Implementation of an Application on Android Devices for Optimal Tour

Vijay Babar, Sonali Jadhav, Pooja Jagtap, Ajendra Joshi

***Abstract*— This paper aims to develop an Android Application that can provide optimal tour (shortest distance) for the selected locations using google maps. Using the optimal tour results in efficient use of time and fuel. In current situation of ever rising petrol prices and waste of valuable time in travelling, it's wise to schedule and plan the tour such that it uses minimum resources. The Application tries to address this requirement. And since android has become a door-to-door technology with high processing power and is extremely mobile, it provides a perfect platform for such an Application. The clustering method for increasing number of nodes is also proposed in this paper, to reduce computation time complexity of the algorithm.**

***Index Terms*— Android, Clustering, TSP Branch and Bound, TSP Dynamic Programming, TSP Greedy method.**

I. INTRODUCTION

The "Travelling Salesman Problem" (TSP) is a common NP hard problem that can be used to test the effectiveness of Genetic Algorithm. The travelling salesman problem is defined in simple term as: "If there are n cities and we have given path between all of them, and we want to find the cheapest and shortest path, coming back to the point from where we have started to travel, provided that all the cities involved in the problem are travelled exactly once". [1]

II. RELATED WORK

There are following different algorithmic strategies by which we can solve the TSP problem.

A. Greedy Strategy:

Greedy algorithm is the simplest improvement algorithm. It starts with the departure Node. Then the distances to other n-1 nodes is calculated. Then go to the next closest node. The current node is taken as the departing node, and selects the next nearest node from the remaining n-2 nodes. The process will continue until all the nodes are visited once and only once then back to Node 1. When the algorithm is terminated, the sequence is returned as the best tour. [2]

TSP using Greedy Approach:

O (n), n =|V|.

• Number of loop iterations: n−1 (0 to n−2edges).

• Cost for each time through loop: O (degree (cur))

• Time for steps after the loop: O (1)

Total time is: $n + \sum_{cur}$ Degree (cur) = n+ (n-1).

O (m) =O (nm) =O (|V|.|E|)

Where m=|E| [4]

B. Branch and bound:

The Branch and Bound strategy divides a problem which we want to solve into a number of sub-problems and then the result of these sub problems is combined to get final result. It is a system for solving a sequence of sub-problems each of which may have multiple possible solutions and the solution chosen for one sub-problem may affect the solutions of later sub-problems. [3]

TSP by Branch and Bound Algorithm:

Step 1: Choose a start node.

Step 2: Set bound to a very large value, for example infinity.

Step 3: Choose the shortest arc between the current and unvisited node and add the distance to the current distance and repeat while the current distance is less than the bound.

Step 4: If current distance is less than bound, then we are done

Step 5: Add up the distance and bound will be equal to the current distance.

Step 6: Repeat step 5 until all the arcs have been covered

Time complexity: O (n!) [3]

C. Dynamic programming:

Dynamic programming (DP) is algorithm in which problem is broken up into a series of overlapping sub-problems, and solution is built to larger sub-problems. Unlike the divide-and-conquer paradigm, DP solves all possible sub-problems rather than a small portion. [2]

Manuscript received January 20, 2014.

Vijay Babar, BE (Computer), Pimpri Chinchwad College of Engineering, Pune-411044,India

Sonali Jadhav, BE (Computer), Pimpri Chinchwad College of Engineering, Pune-411044, India (e-mail: pooja.jagtap21@gmail.com)

Pooja Jagtap, BE (Computer), Pimpri Chinchwad College of Engineering, Pune-411044,India. (e-mail: pooja.jagtap21@gmail.com)

Ajendra Joshi, BE (Computer), Pimpri Chinchwad College of Engineering, Pune-411044, India (e-mail: joshi.ajendra@gmail.com)

TSP by Dynamic Programming:

Time Complexity = $O(n^2 2^n)$

Minimum Distance is given by:

$g(1, V-\{1\}) = \min (C_{1k} + g(k, V - \{1, k\}))$

Generalized solution:

$g(i, S) = \min_{j \in S} (C_{ij} + g(i, S-\{j\}))$

To solve TSP using these formulas:

- First compute g (i,S) with |s| =0 i.e. Compute $g(i,\Phi) = C_{i1}$ for $1 \leq i \leq n$.
- Then compute g (i, S) with |S| = 1.
- Then compute g (i, S) with S=2 and so on.
- Finally compute g(1,V-{1})

At each stage which computes g(i, S) remember the value of j that minimizes the cost of g (i, S). Let J (i, S) denotes this value. These J values give us the optimal tour.

