# Walk

Statt 2-opt, 3-opt. 3-opt gives the best results considering runtime (<https://de.wikipedia.org/wiki/K-Opt-Heuristik>) .

Algorithm use the 2-opt heuristic as a local search. However, the authors use a modified version, in which the offset delta, denoting at which distance from the first opened edge the second edge the tour is opened. The delta is calculated as follows:

First a random city k is taken, the outgoing edge from k is the first point where the tour is opened. Then, another random bird j is taken and delta’ is calculated as the difference between the position of the same city k for bird j and the position of the, for the current bird previous city of k, k-1 is taken. If between city k and previous city k-1, in the tour of i, are at are at least two other cities, then the number of cities between city k and k-1 in j is taken as the offset l. Based on the city k, we then jump l cities forward and open the tour at the city k+l. We then connect cities k and k+l.

To the process of finding the right offset through delta’ is done 100 times: Either we find a delta’ that is at least 2 or we select a random offset between 2 and the number of cities -1, to avoid that we open the tour at the same edge, which would equal no change.

Because the probability that we don’t find another bird for which abs(delta’) is 1 or |T|-1, which would mean that “l” would be the exact city before city “j”, on the first try, we decide to remove the loop and only do the local similarity computation one time. If delta turn out to be 0, we, as described in the algorithm, also draw a random integer between 2 and n-1.

This does not improve the algorithm (figure …), so we refrain from using this modification.

We also, instead of the 2-opt approach, test a 3-opt variant for the walk.

A diagram of different types of types of type

Description automatically generated

( <https://proceedings.mlr.press/v157/sui21a/sui21a.pdf> )

Here the tour is not broken up at two places, but instead at three.

The results can be seen in Table <…>.

Because we are not implementing the 3-opt algorithms by itself, but rather in a swarm algorithm, in which each bird (or agent respectively) can perform this action, the computational complexity will rise by a margin as shown in Table <…>. This is why we test the case where only big birds are able to perform a 3-opt walk, while smaller birds are only capable of the usual 2-opt walk as specified in the paper. This should not only reduce the computational complexity but also pairs well with the assumption that big birds are superior, as only they can join other birds.

Furthermore, we also test the inverse approach: Only small birds can perform 3-opt.

Based on the results we decide to use 3-opt for big birds from now on, as this provides the best average improvement in performance with a reasonable increase in computation time.

(The question is, why this is done. The authors argue is to “account for the local similarity of the bird’s position with another random bird“. However, then don’t explain what practical use this has.)