Implementation of Greedy Algorithm in Travel Salesman Problem

Research · September 2016

DOI: 10.13140/RG.2.2.23921.48485

CITATIONS
2

READS
22,504

1 author:

Samson Ejim
Abubakar Tafawa Balewa University
7 PUBLICATIONS **4** CITATIONS

SEE PROFILE

Implementation of Greedy Algorithm in Travel Salesman Problem

Samson Ejim, Reg. No: 12/28213/D/1

Abubakar Tafawa Balewa University, Bauchi
Department of Mathematical Sciences.

1.0 INTRODUCTION

1.0.1 GREEDY ALGORITHM

A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

For example, a greedy strategy for the traveling salesman problem (which is of a high computational complexity) is of the following heuristic: "At each stage visit an unvisited city nearest to the current city". This heuristic need not find a best solution, but terminates in a reasonable number of steps; finding an optimal solution typically requires unreasonably many steps.

If a Greedy Algorithm can solve a problem, then it generally becomes the best method to solve that problem as the Greedy algorithms are in general more efficient than other techniques like Dynamic Programming. But Greedy algorithms cannot always be applied. For example, Fractional Knapsack problem can be solved using Greedy, but 0-1 Knapsackcannot be solved using Greedy.

1.0.2 PSEUDOCODE OF GREEDY ALGORITHM

- At each step
    - Item will be added in a solution set by using selection function.
    - If the set would no longer be feasible
    - reject items under consideration (and is never consider again).
        - ELSE IF set is still feasible THEN
            - Add the current item

1.0.3 PSEUDOCODE FOR TRAVEL SALESMAN PROBLEM

```
GreedyTSP (V,E, home): RETURN T
X[1]◄— home;
FOR I◄— 1 TO N-1 DO
Select (u,v) with min weight such that (u,v)ϵE,  uϵS, vϵS
     X[…… S◄—S∪{v};…
UNTIL V≠{}
Return something;
```

1.0 STANDARD ALGORITHMS THAT EMPLOY THE CONCEPT OF GREEDY ALGORITHM

The following are some standard algorithms that are of Greedy algorithm in nature.

1) **Prim's Minimum Spanning Tree**: In Prim's algorithm also, we create a MST by picking edges one by one. We maintain two sets: set of the vertices already included in MST and the set of the vertices not yet included. The Greedy Choice is to pick the smallest weight edge that connects the two sets.

2) **Dijkstra's Shortest Path:** The Dijkstra's algorithm is very similar to Prim's algorithm. The shortest path tree is built up, edge by edge. We maintain two sets: set of the vertices already included in the tree and the set of the vertices not yet included. The Greedy Choice is to pick the edge that connects the two sets and is on the smallest weight path from source to the set that contains not yet included vertices.

3) **Huffman Coding:** Huffman Coding is a loss-less compression technique. It assigns variable length bit codes to different characters. The Greedy Choice is to assign least bit length code to the most frequent character.

4) **Kruskal's Minimum Spanning Tree (MST):** In Kruskal's algorithm, we create a MST by picking edges one by one. The Greedy Choice is to pick the smallest weight edge that doesn't cause a cycle in the MST constructed so far.

2.0 TRAVELLING SALESMAN PROBLEM

3.0.1 OVERVIEW

The travelling salesman problem (TSP) asks the following question:

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

TSP is important in operations research and theoretical computer science.
TSP is a special case of the travelling purchaser problem and the vehicle routing problem.

In the theory of computational complexity, the decision version of the TSP (where, given a length L, the task is to decide whether the graph has any tour shorter than L) belongs to the class of NP-complete problems. Thus, it is possible that the worst-case running time for any algorithm for the TSP increases super polynomially (perhaps, specifically, exponentially) with the number of cities.

The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, contains a large number of heuristics, an exact greedy algorithms can be formulated, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%.

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept ‘city’ represents, for example, customers, soldering points, or DNA fragments, and the concept ‘distance’ represents travelling times or cost, or a similarity measure between DNA fragments. The TSP also appears in astronomy, as astronomers observing many sources will want to minimize the time spent moving the telescope between the sources. In many applications, additional constraints such as limited resources or time windows may be imposed.

3.0.2 SIMPLE ALGORITHM OF TRAVEL SALEMAN PROBLEM

Given an optimization problem, a greedy algorithm tries “to find an optimal solution by making a sequence of greedy choices”. In each step, it makes the choice that looks best at the moment, according to some local criterion. For example, A greedy algorithm for the travelling salesman problem:

Step 1: Pick an arbitrary city and call it city 1.

Step 2: Find a city with the smallest distance from city 1, and call it city 2.

Step 3: Find a city in the rest of the n - 2 cities with the smallest distance from city 2.

Step 4: Output the tour: City 1 →City 2 → ... → City n → City 1.

