GA Operators and Parameters
Encoding

ENCODING is a process of representing the solution in the form of a string that conveys the necessary information.

- Just as in a chromosome, each gene controls a particular characteristic of the individual, similarly, each bit in the string represents a characteristic of the solution.
Encoding Methods

- **Binary Encoding** – Most common method of encoding.
- **Chromosomes** are strings of 1s and 0s and each position in the chromosome represents a particular characteristic of the problem.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>10110010110011100101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>111111110000000011111</td>
</tr>
</tbody>
</table>
Encoding Methods

- **Permutation Encoding** – Useful in ordering problems such as the Traveling Salesman Problem (TSP)

In TS, every chromosome is a string of numbers, each of which represents a city to be visited.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>1 5 3 2 6 4 7 9 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>8 5 6 7 2 3 1 4 9</td>
</tr>
</tbody>
</table>
Encoding Methods

- **Value Encoding** – Used in problems where complicated values, such as real numbers, are used and where binary encoding would not suffice. Good for some problems, but often necessary to develop some specific crossover and mutation techniques for these chromosomes.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>1.235</th>
<th>5.323</th>
<th>0.454</th>
<th>2.321</th>
<th>2.454</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>(left), (back), (left), (right), (forward)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A fitness function quantifies the optimality of a solution (chromosome) so that that particular solution may be ranked against all the other solutions.

- A fitness value is assigned to each solution depending on how close it actually is to solving the problem.

- **Ideal fitness function** correlates closely to goal and is quickly computable.

- In **TSP**, \( f(x) \) is the sum of distances between the cities in solution.

- The lesser the value, the fitter the solution is.
Recombination

Recombination is a process that determines which solutions are to be preserved and allowed to reproduce and which ones deserve to die out.

- The primary objective of the recombination operator is to emphasize the good solutions and eliminate the bad solutions in a population, while keeping the population size constant.

- “Selects The Best, Discards The Rest”

- “Recombination” is different from “Reproduction”
Recombination

- Identify the good solutions in a population.
- Make multiple copies of the good solutions.
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population.
**Roulette Wheel Selection**

- Each current string in the population has a slot assigned to it which is in proportion to it’s fitness.

- We spin the weighted roulette wheel thus defined $n$ times (where $n$ is the total number of solutions).

- Each time Roulette Wheel stops, the string corresponding to that slot is created.

Strings that are fitter are assigned a larger slot and hence have a better chance of appearing in the new population.
Example Of Roulette Wheel Selection

<table>
<thead>
<tr>
<th>No.</th>
<th>String</th>
<th>Fitness</th>
<th>% Of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01101</td>
<td>169</td>
<td>14.4</td>
</tr>
<tr>
<td>2</td>
<td>11000</td>
<td>576</td>
<td>49.2</td>
</tr>
<tr>
<td>3</td>
<td>01000</td>
<td>64</td>
<td>5.5</td>
</tr>
<tr>
<td>4</td>
<td>10011</td>
<td>361</td>
<td>30.9</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>1170</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Roulette Wheel For Example
Crossover

Crossover is the process in which two chromosomes (strings) combine their genetic material (bits) to produce a new offspring which possesses both their characteristics.

- Two strings are picked from the mating pool at random to cross over

- The method chosen depends on the Encoding Method
Crossover Methods

- **Single Point Crossover** - a random point is chosen on the individual chromosomes (strings) and the genetic material is exchanged at this point.
Crossover Methods

- **Single Point Crossover**

<table>
<thead>
<tr>
<th>Chromosome1</th>
<th>11011</th>
<th>00100110110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2</td>
<td>11011</td>
<td>11000011110</td>
</tr>
<tr>
<td>Offspring 1</td>
<td>11011</td>
<td>11000011110</td>
</tr>
<tr>
<td>Offspring 2</td>
<td>11011</td>
<td>00100110110</td>
</tr>
</tbody>
</table>
Crossover Methods

- **Two-Point Crossover**: two random points are chosen on the individual chromosomes (strings) and the genetic material is **exchanged** at these points.

