Mutually Nearest Vertex Clusters for Solving TSP

Govindaraj Pandith T. G.
Department of Computer Applications
AIMS Institute of Higher Education
Bangalore, India
Registered research candidate in Rayalaseema University, Kurnool

Siddappa M., PhD
Department of Computer Science and Engineering
Sri Siddartha Institute of Technology
Tumkur, India

ABSTRACT
The Travelling Salesman Problem (TSP) is a classical problem in the field of combinatorial optimization. Main objective of TSP is to find an optimal tour which starts from an arbitrary city (vertex), visits remaining cities exactly once and returns back to the city at which tour commenced. TSP belongs to the class of NP Complete problems, has been studied for many years and is still being studied; a general solution is yet to be reported.

Proximity of cities plays a vital role while traversing a set of cities. Proximity can be traced to a similar measure called distance measure, used in Pattern Recognition (PR). Newer and newer distance measures are proposed in PR for cluster analysis and classification. The sole aim of these measures is to cluster those sets of vertices, which are highly similar or in other words form group of vertices by taking into account distance as a metric of PR from the given instance of complete graph. The objective of this study is to construct a round tour by utilizing the mutually nearest vertices.

General Terms
Graph, Hamiltonian Circuit, Cost adjacency matrix algorithm and time complexity.

Keywords
TSP, PR, distance measure, cluster analysis and cluster classification.

1. INTRODUCTION
Objective: To design an algorithm to find the nearest vertex clusters to obtain solution for TSP

Travelling Salesman Problem (TSP) is one among the distinguished problem in the areas of graph theory and operations research. TSP is an interesting problem whose statement is quite simple, where as obtaining the optimal solution for larger instances is extremely cumbersome. The aim of TSP is find the least cost Hamiltonian Circuit.

TSP has many practical applications. Important applications are

- Many real – world problems, ranging from establishing an optimal route for milk van, mail collection / distribution, news paper distribution, school buses can be solved easily.
- TSP plays important role in solving industrial applications like PCB drilling, computer wiring, message routing and computer networking.
- Many related problems, such as problem of repetition of cities, multiple salesman, stochastic TSP, which can be solved by bringing in minor changes in the TSP.

Formally, goal of clustering is to segregate given a data set into different groups based on some predefined criteria. Several clustering techniques use distance measure to analyze the similarity or dissimilarity between any pair of vertices [1]. Distance between the pair of vertices v and w can be denoted as dist(v,w). Distance as a measure has to satisfy following criteria

i) Distance from a vertex to itself is zero.i.e dist(v,v)=0

ii) If the distance between vertices are symmetrical then dist(v,u)=dist(u,v) else dist(v,u)≠dist(u,v)

iii) The triangle inequality holds: dist(v,u)≤ dist(v,w)+dist(w,v)

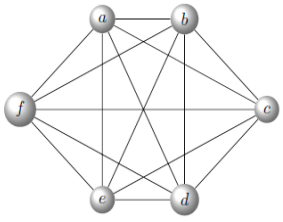
There are numerous methods for solving TSP[2-6]. Broadly algorithms can be classified into following categories:

- Exact Solvers
- Non-Exact Solvers

Exact solvers guarantee finding optimal solutions. These algorithms consume huge amount of running time and space requirements. On the other hand Non-Exact solvers offer potentially non-optimal but typically faster solutions. Many heuristic methods which can give solutions with 2-3% error margin from the optimal solution have been developed

Cluster analysis widely used in many fields viz., machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. An attempt has been made to obtain a sub-optimal solution for given instance of TSP by finding clusters.

2. CLUSTER OF VERTICES FOR SOLVING INSTANCE OF SYMMETRIC TSP
Procedure to form the cluster of vertices is illustrated by taking following instance of TSP shown in Figure 1.



$$\begin{pmatrix} 0 & 6 & 5 & 8 & 2 & 5 \\ 6 & 0 & 7 & 6 & 3 & 1 \\ 5 & 7 & 0 & 10 & 10 & 9 \\ 8 & 6 & 10 & 0 & 8 & 4 \\ 2 & 3 & 10 & 8 & 0 & 4 \\ 5 & 1 & 9 & 4 & 4 & 0 \end{pmatrix}$$

Fig1: Instance of symmetric TSP

Initially consider all single vertex clusters. Stack data structure is used for the formation of clusters. Consider {d} as a initial vertex cluster. Push {d }to the stack. Next nearest cluster to {d} is {f}.Push {f} to the stack. Next {b} cluster is nearest to vertex {f}. Push {b} to the stack as shown in the Figure 2.

