Nearest Neighbor 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

The "Nearest Neighbor" heuristic is a "greedy" algorithm which constructs a tour by adding, at each step, the node which is nearest (among those not yet added to the tour) to the node which was added at the previous step.
When all nodes have been added, the tour is completed by returning to the beginning node.

The "Nearest Neighbor" heuristic algorithm constructs a tour as follows:

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

step 1: Let $\hat{j} = \underset{j \in N'}{\operatorname{argmin}} \{ d_{\hat{i}j} \}$

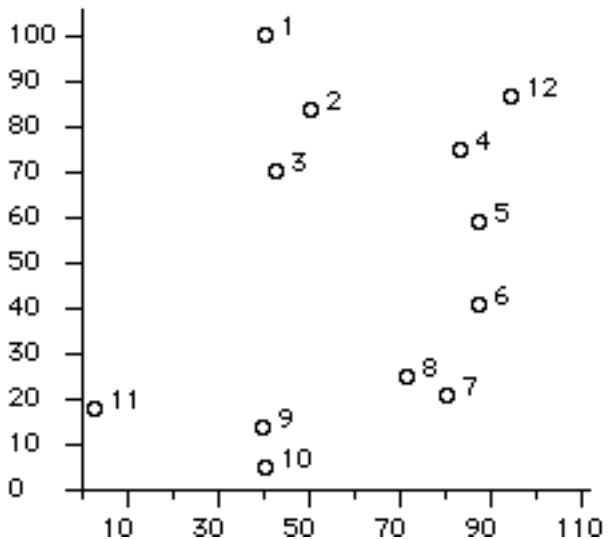
step 2: Add arc $(\hat{i}, \hat{j})$ to the tour T.
Let $N' = N' - \{ \hat{j} \}$ and $\hat{i} = \hat{j}$.

step 3: If $N' = \emptyset$, STOP. Else return to step 1.

Note that the resulting tour T depends upon the initial arbitrarily-selected node with which to begin the tour.

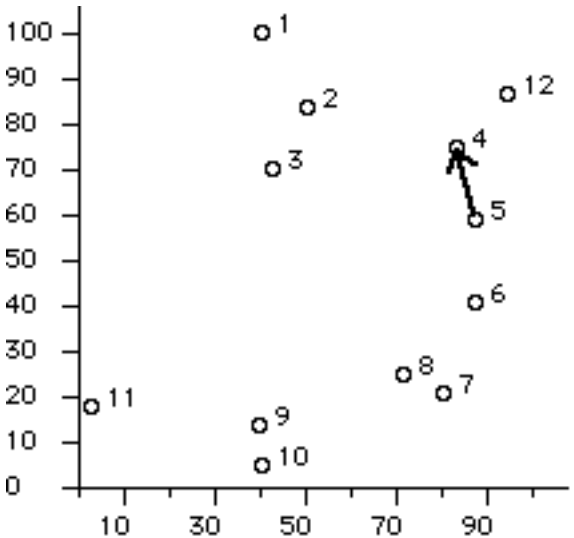
Example

Random Symmetric TSP
(seed= 133398)



Distances

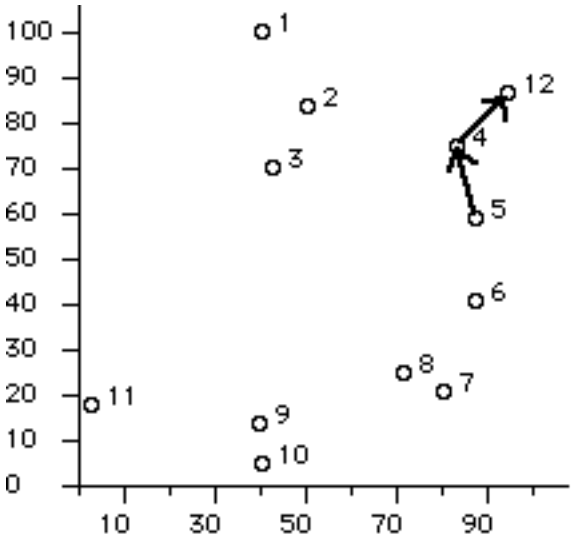
from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56
2	19	0	16	34	45	57	70	63	71	80	82	44
3	30	16	0	41	46	54	62	54	56	65	66	55
4	50	34	41	0	16	34	54	51	75	82	99	16
5	62	45	46	16	0	18	39	38	66	72	94	29
6	75	57	54	34	18	0	21	23	55	59	88	47
7	89	70	62	54	39	21	0	10	42	43	78	67
8	81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

