Genetic Optimisation Algorithms notes from AI course

Filip Dabkowski, Tomasz Kruczalak, ChatGPT

April 7, 2025

1 Introduction

Genetic algorithm:

Search algorithms based on "natural evolution". They are used to solve complex problems by evolving solutions through selection, crossover, and mutation.

Previous algorithms began the search for the optimum in a single point, and intended to find a better solution for this single point. Gen-alg starts with N starting points, and tries to improve "worse" points by making them more similar to "better" points.

Steps of GA:

- **Initialization:** Starts with a population of N potential solutions (chromosomes)
- **Evaluation:** Computes a fitness score for each chromosome (individual) based on a defined objective
- **Selection:** Choose the best individuals to create the next generation
- **Crossover (Recombination):** Combine selected individuals to produce new offspring
- **Mutation:** Introduce random changes to maintain genetic diversity
- **Replacement:** Form the new population and repeat the process

Terms in genetic algorithms. *Maybe not the most important.*

- **phenotype:** physical expression or real-world behavior of a genotype.
- **genotype:** internal representation of a solution in a GA, usually represented as a binary string, list of numbers or other data structures.
- **chromosome:** complete set of genes representing a candidate solution in the population (e.g.:[101011])
- **allele:** specific value of a gene within a chromosome (e.g.: 1, 0)
- **locus:** gene position in chromosome.
- **fitness:** measure of how good a solution (chromosome) is, goal of GA is to maximize fitness.

Basic genetic operators:

- **Selection** (Reproduction)
- **Crossover** (Recombination)
- **Mutation**

Selection (Reproduction):

- Determines which individuals "pass their genes" to the next generation
- Higher fitness individuals have a better chance of being selected

Crossover (Recombination):

- Combines genes of two parents (individuals) to create offspring
- Mimics genetic recombination in nature
- **Common types:** Single-point, Multi-point, and Uniform crossover

Mutation:

- Randomly alters genes in an individual to maintain diversity
- Helps avoid local optima by introducing new traits
- **Types:** Bit Flip, Swap, Scramble, Gaussian mutation (for continuous values)

Reproduction types:

- **Roulette Wheel Selection:** chose individuals proportionally to the fitness value.
- **Rank-Based:** individuals are ranked based on fitness value, instead of using raw values.
- **Tournament:** multiple selection of the best individual from a randomly selected subpopulation.

Actions related to genetic algorithm implementation:

- Choice of coding method phenotype representation (solutions)
- Selection of genetic operators depending on the problem and the coding method used
- Selection of evolution parameters values
- Selection of fitness evaluation function

Genetic Algorithm Schema:

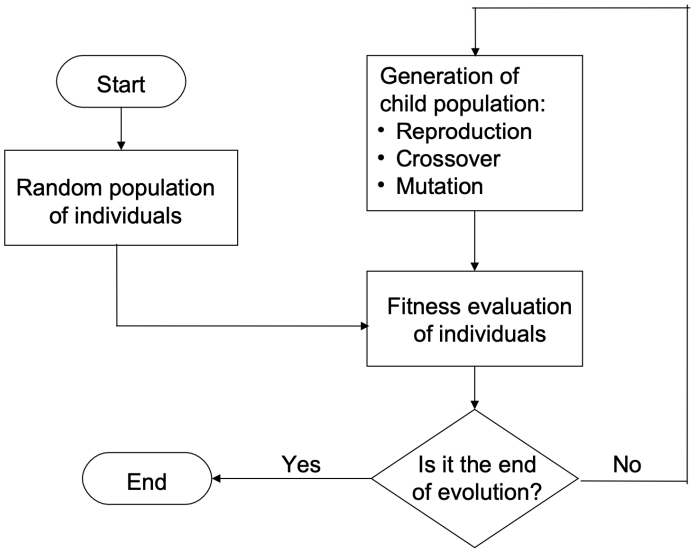

Figure 1: Life cycle of GA

2 Implementation Details

2.1 Roulette Selection

Lets use a simple example we have a chromosome $v = [x_1, x_2]$, and our fittness function:

$$f(v) = -(x_1 + x_2 - 5)^2 + 25 \tag{1}$$

The closer the sum of x_1 and x_2 is to 5 the higher the fittnes. As the fittness should always be non-negative, lets say we only conder values in range [0, 5]

Now the population with $N = 4$ individuals is:

Individual	v	**Fittness**
A	$[1, 2]$	21
B	$[1, 3]$	24
C	$[0, 0]$	0
D	$[4, 4]$	16

Now we determine the probability of being picked for reproduction, with Roulette the probability is a simple scaling calculation. As a population can be large and some individuals with a lot higher fitness may overwhelm the calculation, you may decide to assign some minimal probability. This can help in not being stuck in local minima, by allowing more exploration and ability to make not optimal decisions.

$$p(v) = \frac{f(v)}{\sum_{i=1}^{N} f(v_i)} \tag{2}$$

Here v is the individual, while v_i means iterating over all individuals in the popualtion.

