The Traveling Salesman Problem Group Project

CS352 – Winter 2018

Group 45

Marc Tibbs, Brad Besserman, Michael Chan

Due Date: March 16, 2018

Summary

1 Abstract

2 Introduction

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

```
def distance_squared(c1, c2):
  return (c1['x'] - c2['x'])**2 + (c1['y'] - c2['y'])**2

# cities is an array of city objects which have an id, x-coordinate, and y-coordinate properties.
def get_nearest_neighbor(cities, city):
  # Dictionary for selecting nearest neighbor
  neighbors = {}

  # Add all distances_squared to neighboring cities to dictionary
  for neighbor in cities:
    neighbors[distance_squared(city, neighbor)] = neighbor

  # Return neighbor with least distance
  nearest_neighbor = neighbors[min(neighbors)]
  distance = int(round(sqrt(min(neighbors))))

  return nearest_neighbor, distance

def TSP_nearest_neighbor(cities):
  tour = []
  min_distance = infinity

  for city in cities:
    total_distance = 0

    # Start on arbitrary vertex.
    visited = [city]
    unvisited = []

    # Add all cities to unvisited list except the starting city.
    for city in cities:
      if city is not city:
        unvisited.append(city)

    # find an unvisited nearest neighbor, marked it visited, and add it's distance.
    while len(unvisited) > 0:
      nearest_neighbor, neighbor_distance = get_nearest_neighbor(unvisited, visited[-1])
      visited.append(nearest_neighbor)
      unvisited.remove(nearest_neighbor)
      total_distance += neighbor_distance

    # add the distance between the first and last city to complete the tour.
    total_distance += round(sqrt(distance_squared(visited[0], visited[-1])))

    if total_distance < min_distance:
      tour = visited
      min_distance = total_distance

  return tour, min_distance
```

Algorithm Name

Algorithm Description

Justifications

Pseudocode