#### FUTURE WORKS:

* CONSTRAIN FOR THE AMOUNT OF FUEL A TRUCK HAVE AND HOW MUCH IT CAN TRAVEL UNTIL IT IS EMPTY

#### Crossover

The crossover operation, as one of the most important operations in a GA, must work with the representation chosen on the chromosomes. In this project acts differently of the common crossover from most of the literature, the genetic material is not exchanged between parents, but a piece of route is given by one of the parents, while the remaining genetic material is totally got from the other parent. This generates only one child in the crossover operations, because of that, more crossovers must be done to complete the next population. This crossover methods follows the one proposed by Pereira, Tavares, Machado & Costa (2002) but applying small changes on how to generate new routes on truck’s capacity overflow.

The crossover randomly select a sub-route of one of the parents and insert it on the other parent. The insertion point is defined to be just after the way which has the minimum distance between itself and the first way in the sub-route. This operation must guarantee that every solution generated from it is a valid solution.

Summarizing the crossover process, two parents previously selected in the selection step of the algorithm and a new chromosome is created to store the data generated by the crossover. A sub-route is randomly selected from one of the trucks of the second parent, this sub-route must have at least one way. The selected sub-route weight is calculated and stored in a variable. One time only, the algorithms verify which edge has the lower distance between itself and the first edge of the sub-route, this verification is done accessing the distance matrix. From now one, the method will go through every truck from the parent 1, and consequently every edge each truck servers. This iteration will append new edges in the child path, and new trucks on the child used trucks array. Its important to notice that if the current edge of the iteration is presented inside the sub-route selected in the first parent, the edge will be not considered, because it would generate duplicated edges when the sub-route is inserted in the new solution. When the iteration finds the edge that is closest to the first edge of the sub-route, the edge will be normally appended in the solution and the sub-route will be inserted just after it. After the insertion of the sub-route in the child chromosome the process continue until every edge is served by the child.

IMAGE OF THE CROSSOVER

Capacities overflow can happen multiple times in this algorithm, to manage that, there is a list that stores every edge that could not be served for this reason. Once the previous process is complete, an iteration over the edges not served will take place assigning each edge to a truck as it is done in the initialization process of the GA. This guarantee that all edges are served by a truck.

#### Mutations

As the crossover, the mutation step must be specific for this problem being able to deal with the chromosome representation. The mutation is applied to every member of a new population with a low probability. This project implements two kinds of mutation, called swap mutation and inverse mutation, each of them have their own probability rate. Both mutation operators as the crossover can only generate valid solutions.

In the swap mutation, a truck is randomly select from the chromosome, then a random edge served by this truck in the solution is selected. The same occurs once more in the chromosome. Even if the edges are inside an array without the truck’s array influence, is crucial to first select the truck so the capacity and the load of the truck are previously known for the mutation, this information is then utilized to only generate a valid solution. Having that two edges from the chromosome were randomly selected, a swap operation is done, the first edge takes the place of the second edge and the second edge is placed on the first’s spot. The swap is done with validation to stop overflows from occurring. In the case of the validation fails, the swap process is repeated until the validation is successful, with a maximum of ten attempts. After that if the validation fails more than ten time, the mutation swap is not done.

The inverse mutation, acts just in a sub-route of a truck in the solution. Like the sub-route selection in the crossover step, a sub-route is selected in this mutation. Then the order of the sub-route in inverted and inserted again in the path. This type of mutation differently of the swap mutation can not generate invalid solutions, as the number and weight of the ways are not changed and only the distance can change, if this GA had other constrains like the total length a truck can travel because of its fuel, this could generate an invalid solution and would require a validation, new constrains can be developed in future works and will be discussed in the future works section.