D. ITS-TSP algorithm:

Intelligent Transport System Travelling Salesman Problem (ITS-TSP) has here variations on the traditional TSP:

1. Edge weights can change constantly in transportation-based graphs when they are based on the time to traverse the roadway they represent.
2. Not every node in a transportation graph will need to be visited on every route, so a route that visits only a predefined subset needs to be determined.
3. A node can be visited more than once if that provides a faster route, since the edge weights can change while travelling. [1]

Considering these variations, they have designed their algorithm:

ITS-TSP Algorithm:

Step 1 – Create a sub-graph of the original graph with nodes comprised of the start node, intermediate nodes, and end node. The weights on the edges will correspond to the minimum cost path between the nodes.

Step 2 – Iterate over all of the intermediate nodes

a) Determine the minimum weight path from the start node to the intermediate node and add it to the potential fastest route

b) From the original set of paths found in step 1, remove all paths that have a source of the start node

c) From the original set of paths found in step 1, remove all paths that have a destination of the intermediate node

d) As long as the set of paths still contains more paths

1. Determine the path with the minimum overall cost that still remains
2. As long as that path does not create a cycle in our potential fastest route, add it to the potential fastest route
3. From the original set of paths found in step 1, remove all paths that have a source of the source node.
4. From the original set of paths found in step 1, remove all paths that have a destination of the destination node.

e) Order the paths in the potential fastest route from the start node where the destination of one path is the source of the next path

f) Add the fastest path from the destination of the route back to the start node to the potential fastest route.

g) Store the potential fastest route in an array and reset the potential fastest route to be empty for the next iteration.

Step 3 – From all the routes found in Step 2, find the route with minimum weight and return it. [ITS-TSP]

Using ITS_TSP they were able to reduce the running time to polynomial-time. [1]

E. Comparison:

After executing the program to solve TSP by these methods, we got following output. These results are obtained after executing algorithms on Linux OS and i5 core processor. Comparison between computational times required to get output of TSP using different algorithms:

Table 1: Time required getting output of TSP using different algorithmic strategies

Nodes	Backtracking	Dynamic Programming	Depth-First Branch and Bound
7	~ 0.0001 s	~ 0.0000 s	~ 0.0000 s
8	~ 0.4 s	~ 0.0000 s	~ 0.0000 s
9	~ 3.54 s	~ 0.0000 s	~ 0.0000 s
10	~ 32.681 s	~ 0.05 s	~ 0.03 s
11	~ 329.1 s	~ 0.42 s	~ 0.52 s
12	~ 3624.54 s	~ 4.82 s	~ 1.43 s

Among all these strategies, we found that TSP can be better implemented on android using dynamic programming strategy, as dynamic programming algorithm stores results or solutions for small sub-problems. The technology used is android. For the nodes which user wants to travel are shown on a map in this application. For this purpose, google map and its services are used.

III. TECHNOLOGY USED

A. Android:

Android is a Linux-based operating system designed primarily for touch screen mobile devices such as smart phones and tablet computers. Android is open source and Google releases the code under the Apache License. This open-source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. [10]

B. Google Maps:

Google maps provide an intuitive and highly responsive mapping interface with aerial imagery and detailed street data. In addition, map controls and overlays can be added to the map so that users can have full control over map navigation. Map panning can also be performed by dragging the map via the mouse or by using “arrow” keys

on a keyboard. Google maps can be customized according to application specific needs. Various web-based application and the results can be displayed on Google maps. [9][10]

Geo-Coder class:

1) Public Geocoder (Context context, Locale locale): The response of the above constructor will be localized for the given Locale. Where, Context is the Context of the calling activity and Locale is the desired Locale for the query results.

2) Public Geocoder (Context context): The response here is localized for the default system Locale. Where, Context is the Context of the calling activity

3) Public Methods: Public List<Address> getFromLocation (double latitude, double longitude, int maxResults): It returns an array of addresses that describe the area that are surrounded by the given latitude and longitude. The parameters are:

- Latitude: the latitude a point for the search
- Longitude: the longitude a point for the search

maxResults: max number of addresses to return.

