International Research Journal of Advanced Engineering and Science
ISSN: 2455-9024

Automatic Itinerary Planning for Traveling Service Based on Budget using Spatial Datamining with Hadoop

Y. I. Jinesh Melvin[1], K. S. Charumathi[2], J. Teena[3]
[1, 3]Computer Engineering, Mumbai University, New Mumbai, Maharashtra, India
[2]Information Technology, Mumbai University, New Mumbai, Maharashtra, India
Email address: [1]yijmelvin@mes.ac.in

***Abstract** -To plan effective and economic successful trip is a very big challenge for a traveler. Even-though the travel agency can provide some arrangements for customer as per there schedule. But most of the customer will not satisfy there predefined itinerary. Existing system provides the automatic itinerary planning, which assumes the point of interest. In the previous system there are two stages to reduce the processing cost. In which offline stage for one day itineraries are pre-computed using map reduced and online stage for those one day itineraries are combined using a search algorithm. In our paper we consider the multi-day itinerary using nearest neighbor chain algorithm with budget based technique to get geographical information from spatial database and store the planed trip using Hadoop and also to reduce the complexity for a successful and satisfied trip for customer.*

***Keywords** -Spatial database, google map, Hadoop.*

I. INTRODUCTION

Travailing arrangements for a traveler is two ways, For a casual traveler they list out the local travel agency and pick a package from them, in fact it represents a pre-generated itinerary. Agency will help for booking hotels arranging transports, tickets of museums/parks etc. It prevent the customer from constructing their personalized itinerary which is very time consuming and inefficient. Although the travel agency make trying to give the effective itinerary to experience traveler but they cannot satisfy with the lake of customization from travel agency for providing these itinerary.

Here some point of Interests are missing in the itinerary and the packages are too expensive for a backpack traveler. Therefore they have to plan the trips in efficient and effective successful. So travelers will find-out all the details about there trip such as booking, car rental, Contacting, etc. Economic is a major problem for arranging trip, so to overcome all these problem we used budget based multidays itinerary planning.

The design of our approach is to generate the Spatial Database, which is include for find-out the place, hotels, roads, tourist-spot and all other details about there trip. Spatial database is very useful technology for multi-day trip [3]. In which Geographical Information System stores data collected from heterogeneous source in various format. GIS has emerged as a new discipline due to the development of communication technology. It generate amount of data as image, fat-file from sources like satellite imaginary sensors and other devices. In GIS the data representation from these device is classified into two main categories as Raster and vector data type. Raster is a two dimensional data type, which stores the value of pixel color of image in a cell. Vector data type represents as point, line and polygon. Vector data types are used to represents information from source such as roads, rivers, city, lakes, park boundaries with a layered hierarchy. This GIS is very useful for traveler because some traveler travels in unknown places. So that GIS will help to find all the information about travel itinerary. Spatial Datamining updates the information so that it requires large data-storage, and it face big data problem to overcome the problem in early stage. Hadoop is a technique to reduce the storage space in our system and take more information and gets high speed process from it. The distance of two POI, are evaluated by Google map's API's.

When you submit your paper print it in two-column format, including figures and tables. In addition, designate one author as the "corresponding author". This is the author to whom proofs of the paper will be sent. Proofs are sent to the corresponding author only.

II. PROBLEM STATEMENT

The previous problem statement in itinerary planning system, the user select their interested POI's and asks the system to generate k-day itinerary. To model the planning problem, we organize the POI's into a complete graph, there are two properties to denote the POI's into complete graph node (i) weight (ii) travel time.

The nodes are connected via weighted edges. The edge's weight is set to the average traveling time for the shortest path between the corresponding POI's in the map. There are two types of edge, The first type represents that the two nodes are directly connected with map. The second type contains multiple shortest path in the map [5]. Here it takes more time to find the shortest path.

