Student Name : Yixiang Fan

Subject Code : CSCI803

Student Number : 5083898

Email ID : yf874@uowmail.edu.au

Data structure:
Each node is a complicated structure, including the start vertice and end vertice of this step, the weight from start vertice and end vertice, the cost which is the sum of weight from vertice '0' to the end vertice, the lb that is the lower bound from the end vertice, the tour that is the collection of end vertice of this route and the next which contains the child nodes of this node.

Algorithm:
Greedy: In each loop, program select the nearest vertice based on the vertice selected in the last loop until all vertices are selected.

Branch and bound(Depth): In function BABDepth, program initiates the algorithm and calls the function BABDIteration which is a iterative function. In each iteration, program generates all possible nodes whose lowerbound is smaller than upperbound. Then sort these new nodes and choose the node which has the smallest lowerbound as the parent node in the next iteration. When it comes to the last iteration, program finds the node generated in this iteration which has the smallest cost and replaces upperbound with that cost and record the tour of that node. In the last iteration, the cost of each node equals to the lowerbound of each node. Then the program returns to the upper iteration and checks the other nodes whose lowerbound are smaller than upperbound. When the iteration finishes, the program can get the best tour and the minimum cost.

Branch and bound(Breadth): In function BABBreadth, program initiates the algorithm. It will create a head node. In the function BABBLoop, program generates all nodes based on vertice '0' which represents Wollongong and store them as a minimum node heap. In the following each loop, program generates all nodes based on the top node of the heap. Push all new nodes into the heap and remove the top node at last. The way of generating new node and update upperbound is the same as branch and bound(depth). If the lowerbound of top node was bigger than the common upperbound, this node will be discarded directly and update the heap. In next loop, check the new top node. Loop stops while the heap is empty.

Structure of my code: My code read the city names and the two matrix into the program. Then calculate the lowerbound of the two matrix. After that, run the greedy algorithm to calculate the tour and the cost of the flight and the road, respectively. Then call BABDepth to handle the assignment problem of flight and road. So does BABBreadth.

Analysis and discuss:

Nodes generated:

Depth	flight	71706
	road	313610
Breadth	flight	131761
	road	349776

As to greedy algorithm, most of the time it does not get the best solution. But it can get a not bad solution, which can be used as a premise of other algorithm. Besides, speed is another advantage.

As to branch and bound algorithm, although the lecturer said generally the breadth is better than depth, but in this case, depth is much better. In my views, the key of branch and bound algorithm is to find a smaller upperbound as soon as possible. Then depth is better than breadth as this point.

All in all, greedy algorithm can get a not bad solution with a highly speed. Branch and bound can get the best solution but time-consuming. And in this experiment, depth performances better than breadth.