From here as we want to keep the population the same size, we draw N individuals. There is a possibility of the same individual being picked for reproduction, whether it is allowed with mutation, or not, then the selection is run again. Two individuals picked in the selection, will be then handled by crossover and mutation.

2.2 Rank Selection

Another method for selecting parents is based on ranks, rather than scaling the percentages based on raw fitness values. An advantage of this is individuals with much higher do not completely dominate the population, more stable than roulette, however it is slower to converge.

Equation for Linear Rank Selection

$$p(x) = a + k\left(1 - \frac{r(x)}{N}\right) \tag{3}$$

Legent for variables:

- $p(x)$: probability for individual x
- x: single individual
- $r(x)$: rank of individual x ($r = 1$: best individual, $r_{\max}$: worst individual)
- N: population size
- a: base selection probability
- k: scaling factor to ensure probability sums up to 1

Equation for Exponential Rank Selection

$$p(x) = a + k\left(N - r(x)\right)^b \tag{4}$$

Legent for additional variables:

- b - **power exponent** controlling **selective pressure**

Note: There exist more formulas for example with selection pressure for linear rank selection, and so on.

Now continuing the example from above, lets calculate the probabilities using the simple Linear Rank Selection.

Individual	v	**Fittness**	**Rank**	$p(v), k = \frac{2}{3}, a = 0$
A	$[1, 2]$	21	2	0.33
B	$[1, 3]$	24	1	0.5
C	$[0, 0]$	0	4	0
D	$[4, 4]$	16	3	0.17

3 Crossover

Crossover is a stage in GAs when from two selected parents a new instance is created.

3.1 Single-Point Crossover (for vectors, and lists like structures)

Randomly picking a split point in $[1, d]$, where d is number of dimension of some vector x. Then taking the first part from the beginning of a vector to a split point from x_1 and from split point to the end form x_2 For loger gene sequences you can use Multi-Point Crossover where you define multiple points to divide the parents.

```
def cross_single_point(x_1, x_2):
    split_point = randint(0, len(x))
    x_new = x_1[:split_point] + x_2[split_point:]
```

3.2 Average Crossover (when genes are continuous)

To create a new individual we take a simple average of its parents.

```
def cross_average(x_1, x_2):
    x_new = (x_1 + x_2) / 2
```

3.3 Uniform Crossover

To create a new individual, for each gene (*element in vector*) choose random from parent genes.

```
def cross_uniform(x_1, x_2):
    x_new = [
        random.choice(x_1[0], x_2[0]),
        random.choice(x_1[1], x_2[1])
    ]
```

3.4 Heuristic Crossover (when genes are continuous)

This can be thought about as an extension of average crossover, however with preference towards the better (judged by fitness) parent. *Here alpha is random number from 0 to 1, and x_1 is the parent with higher fitness.*

```
def cross_heuristic(x_1, x_2):
    alpha = random()
    x_new = x_1 + alpha * (x_1 - x_2)
```

4 Mutation

Mutation is the last step in creating new population, after crossover when new individuals are created, there is a chance to add randomness to newly created individual. This help with exploration and finding new areas of possible solutions.

4.1 Gaussian Mutation (when genes are continuous)

Adding a small noise to a single gene, this method is great for fine-tuning.

```
def mutate_gaussian(gene):
    gene_new = gene + random.gauss(0, 0.1)
```

4.2 Creep Mutation

Similar to Gauss Mutation however bounded to integers, making it more deterministic and suitable for discrete variables.

```
def mutate_creep(gene):
    gene_new = gene + random.choice(-1, 1)
```

4.3 Uniform Mutation (requires knowledge of value range of the gene)

Here you replace given gene with a completely random value from the range of this gene.

```
def mutate_gaussian(range_min, range_max):
    gene_new = randint(range_min, range_max)
```

5 Additional Info

5.1 Important terms

- **Selective Pressure**: action of increasing probability of being picked in the next population with respect to fitness.
- **Elitism**: the best individuals in the population go to the next generation without any change

5.2 Assumptions

- Fitness value should be non-negative and for better performing individuals higher.
- Probabilities of picking an individual during, reproduction selection should be between $[0, 1]$ and sum up to 1.

5.3 Notes

As usual the best solution for picking the correct reproduction selection, cross-over method and mutation is problem dependant.

As well as picking the best hyperparameter values.