Itinerary planning process has takes more spaces to store the relevant data into it. Selectivity of the planning is not perfected and user also not satisfied with it. Previous technique was not supported for budget planning trip. The elapsed time is estimated as in the rest discussion, we remove the hotel part and focus on how to merge the POI's into itinerary after the other POI's are fixed, we will solve the hotel selection problem.

III. System Architecture

In our proposed system trying a new ideas about a good and satisfied trip planning for customers. In previous system it get more affected and customers are not satisfied with it because of budget problem, data storage, no proper planning for trip and so on. To overcome this problem we use budget based itinerary planning with spatial datamining technique and to find shortest distance we choose Nearest Neighbor Chain Algorithm and store the information about the trip in to Hadoop. Figure 1 shows the architecture of our trip-planning system. In the first step, It collect the trip information from spatial database using spatial datamining techniques The Google Map's are used to evaluate the distance between POI's. Algorithm will find the shortest path between POI's and there budget based accommodation, then it display the K-day Itinerary and stored it into Hadoop for next same trip from other customer. Spatial Datamining technique will guide the travelers and it gives more information about the places like zoo, cities, villages, mountains, forest, etc.

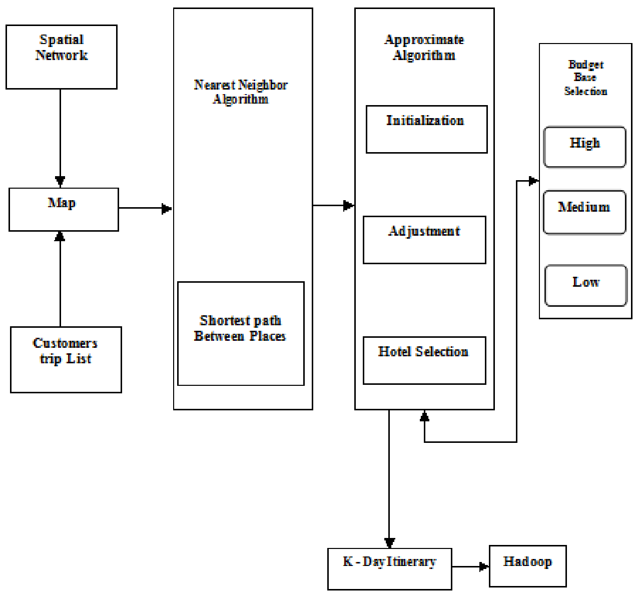

Fig. 1. System architecture

IV. Nearest Neighbor Chain

The nearest-neighbor chain algorithm is a method that can be used to perform several types of agglomerative hierarchical clustering, using an amount of memory that is linear in the number of points to be clustered and an amount of time linear in the number of distinct distances between pairs of points. The main idea of the algorithm is to find pairs of clusters to merge by following paths in the nearest neighbor graph of the clusters until the paths terminate in pairs of mutual nearest neighbors. In this proposed system nearest-neighbor chain algorithm is used to find shortest distance between tourist place that can easy for customers to travel and it will become a well planed trip.

V. Working Principle of NNC Algorithm

The input to a clustering problem consists of a set of points. A cluster is any proper subset of the points, and a hierarchical clustering is a maximal family of clusters with the property that any two clusters in the family are either nested or disjoint. Alternatively, a hierarchical clustering may be represented as a binary tree with the points at its leaves; the clusters of the clustering are the sets of points in subtrees descending from each node of the tree.

In agglomerative clustering methods, the input also includes a distance function defined on the points, or a numerical measure of their dissimilarity that is symmetric (insensitive to the ordering within each pair of points) but (unlike a distance) may not satisfy the triangle inequality. Depending on the method, this dissimilarity function can be extended in several different ways to pairs of clusters; for instance, in the single-linkage clustering method, the distance between two clusters is defined to be the minimum distance between any two points from each cluster. Given this distance between clusters, a hierarchical clustering may be defined by a greedy algorithm that initially places each point in its own single-point cluster and then repeatedly merges the closest pair of clusters.[6]

However, known methods for repeatedly finding the closest pair of clusters in a dynamic set of clusters either require super-linear space to maintain a data structure that can find closest pairs quickly, or they take greater than linear time to find each closest pair.[7], [8]. The nearest-neighbor chain algorithm uses a smaller amount of time and space than the greedy algorithm by merging pairs of clusters in a different order. However, for many types of clustering problem, it can be guaranteed to come up with the same hierarchical clustering as the greedy algorithm despite the different merge order.