<table>
<thead>
<tr>
<th>Chromosome 1</th>
<th>11011</th>
<th>00100</th>
<th>110110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2</td>
<td>10101</td>
<td>11000</td>
<td>011110</td>
</tr>
<tr>
<td>Offspring 1</td>
<td>10101</td>
<td>00100</td>
<td>011110</td>
</tr>
<tr>
<td>Offspring 2</td>
<td>11011</td>
<td>11000</td>
<td>110110</td>
</tr>
</tbody>
</table>

**NOTE**: These chromosomes are different from the last example.
Crossover Methods

- **Uniform Crossover** - each gene (bit) is selected randomly from one of the corresponding genes of the parent chromosomes

<table>
<thead>
<tr>
<th></th>
<th>Chromosome1</th>
<th>11011</th>
<th>00100</th>
<th>110110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome 2</td>
<td>10101</td>
<td>11000</td>
<td>011110</td>
<td></td>
</tr>
<tr>
<td>Offspring</td>
<td>10111</td>
<td>00000</td>
<td>110110</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** Uniform Crossover yields ONLY 1 offspring
Crossover

- **Crossover** between 2 good solutions
- **MAY NOT ALWAYS** yield a better or as good a solution

- Since parents **are good**, probability of the child being good **is high**

- If offspring **is not good** (poor solution), it will be removed in the next iteration during “Selection”
Elitism

Elitism is a method in which copies the best chromosome are added to the new offspring population before crossover and mutation.

- When creating a new population by crossover or mutation the best chromosome might be lost.
- Elitism lets GA to retain some number of the best individuals at each generation.
- It has been found that elitism significantly improves performance.
**Mutation**

*Mutation* is the **process** by which a string is deliberately changed so as to maintain diversity in the population set.

We saw in the giraffes’ example, that mutations could be beneficial.

*Mutation Probability* - determines how often the parts of a chromosome will be mutated.
Example Of Mutation

- For chromosomes using **Binary Encoding**, randomly selected bits are inverted.

<table>
<thead>
<tr>
<th>Offspring</th>
<th>11011 00100 110110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutated Offspring</td>
<td>11010 00100 100110</td>
</tr>
</tbody>
</table>

**NOTE:** The number of bits to be inverted depends on the Mutation Probability
Advantages Of GA

- **Global Search Methods**: GAs search for the function optimum starting from a population of points of the function domain, not a single one.

- This characteristic suggests that GAs are global search methods.

- They can, in fact, climb many peaks in parallel, reducing the probability of finding local minima, which is one of the drawbacks of traditional optimization methods.
Advantages of GA

- **Blind Search Methods:**
  - GAs only use the information about the objective function
  - GAs do not require knowledge of the first derivative or any other auxiliary information
  - They allow a number of problems to be solved without the need to formulate restrictive assumptions
  - For this reason, GAs are often called blind search methods.
Advantages of GAs

- GAs use probabilistic transition rules during iterations, unlike the traditional methods that use fixed transition rules.

This makes them more robust and applicable to a large range of problems.
Advantages of GAs

- GAs can be easily used in parallel machines
- Since in real-world design optimization problems, most computational time is spent in evaluating a solution, with multiple processors all solutions in a population can be evaluated in a distributed manner
- This reduces the overall computational time substantially
Genetic Algorithms To Solve The Traveling Salesman Problem (TSP)
The **Traveling Salesman Problem** is defined as:

‘We are given a set of cities and a symmetric distance matrix that indicates the cost of travel from each city to every other city. The goal is to find the shortest circular tour, visiting every city exactly once, so as to minimize the total travel cost, which includes the cost of traveling from the last city back to the first city.’
Encoding

- We represent every city with an integer

- Consider 6 Indian cities –
  Mumbai, Nagpur, Calcutta, Delhi, Bangalore and Chennai and assign a number to each.

  Mumbai  →  1
  Nagpur   →  2
  Calcutta →  3
  Delhi    →  4
  Bangalore →  5
  Chennai  →  6
Encoding

- Thus a path would be represented as a sequence of integers from 1 to 6

- The path [1 2 3 4 5 6] represents a path from Mumbai to Nagpur, Nagpur to Calcutta, Calcutta to Delhi, Delhi to Bangalore, Bangalore to Chennai, and finally from Chennai to Mumbai

- This is an example of Permutation Encoding as the position of the elements determines the fitness of the solution.
Fitness Function

- The **fitness function** will be the **total cost of the tour** represented by each chromosome.

- This can be calculated as the **sum of the distances** traversed in each travel segment.

The **Lesser The Sum, The Fitter The Solution** Represented By That Chromosome
## Distance/Cost Matrix For TSP

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>863</td>
<td>1987</td>
<td>1407</td>
<td>998</td>
<td>1369</td>
</tr>
<tr>
<td>2</td>
<td>863</td>
<td>0</td>
<td>1124</td>
<td>1012</td>
<td>1049</td>
<td>1083</td>
</tr>
<tr>
<td>3</td>
<td>1987</td>
<td>1124</td>
<td>0</td>
<td>1461</td>
<td>1881</td>
<td>1676</td>
</tr>
<tr>
<td>4</td>
<td>1407</td>
<td>1012</td>
<td>1461</td>
<td>0</td>
<td>2061</td>
<td>2095</td>
</tr>
<tr>
<td>5</td>
<td>998</td>
<td>1049</td>
<td>1881</td>
<td>2061</td>
<td>0</td>
<td>331</td>
</tr>
<tr>
<td>6</td>
<td>1369</td>
<td>1083</td>
<td>1676</td>
<td>2095</td>
<td>331</td>
<td>0</td>
</tr>
</tbody>
</table>