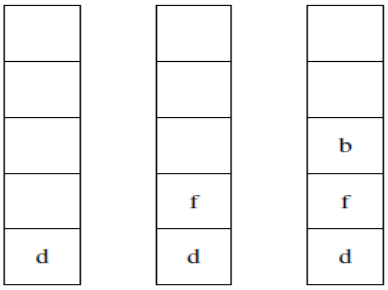


Fig 2: Formation of cluster using stack

Next nearest vertex for b is f which is the immediate predecessor, therefore pop all the vertices to get the cluster of vertices d f b(Means the vertices d f and b are mutually nearest vertices). Similarly the cluster of vertices {e, a} is obtained. Finally the single cluster {d,f,b,e,a,c} is obtained by merging the three sets {d,f,b},{e,a} and vertex {c}. Vertex clusters are represented in the Figure 3 and Figure 4

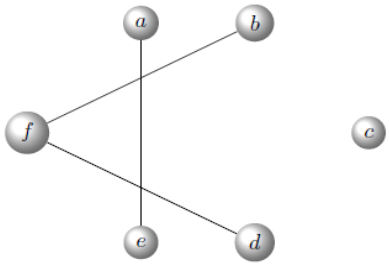


Fig 3: Different clusters

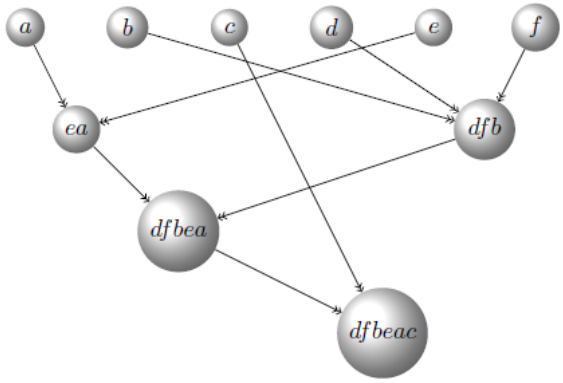


Fig 4: Merging the clusters

3. NEAREST-NEIGHBOR CHAIN ALGORITHM

In the theory of cluster analysis, the nearest-neighbor chain algorithm [7] takes a collection of points as input, and creates a hierarchy of clusters of vertices by repeatedly merging pairs of smaller clusters to form larger clusters.

The main idea of the algorithm is to find pairs of clusters to merge by following paths in the nearest neighbour graph of the clusters. Every such path will eventually terminate at a pair of clusters that are nearest neighbors of each other, and the algorithm chooses that pair of clusters as the pair to merge. In order to save work by re-using as much as possible of each path, the algorithm uses a stack data structure to keep track of each path that it follows. By following paths in this way, the nearest-neighbor chain algorithm merges its clusters in a different order than methods that always find and merge the closest pair of clusters. However, despite that difference, it always generates the same hierarchy of clusters.

The time complexity of nearest-neighbor chain algorithm is O(n2), where n is the number of vertices to be clustered, when the input is provided in the form of an explicit distance matrix.

The algorithm performs the following steps:[8][9]

- Initialize the set of active clusters to consist of n one-vertex clusters, one for each input vertex.
- Let T be a stack for the tour, initially empty, the elements of which will be active cluster of vertices.
- While still there is more than one cluster in the set of clusters:
  - If T is empty, choose the first active cluster and push it onto T.
  - Let C be the active cluster on top of S. Compute the distances from C to all other clusters, and let D be the nearest other cluster.
  - If D is already in T, it must be the immediate predecessor of C. Pop both clusters from T and merge them.
  - Otherwise, if D is not already in T, push it to T.
- Repeat the above steps for obtaining all routes starting from respective vertices
- Select the least cost route as solution for the given instance of TSP.

In the case if one cluster has multiple equal nearest neighbors, then the algorithm has to adopt a consistent tie-breaking rule. For instance, one may assign priority index numbers to all of the clusters, and then select the one with the smallest priority index number [10]. In the case of directed graphs a search operation on stack is necessary to check whether a newly selected vertex cluster is present or not. New cluster is pushed to the stack if it is not present in the stack. If the cluster already exists in the stack then form the cluster of vertices by popping all the vertices in the stack. Linked lists can be used for merging the clusters.