VI. NNC Algorithm

Intuitively, the nearest neighbor chain algorithm repeatedly follows a chain of clusters $A \rightarrow B \rightarrow C \rightarrow \ldots$ where each cluster is the nearest neighbor of the previous one, until reaching a pair of clusters that are mutual nearest neighbors.

More formally, the algorithm performs the following steps:

- Initialize the set of active clusters to consist of n one-point clusters, one for each input point.
- Let S be a stack data structure, initially empty, the elements of which will be active clusters.
- While there is more than one cluster in the set of clusters:
- If S is empty, choose an active cluster arbitrarily and push it onto S.
- Let C be the active cluster on the 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 S, it must be the immediate predecessor of C. Pop both clusters from S and merge them.
- Otherwise, if D is not already in S, push it onto S.

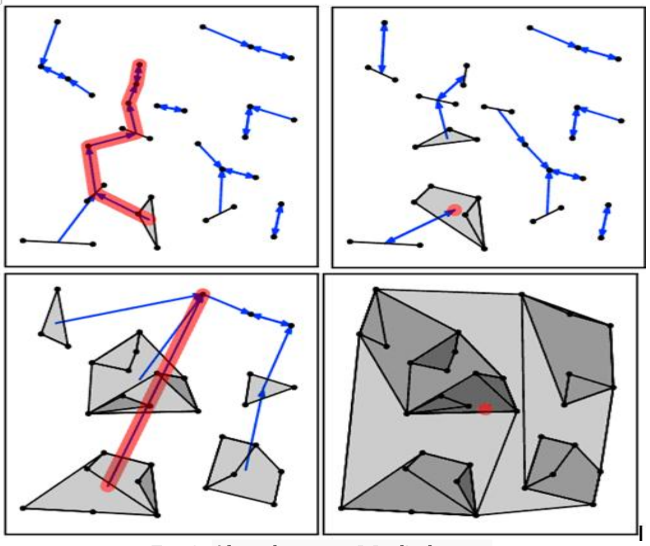

Fig. 2. Algorithm using Ward's distance

From figure 2 black dots are points, gray regions are larger clusters, blue arrows point to nearest neighbors, and the red bar indicates the current chain. For visual simplicity, when a merge leaves the chain empty, it continues with the recently merged cluster.

If there may be multiple equal nearest neighbors to a cluster, the algorithm requires a consistent tie-breaking rule: for instance, in this case, the nearest neighbor may be chosen, among the clusters at equal minimum distance from C, by numbering the clusters arbitrarily and choosing the one with the smallest index. The correctness of this algorithm relies on a property of its distance function called reducibility, connection with an earlier clustering method that used mutual nearest neighbor pairs but not chains of nearest neighbors. [4] A distance function d on clusters is defined to be reducible if, for every three clusters A, B and C in the greedy hierarchical clustering such that A and B are mutual nearest neighbors, the following inequality holds:

$$d(A \cup B, C) \geq \min(d(A,C), d(B,C)) \quad (1)$$

If a distance function has the reducibility property, then merging two clusters C and D can only cause the nearest neighbor of E to change if that nearest neighbor was one of C and D. This has two important consequences for the nearest neighbor chain algorithm: first, it can be shown using this property that, at each step of the algorithm, the clusters on the stack S form a valid chain of nearest neighbors, because whenever a nearest neighbor becomes invalidated it is immediately removed from the stack.

