# Improving Swarm

If we study the effects of how different number of birds change the solution of different tours, we learn that less birds seem to yield better solutions. At first this may seem counterintuitive, but there is an intuitive explanation to this:

The basic algorithm defines one iteration as the calculation for the cost of one tour. Now let a phase of the algorithm denote each bird performing one move. So, in each phase each bird performs one move.

If we now use more birds the number of phases the algorithm will run through will decrease, as in each phase the length of more tours will be calculated, which will use up more iterations. Therefore, the more birds we have, the more iterations we will use up per phase, which means the algorithm will run through less phases until it stops.

Because in one phase each bird can perform one move, less phases mean each bird can perform less moves. Less moves in turn leads to a less pronounced search of possible solutions, which will make the algorithm worse. This is why the less birds we have, the better the results will be.

Consequently, we aim to improve the bird behavior, so that each bird needs fewer overall steps to achieve a good solution.

Currently, if one big bird made the decision to join another bird, he picks one randomly.

This means joining any bird without considering how good the position of that bird might be. This contradicts the original idea of the authors that big birds tend to join others that have found a good food source (current solution seems promising).

Therefore, we propose that a big will only be able to join the top-b percent of birds that have the lowest current cost. If one chooses the right ratio, we assume that it will automatically nudge the swarm in the direction of the global minimum.

We implement this by storing the indices i of the birds in an ordered integer array “ord” and introducing a new hyperparameter call b, denoting which of the top-b percent to join. When we select the bird to join to, we draw a random uniform number j between 1 and b\*n\_birds and get the index of the bird to join from the ordered array (ord[j]).

The main disadvantage this approach has is that at we need to continuously update “ord” so that we only join the top-b percent at the moment of the move. We decide to update the list after each bird has performed one move, so after each phase.

We previously argued that will yields less phases, so the more birds we take, the less often we will have to update “ord”, which will make the algorithm faster. At the same time this will make “ord” less up to date, as for many birds the true order of the best performing birds will change more often per phase. This provides us with a trade-off

Authors didn’t use this approach, because they argue it is not usual with pigeons, as they rather behave individualistic: A pigeon does not call other pigeons if it found food. However, another pigeon might join another pigeon if the other found food!

We further believe one should not be constrained by the behavior being used as inspiration for the algorithm. After all, the goal is to find the algorithm that produces the best results in a reasonable amount of time, not the exactly model the behavior of the respective swarm.

They did try it!?

However, one must be careful when selecting the ratio, as a number too low will increase the probability that all birds get stuck in a local minimum. A number too high will in turn lower this probability and birds can then also joint birds which are not performing as well.