[redacted] [redacted]

Algorithm A: US (Uniformed Search - Depth First Search)

Algorithm B: AS (A-Star Search)

Description of enhancement of Algorithm A:

Instead of starting on an empty root node, as outlined in the pseudocode, we can define our root node as the first city in the list (0). This significantly reduces the number of tours we visit, since it removes many duplicates that the depth first search algorithm would originally run through. For example, say we have tour ABCD, this is the same distance as tours BCDA, CDAB and DABC, since they are all the same tour but one that begins on a different city.

To remove these nodes from our graph, the initial variables were changed from:
state 0, parent_id None, action [None], path_cost 0 and depth 0
to …
state 1, parent_id 0, action [1], path_cost 0 and depth 1

The other repeated tours are any tours which happen to be the same forwards as they are backwards, such as ABCD and ADCB, however you would have to check whether one of these had been encountered on every iteration, which would reduce efficiency.

Although depth first search is always implemented to find the optimal solution, if left running until the program terminates, performing this enhanced algorithm reduces the time it takes to find a shorter tour when compared to performing Algorithm A within the same timeframe.

Description of enhancement of Algorithm B:

I applied the same simple enhancement as above, where I defined the root node as the first city, rather than leaving it empty. As stated previously, this reduces the number of tours that the algorithm visits.

To remove these tours from our graph, the initial variables were changed from:
state 0, parent_id None, action [None], path_cost 0 and depth 0
to …
state 1, parent_id 0, action [1], path_cost 0 and depth 1

Ideally, the distance heuristic I defined would be the Euclidean distance, however I could not make this change as this cannot be calculated from the data.

This enhancement generally improved the tour for A search, as it reduces the number of nodes for the algorithm to search through. Hence, in most cases, it can find a shorter tour compared to Algorithm B within the same timeframe.*