Name: Joshua Fitch User-ID: mccb22

Algorithm A: Greedy best first search

Algorithm B: Ant colony optimisation

Description of enhancement of Algorithm A:

I added a local search parameter to my greedy best first search algorithm that travelled to the nodes nearest neighbour at each transition. This parameter could take 2 values specifying either 2(better for the largest city sets) or 3 opt local search. To speed up runtime of 2/3 opt search that has a time complexity of O(n^2)/O(n^3) I used adjustable candidate lists. Only cities in each nodes nearest neighbour list were considered for swapping. 8 city lists balanced solution quality and time(although I used larger lists on smaller city sets where runtime was less of an issue). I also only actually constructed the new tour when it had been calculated swapping these edges would reduce tour length. Using these speed up techniques I reduced runtime, especially on the larger city sets, on the 175/180 city sets these techniques reduced runtime by an average of ~30% for 3 opt, and made it just over 10x faster on the 535 set for 2opt(3opt runtime very long on this set).

Despite being relatively simple in nature this algorithm was very powerful, competing(although rarely beating) my AS ranked algorithm despite much faster run times. This shows the power of applying local search to global search. Having an already quite good tour as a starting point means we know we are locally searching in a part of the state space with good solutions.

On larger sets(58+) this algorithm produced on average ~20% better tours than the base greedy algorithm.

Description of enhancement of Algorithm B:

I upgraded my simple ACO to a ranked ant system as described in lectures. Along with this I also added "smoothing" pheromone level on edges. Whenever a value got excessively low (upperLimit_deposit / 2*num_cities) or high(1/p * 1/best_tour_score) I would not update this value until it moved back in range. I also applied a local 2/3 opt(controlled by a parameter) search using the speed up techniques described above to an adjustable number of ants. Finally I used candidate lists(adjustable) to speed up the search, each ant would pick a node to move to based on the nearest neighbours of that node. Only if this list had been exhausted would the ant consider other nodes.

The enhancements applied meant this algorithm produced the best tours on the city sets. Smoothing stopped stagnation and made the algorithm less likely to be stuck in local minima and candidate lists sped up the algorithm. By far the most effective enhancement however was local search, I believe this is because it drastically increased the amount of the search space that was explored each iteration. Also, being guided by ants that are already likely to have pretty good tours meant a large portion of a part of the state space with good solutions was explored.

The other useful feature about this algorithm was the number of adjustable parameters, being able to adjust the iterations, length of candidate lists, number ants locally searched and 2 or 3 opt meant this algorithm was able to perform better on all city sets. Using 3 opt and ants = num_cities and all ants locally searched explored the maximum amount of the state space in smaller city sets. On larger sets using 2 opt and 25 ants all with local search applied worked best. In general I found parameter settings that explored a large portion of the search space, but only allowed the best few ants to influence it(w = 6) worked best.

On larger sets(58+) this algorithm produced on average ~10% better tours than the base ACO algorithm.