4) Public static Boolean isPresent (): The above method returns true if the methods getFromLocation and getFromLocationName are implemented. Due to lack of network connectivity, the method may return null or empty lists.[10]

IV. CLUSTERING

As the time complexity of dynamic algorithm increases exponentially, it takes a lot of time to solve the problem having 10 or above that number of nodes. So to overcome this problem, we can use clustering method for this. There are various clustering methods proposed.

A. Genetic algorithm for clustering:

Clustering Genetic Algorithm is proposed which tends to perform better for TSP when compared with existing algorithms. The methodology of CGA is simple which effectively groups chromosomes in the population into clusters using K- means clustering and then applies genetic operators.

- If there are n cities in the chosen problem, it has to be divided into x/n subgroups (clusters) where x represents the number of chromosomes in the entire population and n represents the size of clusters.
- Parent selection, genetic operators like crossover and mutation are applied and best ordering of the cities within each group are calculated independently.
- Finally the results from each group are combined to form a single solution. Thus it will reduce overhead for the algorithm to apply genetic operators as a whole which takes more time to execute.[7]

Following are the operators used to order the cities:

- Crossover operator :

Crossover operator generates 2 offspring chromosomes from parents' chromosomes which bring about diversity characteristic of new child.

- Mutation operator :

Mutation is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly the new offspring.

- Selection Operation :

Selection operation is a selection of chromosomes to be the chromosome in the next generation. According to Darwin's evolution theory, the best ones should survive and become the original breed for the next generation. Chromosomes with good fitness are probably selected. "Unpublished" [11]

B. K-means clustering:

K-Means clustering is an exclusive clustering algorithm. Each object is assigned to precisely one of a set of clusters. For this method of clustering we start by deciding how many clusters we would like to form from our data. We call this value k. The value of k is generally a small integer, such as 2, 3, 4 or 5, but may be larger.

We next select k points. These are treated as the centroids of k clusters, or to be more precise as the centroids of k potential clusters, which at present have no members.

Let us take an example. Suppose following are the nodes on which we have to apply K-means clustering method.

Step1: Locate the nodes.

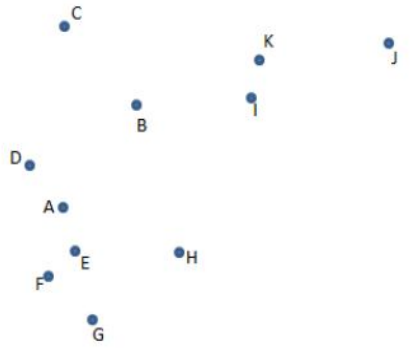

Fig 1.Example of nodes

Step 2: Take k=3 and form three clusters. Find their centroids: centroid 1, centroid 2 and centroid 3.

Step 3: Apply TSP in each cluster to find optimal path between the nodes in each clusters.

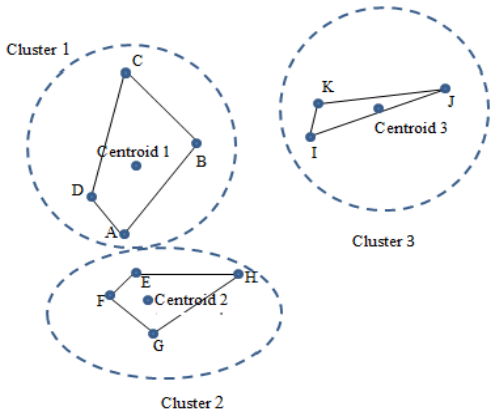

Fig 2.Finding clusters using K-means clustering

Step 4: Now to connect the clusters get the shortest distance between two centroids. The distances can be compared by using formula:

Round to compare = Factorial (number of centroid)/ Factorial (number of centroid - 2) * Factorial (2).

For our example, we have three centroids and rounding to compare answer is three. The result found that centroid 1 and centroid 2 are nearest centroids. To get the nearest nodes between centroid 1 and centroid 2, find the distance between nearest point to cluster and centroid of other cluster. The nearest point between cluster 1 and centroid 2 is A and the nearest point between cluster 2 and centroid 1 is E. So join these two nodes.

