

The Nearest Neighbor Algorithm

Algorithm Description

The nearest neighbour algorithm is a greedy algorithm that solves the traveling salesman problem by continually choosing the nearest unvisited city on his tour. True to its nature the algorithm works quickly and effectively. Given a randomly distributed number of cities this algorithm will yield a tour that is on average 25% longer than the shortest possible path.[1]

However, there exist many specially arranged city distributions which make the NN algorithm give the worst route.[18] This is true for both asymmetric and symmetric TSPs.[19] Rosenkrantz et al.[20] showed that the NN algorithm has the approximation factor $\Theta(\log |V|)$ for instances satisfying the triangle inequality.

A variation of NN algorithm, called Nearest Fragment (NF) operator, which connects a group (fragment) of nearest unvisited cities, can find shorter route with successive iterations.[21] The NF operator can also be applied on an initial solution obtained by NN algorithm for further improvement in an elitist model, where only better solutions are accepted.

[1] Johnson, D. S.; McGeoch, L. A. (1997). "The Traveling Salesman Problem: A Case Study in Local Optimization" (PDF). In Aarts, E. H. L.; Lenstra, J. K. *Local Search in Combinatorial Optimisation*. London: John Wiley and Sons Ltd. pp. 215–310.

Justifications

Pseudocode

```

1 def distance_squared(c1, c2):
2     return (c1['x'] - c2['x'])**2 + (c1['y'] - c2['y'])**2
3
4 # cities is an array of city objects which have an id, x-coordinate, and y-coordinate properties.
5 def get_nearest_neighbor(cities, city):
6     # Dictionary for selecting nearest neighbor
7     neighbors = {}
8
9     # Add all distances_squared to neighboring cities to dictionary
10    for neighbor in cities:
11        neighbors[distance_squared(city, neighbor)] = neighbor
12
13    # Return neighbor with least distance
14    nearest_neighbor = neighbors[min(neighbors)]
15    distance = int(round(sqrt(min(neighbors))))
16
17    return nearest_neighbor, distance
18
19 def TSP_nearest_neighbor(cities):
20     tour = []
21     min_distance = infinity
22
23     for city in cities:
24         total_distance = 0
25
26         # Start on arbitrary vertex.
27         visited = [city]
28         unvisited = []
29
30         # Add all cities to unvisited list except the starting city.
31         for city in cities:
32             if city is not city:
33                 unvisited.append(city)
34
35         # find an unvisited nearest neighbor, marked it visited, and add it's distance.
36         while len(unvisited) > 0:
37             nearest_neighbor, neighbor_distance = get_nearest_neighbor(unvisited, visited[-1])
38             visited.append(nearest_neighbor)
39             unvisited.remove(nearest_neighbor)
40             total_distance += neighbor_distance
41
42         # add the distance between the first and last city to complete the tour.
43         total_distance += round(sqrt(distance_squared(visited[0], visited[-1])))
44
45         if total_distance < min_distance:
46             tour = visited
47             min_distance = total_distance
48
49     return tour, min_distance

```
