Name: Zac Robinson User-ID: qrgk42

Algorithm A: Simulated Annealing

Algorithm B: A* Search

Description of enhancement of Algorithm A:

The definition of a successor node has been improved between the algorithm versions. The basic SA algorithm creates successors by randomly swapping the locations of two cities in the tour, while the enhanced SA algorithm creates successors by selecting a random slice of the list and reversing it. This improves tours by allowing low-cost sections within the slice to be preserved.

The enhanced version also runs the SA algorithm multiple times, as the random choice of successor means that the same input can produce better or worse tours for consecutive runs on the same cityset. The optimal tour found is then chosen as the tour to use. For citysets larger than 200, time constraints become more significant, so the number of SA iterations is reduced in order to ensure the program decides on a tour and terminates in a reasonable amount of time.

The enhanced SA algorithm uses a better temperature function. For the basic version, temperature decreases linearly with time; in the enhanced, it decreases exponentially. This allows the algorithm to start prioritising finding a maximum earlier without completely removing its ability to exit a non-optimal local maximum.

Description of enhancement of Algorithm B:

The enhanced A* algorithm uses the greedy completion distance as the heuristic value for each node instead of nearest neighbour distance, which results in routes being found much faster; with this improvement alone, the A* search can find a tour for smaller (≤100) city-sets before the 110-second timer finishes and the rush algorithm kicks in. Once a tour has been found, if the timer has not finished, the algorithm runs again with both heuristics summed together. Although this takes longer, this combination finds better tours than either individual heuristic.

Instead of a list, the enhanced A* algorithm stores the fringe as a min-heap, sorted by (in order of descending priority): least f(z), greatest depth, least list as defined by Python (example: [1,2,4] < [1,3,4]). This means that choosing the next node on the fringe is significantly faster, as it just requires popping the first element of the heap.

Instead of starting with a single root node starting at a fixed city, the algorithm has been enhanced by starting with a fringe that contains one node for each possible starting city. The choice of starting city has a significant impact on the final tour length; adding the possibility to start at any city has allowed for much better tours.

The state for each node has been modified to now include the set of unvisited cities. This allows them to be iterated through faster without having to be recalculated each time.

The enhanced algorithm also includes several other smaller speed enhancements that were made possible due to the refactored data structures used. Along with the previously described speed enhancements, these make the algorithm operate faster, meaning the rush algorithm will have a better optimal node to work on if it is still needed.