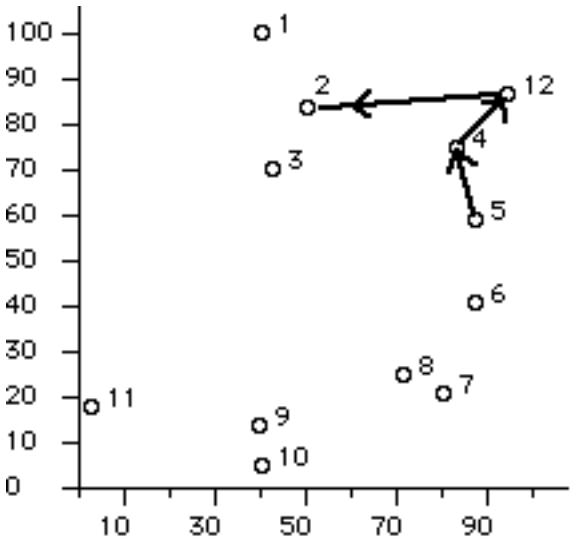
It's nearest neighbor is node #4

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
5	62	45	46	16	0	18	39	38	66	72	94	29



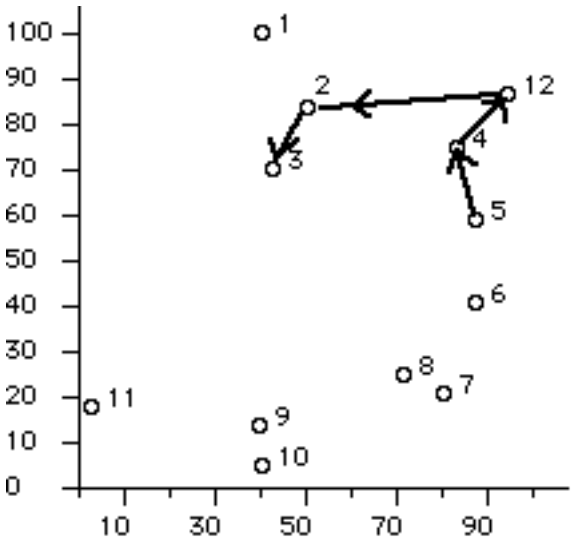
The nearest unvisted neighbor of node #4 is node #12

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
4	50	34	41	0	16	34	54	51	75	82	99	16



The nearest unvisited neighbor of node #12 is node #2

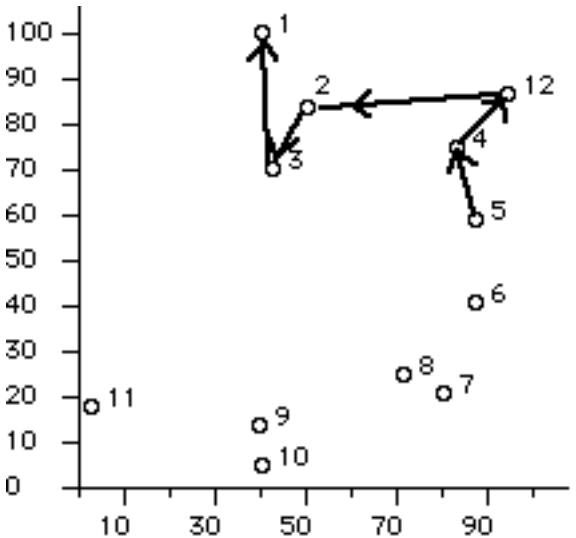
from \ to	1	2	3	4	5	6	7	8	9	10	11	12
12	56	44	55	16	29	47	67	66	91	98	115	0



The nearest unvisited neighbor of node #2 is node #3

(Clearly the better node to visit next would be #1, but this algorithm lacks foresight!)