4. CLUSTER OF VERTICES FOR SOLVING INSTANCE OF ASYMMETRIC TSP

Similarly Nearest Neighbor chain algorithm can be used for solving asymmetric TSP. Illustration of the method for the instance of 10 cities TSP represented by cost adjacency matrix follows

$$\begin{pmatrix} 0 & 11 & 7 & 6 & 16 & 8 & 9 & 4 & 2 & 9 \\ 14 & 0 & 4 & 12 & 5 & 15 & 22 & 23 & 18 & 21 \\ 2 & 3 & 0 & 1 & 4 & 2 & 21 & 2 & 34 & 10 \\ 6 & 32 & 21 & 0 & 5 & 2 & 8 & 7 & 12 & 15 \\ 5 & 9 & 20 & 10 & 0 & 16 & 11 & 19 & 4 & 7 \\ 14 & 22 & 20 & 15 & 11 & 0 & 4 & 2 & 21 & 14 \\ 3 & 29 & 4 & 19 & 12 & 11 & 0 & 3 & 21 & 4 \\ 20 & 15 & 11 & 9 & 6 & 12 & 14 & 0 & 13 & 17 \\ 9 & 11 & 14 & 16 & 18 & 24 & 28 & 20 & 0 & 6 \\ 11 & 7 & 8 & 4 & 31 & 27 & 29 & 32 & 9 & 0 \end{pmatrix}$$

choosing a as the initial vertex, following the similar procedure as symmetric TSP results in vertex clusters {a,i,j,d,f,h,e},{b} and {g,c}. In the case of asymmetric TSP the recently selected vertex may be present in any part of the

stack. So searching the stack becomes a mandatory requirement. If the recently selected new vertex already exists in the stack then pop all the elements of the stack to form the cluster. Merging all the clusters gives the single cluster {a,i,j,d,f,h,e,g,c,b}.

5. RESULT OF SYMMETRIC TSP

An attempt has been made to obtain most appropriate solution for symmetric TSP by clustering the vertices based on nearest neighbor chain algorithm. Stage by stage clusters are formed and merged to get the larger clusters. The route can be decided by the final set of cluster i.e. {d f b e a c}.

The tour obtained is

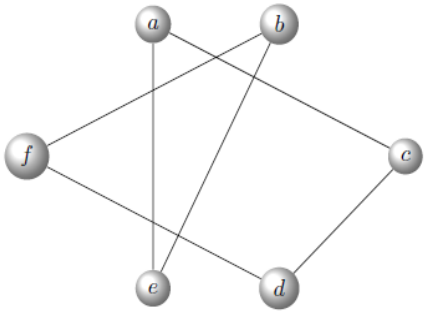


Fig 5: The optimal tour: d→ f→b→e→a→c→d

The total cost of the tour is 25 which is the least cost route obtained by this method. If the vertex a is considered as the arbitrary staring vertex the route obtained is a→e→b→f→c→d→a and cost of the tour is 33. If vertex b is considered as starting node then the route obtained is b→f→d→e→a→c→b and the cost of the tour is 27. Similarly route obtained by starting from vertex c is c→a→e→b→f→d→c. This route cost is 25 and which is another least cost route. Route starting from vertex e is e→a→b→f→d→c→e and its cost is 33. Finally route starting from f is f→b→d→e→a→c→f whose cost is 31.

6. RESULT OF ASYMMETRIC TSP

The result for the above instance of asymmetric TSP can be obtained from following tours. Round tour obtained starting from vertex a has total cost 54. Following is the round tour.

a →i → j →d →f →h →e →g →c →b →a

Round tour obtained starting from vertex b is

b →c →d →f →h →e →i →j →g →a →b with total cost 68.

Round tour obtained starting from vertex c is

c →a →i → j →d →f →h →e →b →g →c with total cost 59.

Round tour obtained starting from vertex d is

d →f →h →e →i →j →b →a →g →c →d with total cost 59.

Round tour obtained starting from vertex e

e →i →j →d →f →h →g →a →b →c →e with total cost 52.

Round tour obtained starting from vertex f is

f →h →e →i →j →d →g →a →b →c →f with total cost 50.

Round tour obtained starting from vertex g is

g →h →e →i →j →d →f →b →c →a →g with total cost 62.

Round tour obtained starting from vertex h is

h →e →i →j →d →f →g →a →b →c →h with total cost 46.

Round tour obtained starting from vertex i is

i →j →d →f →h →e →g →c →b →a →i with total cost 54.

Round tour obtained starting from vertex j is

j →d →f →h →e →i →b →c →a →g →j with total cost 48.

Therefore the optimal tour among the tours mentioned above is **h→e→i→j→d→f→g→a→b→c→h** with total cost 46.

7. CONCLUSION

TSP can be solved by many methods. The methods are broadly classified into two categories of algorithms viz., Exact solvers and Non-exact solvers. Even though exact solvers produce optimal solution they are highly in efficient with respect to space and time trade off for larger instances. In this study a Non-exact solver algorithm is proposed based on the nearest neighbour chain algorithm. The advantage is in terms of simplicity and faster execution. The disadvantage is that algorithm always may not give exact solution. However select the least cost cluster as an effective estimate for the optimal tour. In the case of symmetric TSP execution time of the algorithm is $O(n^3)$. A slight modification of the algorithm is required in the case of asymmetric TSP (Since there is requirement of search operation for iteration). The time complexity of the modified algorithm will be $O(n^4)$. As far as data structure is concerned one can use either arrays or linked list for stack implementation. Merging of clusters can be achieved by using linked lists. Several variations are possible with respect to the proposed algorithm which provides scope for further investigation.

