Genetic algorithm

Genetic algorithms, introduced by John Holland (Kumar, Husian, Upreti, & Gupta, 2010) in [YEAR], belong to the class of evolutionary algorithms and are meta-heuristics based that imitate the biological process of reproduction and natural selection (Carr, 2014). These algorithms are commonly used on search problems and functions optimizer, and it has been applied in a broad range of known problems (Whitley, 1994). One of the greatest barriers of software design, that is to fully understand the structure of complex problems can be solved mimicking natural selection, the specification of every feature of the problems and how to deal with them are not an impediment to search for a solution using this approach (Holland, 1992).

Given its nature, genetic algorithms have been used to find solution for hard problems, like the Travelling Salesman Problem (TSP), VRP, ARP [REFERENCE: ???] and many other problems that due to its complexity don’t have an algorithm to give exact solutions. This is possible because these algorithms tends to explore a far greater range of potential solutions in the search space (Holland, 1992).

Because genetic algorithms are based on biological evolution, the terminology used are the same as the one used in biology, although representing fairly simpler concepts than their biological counterpart. To move forward on understanding genetic algorithms, a few concepts with its nomenclature must be defined, this are the most common components find in most genetic algorithms implementations:

* Gene: a variable (parameter) of the chromosome;
* Chromosome: a set of genes, is a candidate solution for the problem, is the representation of the phenotype on a data structure that can be understood by the algorithm;
* Fitness function: a function to measure the fitness of a solution compared with others, this is the function that must be maximized or minimized depending on the algorithm objective;
* Population: a set of chromosomes that are used to evolve to the next population;
* Crossover: combination of chromosomes to generate offspring for the next generation;
* Mutation: random changes of genes in the chromosome.

Parameters:

* Population size
* Mutation rate
* Crossover rate

[REVISAR ESSA PARTE] In principle, a population of individuals selected from the search space, often in a random manner, serves as candidate solutions to optimize the problem [3]. The individuals in this population are evaluated through ("fitness") adaptation function. A selection mechanism is then used to select individuals to be used as parents to those of the next generation. These individuals will then be crossed and mutated to form the new offspring. The next generation is finally formed by an alternative mechanism between parents and their offspring [4]. This process is repeated until a certain satisfaction condition (Jebari & Madiafi, 2013).

The word fitness come from the evolutionary theory (Carr, 2014).

### Fitness Function

### Before addressing each step presented at most genetic algorithms, the fitness function must be defined as it is the most important piece used in almost every step of the cycle.

### The fitness function is one of the most important part of the genetic algorithm approach as it is the only one method of evaluating the quality of the solution and measure the improvement of through the generations. It must be more sensitive than just measuring good or bad results, it needs to be able to define where the chromosomes stands in the fitness range and compare it with other solutions presented in the population (Carr, 2014).

### Fitness function can be the most limitation factor to a genetic algorithm. As addressed above, the fitness function must to translate how to solution performs, this in most cases is not straightforward. Generating complex and expansive fitness functions that are not computational efficient and require minutes, hours of even seconds to complete can be prohibitive in the development of the genetic algorithm. Need to be considered that the fitness function will be evaluated for each chromosome of the population for every new generation produced.

### Initialization

The implementation of a genetic algorithm begins with a population with random chromosomes. The size of the population depends on the previous selected size for the population. This size is preserved through the entire life of the algorithm. The initialization can be done totally random or applying some previous knowledge of the problem, in this case, some chromosomes can be included with known genes that makes sense to the problem (Kumar et al., 2010), this can lead the algorithm to converge faster to areas where optimal solutions are more likely to be found. From this early step, the evolutionary process begins.

### Selection

A subset of the population is then selected and will be used to breed a new generation, that said, this step is critical since it need to select good individuals trying to keep the diversity of the selected chromosomes. The subset size is also a parameter that need to be set into the algorithm.

The selection step can take place using a variety of techniques. Some methods focus on the fitness of the individual, where chromosomes with best fitness are the one to be selected. Other methods are based on randomness selection or combination of these techniques. No method is guaranteeing to be the best one, and the choice must be problem specific.

There are many selection methods, the most used are roulette wheel and tournament selection (Saini, 2017). But other methods like Stochastic Universal Sampling, Rank Selection and Random Selection can be found in the literature.

The tournament selection and roulette wheel will be addressed bellow. Both methods provide good and diverse parents in most cases, because they give possibility of poorer fit chromosomes to be chosen and still rely on the fitness value to make decision on which individual to choose in their deterministic steps.

#### Tournament Selection

In the tournament selection, the selection process used in this project, K different individuals are randomly selected from the population. Within this set, the chromosome with the best fitness is then selected to reproduce. This process is done once more to select the next parent.

