Farthest Insertion Algorithm for the Traveling Salesman Problem

This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dennis-bricker@uiowa.edu

author

The "Farthest Insertion" heuristic algorithm constructs a tour, starting with an arbitrary node.
Each step begins with a subtour, and selects the node which is *farthest* from the set of nodes on the subtour to be added to the subtour.
After selecting the node k to be added, an edge (i,j) is selected and the edges (i,k) and (k,j) then replace the edge (i,j).
The edge (i,j) is selected so as to minimize the increase in the length of the subtour, i.e.,

$$d_{ik} + d_{kj} - d_{ij}$$

The "Farthest Insertion" heuristic constructs a tour for the TSP as follows:

step 0: Select an initial node $\hat{i}$.
Let N' denote the set of nodes $N - \{\hat{i}\}$
Let $T = \{(\hat{i}, \hat{i})\}$

step 1: Let $\hat{j} = \operatorname{argmax}_{j \in N'} \left[\min_{i \in T} \{d_{ij}\} \right]$

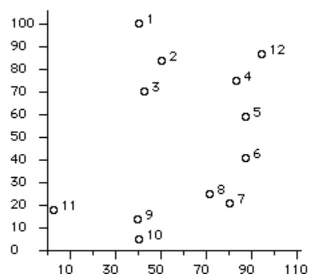
step 2: Let $(i', i'') = \operatorname{argmin}_{(i_1, i_2) \in T} \{d_{i_1 j} + d_{j i_2} - d_{i_1 i_2}\}$

step 3: Replace arc (i_1, i_2) in the tour T with the pair of arcs $(i_1, \hat{j})$ and $(\hat{j}, i_2)$.
Let $N' = N' - \{\hat{j}\}$ and $\hat{i} = \hat{j}$.

step 4: If $N' = \emptyset$, STOP. Else return to step 1.

Example

Random Symmetric TSP
(seed= 133398)



Distances

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56
2	19	0	16	34	45	57	70	63	71	80	82	44
3	30	16	0	41	46	54	62	54	56	65	66	55
4	50	34	41	0	16	34	54	51	75	82	99	16
5	62	45	46	16	0	18	39	38	66	72	94	29
6	75	57	54	34	18	0	21	23	55	59	88	47
7	89	70	62	54	39	21	0	10	42	43	78	67
8	81	63	54	51	38	23	10	0	34	37	69	66
9	86	71	56	75	66	55	42	34	0	9	37	91
10	95	80	65	82	72	59	43	37	9	0	40	98
11	90	82	66	99	94	88	78	69	37	40	0	115
12	56	44	55	16	29	47	67	66	91	98	115	0

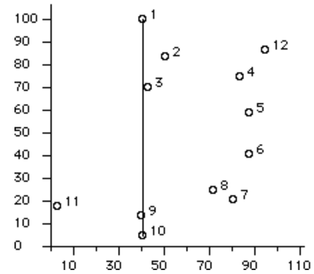
Let's arbitrarily begin the tour with node #1, i.e., T = {1}

We select the FARTHEST node from the tour, i.e., node #1. This is node #10

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56

Farthest Insertion Heuristic

(Beginning with node #1)



Insert node 10

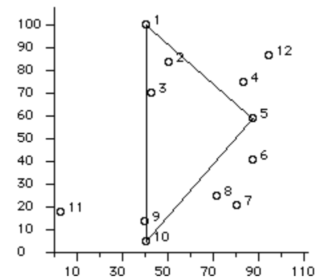
T = {1, 10}

Distances

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56
10	95	80	65	82	72	59	43	37	9	0	40	98

We compute the distance from each node to the nearest node already on the tour. The node selected to be inserted is that node which is FARTHEST from the tour, namely node #5