3.0.3 ADVANCE ALGORITHM OF TRAVELLING SALESMAN PROBLEM

The following are the steps of the greedy algorithm for a travelling salesman problem:

Step 1: input the distance matrix, $[D_{ij}]i = 1, 2, 3, ..., n$, where n is the number of nodes in the distance network.

Step 2: Randomly select a base city, let it be X and delete the column X of the distance matrix

Step 3: Include X as the first city in the tour.

Step 4: in the row X, find the least undeleted matrix cell entry and identify the corresponding column (break tie randomly), let this column be Y.

Step 5: include Y as the next city to be visited in the tour.

Step 6: Delete the column Y of the distance matrix.

Step 7: Check whether all the columns of the distance matrix are deleted. If yes, to *Step 9:* otherwise go to step 8.

Step 8: Set X = Y and go to step 4.

Step 9: include the first city as the last city in the tour.

Step 10: list the cities in the tour along with the corresponding total distance of travel.

4.0 IMPLEMENTATON

4.0.1 ENVIRONMENT VARIABLES

Operating System Version = Windows 8.0

System RAM: 2.0 GB

Processor Speed: 2.0 GHZ

4.0.2 SOFTWARE TOOLS USED

Development Platform = Microsoft Visual Studio 2015

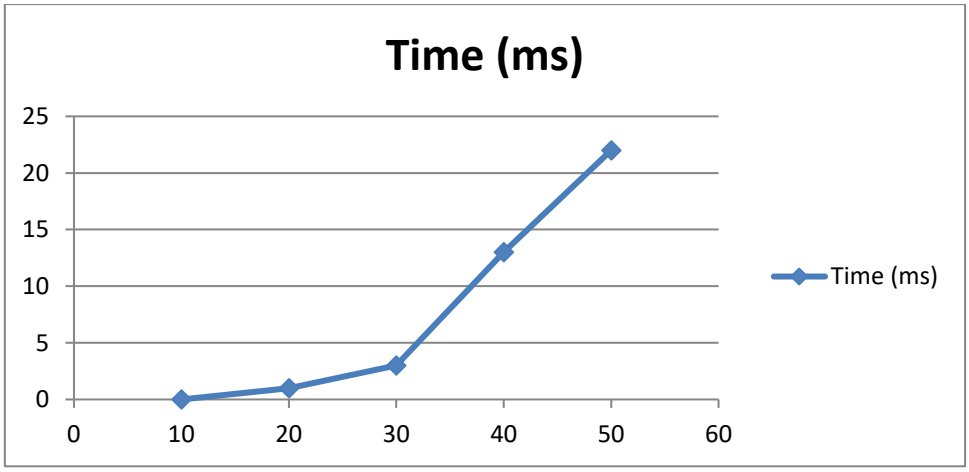
Programming Language = C# (C Sharp)

4.0.3 PROFILING

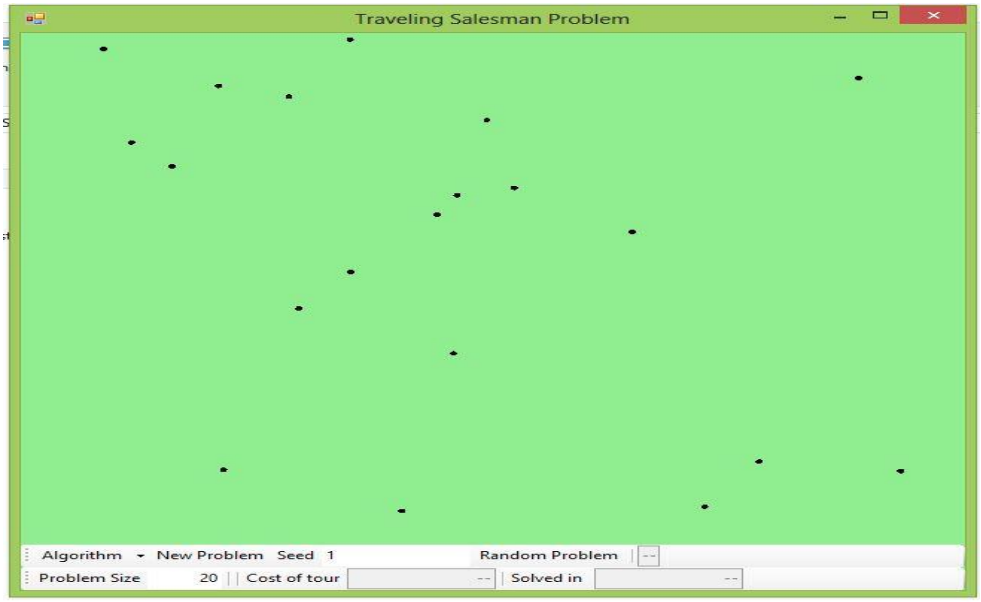
S/N	Problem Size	Time (ms)
1	10	0
2	20	1
3	30	3