Figura com tournament selection

#### Roulette Wheel Selection

Roulette wheel selection give each chromosome i in the population a probability p(i) of being selected. This probability is proportional with its fitness. To select one individual, a random number is generated, simulating a roulette, and the generated number will define which chromosome will be chose to produce the offspring. Again, this process is done one more time to select the next parent.

### Reproduction

Once the parents are selected, the reproduction step takes place. The parents are combined using crossover to generate the offspring. Then, the generated chromosomes can have its genes randomly mutated by the mutation process at a certain rate, this helps the algorithm to run away from local optimum and have a broader exploratory range in the search space. These steps are also problem specifics, giving that each problem will use the crossover and mutation methods that make sense, also their rates.

#### Crossover

Crossover is a vital process in generation new chromosomes. It exchange genetic material (genes) from two or more chromosomes hoping that can generate individuals with better fitness in the next population (A Study of Crossover Operators for Genetic Algorithms to Solve VRP and its Variants and New Sinusoidal Motion Crossover Operator). Usually crossover is applied with a high probability, it means that in most cases the genetic material of the parents will be recombined to generate the children, less likely, they will just be copied to the next generation as they are. Using a crossover rate of 100% means that every chromosome in the offspring were generated by crossover at least.

As the selection phrase, there are multiple methods to apply crossover, some of the most known and generic are one point crossover and two point crossover, among others. This project uses a different type of crossover that don’t share the behavior of these generic methods and will be further explained in the chapter 4. Because of that, only these two crossover techniques will be explained. These crossover methods will be addressed here to give an overview on how this process take place in the majority of the cases and illustrate the crossover operation.

***One point crossover (A Study of Crossover Operators for Genetic Algorithms to Solve VRP and its Variants and New Sinusoidal Motion Crossover Operator)***

The simplest crossover operator. In this type of crossover, a random point is selected within the limits of the parent, this point is called the cut point. Every point possible to be selected have an equal chance of being selected. To illustrate, in the FIGURE X, two parents chromosomes represented by an array of 10 integers, the cut point would be any number between 0 and 8, in this case, the point 5 was selected. The cut point splits the parents in two half each, the first part are every array element which its index in the array is less or equal the cut point, the second part are the opposite, the elements with index greater than the point. To generate the children, copy the first part of the parent one and insert in the offspring, then get the second part of the other parent and insert in the offspring. Change the order of the parents and do the same operation to generate the second child [REFERENCE: ???].

Figura com one point crossover here

In cases where elements cannot be repeated, the copy of the second parent become a copy of genes one by one in order, avoiding the elements that are already in the child until the child is fulfilled.

***Two point crossover***

This one is a generalization of one point crossover. The difference between them is that this method choose two cut point instead of just one, this will split the parents in 3 parts. As a reference, multi point crossover also exists, everything depends on the number of cut points selected.

Two point crossover mix the parts of each parent in the child, the first part of the parent 1 goes first in the child, then the second part of the parent 2 is then inserted, finally, the last part of the parent 1 is inserted. Repeating this operation interchanging the order of the parents generate the second offspring.

Figura com two point crossover here

#### Mutation

* + - 1. Mutation are small random changes in the genetic material of a chromosome. Mutations itself is not supposed to carry the solution to a better fitness in purpose, but they provide an insurance policy against the development of uniform populations that are less likely to improve themselves in the next iteration (Holland, 1992). Typically, the mutation rate is applied with low probability of 1% or less in many cases (Whitley, 1994), because with very right probability, the algorithm could be reduced to a random search over the space.
      2. Common mutation methods are bit flip, swap, inversion, among many other that can be found in the literature. In the example bellow in FIGURE X, the swap mutation is shown. In this mutation technique, two random genes of the chromosome are selected and swapped between them, these cases are useful when repeating a gene is not allowed, as the case of the TSP. In the bit flip mutation, for each gene in the solution, there is a change of change the data in the gene to other random data with possibility to be inserted in the chromosome.
      3. Figura com two point crossover here

### Termination

Genetic algorithm creates population after population iterating until some condition, or conditions, have been reached. Conditions must be pre-defined by the developer of the algorithm, usually they rely on time, number of iterations, minimum criteria found. Kumar et al. [TA CERTO ESSA CITACAO?] (2010) described some techniques used as stop conditions, they are:

* Found a solution that satisfy a minimum criterion;
* Number of iterations reached;
* Computational time reached (budged);
* The algorithm has reached a highest fitness solution and no longer is producing better solution for an amount of iterations;
* Manual inspection;
* Combination of the previous methods, or any other method created.