CT421 Assignment 1 Part A

Eoghan O'Brien ID:20451106 (Github Repository- https://github.com/Eoghan243/CT421Project1)

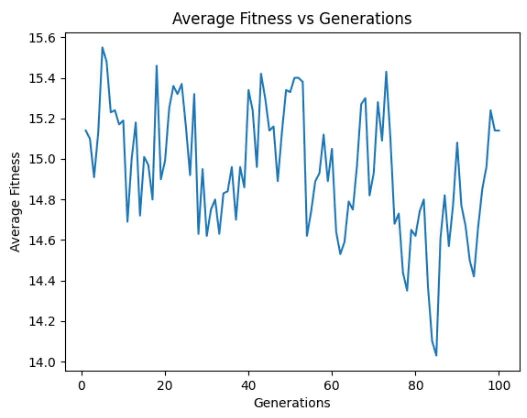
1.1 – One max problem

- Representation: Each individual in the population is represented as a binary string of length 'dna_size'. This binary string acts a potential solution to the issue being addressed by the genetic algorithm.
- Fitness function: An individual's fitness is calculated by adding up all of its binary values. The fitness function specifically calculates the binary string's sum of ones. Maximising this total is the aim as it shows how effective the solution is.
- Crossover: A single-point crossover technique is used to carry out the crossover process. From the population, two parent individuals are chosen at random, and a random crossover point within the binary string is selected. Offspring are then created by swapping the genetic material beyond the crossover point between the parents.
- Mutation: Individuals go through random changes through mutation in order to maintain genetic variety and discover new areas of the solution space. There is a probability (mutation_rate) that every bit in the binary string will be flipped. The bit value is inverted and variation is introduced into the population if the mutation rate is achieved.
- Initialisation: The algorithm starts by producing a population of binary strings with a total of population_size individuals, each string having a length of dna_size.
- Evolution: Over a number of generations, the process of evolution takes place. In every generation:

Other operations:

- Survival: To create the next generation, 10% of the fittest individuals are kept and chosen at random.
- Calculation of Average Fitness: The population's average fitness is calculated and stored for evaluation.
- Completion: When an individual's binary string has a sum of ones equal to dna_size, the algorithm finds a solution and ends, displaying a success message. The method ends automatically after a certain number of generations if a solution can't be found.

Plot:



Results:

- The average fitness varies at the start as the algorithm looks at various areas of solution space.
- Average fitness often shows a rise as the algorithm runs, showing that the population is moving towards better solutions.
- The average fitness stabilises towards the end of the evolution, showing that the algorithm has either declined in the search space or found optimal or near optimal solutions.

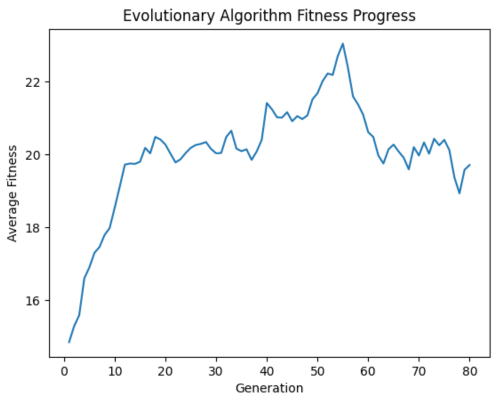
1.2 – Evolving to a target string

- Representation: Every individual of the population is represented by a binary string, in which every bit signifies a gene.
- Fitness function: By comparing an individual’s binary string with a target string and counting the number of bits that match, the fitness function measures an individual’s level of fitness.
- Selection: Based on their fitness scores, parent individuals are chosen at random from the population to carry out the selection process.
- Crossover: To create offspring, a crossover point is randomly selected for each pair of parent individuals. After that, parts of the binary strings of the parent individuals are swapped.
- Mutation: Every member of the population is affected by mutation with a given probability (mutation_rate). It involves flipping individual bits in the binary string.

Other operations:

- The algorithm stops when the target string is identified or after a specified number of generations, tracking the population’s average fitness over time.

Plot:



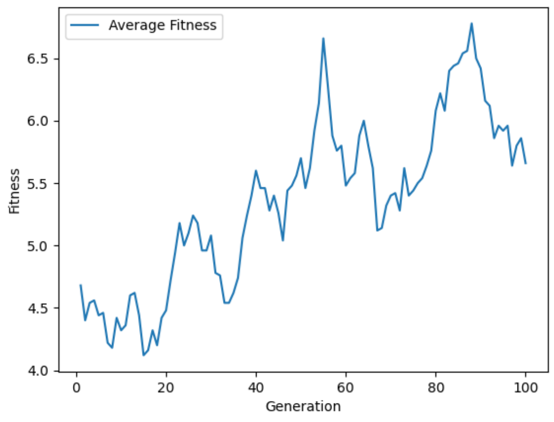
Results: The trend of average fitness over 80 generations is shown in the plot. The average fitness starts a low level in the first generation and rises gradually to a high at the midway point of the evolutionary process. Then it settles, showing the algorithm's progress towards its target fitness without making any use of specific numbers.

1.3 – Deceptive Landscape

- Representation: A binary array, with each element representing a gene, is used to represent each individual of the population.
- Fitness Function: An individual's binary array's elements are added together to indicate their level of fitness. The fitness is set to twice the length of the array if the sum is zero, otherwise it is the sum of the items.
- Selection: Individuals with replacements are chosen at random from the population based on their fitness scores.
- Crossover: Pairs of selected parents undergo one-point crossover with a certain probability (crossover_rate). To make two offspring, the genetic material beyond a randomly selected crossing point is shared by the parents.
- Mutation: Mutation is applied to each gene of the offspring with a certain probability (mutation_rate). Gene values are flipped (from 0 to 1 or the other way around) when a mutation takes place.

Other operations: The fitness of each individual is determined at each generation, and the next generation of individuals is then produced by using selection, crossover, and mutation. Until the specified number of generations is reached, the process is repeated.

Plot:



Results: The trend of average fitness over 100 generations is shown in the plot. The average level of fitness starts out low and rises gradually, peaking around the 80th generation. It then settles at a higher point, showing a general increase in population fitness. There are a few minor differences towards the end, which may indicate that population patterns are still changing.

References

Bin Packing Problem (Minimize number of used Bins). (n.d.). Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/bin-packing-problem-minimize-number-of-used-bins/

Genetic Algorithms. (n.d.). Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/genetic-algorithms/

One Max Problem. (n.d.). Retrieved from Deap: https://deap.readthedocs.io/en/master/examples/ga_onemax.html

What Are Genetic Algorithms? Working, Applications, and Examples. (n.d.). Retrieved from Spiceworks: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-are-genetic-algorithms/