HEURISTIC APPROXIMATION!!!!!!!!

In computer science, artificial intelligence, and mathematical optimization, a heuristic is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution.

- Provide a feasible solution without guarantee on its optimality, or even quality, or algorithm running time.
- Well-designed heuristics for particular problem classes have been empirically shown to find good solutions fast.
- Examples of heuristic algorithms:

  - Lagrangian

    -Greedy Construct

    -Local Search

Problem: Implement a travelling Salesman program (TSP)

- A list of cities to be visited
- Pairwise distances between the cities
- Must return to the home/starting city

SOLUTION:

- Cities could be thought of as → Vertices
- Distances could be described as → Edges (weighted)

➔ Then we have reformulated the problem into something, on which we could apply Graph Theory.

ALGORITHM:

T (i , s) = min ((i , j) + T (j , S – { j })) ; S!= Ø ; j € S ;

S is set that contains non visited vertices

= (i, 1) ; S=Ø, This is base condition for this recursive equation.

Here,

T (i, S) means We are travelling from a vertex “i” and have to visit set of non-visited vertices “S” and have to go back to vertex 1 (let we started from vertex 1).

(i, j) means cost of path from node i to node j

If we observe the first recursive equation from a node we are finding cost to all other nodes (i,j) and from that node to remaining using recursion (T (j , {S-j}))

But it is not guarantee that every vertex is connected to other vertex then we take that cost as infinity. After that we are taking minimum among all so the path which is not connected get infinity in calculation and won’t be consider.

If S is empty that means we visited all nodes, we take distance from that last visited node to node 1 (first node). Because after visiting all he has to go back to initial node.