Similarly, to join cluster 2 and cluster 3, join points H and I and to join cluster 1 and cluster 3, join points B and K. Hence we get the final path which covers all the nodes and gives optimal path to travel to all the nodes. “unpublished” [11]

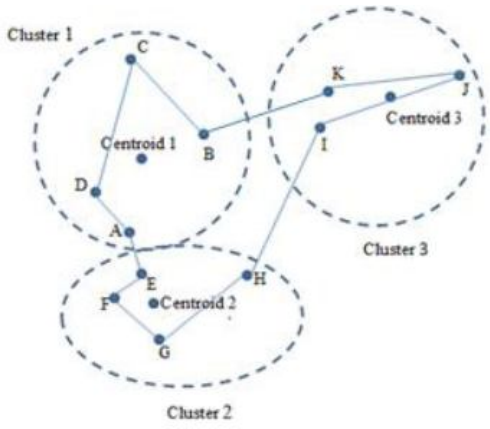

Fig 3.Optimal Path Using TSP

C. Adaptive Clustering:

The key process of this study is decomposing the large- scale data, the result of which may impact the final outcome.

We use the adaptive data clustering approach to decompose the data; the algorithm in details is followed. Supposing the number of cities is N and the decomposition must be working when the number of cities is larger than m.

Decomposing the large scale data task into several sub-tasks and computing each of the sub-tasks independently is an efficient approach to solve the large-scale data task. But the data of TSP are not easy to decompose because of its complexity. This algorithm decomposes the data by their relativity which can reflect the main character of the relationship between the data of TSP. The method was proved to have good universality, and the key step of the algorithm was to decompose the data. They made use of the k- mean algorithm, which was a familiar and simple algorithm for clustering. The results of their experiments showed that this method was more effective than the ordinary GA, especially when the cities are more than 30000 the memory of ordinary computer can still satisfy the require of computing by data decomposition. K-mean algorithm is a simple clustering algorithm. [5]

Algorithm:

1) If N<m, not need to decompose the data;
2) If N>km, (1<k<m), decompose the N cities to k groups, if km<N<m2, decompose the N cities to [N/m] groups, and if N>m2, decompose the N cities to m groups.
3) If one of m groups in step 2) has more than m cities, then Go to step 2), else stop.

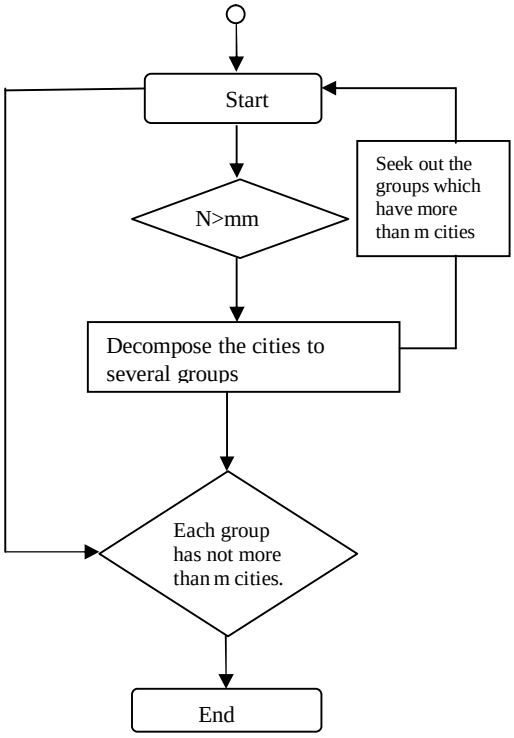

Fig 4: Flowchart for adaptive clustering

Comparison between different types of clustering methods according to time required to get the output is:

Table 2: Comparison chart for clustering methods

No of cities	Adaptive clustering algorithm	Computing Time	Ordinary GA	Computing Time
532	57680.9	1.641	1495160	4.219
1291	967533.4	3.125	1744777	7.094
2391	8161779	6.376	1.55E+07	25.000
5932	1.72E+07	28.125	4.24E+07	363.547
14051	1.74E+07	88.984	4.14E+07	7804.766
33810	4.84E+09	405.312	Memory Overflow	-
85900	1.52E+10	2424.297	Memory Overflow	-

V. PROPOSED ALGORITHM FOR CLUSTERING

We use dynamic programming for nodes less than 9 as it gives optimal solution within 3 seconds on android platform. For nodes more than 9, we use the following algorithm:

Step 1: If node count <= 9 .Implement TSP by Dynamic Programming.

Step 2: Else

a. Find all extreme nodes (East most, West most, North most, South most) by comparing latitudes and longitudes.

b. Using latitudes and longitudes of extreme nodes find centroid. The area between two adjacent extremes and centroid becomes cluster. Hence four clusters are formed.

c. Compare every other node with centroid and put them in respective cluster.

d. If start node ε extreme nodes,

i. Set direction clockwise or anti-clockwise.