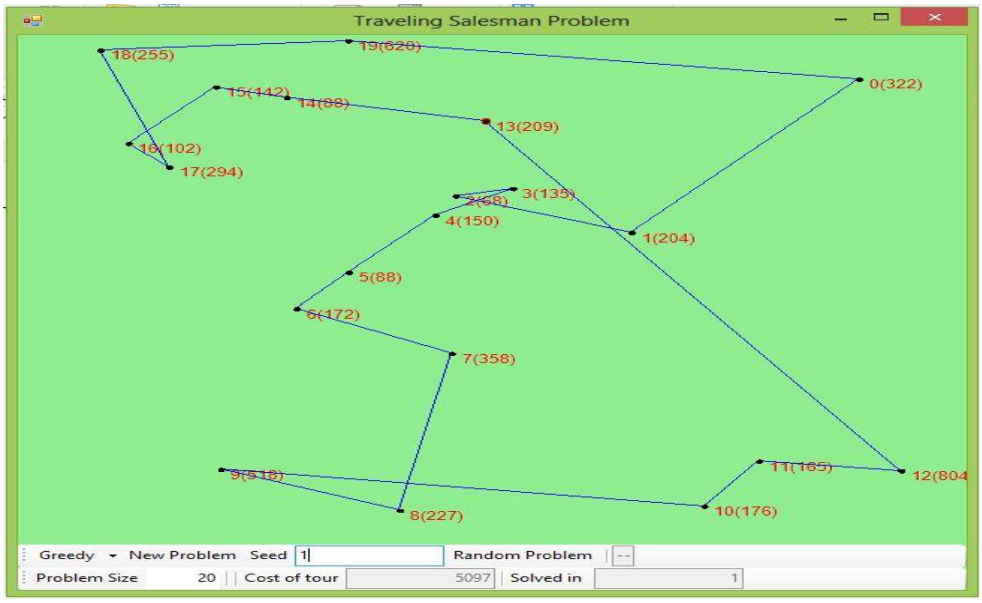
4	40	13
5	50	22

NOTE: According to the profiling of number of input (n) against time in milliseconds, the order of magnitude is of the order O (N),



SCREEN SHOTS OF THE TRAVEL SALESMAN PROBLEM(TSP)





5.0 SOURCE CODE

//This is the class that implements the greedy solution for TSP

```
private class GreedyRoute
        {
publicGreedyRoute(City[] list)
            {
                Cities = list;
visited = new bool[Cities.Length];
for (int i = 0; i <visited.Length; i++)
visited[i] = false;
                r = new Random();
startIndex = r.Next() % Cities.Length;
numVisited = 1;
route = new ArrayList();
            }

public void SetRoute()
            {
if (startIndex>= Cities.Length)
startIndex = startIndex % Cities.Length;

                //start with a random city
                City currentCity = Cities[startIndex];
route.Add(Cities[startIndex]);
visited[startIndex] = true;
                City nextDest = null;
while (numVisited<Cities.Length)
                {
doubleminCost = Double.PositiveInfinity;
intcurrentIndex = -1;

                    //look for the city that has the lowest cost to get to
```

```
for (int i = 0; i <Cities.Length; i++)
                    {
if (!visited[i])
                        {
      doubletempCost = currentCity.costToGetTo(Cities[i]);
if (tempCost<minCost)
                            {
minCost = tempCost;
nextDest = Cities[i];
currentIndex = i;
                            }
                        }
                    }

                    //break if a dead-end city is reached
if (minCost == Double.PositiveInfinity)
break;
else
                    {
route.Add(nextDest);
visited[currentIndex] = true;
currentCity = nextDest;
                    }
                }

//set the route to null if not completed
if (currentCity.costToGetTo(Cities[startIndex]) == Double.PositiveInfinity)
route = null;
else if (route.Count<Cities.Length)
route = null;
            }

publicArrayListGetRoute()
            {
return route;
            }

            //get ready to form a new route
public void Reset()
            {
startIndex = r.Next();
route = new ArrayList();
            }

private Random r;
private City[] Cities;
private bool[] visited;
privateintstartIndex;
privateintnumVisited;
privateArrayList route;
        }
```

REFERENCES

Basic Fundamentals of Algorithm, Design and Analysis, *A Lecture Note prepared by Prof. Souley Boukari, for the Course Algorithm Design and Analysis (CS531)*, Department of Mathematical Science, AbubakarTafawa University, Bauchi, 15th May, 2015.

Black, Paul E. (2 February 2005). "Greedy Algorithm". Dictionary of Algorithms and Data Structures. U.S. National Institute of Standards and Technology (NIST). Retrieved 1st August, 2016.

G. Gutin, A. Yeo and A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discrete Applied Mathematics 117 (2002), 81–86.

Hazewinkel, Michiel, ed. (2001), "Greedy algorithm", Encyclopedia of Mathematics, Springer, ISBN 978-1-55608-010-4