*Q1. Describe shortly step-by-step how a genetic algorithm works according to Anders’ lecture on this. You can chose to either describe a binary or a continuous GA.*

A genetic algorithm (GA) is a heuristic designed to find, generate, tune, that may provide a sufficiently good solution to an optimization problem. GA is inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to search problems by relying on biologically inspired operators such as mutation, crossover and selection.

I would like to provide step-by-step explanation of how a genetic algorithm (GA) works with binary representation. According to Anders’s lecture, there are eight steps of GA binary solution. There are 1) Represent your problem as genes in a chromosome, 2) Random generation of a population of chromosomes, 3) Evaluate fitness of individual chromosomes,

4) Sort after rank, 5) Pairing, recombination, 6) Insert offspring, 7) Mutation and 8) Termination.

1. Represent your problem as genes in a chromosome: This step starts by creating the decision variables that are represented as Chromosomes, for example MyChromosome <- c(1, 0, 0, 0, 1, 0, 1, 1, 0, 0 ). Each gene in the chromosome is coded by m bits (binary representation). A single decision variable can be referred to as a gene.

2) Random generation of a population of chromosomes: Binary GA uses a group of chromosomes called population and the cost function can be represented as f(chromosome). The population goes through a series of updates or generations to evolve. The initial population serves as potential solutions. Each solution is represented as a binary string, often of fixed length, where each bit corresponds to a decision variable or a part of the solution space. The population size is predetermined and could range from tens to thousands of individuals.

3) Evaluate fitness of individual chromosomes: Evaluate the fitness of each solution in the population. Assess the fitness of each solution in the population using a fitness function.

This function takes the chromosome as input and outputs a fitness score.

Calculate the fitness of each chromosome in the population based on this function. Higher fitness values indicate better solutions.

4) Sort solution after rank: Rank the chromosomes based on their fitness scores, ordering them from the highest to the lowest fitness.

This step helps in subsequent selection processes by identifying the best-performing chromosomes.

5) Pairing, recombination: Select pairs of chromosomes from the sorted population to act as parents for producing offspring. Use crossover (recombination) techniques such as single-point, two-point, or uniform crossover to create new offspring from the selected parents. For binary strings, this involves swapping or combining bits between parents to generate new chromosomes. Ander’s lecture provides examples like elitistic pairing and random cut. Elitistic pairing is a selection strategy that ensures the best individuals (elite individuals) from the current generation are preserved and allowed to directly produce offspring. In elitistic pairing, a certain portion (usually a small percentage) of the best-performing individuals from the current generation are selected without undergoing any crossover or mutation. They are directly carried over to the next generation as parents for the new offspring. Random cut, often used interchangeably with single-point crossover, is a type of crossover operation in genetic algorithms. In a binary genetic algorithm, random cut involves selecting a random point along the binary strings (chromosomes) of the parents. The bits at and after this randomly chosen point are swapped between the parents to create new offspring.

6) Insert offspring: Create a new population by combining the offspring generated through crossover with the existing population. Binary GA first selects a set of fittest chromosomes from which the parents will be selected, and all other chromosomes are discarded and be replaced by generated offspring. This step will keep the number of chromosomes at constant as the original input chromosomes.

7) Mutation: Apply mutation to some of the offspring solutions. Mutation involves randomly flipping bits in the binary strings with a low probability. This step introduces diversity in the population and helps prevent premature convergence to a suboptimal solution. This random alteration helps prevent the algorithm from getting stuck in local optima and explores different regions of the solution space. A new population is created by combining the original population with the offspring generated through crossover and mutation. This new population replaces the old population.

8) Termination: GA cycle is started by repeating the process (from step 3) for a certain number of generations or until a termination criterion is met. Termination conditions could include finding a solution that satisfies predefined criteria, reaching a maximum number of iterations, or no significant improvement in successive generations.

The final solution or the best solution found in the last generation is considered the output of the GA. The output of the GA is typically the best solution found across all generations or the final generation's best solution, determined by the fitness function.

Throughout these steps that mentioned as above, simulate the population towards better solutions over successive generations, mimicking the principles of natural selection and evolution to solve optimization or search problems.

A binary GA involves representing solutions as binary strings, evaluating their fitness using a goal function, employing basic crossover techniques (such as simple crossing or traditional single/two-point crossover), and operating within the realm of discrete representations to optimize solutions towards a specific goal or objective.