Evolutionary Algorithms: Peer review report

For the group of Benois Ardenois, Tomas Van Den Noortgate, and Arno Van Eetvelde

Orestis Lomis, Thibeau Wouters, and Jingyu Zhang

November 15, 2022

1 One strong aspect

We think the recombination operator is rather good. It allows for enough exploration while also taking into account the good aspects present in both parents. It also does not have a bias towards one of the parents, which is something we saw in a different group. We would argue that there is maybe too little exploration when both parents go to two different cities when starting from a specific city. Both may be making bad choices, so it might be worth it to allow random cities to be chosen as the next destination as well in those cases.

2 Two weak aspects

The two aspects which we considered to be a bit weaker and may have to receive additional attention in this implementation are the stopping criterion and the initialization.

2.1 Vague stopping criterion

The evolutionary algorithm stops when no improvement was detected in the final 25 generations. First of all, it seems to us that the stopping criterion can disrupt the convergence of the algorithm if only the fitness of the best individual is considered. The best individual may be stuck in a local minimum, while the rest of the population is still exploring and might eventually catch up, as measured by the mean fitness value. Considering the example from Figure 4 of the report, we observed that the best fitness value was constant for around 15 iterations, namely between generation 20 and generation 35. Such plateaus in the fitness value are dangerous for this specific stopping criterion, since a lengthy plateau (or equivalently, a low value for the final amount of generations before exiting the algorithm) can make the algorithm stop while improvement may still be possible afterwards. In case the mean fitness value is used for this stopping criterion, we believe that this requires a clear definition of what is meant by *no improvement*. Because alternatively you likely will not want to keep running the algorithm when improvements are only marginal. For example, if in 500 generations the mean fitness values slightly improves every 25 generation by only 10^{-5}, you could obviously stop earlier on without losing too much improvement. Concretely, one can terminate the algorithm when the fitness value should not improve by Δ for N generations, where Δ is now an additional hyperparameter to be tuned and N can be 25 as is mentioned for example in the report. While the idea behind the stopping condition allows for flexibility, we think the group has to be careful in its implementation.

2.2 Computationally expensive initialization

We believe that the initialization is computationally expensive, which may be a disadvantage when running the algorithm for larger tours. This is also stated in the report. Generating whole cycles which can possibly be discarded wastes a lot of computational resources. Especially given the fact that the larger tours for the TSP have many more infinite values to be dealt with, we believe that initialization will take quite a long time during this algorithm. This implies that most of the randomly generated cycles will give an invalid path, and will hence be discarded, such that the amount of valid cycles generated randomly will be quite small. We think it would be a good idea to consider another way of generating the initial population by adding cities one by one, and rejecting an individual city when the corresponding value is infinite. Moreover, one can filter rows in the distance matrix to remove the infinite values to bypass the rejection step and make sure that generated cycles are guaranteed to be valid. After filtering, one can then select a random city from the remaining cities, such that we still keep enough randomness in the initialization phase. Moreover, we would make a small comment that the probability of generating duplicates is likely quite low. Besides, a few duplicate individuals in the beginning of the algorithm are probably not very problematic, as they will be mutated and possibly removed later on. The report also mentions that this can be resolved by initializing individuals in a greedy manner, but we think one needs to be careful when doing this, as this might lead to a lot of similarity between the individuals early on, potentially causing them to end up in a local minimum.

3 One suggestion

We would like to suggest, as already mentioned above, to redesign the initialization of the algorithm, as we believe that the computational resources which are being wasted here, which will certainly increase for the larger tours, can be used for more useful calculations in the remainder of the evolutionary algorithm. We would propose to randomly select from the cities which do not give infinite values, as detailed above, to still ensure that the initialization phase does not generate invalid paths. We want to stress that the filtering can be done on infs rather than simply mapping these values to `None`, which could lead to `NoneType` exceptions which is never pleasant.

A different approach is to allow invalid paths, but simply punish these in a good way, because it will also be difficult and computationally expensive to validate all the mutations and recombinations that are being created.