Second, and even more importantly, it follows from this property that, if two clusters C and D both belong to the greedy hierarchical clustering, and are mutual nearest neighbors at any point in time, then they will be merged by the greedy clustering, for they must remain mutual nearest neighbors until they are merged. It follows that each mutual nearest neighbor pair found by the nearest neighbor chain algorithm is also a pair of clusters found by the greedy algorithm, and therefore that the nearest neighbor chain algorithm computes exactly the same clustering (although in a different order) as the greedy algorithm.

VII. Application for Specific Clustering Distance

There are three different distance measure application

A. Ward's method
B. Complete linkage and average distance
C. Single linkage
D. Centroid Distance

A. Ward's Method

Ward's method is an agglomerative clustering method in which the dissimilarity between two clusters A and B is measured by the amount by which merging the two clusters into a single larger cluster would increase the average squared distance of a point to its cluster centroid. [9] That is,

$$d(A,B) = \sum_{x\in A, y\in B} \frac{d^2(x,y)}{|A|+|B|} - \sum_{x,y\in A} \frac{d^2(x,y)}{|A|} - \sum_{x,y\in B} \frac{d^2(x,y)}{|B|}. \quad (2)$$

Expressed in terms of the centroid CA and cardinality nA of the two clusters, it has the simpler formula

$$d(A,B) = \frac{d^2(c_a, c_b)}{1/n_A + 1/n_b} \quad (3)$$

allowing it to be computed in constant time per distance calculation. Although highly sensitive to outliers, Ward's method is the most popular variation of agglomerative clustering both because of the round shape of the clusters it typically forms and because of its principled definition as the clustering that at each step has the smallest variance within its clusters.[10] Alternatively, this distance can be seen as the difference in k-means cost between the new cluster and the two old clusters.

Ward's distance is also reducible, as can be seen more easily from a different formula of Lance–Williams type for calculating the distance of a merged cluster from the distances of the clusters it was merged from [9], [11]:

$$d(A \cup B, C) = \frac{n_A + n_C}{n_A + n_B + n_C} d(A,C) + \frac{n_B + n_C}{n_A + n_B + n_C} d(B,C) - \frac{n_C}{n_A + n_B + n_C} d(A,B) \quad (4)$$

If d(A, B) is the smallest of the three distances on the right hand side (as would necessarily be true if A and B are mutual nearest-neighbors) then the negative contribution from its term is canceled by the nC coefficient of one of the two other terms, leaving a positive value added to the weighted average of the other two distances. Therefore, the combined distance is always at least as large as the minimum of d(A, C) and d(B, C), meeting the definition of reducibility.

Therefore, the nearest-neighbor chain algorithm using Ward's distance calculates exactly the same clustering as the standard greedy algorithm. For n points in a Euclidean space of constant dimension, it takes time O(n2) and space O(n) [1].

B. Complete Linkage and Average Distance

Complete-linkage or furthest-neighbor clustering is a form of agglomerative clustering that uses the maximum distance between any two points from the two clusters as the dissimilarity, and similarly average-distance clustering uses the average pairwise distance. Like Ward's distance, these forms of clustering obey a formula of Lance-Williams type: in complete linkage, the distance $d(A \cup B, C)$ is the average of the distances d(A, C) and d(B, C) plus a positive correction term, while for average distance it is just a weighted average of the distances d(A, C) and d(B,C). [9], [11] Thus, in both of these cases, the distance is reducible.

Unlike Ward's method, these two forms of clustering do not have a constant-time method for computing distances between pairs of clusters. Instead it is possible to maintain an array of distances between all pairs of clusters, using the Lance–Williams formula to update the array as pairs of clusters are merged, in time and space O(n2). The nearest-neighbor chain algorithm may be used in conjunction with this array of distances to find the same clustering as the greedy algorithm for these cases in total time and space O(n2). The same O(n2) time and space bounds can also be achieved by a different and more general technique that overlays a quadtree-based priority queue data structure on top of the distance matrix and uses it to perform the standard greedy clustering algorithm, avoiding the need for reducibility, [7] but the nearest-neighbor chain algorithm matches its time and space bounds while using simpler data structures [12].