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
2	19	0	16	34	45	57	70	63	71	80	82	44

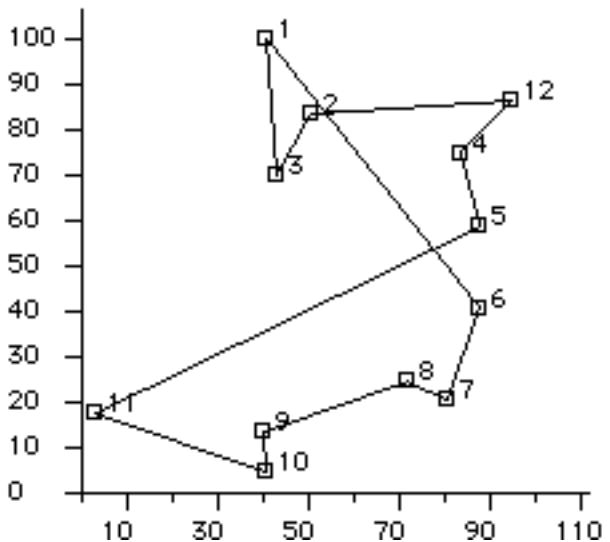


The nearest unvisited neighbor of node #3 is node #1

... etc.

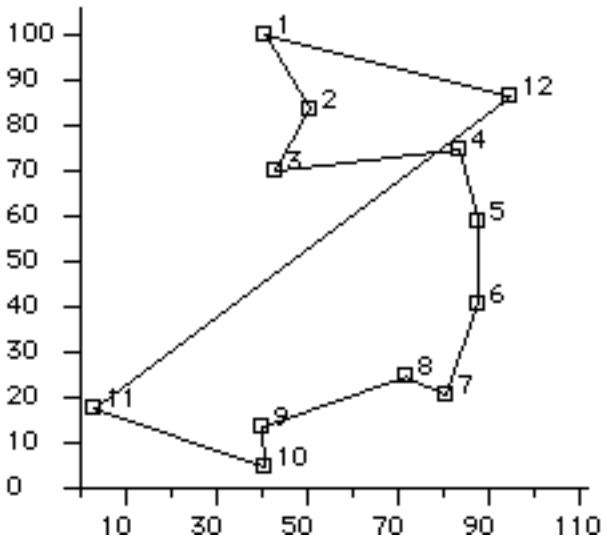
from \ to	1	2	3	4	5	6	7	8	9	10	11	12
3	30	56	0	41	46	54	62	54	56	65	66	55

Nearest Neighbor Tour: 5 4 12 2 3 1 6 7 8 9 10 11 5,
with length 405



(Starting with node #5)

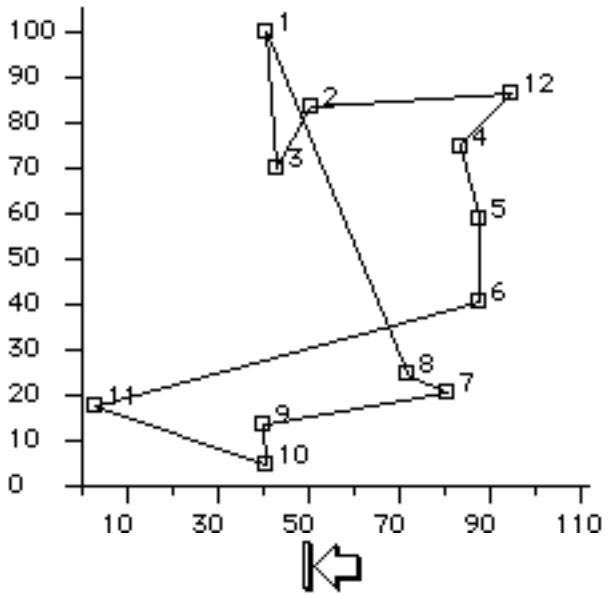
Nearest Neighbor Tour: 1 2 3 4 5 6 7 8 9 10 11 12 1,
with length 395



(Starting at node #1)

Different starting nodes result in different tours!

Nearest Neighbor Tour: 6 5 4 12 2 3 1 8 7 9 10 11 6,
with length 410



(Starting with node #6)