Cost matrix for six city example.  
*Distances in Kilometers*
Fitness Function

- So, for a chromosome [4 1 3 2 5 6], the total cost of travel or fitness will be calculated as shown below:

  \[
  \text{Fitness} = 1407 + 1987 + 1124 + 1049 + 331 + 2095 = 7993 \text{ kms}.
  \]

- Since our objective is to Minimize the distance, the lesser the total distance, the fitter the solution.
Selection Operator

We use **Tournament Selection**

As the name suggests *tournaments* are played between two solutions and the better solution is chosen and placed in the *mating pool*

Two other solutions are picked again and another slot in the *mating pool* is filled up with the better solution
Tournament

Mating Pool
Why we cannot use single-point crossover:

- Single point crossover method randomly selects a crossover point in the string and swaps the substrings.
- This may produce some invalid offsprings as shown below.
Crossover Operator


- This operator is different from other genetic sequencing operators in that it emphasizes *adjacency information* instead of the order or position of items in the sequence.

- The algorithm for the Edge-Recombination Operator involves constructing an Edge Table first
Edge Table

The *Edge Table* is an *adjacency table* that lists links *into* and *out of a city* found in the two parent sequences.

If an item is already in the edge table and we are trying to insert it again, that element of a sequence must be a *common edge* and is represented by inverting it's sign.
Finding The Edge Table

Parent 1

|   | 4 | 1 | 3 | 2 | 5 | 6 |

Parent 2

|   | 4 | 3 | 2 | 1 | 5 | 6 |

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-3</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-6</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-5</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Enhanced Edge Recombination Algorithm

1. Choose the initial city from one of the two parent tours. (It can be chosen randomly as according to criteria outlined in step 4). This is the current city.

2. Remove all occurrences of the current city from the left hand side of the edge table. (These can be found by referring to the edge-list for the current city).

3. If the current city has entries in it's edge-list, go to step 4. Otherwise go to step 5.

4. Determine which of the cities in the edge-list of the current city has the fewest entries in it's own edge-list. The city with fewest entries becomes the current city. In case a negative integer is present, it is given preference. Ties are broken randomly. Go to step 2.

5. If there are no remaining unvisited cities, then stop.

6. Otherwise, randomly choose an unvisited city and go to step 2.
Example Of Enhanced Edge Recombination Operator

Step 1

```
1 | 4 | 3 | 2 | 5
2 | -3| 5 | 1 |
3 | 1 | -2| 4 |
4 | -6| 1 | 3 |
5 | 3 | 2 | -6|
6 | -5| -4|
```

Step 2

```
1 | 3 | 2 | 5
2 | -3| 5 | 1 |
3 | 1 | -2|   |
4 | -6| 1 | 3 |
5 | 3 | 2 | -6|
6 | -5|   |   |
```
Example Of Enhanced Edge Recombination Operator

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>-3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 6 5

---

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 6 5 1
Example Of Enhanced Edge Recombination Operator

Step 5

Step 6
Mutation Operator

- The mutation operator induces a change in the solution, so as to maintain diversity in the population and prevent **Premature Convergence**

- We **mutate the string** by randomly selecting any two cities and interchanging their positions in the solution, thus giving rise to a new tour.

```
4 1 3 2 5 6
```

```
4 5 3 2 1 6
```
### Travelling Salesman Problem using Genetic Algorithm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter population size</td>
<td>20</td>
</tr>
<tr>
<td>Enter number of cities</td>
<td>20</td>
</tr>
<tr>
<td>Enter maximum generations</td>
<td>10000</td>
</tr>
<tr>
<td>Enter mutation probability</td>
<td>0.09</td>
</tr>
</tbody>
</table>
Initial Output For 20 cities: Distance=34985 km
Initial Population
Final Output For 20 cities : Distance=13170 km
Generation 4786
Summary
Genetic Algorithms (GAs) implement optimization strategies based on simulation of the natural law of evolution of a species by natural selection.

The basic GA Operators are:
- Encoding
- Recombination
- Crossover
- Mutation

GAs have been applied to a variety of function optimization problems, and have been shown to be highly effective in searching a large, poorly defined search space even in the presence of difficulties such as high-dimensionality, multi-modality, discontinuity and noise.