Name: Hanna Foerster User-ID: lzhc88

Algorithm A: Ant-Colony-Optimisation

Algorithm B: Particle-Swarm-Optimisation

Description of enhancement of Algorithm A:

The first enhancement I made for the Ant Colony Optimisation was to implement the rank-based Ant-System. After that I experimented a lot with different parameters. Generally, it turned out that for a smaller number of iterations a large decay rate yielded better results. Hence, I changed the pheromone decay rate to be 0.85 initially and after 0.8 times the iterations that the algorithm is supposed to run for, it is changed to 0.5. Moreover, for the other parameters alpha=1, beta=3 and w=6 yielded the best results. After playing with the parameters, I noticed that the algorithm might yield better results, if the randomness in choosing the next vertex was sometimes eliminated and the vertex with the highest probability was chosen. Furthermore, eliminating more of this randomness in the first few iterations turned out to be especially important. Hence, I included an if statement that would choose the vertex with the highest probability, for every second iteration t if t were smaller than 6, and after the sixth iteration depending on the size of the city set, for every 2, 4 or 10 iterations. There are a few other cases I included after experimenting to get the right amount of randomness into the ACO. Another result concerning the iteration parameter t and number of ants parameter N was that the ACO would generally yield the best results if N was the number of cities in the city sets, but only if the ACO was able to run at least 10 iterations. So, I made a regulator to regulate the size of N according to the number of cities. In this way the ACO will run well in under a minute for any city set.

Description of enhancement of Algorithm B:

The first enhancement I made for the Particle Swarm Optimisation was to change the randomly initialised tour of the first particle into the tour obtained by the nearest-neighbour algorithm. In that way the worst my Particle Swarm Optimisation can perform changed to be that of the nearest neighbour algorithm. Furthermore, for every fourth particle from the N particles initiated, I changed the code to go near the vicinity of the best tour obtained globally, if the difference of tour length between the best global tour and the current tour of that particle was bigger than the difference of tour length between the best global tour and the worst tour from the initial particles divided by two. In that way every fourth particle would flock around the vicinity of the first particle and explore this area if their own tours were not that good. Additionally, for every two in three tours I changed the algorithm to go in the direction of the global best tour if their best tour was not very good, but in this case only 0.85% of the swaps to global best should be implemented, making the tours obtained not quite as strictly neighbours of the global best tour. In all other cases the algorithm should explore the vicinity of their own best tour if the current tour was much worse than its own best tour. To make this exploring the vicinity more exact than it was with epsilon, alpha and beta, I implemented a new function take(int x) that would take exactly x swaps of a velocity. In this way, if a current tour is much worse than the best tour of the particle, the distance to this best tour minus a few swaps is taken as velocity. To aim at the vicinity an extra few swaps are added. Moreover, if the current tour was near the best tour of the particle it should only change the tour by a few swaps to explore the vicinity for a better tour. Only if all these cases do not apply should the basic case run.