EFFICIENT MULTIUSER ITINERARY PLANNING FOR TRAVELLING SERVICES USING FKM-CLUSTERING ALGORITHM

R.Rajeswari[1], J. Mannar Mannan[2]

[1] III-M.tech(IT) , Department of Information technology, Regional Centre, Anna University, Coimbatore, India
[2] Teaching Assistant, Department of Information technology, Regional Centre, Anna University, Coimbatore, India

---***---

Abstract - *This paper proposes, previous efforts address the problem by providing an automatic itinerary planning service that organizes the points-of-interests (POIs) into a customized itinerary. The search space of all possible itineraries is too costly to fully visit, to simplify the complexity, most work assume that* ***user's trip is limited to some important POIs and will*** *complete within one day. To address the above Problem, design a more general automatic itinerary planning service, this generates multiday itineraries for the users. All POIs are considered and ranked based on* ***the user's preference. Since the many users are*** *planning for a trip with various requirements then their search complexity is increased. To overcome this limitation using the concept of grouping or clustering* ***the users based on user's requirement similarity. The*** *novel modified clustering is using for grouping the user according to their requirement similarity. The membership values are defined for clustering process* ***and it's based on the users. It is*** *used to reduce the search complexity as well as time complexity of the system.*

Key Words: POI, Grouping Clustering, Itinerary

1. Introduction

Transportation systems have played an important role in real applications such as the traffic control, location-based services (LBS), trip planning, and geographical data management. One typical example of such systems is the European Traffic Message Channel (TMC), which has been operated in many European countries, North America, and Australia. With the increasing interest in the management of transportation systems, recently, the spatial road network has received much attention from the database community.

Specifically, a spatial road network can be modeled by a large graph in a 2-dimensional geographical space whose edges correspond to road segments, and are associated with weights related to the traffic information (e.g., road-network distance, speed of vehicles, or the delay time). Over such road networks, a wide spectrum of practical problems have been extensively studied, including range queries, *k*-nearest neighbor (*k*NN) queries, reverse nearest neighbor queries, shortest path queries, multi-source skyline queries and so on.

Traveling market is divided into two parts. For casual customers, they will pick a package from local travel agents. The package, in fact, represents a pre-generated itinerary. The agency will help the customer book the hotels, arrange the transportations, and preorder the tickets of museums/parks. It prevents the customers from constructing their personalized itineraries, which is very time consuming and inefficient. For instance, a four-day package to Hong Kong provided by a Singapore agency is covers the most popular POIs for a first-time traveler. Although the travel agencies provide efficient and convenient services, for experienced travelers, the itineraries provided by the travel agents lack customization and cannot satisfy individual requirements. Some interested POIs are missing in the itineraries and the packages are too expensive for a backpack traveler. Therefore, they have to plan their trips in every detail, such as selecting the hotels, picking POIs for visiting, and contacting the car rental service.

First, current planning algorithms only consider a single day's trip, while in real cases, most users will schedule an n-day itinerary. Generating an n-day itinerary is more complex than generating a single day one. It is not equal to constructing single-day itineraries and combining them together, as a POI can only appear once in the itinerary. It is tricky to group POIs into different days. One possible solution is to exploit the geo-locations, for example, nearby POIs are put in the same day's itinerary. Alternatively, it can also rank POIs by their importance and use a priority queue to schedule the trip.

Second, the travel agents tend to favor the popular POIs. Even for a city with a large number of POIs, the travel agents always provide the same set of trip plans, composed with top POIs. However, those popular POIs may not be attractive for the users, who have visited the city for several times or have limited time budget. It is impossible for a user to get his personal trip plan. The travel agent's service cannot cover the whole POI set, leading to few choices for the users. In our algorithm, we adopt a different approach by giving high priorities to the selected POIs and generating a customized trip plan on the fly.