Else

i. Find distance from start node to extreme nodes of the cluster.

ii. Use minimum distance from start node to extreme node to set direction.

e. Create cluster sequence depending on the selected direction.

f. Implement shortest path single source single destination algorithm in each cluster by using cluster sequence array.

i. The sequence of nodes is determine by shortest path algorithm gives the route.

Step 3: End.

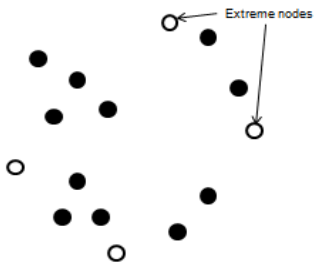

Fig. 5: Nodes to find the optimal path using clustering

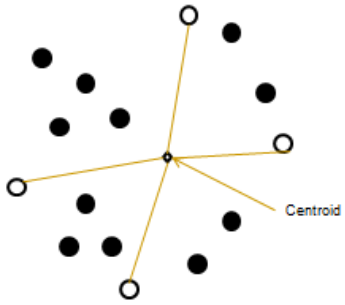

Fig. 6: Centroid of extreme nodes

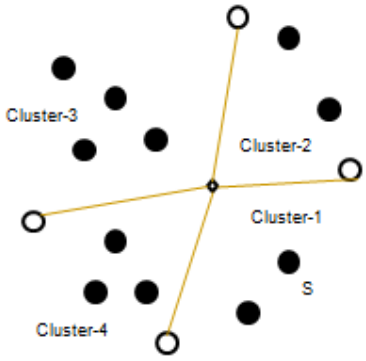

Fig. 7: Clusters formed

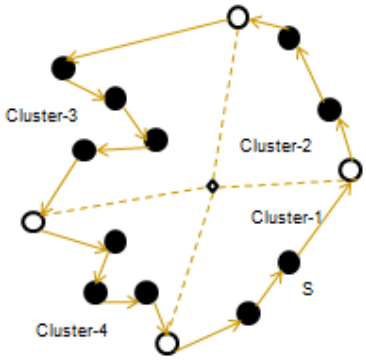

Fig. 8: Expected Output

Output of TSP using dynamic programming on android platform:

Table 3: Result of implementation of TSP using Dynamic Programming

Number of nodes	Time Required (sec)
4	~0.0000
6	~0.0000
8	~0.04
9	~2.031
10	~17.68

ACKNOWLEDGMENT

We express our sincere thanks to our project guide, Mr. Atul Pawar for his constant encouragement and support for the project, especially for the guidance given during project work.

REFERENCES

[1]Jeffrey Miller, Sun-il Kim, Timothy Menard Intelligent Transportation Systems Travelling Salesman Problem (ITS-TSP) - A Specialized TSP with Dynamic Edge Weights and Intermediate Cities"

[2]"Fundamentals of Computer Algorithms", By Ellis Horowitz, Sartaj Sahani, Sanguthevar Rajasekaran, Published by University Press Limited.

[3] Amanur Rahman Saiyed, The Traveling Salesman problem Indiana State University Terre Haute, IN 47809 , USA

[4]Mark G. Eramian, TSP Problem, Greedy Algorithms, and Backtracking. University of Saskatchewan Based on notes by G. Cheston and J. P. Tremblay

[5] Jin-Qiu Yang,Jian-Ganh Yang, Gen-Lang Chen, "Solving large scale TSP using adaptive clustering methods", IEEE, 2009.

[6] Prateek Agrawal , Harjeet Kaur , Pallavi Arora from Lovely Professional University, INDIA,"analysis and synthesis of enhanced ant colony optimization with the traditional ant colony optimization to solve travelling sales person problem**",**International Journal of Computers & Technology Volume 2 No.2 April 2012.

[7]R.sivaraj, dr.t.ravichandran, r.devi priya,"Solving Traveling Salesman Problem usingClustering Genetic Algorithm"International Journal on Computer Science and Engineering (IJCSE) ,Vol. 4 No. 07 July 2012 .

[8]Developer.android.com

[9]"Android Cookbook", By Ian F. Darwin, Published By O'Reilly Media

[10]"Professional Android 4 Application Development", By Reto Meier, Published By John Wiely & Sons.

[11] Supawadee Charoenwiengwechakij, Tanasanee Phienthrakul, "Solving traveling salesman problems with k-means clustering genetic algorithm."