Farthest Insertion Heuristic



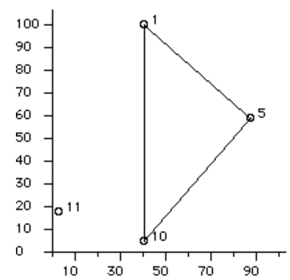
Insert node 5

T = {1,10,5}

Distances

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56
5	62	45	46	16	0	18	39	38	66	72	94	29
10	95	80	65	82	72	59	43	37	9	0	40	98

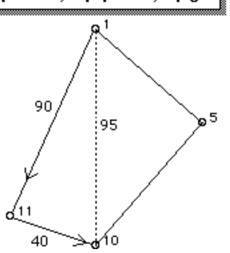
Again, we compute the distance from each node to the nearest node on the tour, and then select the FARTHEST such distance, in this case to node #11, which will be inserted next.



We next need to decide between which 2 nodes on the tour to insert node #11.
There are 3 possibilities:
1 ⟶ 11 ⟶ 10
10 ⟶ 11 ⟶ 5
5 ⟶ 11 ⟶ 1

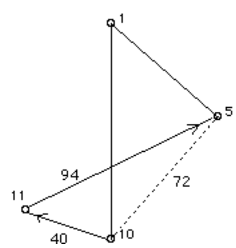
We choose to insert node #11 in such a way that the increase in the tour length is minimized:

1 ⟶ 11 ⟶ 10



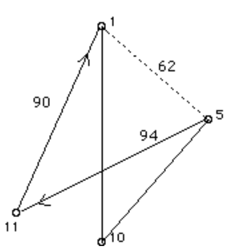
Increase in tour length is
90 + 40 − 95 = 35

10 ⟶ 11 ⟶ 5



Increase in tour length is
40 + 94 − 72 = 62

5 ⟶ 11 ⟶ 1

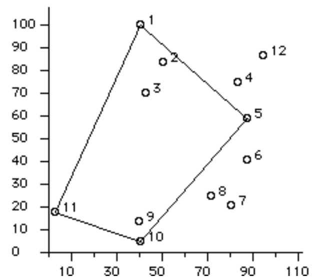


Increase in tour length is
94 + 90 − 62 = 122

Minimum { 35, 62, 122} = 35

and so node #11 is inserted between node #1 and node #10.

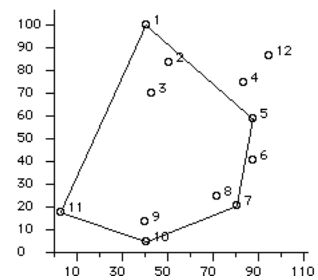
Farthest Insertion Heuristic



Insert node 11

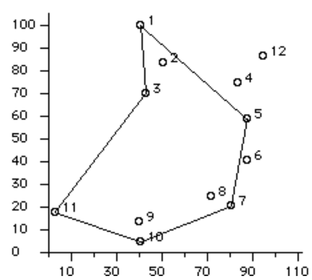
T = {1, 11, 10, 5}

Farthest Insertion Heuristic



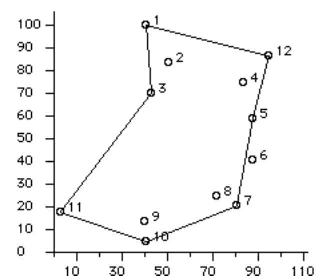
Insert node 7

Farthest Insertion Heuristic



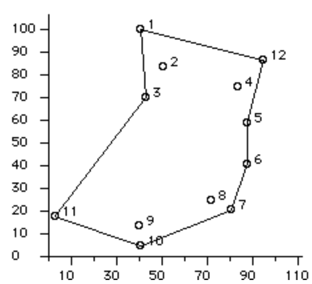
Insert node 3

Farthest Insertion Heuristic



Insert node 12

Farthest Insertion Heuristic



Insert node 6

... etc.

Farthest Insertion Heuristic

Farthest Insertion Tour: 1 12 4 5 6 7 8 9 10 11 3 2 1,
with length 321