8. REFERENCES

[1] Satu Elisa Schaeffer 'Graph clustering', *ELSEVIER C O M P U T E R S C I E N C E R E V I E W 1 (2 0 0 7) 2 7 – 6 4 available at www.sciencedirect.com, journal homepage:* www.elsevier.com/locate/cosrev

[2] Chetan Chauhan, Ravindra Gupta, Kshitij Pathak "Survey of methods of solving TSP along with its implementation using Dynamic Programming approach" International Journal of Computer Applications, Vol-52,No-4, August 2012

[3] Anshul Singh, Devesh Narayan "A survey paper on Solving Travelling Salesman Problem using Bee colony optimization", ISSN 2250-2459, Volume 2, Issue 5, May 2012.

[4] Corinne Brucato, "The Travelling Salesman Problem", University of Pittsburg, 2013.

[5] Anitha Rao, Sandeep Kumar Hegde "Literature Survey On Travelling Salesman Problem Using Genetic Algorithms" International Journal of Advanced Research in Eduation Technology (IJARET) Vol. 2, Issue 1 (Jan. - Mar. 2015) ISSN : 2394-2975 (Online) ISSN : 2394-6814 (Print)

[6] Komal Joshi1, Ram Lal Yadav "A new hybrid approach for solving travelling salesman problem using ordered cross over 1(ox1) and greedy approach" IJRET: International Journal of Research in Engineering and Technology, Volume: 04 Issue: 05 May-2015.

[7] Nearest Neighbour chain algorithm, Wikipedia

[8] Murtagh, Fionn (1983), "A survey of recent advances in hierarchical clustering algorithms" (PDF), The Computer Journal, 26 (4): 354–359, doi:10.1093/comjnl/26.4.354

[9] Murtagh, Fionn (2002), "Clustering in massive data sets", in Abello, James M.; Pardalos, Panos M.; Resende, Mauricio G. C., Handbook of massive data sets, Massive Computing, 4, Springer, pp. 513–516, Bibcode:2002hmds.book.....A, ISBN 978-1-4020-0489-6

[10] Sedgewick, Robert (2004), "Figure 20.7", Algorithms in Java, Part 5: Graph Algorithms (3rd ed.), Addison-Wesley, p. 244, ISBN 0-201-36121-3.

9. AUTHOR PROFILE

Govindaraj Pandith T.G has fourteen years of teaching experience for UG and PG courses for computer applications and is presently working as Assistant Professor in the department of Information Technology, AIMS Institute of Higher Education, Bangalore. He Obtained B.Sc.(Physics, Chemistry, Mathematics) from Sri Bhuvanendra College, Karkala affiliated to Mangalore University in the year 1994. Completed PGDCA from Dr. NSAM First Grade College, Nitte in the year 1995 affiliated to Mangalore University. He completed M.Sc, Mathematics from Mangalagangotri, Mangalore University in the year 1997. He also successfully completed M.Sc, Computer Science from Mangalagangotri, Mangalore University with distinction in the year 2001. His research interests are in the areas of Discrete mathematics, Data structures, Pattern Recognition, Analysis and Design of Algorithms. Currently pursuing PhD from Rayalaseema University, Kurnool under the guidance of Dr. Siddappa M, Head of the department of Computer Science and Engineering, Sri Siddartha Institute of Technology, Tumkur.

Dr. M. Siddappa received B.E and M.Tech degree in Computer Science & Engineering from University of Mysore, Karnataka, India in 1989 and 1993 respectively. He has completed doctoral degree from Dr.MGR Educational Research Institute Chennai under supervision of Dr.A.S.Manjunatha, CEO, Manvish e-Tech Pvt. Ltd., Bangalore in 2010. He worked as project associate in IISc, Bangalore under Dr.M.P Srinivasan and Dr. V.Rajaraman from 1993 – 1995. He has teaching experience of 29 years and research of 15 years. He published 62 Technical Papers in National, International Conference and Journals. He has citation index of 192 till 2016 and h-index of 5 and i10-index of 4 to his credit. He is a life member of ISTE. He is working in the field of data structure and algorithms, Artificial Intelligence, Image processing and Computer networking. He worked as Assistant Professor in Department of Computer Science & Engineering from 1996 to1998 in Sri Siddhartha Institute of Technology, Tumkur. Presently, he is working as Professor and Head, Department of Computer Science & Engineering from 1998 in Sri Siddhartha Institute of Technology, Tumkur. He has visited Louisiana university Baton rouge, California university and Wuhan university China. He bagged “Best Teacher” award from ISTE in the year 2013.