Name: Theo Farrell User-name: nchw73

Algorithm A: Genetic algorithm (GA)

Algorithm B: Ant colony optimisation (ACO)

Description of enhancement of Algorithm A:

Initial population – Instead of generating it randomly, some nearest neighbours (NN) tours are calculated and used in the initial population P. These are generally shorter than randomised ones, so the initial P is somewhat fit. This helps breed even fitter individuals faster than the basic version since short edges already exist in P and don't have to be created by crossover/mutation. Since NN tour generation is much more costly than randomised ones ($O(n^2)$ vs $O(n)$), a user-defined proportion, $0 < m < 1$, of the population is filled with NN tours with unique start cities. Remaining individuals are completely randomised since some randomness reduces risk of getting stuck in local minima. **Sequentially constructed crossover (SCX)** (Ahmed, 2010) – SCX generates a child tour Z from two parent tours X, Y as follows: 1. Z's start city c_Z is the same as X's or Y's. 2. Search through X and Y for the first unvisited cities after p – call these c_X and c_Y. 3. If c_X is nearer to c_Z then add it to the end of Z, otherwise add c_Y. 4. Repeat 2-4 until Z is complete. If a parent tour has a good sequence of cities, SCX makes Z likely to inherit it. However, step 2 means that a new edge will be created if c_X or c_Y does not immediately follow c_Z in X or Y. This supports population diversity. With a randomised initial population, SCX generated a high-quality population in much fewer generations than the basic crossover. This resulted in much better tours for large city sets since more iterations could be used to improve further. **Dynamic mutation probability** – Mutation is essential to escape local fitness maxima that hinder convergence on an optimal tour. If it occurs too often, divergence is possible since good tours can become worse. In the basic GA, this is fixed at 0.2 which works well initially. However, if the best tour has not been updated for many generations it is likely either a local extremum or in fact optimal. In the enhanced GA, the probability of mutation increases after either: many iterations pass without an update to the best tour, or if 90% of the population has the same fitness. Increments and their associated thresholds are user-defined. This appeared successful, with better tours often found immediately after a period of stagnation triggered more mutations.

Description of enhancement of Algorithm B:

2-opt local search - Once ant k has built its tour, 2-opt local search can improve it. This searches the tour for a pair of disjoint edges, (a_0, a_1) and (b_0, b_1), that can be replaced by another pair of disjoint edges, (a_0, b_0) and (a_1, b_1), such that the new pair of edges have a lower cost than the original pair. This is equivalent to reversing the tour between (and including) a_1 and b_0. This significantly reduced each ant's tour length by greedily looking for local optima within the ant's tour, which are easily overlooked in basic ACO. Since this is very expensive, cities considered for a_0 are restricted to edges not already in the best tour found so far, since those have already undergone 2-opt. The cities considered for b_0 are limited to the first k nearest neighbours of a_0, where k is user-defined (k=4 strikes a good balance). **Dynamic candidate lists** - each city has a candidate list, containing its m nearest neighbours where m is user-defined (m=20). During trail building, only the candidate list of the current city, c, undergoes the stochastic process. If all these have been visited, then the next city is deterministically chosen as the unvisited city with the greatest product of pheromone on the connecting edge and heuristic desirability (i.e., the numerator used to calculate probability in the stochastic process). This drastically speeds up the trail-building but increases the risk of converging to a local minimum since the next edge in the optimal tour may not be in the candidate list, limiting output quality. To counteract this, each city c's candidate list is dynamically updated by adding s and b to the front, where s is the city such that the edge c --> s has the most pheromone of c's neighbours, and b is the city visited after c in the best tour found so far. This uses heuristic information that is readily available since it is calculated over the course of the ACO regardless, and it reduces the aforementioned limitations on the tour quality when using candidate lists. **MAX-MIN AS (MMAS)** (Stützle & Hoos, 2000) - The basic ACO uses AS_{rank} which gives generally better results than standard AS because only top ants and the globally best tour found so far deposit pheromone. Using the global best tour to deposit encourages convergence towards it which is important since better tours are assumed to share edges with the current best. Using the current iteration's best tour promotes exploration of the search space, avoiding getting trapped in local minima. MMAS takes AS_{rank} to the extreme by allowing only one of either the global best or current iteration's best tour to deposit. Which one it is depends on a user-defined schedule which makes it more likely the current iteration's best is used early on, but in following iterations the global best is used to encourage convergence. A maximum pheromone level is maintained to avoid premature convergence, and a non-zero minimum level means there is always a chance to visit other cities, promoting exploration. These are defined using the cited paper's equations which are explained in the code.

DESCRIPTION OF ALGORITHM ONLY IF THE ALGORITHM IS NOT COVERED IN LECTURES

Description of *non-standard* Algorithm A:

Describe any non-standard algorithms you have implemented that ***have not been covered in lectures*** *(otherwise these boxes should be blank) You need to convince me that your implementation is indeed that of the named algorithm and you need to* ***provide a full reference to the source for your algorithm****. You should* ***include a pseudocode description****. You can vary the sizes of these boxes but do not change the font (Calabri), font size (11), the paragraph properties (single space) or the header and footer and everything should fit onto one side of A4. (You can delete these instructions.)*

Remember: You need my express permission to implement a non-standard algorithm!

Describe the ***enhancements*** *you have made to your algorithms in the two boxes. You can vary the sizes of these boxes but do not change the font (Calabri), font size (11), the paragraph properties (single space) or the header and footer, and everything should fit onto one side of A4. Do not embed images. You should type into this Word document and save it as a pdf. You may include a commentary on the relative success of your enhancements if you wish and* ***your submitted codes should be well commented****. However,* ***full explanations of enhancements should be provided here*** *with code comments used to show where and how the enhancements are made.*

Save the final document as a pdf. *(You can delete all these instructions.)*

Description of *non-standard* Algorithm B:

Type here, as above.