Third, suppose if have N available POIs and there are m POIs in each single day's itinerary averagely and then end up with N! / (N-m)! m! Candidate itineraries. It is costly to evaluate the benefit of every itinerary and select the optimal one. Therefore, in [9] and [3], some heuristic approaches are adopted to simplify the computation. However, the heuristic approaches are based on some assumptions (e.g., popular POIs are selected with a higher probability).They only provides limited number of itineraries and is not optimized for the backpack traveler, who plans to have a unique journey with his own customized itinerary. Last but not the least, handling new emerging POIs was tricky in previous approaches. The model needs to be rebuilt to evaluate the benefit of including the new POIs into the itinerary. For systems based on the users' feedback [3], they need to collect the comments for the new POIs from the users, which is very time-consuming.

In the preprocessing, POIs are organized into an undirected graph, G. The distance of two POIs is evaluated by Google Map's APIs. Different ranking functions are applied to different types of POIs. The automatic itinerary planning service needs to return an itinerary with the highest ranking. Searching the optimal itinerary can be transformed into the team orienteering problem (TOP), which is an NP-complete problem without polynomial approximations [4]. Therefore, a two stage scheme is applied.

In the preprocessing stage, iterate all candidate single-day itineraries using a parallel processing framework, MapReduce [4]. The results are maintained in the distributed file system (DFS) and an inverted index is built for efficient itinerary retrieval. To construct a multiday itinerary, I need to selectively combine the single itineraries. The preprocessing stage, in fact, transforms the TOP into a set-packing problem [6], which has well-known approximated algorithms. In the online stage, design an approximate algorithm to generate the optimal itineraries.

2. Related Research

2.1. Automatic Construction of Travel Itineraries[3]

To construct itineraries following a two-step approach. Given a city, here first extract photo streams of individual users. Each photo stream provides estimates on where the user was, how long he stayed at each place, and what was the transit time between places. In the second step, here aggregate all user photo streams into a POI graph. Itineraries are then automatically constructed from the graph based on the popularity of the POIs and subject to the user's time and destination constraints.

(1) Introduce a novel end-to-end approach that starts with the analysis of latent information.
(2) Apply a pipeline of multiple heuristics that together extract reliable granular evidence of individual tourists' trips to a destination from Flickr photos.
(3) Aggregate the individual trips to form a graph representing collective touristic behavior, and adapt a solution of the Orienteering problem to efficiently.

2.2 Greedy Local Improvement and Weighted Set Packing Approximation[6]

Here present an approximation algorithm for the weighted k-set packing problem that combines the two paradigms by starting with an initial greedy solution and then repeatedly choosing the best possible local improvement. The algorithm has a performance ratio of 2(k + 1)/3, which here show is asymptotically tight. Here present a natural heuristic BestImp that combines the greedy and local search paradigms by starting with an initial greedy solution and then repeatedly choosing the best possible local improvement. Its performance ratio is at most 2(k + 1)/3, which is asymptotically tight. AnyImp that also combines the greedy and local search paradigms. The difference is that AnyImp just looks for an improvement that leads to a gain bigger than a specified threshold, instead of looking for the best improvement.

The proof technique here use to obtain upper bounds on the performance ratio can be understood in a simpler setting with AnyImp. The performance ratio of AnyImp as a function of the threshold, which for the best choice of a threshold is at most (4k + 2)/5. The results hold equally for the slightly more general problem of approximating maximum weight independent sets in k + 1-claw free graphs.

2.3 Interactive Itinerary Planning[8]

Adopt an interactive process where the user provides feedback on POIs suggested by our itinerary planning system and the system leverages those feedbacks to suggest the next batch of POIs, as well as to recommend the best itineraries so far. The process repeats until the user is satisfied. In other words, instead of asking the user to examine all the POIs before deciding on the itinerary, our goal is to ask the user to examine only a subset of those POIs in multiple steps, each with a small number of increasingly relevant POIs, thereby reducing the overall efforts required on the user to construct the itinerary.

