# Improving Swarm

Because less birds the better (more iters per bird), we should improve the birds moves so that each birds needs less overall steps to move for achieving a good solution, this includes improving the swarm (join) behavior as then maybe more birds will yield better results even though they have less iters available overall. At the same time more birds would mean more big birds with the improved join behavior, so the algorithm would benefit even more?

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

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 or is lower than others).

Therefore, we propose that a big will only joint the top n % birds that have the lowest current cost. If one chooses the right ratio, we assume that it will automatically nudge the swarm in the right direction.

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.

Another thing worth to consider is that this change will for now only affect line 28 of algorithm 1. We mention this, because the same operation of finding another bird (tour) randomly is also executed in line 3 of the walk algorithm (algorithm 3). If that were to change, the candidates with which we would compare the current tour (or bird respectively) would only the top n % ones.

However, this operation, the selection of a random bird other than the current one, is also executed in line 3 of the walk algorithm (algorithm 3) to estimate the similarity of the current tour with that of another bird.

We implement this storing the indices i of the birds in an ordered integer array “ord”. When we select the bird to join, we now draw a random uniform number j between 1 and n\*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 truly only joint the top n% at the moment of the move. We decide that for now we update the list after each bird has performed one move, though this might change later due to the enormous computation effort.

However, as we saw in the section “Iterations” more birds will yield less overall round, as the iterations are used up more quickly. 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 a large number of birds the true order of the best performing birds will change more often. This could provide us with a nice trade-off situation

Normal distributed draw would also be interesting?!