Genetic Algorithms

Presented by : Yash Garg

Index

Evolutionary Computing

1. In computer science, evolutionary computation is the study of algorithms for global optimization inspired by biological evolution. In technical terms, they are a family of population-based trial and error problem solvers with a optimization objectives and several heuristics.

2. In evolutionary computing, an initial set of candidate solutions is generated and iteratively updated. Each new generation is produced by stochastically removing less desired solutions, and introducing small random changes.

3. Examples - Ant Colony Optimization, Genetic Algorithms, Neuroevolution, etc.

Darwinian Natural Selection

1. Heredity - There must be a process in place by which children receive properties of their parents.

2. Variation - There must be a variety of traits present between a population or a means with which to introduce variation.

3. Selection - There must be a mechanism by which some members of a population have the opportunity to be parents and pass down their genetic information and some do not. This typically refers to “survival of the fittest”.

Traditional Genetic Algorithm

i) Initialise :- Create a population of N elements each with randomly generated DNA.

ii) Repeat (each iteration is next generation) :-

1. Selection - Evaluate the fitness of each element and build a mating pool.

2. Reproduction Phase - Repeat N (population size) number of times
    a. Pick multiple parents from mating pool (eg- take 2 parents)
    b. Crossover - Create children from those parents (eg- 2 parents can produce 1, 2 or more children)
    c. Mutation - Mutate the obtained children with mutation rate (probability)
    d. Add new children to next generation population

3. Replace the old population with new population.

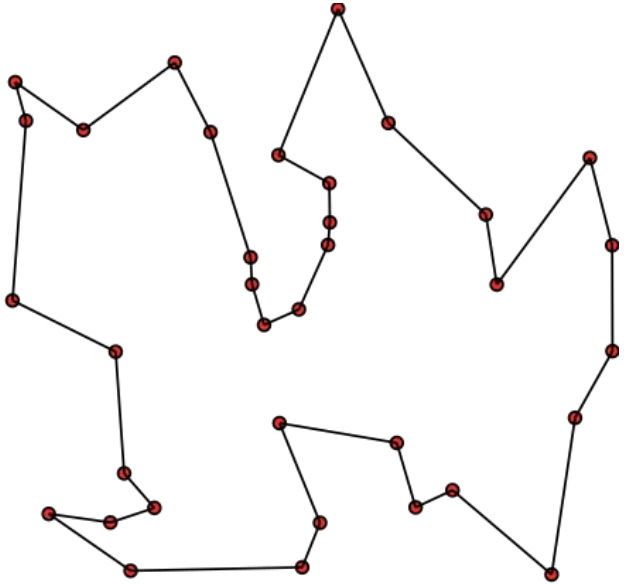
Essential Elements of GA

- Fitness Function - To decide which members are more fit and more likely to become a parent.
- Mating Pool - Set of parents chosen to produce offspring in next generation.
- Crossover Strategy - Heuristic/Algorithm to combine the genetic material of the two parents to produce offsprings.
- Mutation Strategy - Ways of adding little modification/variations to the offspring genes.

Demo and Code Walkthrough

Problem Statement - TSP

- Traveling Salesman Problem
- Given a list of cities on euclidean 2D space, what is the shortest possible route that visits each city and returns to the origin city.
- It is an NP-hard problem in combinatorial optimization, important in theoretical computer science and operations research.

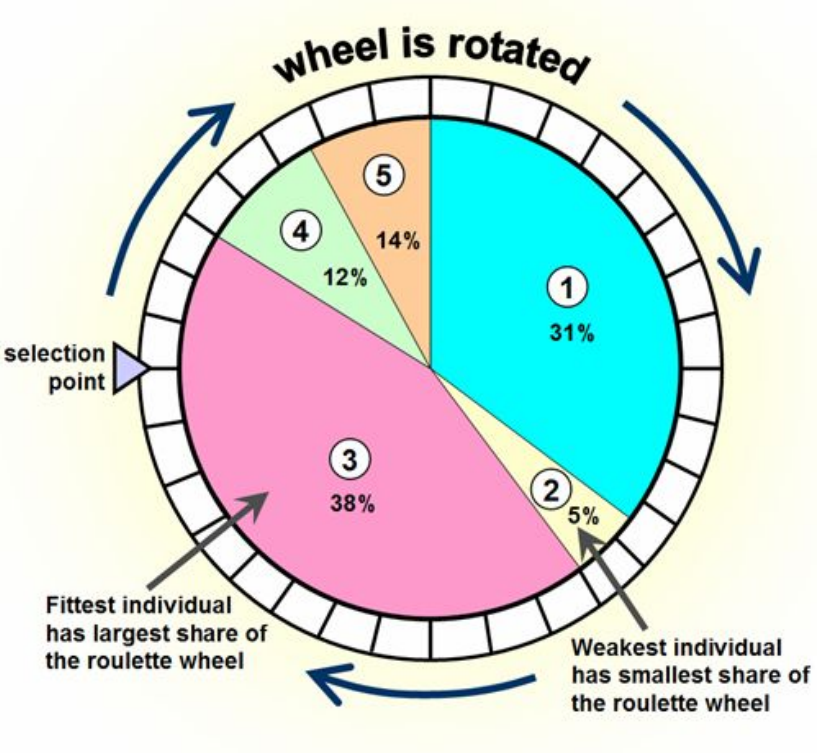


Fitness Function

- temp = sum of distances in the path.
- fitness = take inverse of that value = c / temp (c = some constant)
- normalized fitness = fitness / (total fitness in generation)
- normalized fitness is now between 0 and 1, thus can be used as probability of selecting this DNA.

