Name: Alex Goodall User-ID: clvp22

Algorithm A: A* Search

Algorithm B: Genetic Algorithm

Description of enhancement of Algorithm A:

The enhanced A* Search algorithm models the problem as a list of selected edges. Each state represents a list of selected edges (list of tuples) as opposed to a list of visited cities in the basic version. We start from the empty state (no selected edges), and the algorithm searches the state space by adding every possible edge to the current node state, excluding those edges that violate the constraints of the problem (degrees of cities must not exceed 2, no cycles in non-goal nodes). The heuristic function is the greedy selection of edges, that is, we greedily select the shortest possible edge until we have constructed a tour. This algorithm performs better than the basic one because the greedy selection of edges heuristic is better than the nearest neighbour / greedy completion heuristic. Because there are many more edges than cities the computational complexity of this A* Search algorithm is rather high, as a result it has also been improved with early termination, so that if the algorithm has been running longer than 90 seconds the current node state is greedily completed and returned as the solution.

Description of enhancement of Algorithm B:

The enhanced genetic algorithm incorporates multiple enhancements to the basic algorithm:

1. We start from N greedy / nearest neighbour tours (each starts at a different city), because these give better end results as opposed to 100 random tours.
2. The mating process is less naive; we assign one parent as the “donor” and the other as the “receiver”, we then pick a random number **x** between 10 – N // 2, and this is our crossover point. We then assign the first **x** cities in the “donor’s” encoding to the child’s encoding (unvisited cities are cities not in the child’s encoding). We then look at the last city in the child’s encoding **c** and append an unvisited neighbour of **c** in the “receiver’s” encoding to “child’s” encoding. If this is not possible, we then look for a neighbor of **c** in the “donor’s” encoding. And if this is not possible, we append the next unvisited city to the child’s encoding, and we repeat until the child has an encoding of length N.
3. The mutation process consists of 4-Opt moves, that is, we remove 4 edges from the tour giving us 4 separate paths, we then reconnect the paths in some different way. This is achieved by splicing the child’s encoding at 4 random positions and reconnecting them in some way that resembles a 4-Opt move. We extend this process to use a randomized acceptance criterion using temperature. After we apply a random 4-Opt move we check to see if the child’s encoding is better than both the parent’s encoding, if it is, we accept it immediately, if it isn’t we accept it with probability e^delta/T. If we fail to accept, we increase the temperature and try another random 4-Opt move until we accept. This is a similar process to simulated annealing I suppose.
4. The final improvement is we apply the 3-Opt local optimization algorithm to the encoding of the child before we finish, so it is locally optimal w.r.t 3-Opt moves. We only do this for N <= 90 for computational reasons.

Note: the population is only evolved once – this is sufficient to get good tours, and evolving it further seems to make little difference and just increases computation time.