(1) It starts with a user providing a time budget and a starting point of the itinerary.
(2) At each step, the system presents the user with a small fixed number of POIs that are most probably liked by the user, based on feedback provided by the user so far.
(3) The system also recommends highly ranked itineraries to the user based on the feedback;

(4) The user provides feedback on suggested POIs to indicate process continues.
(5) The user can also choose to pick one of the recommended itineraries, at point, the process stops.

2.4 Using Flickr Geotags to Predict User Travel Behavior[4]

Here using a method to predict a user's favorite's locations in a city, based on his Flickr geotags in other cities. Here define a similarity between the geotag distributions of two users based on a Gaussian kernel convolution. The geotags of the most similar users are then combined to rerank the popular locations in the target city personalized for this user. Here show that this method can give personalized travel recommendations for users with a clear preference for a specific type of landmark.

Here use to predict a user's favorites locations in a city based on his travel behavior in previously visited cities. On social photo sharing websites like www.flickr.com people can annotate their photos, including the geographical location where the photo was made. Also, increasingly more cameras and smart phones are automatically storing the GPS coordinates when a photo is made. These *geo tags* give an accurate indication of the user's preferred landmarks. Based on a set of collected geotags, define a measure to identify similar users in previously visited cities. Then we aggregate these user opinions in a different city to obtain a personalized travel recommendation for the target user.

2.5 Metaheuristics for the Team Orienteering Problems [1]

Investigate the VRP with profit which is the extension to the case of multiple tours of the most studied TSP with profits, namely the OP. In the OP, given a set of potential customers with associated profit and given the distances between pairs of customers, the objective is to find the subset of customers for which the collected profit is maximum, given a constraint on the total length of the tour. The OP is also called the Selective Traveling Salesman Problem (STSP). The name orienteering comes from an outdoor sport usually played on mountains or forest areas. Given a specified set of points, each competitor, with the help of a map and a compass, has to visit as many points as possible within a specified time limit.

The competitor starts at a given point and has to return to the same point. The extension of the Orienteering Problem to the case of multiple tours is known as the Team Orienteering Problem (TOP). Among the metaheuristics proposed for the solution of combinatorial optimization problems, tabu search has been shown to be very effective for vehicle routing problems. Another interesting metaheuristic is the variable neighborhood search.

3. Architecture

3.1 System Architecture

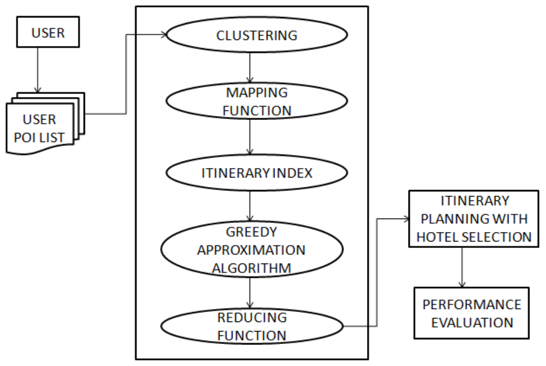

Fig -1 System Overview

Initially, the place of interests that user wants to visit will be gathered. After gathering the location information from the users, in the proposed work, k-means clustering will be applied to group the nearest locations together. Then map-reduce process will be applied on the data's in order to make them process in the efficient manner. The location will be selected in order based on the indexes assigned. Then greedy approximation algorithm to create the k-day itinerary plan. After creating the plan, the hotel selection will be done in order to effectively handle the user selection.

3.2 Hadoop File System Architecture

Hadoop consists of the Hadoop Common package, which provides files system and OS level abstractions, a MapReduce engine and the Hadoop Distributed File System (HDFS). The Hadoop Common package contains the necessary Java Archive (JAR) files and scripts needed to start Hadoop. The package also provides source code, documentation and a contribution section that includes projects from the Hadoop Community.

