Name: Hailing Cai User-name: llpw83
Algorithm A: GA, genetic algorithm
Algorithm B: AC, ant colony optimization
Description of enhancement of Algorithm A:

Analysis of the results of the AlgAbasic run showed that increasing the number of iterations not only fails to improve the path length, but also increases the execution time. Therefore, I implemented the following optimisations focusing on improving initial path quality and enhancing path exploration:

***Initial method changed to multi-start nearest-neighbour initialisation:** Abandoning complete random permutations and instead using a multi-start nearest-neighbour algorithm to generate initial paths. This produces a higher quality and more diverse initial population, accelerating convergence and improving the quality of the final solution. This improvement saves about 25% of the runtime and reduces the path length of large-scale city maps by 50%.*

In the second half of the execution, the algorithm switches to elite selection (retaining the best elite individuals in each generation without crossover and mutation on them, thus preventing the loss of the optimal solution) and top-k selection strategies to ensure that high-quality solutions are retained and to speed up convergence. The first half retains tournament selection, which maintains population diversity by randomly selecting the best individuals.

*Partial mapping crossover replaces **sequential crossover** to better retain valid genetic information from the parents, while ensuring uniqueness of cities and reducing the generation of invalid solutions.*

The original exchange mutation is upgraded to a reverse mutation. By selecting and flipping path segments, this method produces more significant changes that help to get rid of local optima and improve the quality of the solution. This mutation approach is particularly effective for optimising inefficiently sorted paths.

Description of enhancement of Algorithm B:

The test showed that although AlgBbasic runs fast, its path optimisation performance is not satisfactory. For this reason, we implemented two improvements:

rank_based_update: After sorting by path length, only the top-ranked paths participate in the pheromone update, with higher rankings contributing more.

elite_strategy: In the later stages, the globally optimal paths are additionally reinforced with pheromones computed based on path lengths and distributed uniformly around the edges of the paths.

These enhancements strengthen the guidance to the optimal path and reduce inefficient searches, allowing the algorithm to find shorter paths faster and use the remaining time to explore promising regions in depth.

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* <u>*do not change the font (Calabri), font size (11), the paragraph properties (single space) or the header and footer*</u> *and everything should* <u>*fit onto one side of A4*</u>*. (You can delete these instructions.)*

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

Description of *non-standard* Algorithm B:

Type here, as above.