Roulette Selection

- Suppose we have 5 genes with normalized fitness (probability) as 0.38, 0.12, 0.14, 0.31, 0.05
- We can compute running sum 0.38, 0.50, 0.64, 0.95, 1.0
- Generate random num between 0 and 1
- Based on that value, select parent.

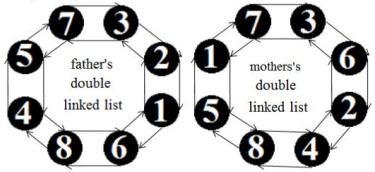


Improved Greedy Crossover (IGX)

Proposed in - H. Ismkhan, K. Zamanifar, “Developing Improved Greedy Crossover to Solve Symmetric Traveling Salesman Problem.” https://arxiv.org/ftp/arxiv/papers/1209/1209.5339.pdf

Use Fig.2’s father and mother:

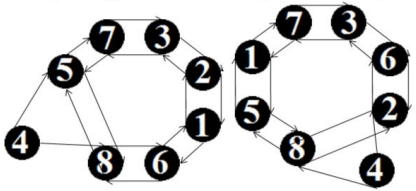
Step 0: Double-linked list construction



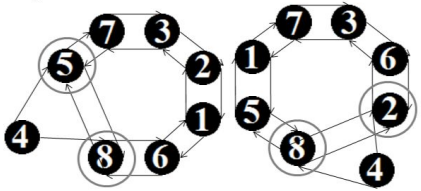
Step 1: First node selection: It is selected randomly. Please suppose it is 4.

child:	4							

lists updating: Eliminates two pointers that point to 4.



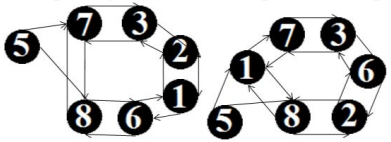
Step 2: Select 2’th node



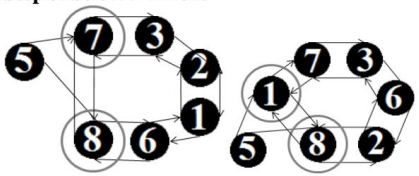
Among 5, 8, 8 and 2, 5 is closer to 4 so it selected as next node.

child:	4	5						

lists updating: Eliminates two pointers that point to 5.



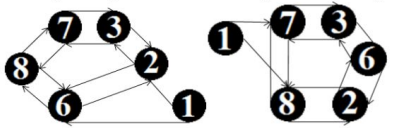
Step 3: Select 3’th node



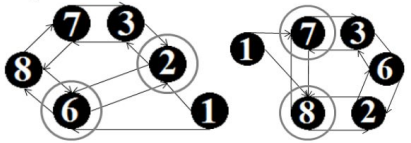
Among 7, 8, 1 and 2, 1 is closer to 5 so it selected as next node.

child:	4	5	1					

lists updating: Eliminates two pointers that point to 1.



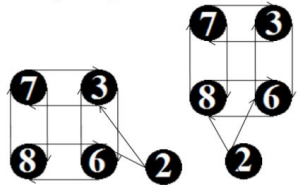
Step 4: Select 4’th node



Among 2, 6, 7 and 8, 2 is closer to 1 so it selected as next node.

child:	4	5	1	2				

lists updating: Eliminates two pointers that point to 2.



Mutation Strategy

- Three mutation strategy used with one-third probability each. Whether to apply mutation on a DNA depends on mutation rate probability.
- 2 Point Swap : (a -> b) (b -> a)
- 3 Point Swap : (a -> b) (b -> c) (c ->a)
- Reverse Segment : Choose a segment [a..b] and reverse that segment.

Going Further

- NeuroEvolution - Combination on neural networks and genetic algorithms. For finding ideal weights and topology of neural network for a given problem/task. Eg - Agent learning to drive a car, agent trying to do automated trading.

- Interactive Selection - No direct way of evaluating fitness function, apart from a user's feedback/response, etc. Eg - Time spent looking at a painting, Rating a melody.

- Ecosystem Simulation - Not all system can be simulated as full population spawned in each generation. So ecosystem evolution works in dynamic and continuous manner.

Applications

- Approximately Solving Optimization Problems
- Automated Trading Systems and Portfolio optimizations
- Self acting agents in unknown environments
- Bioinformatics - Structure prediction and discovery
- Artificial art creation
- BoxCar2D
- Novel Startup Products - SlideBean

Acknowledgement

- Daniel Shiffman and The Coding Train - For all the amazing videos explaining Genetic Algorithms from scratch.

- Processing Foundation - For the processing engine and framework which I used to make this project.

Thank You