C. Single Linkage

In single-linkage or nearest-neighbor clustering, the oldest form of agglomerative hierarchical clustering, [11] the dissimilarity between clusters is measured as the minimum distance between any two points from the two clusters. With this dissimilarity,

$$d(A \cup B, C) = \min(d(A, C), d(B, C)), \tag{5}$$

meeting as an equality rather than an inequality the requirement of reducibility. (Single-linkage also obeys a Lance–Williams formula, [9] [11] but with a negative coefficient from which it is more difficult to prove reducibility.)

As with complete linkage and average distance, the difficulty of calculating cluster distances causes the nearest-neighbor chain algorithm to take time and space O(n2) to compute the single-linkage clustering. However, the single-linkage clustering can be found more efficiently by an alternative algorithm that computes the minimum spanning treeof the input distances using Prim's algorithm (with an unsorted list of vertices and their priorities in place of the usual priority queue), and then sorts the minimum spanning tree edges and uses this sorted list to guide the merger of pairs of clusters. This alternative method would take time O(n2) and space O(n), matching the best bounds that could be achieved with the nearest-neighbor chain algorithm for distances with constant-time calculations [2].

D. Centroid Method

Another distance measure commonly used in agglomerative clustering is the distance between the centroid of pairs of clusters, also known as the weighted group method. [9], [11] It can be calculated easily in constant time per distance calculation. However, it is not reducible: for instance, if the input forms the set of three points of an equilateral triangle, merging two of these points into a larger cluster causes the inter-cluster distance to decrease, a violation of reducibility. Therefore, the nearest-neighbor chain algorithm will not necessarily find the same clustering as the greedy algorithm. A different algorithm by Day and Edelsbrunner can be used to find the clustering in O(n2) time for this distance measure [8].

VIII. Data Storage for K - Days Itineraries

Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

A. Low-Cost Storage and Active Data Archive

The modest cost of commodity hardware makes Hadoop useful for storing and combining data such as transactional, social media, sensor, machine, scientific, click streams, etc. The low-cost storage lets you keep information that is not deemed currently critical but that you might want to analyze later.

B. Staging Area for a Data Warehouse and Analytic Store

One of the most prevalent uses is to stage large amounts of raw data for loading into an enterprise data warehouse (EDW) or an analytical store for activities such as advanced analytic, query and reporting, etc. Organizations are looking at Hadoop to handle new types of data (e.g., unstructured), as well as to offload some historical data from their enterprise data warehouses.

C. Data Lake

Hadoop is often used to store large amounts of data without the constraints introduced by schemas commonly found in the SQL-based world. It is used as a low-cost compute-cycle platform that supports processing ETL and data quality jobs in parallel using hand-coded or commercial data management technologies. Refined results can then be passed to other systems (e.g., EDWs, analytic marts) as needed.

D. Sandbox for Discovery and Analysis

Because Hadoop was designed to deal with volumes of data in a variety of shapes and forms, it can run analytical algorithms. Big data analytic on Hadoop can help your organization operate more efficiently, uncover new opportunities and derive next-level competitive advantage. The sandbox approach provides an opportunity to innovate with minimal investment.

E. Recommendation Systems

One of the most popular analytical uses by some of Hadoop's largest adopters is for web-based recommendation systems. Facebook – people you may know. Linked-in – jobs you may be interested in. Netflix, eBay, Hulu – items you may be interested in. These systems analyze huge amounts of data in real time to quickly predict preferences before customers leave the web page.

IX. Related Work

Most existing work on itinerary generation take a two-step scheme. They first adopt the data mining algorithms to discover the users' traveling patterns from their published images, geolocations and events [11-13]. Based on the relationships of those historical data, new itineraries are generated and recommended to the users. This scheme leverages the user data to retrieve POI's and organize the POI's into itinerary, which is based on a different application scenario to ours. We help the traveler to get easy and successful trip with there expectations.

In our system we use Nearest Neighbor algorithm, to find shortest distance between POI's for multi-day itinerary. It give more information about there POI's, and it is budget based system. Some travelers like to select high level accommodation, some medium and low. Any how it will select according to its budget and give good packages. System will collect POI's and geographical information about POI's from spatial Database and connected with Google map, NNA will help to find nearest path for travel the place which they choose and it save the time, the it is easy to travel more place. From the above topic we would like to find the shortest distance and form the cluster. Finally it produce the k-day itinerary and stored it into Hadoop. It will save the storage space from our database and it is easy to retrieve in the next time. Some interactive search algorithms, are proposed in recent years. These algorithms still focus on optimal single day itinerary planning. To reduce the computation overhead and improve the quality of generated itineraries, users' feedback's are integrated into the search algorithm. Here we focusing on multi-day itinerary saving time, traveling more place, get more information without the help of guide, budget based accommodation save storage space etc. So this system will provide more facilities to customer.

To the best of our knowledge, no previous work studied the problem of generating in multidays itinerary with budget base system with new algorithm and new storage application . This system is more challenging than the previous system, The multi-day itinerary, as shown in this paper, can be reduced the traveler problem. The beauty of our approach is that after the transformation, the itinerary planning problem is reduced to the weighted set-packing problem, which has approximate solutions under some constraints.

X. Conclusion

In this paper, we present an automatic itinerary generation service with budget based for the backpack travelers. The service creates a best itinerary based on the user's preference. To search for an optimal solution, we use Nearest Neighbor Algorithm with some application for finding shortest path and optimize trip in successful manner. In future we use more optimize technique for better itineraries with the help of smart applications.

References

[1] F. Murtagh, "Clustering in massive data sets," in J. Abello, P. M. Pardalos, M. G. C. Resende, *Handbook of massive data sets*, Massive Computing, Springer, pp. 513–516, 2002.

[2] J. C. Gower and G. J. S. Ross, "Minimum spanning trees and single linkage cluster analysis," *Journal of the Royal Statistical Society, Series C*, vol. 18, issue 1, pp. 54–64, 1969.

[3] M. Perumal, "A survey on geographical information system, spatial datamining and ontology," Conference Paper, 2014.

[4] M. Bruynooghe, "Méthodes nouvelles en classification automatique de données taxinomiqes nombreuses," *Statistique et Analyse des Données*, vol. 3, pp. 24–42, 1977.

[5] S. D. Antelson and K. Sreekandan, "Budget based automatic itinerary planning for traveling services," *International Journal of Research in Engineering and Bioscience*, vol. 2, issue 3, pp. 37-43, 2014.

[6] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *The Computer Journal*, vol. 26, issue 4, pp. 354–359, 1983.

[7] D. Eppstein, "Fast hierarchical clustering and other applications of dynamic closest pairs," *Journal of Experimental Algorithmics (ACM)*, vol. 5, issue 1, pp. 1–23, 2000.

[8] W. H. E. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, vol. 1, issue 1, pp. 7–24, 1984.

[9] B. Mirkin, "Mathematical classification and clustering," Nonconvex Optimization and its Applications 11, Dordrecht: Kluwer Academic Publishers, pp. 140–144, 1996.

[10] S. Tuffery, "9.10 Agglomerative hierarchical clustering," *Data Mining and Statistics for Decision Making*, Wiley Series in Computational Statistics, pp. 253–261, 2011.

[11] G. N. Lance and W. T. Williams, "A general theory of classificatory sorting strategies. I. Hierarchical systems," *The Computer Journal*, vol. 9, issue 4, pp. 373–380, 1967.

[12] I. Gronau and S. Moran, "Optimal implementations of UPGMA and other common clustering algorithms," *Information Processing Letters*, vol. 104, issue 6, pp. 205–210, 2007.