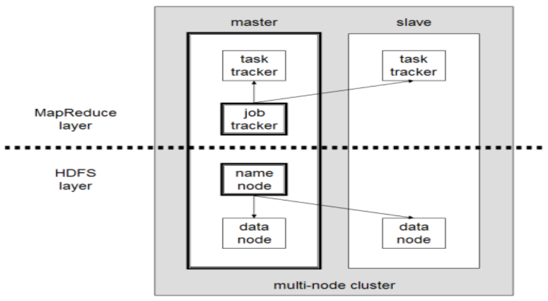

Fig- 2 Hadoop System Overview

HDFS uses this method when replicating data to try to keep different copies of the data on different racks.

The goal is to reduce the impact of a rack power outage or switch failure, so that even if these events occur, the data may still be readable. A small Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a Job Tracker, Task Tracker, Name Node and Data Node. A slave or worker node acts as both a Data Node and Task Tracker, though it is possible to have data-only worker nodes and compute-only worker nodes. These are normally used only in nonstandard applications. Hadoop requires Java Runtime Environment (JRE) 1.6 or higher. The standard start-up and shutdown scrip ts require Secure Shell(ssh) to be set up between nodes in the cluster.

4. Implementation and Results

4.1 Clustering

Clustering is the process of grouping data objects into a set of disjoint classes, called clusters, so that objects within a class have high similarity to each other, while objects in separate classes are more dissimilar. Clustering is the assignment of objects into groups (called clusters). Clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning ,data mining, pattern recognition, image analysis, and bioinformatics.

Partitional Clustering

It attempts to directly decompose the data set into a set of disjoint clusters. The criterion function that the clustering algorithm tries to minimize may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Typically the global criteria involve minimizing some measure of dissimilarity in the samples within each cluster, while maximizing the dissimilarity of different clusters.

K-means Clustering Algorithm is a simple and fast clustering method, which has been popular used. So we apply it to group the items with some adjustments. The difference is, apply the fuzzy set theory to represent the affiliation between an object and a cluster. Firstly, users are grouped into a given number of clusters. After completion of grouping, the possibility of one object (here one object means one item) belonging to a certain cluster is calculated as follows.

$$Pro(j, k) = 1 - \frac{CS(j,k)}{MaxCS(i,k)}$$

where ,

Pro(*j*, *k*) means the possibility of object *j* belonging to the cluster *k*; The *CS*(*j*, *k*) means the counter-similarity between the object *j* and the cluster *k*, which is calculated based on the Cosine method; *MaxCS*(*i*, *k*) means the maximum counter-similarity between an object and cluster *k*.

4.2. Algorithm

Algorithm 1 : Fuzzy K-means Clustering

```
Input: the number of clusters k and user's requirements attribute features.

(1) Initialize the parameters, and membership between objects and clusters;
(2) Repeat (a) and (b) until global cost function has small change;
        a) Recompute the mean value of each cluster.
        b) Recompute the membership of each object.
(3) Return the membership.
```

In our fuzzy k-means(FKM) algorithm, the fuzzy membership in a cluster is only assigned at the last step. It seems to un essentially represent the fuzzy memberships of objects. So the FKM algorithm is also applied to group the requirements, in which each object is assigned a fuzzy membership during each iteration. The global cost function, membership between an object and a cluster, and the mean value of one cluster

4.3 Module Description

Single-Day Itinerary

The preprocessing includes two steps. In the first step, a set of MapReduce jobs are submitted to produce all possible single-day itineraries. The basic idea of transformation is to iterate all possible single-day itineraries. In the first job, we generate |P| initial itineraries for the POI set P. Each initial itinerary only consists of one POI. MapReduce job tries to add one more POI to the itineraries. If no more single-day itineraries can be generated, the process terminates. In current implementation, we allow maximally m MapReduce jobs in the transformation process to reduce the overheads.

Itinerary Index

To efficiently locate the single-day itineraries, an inverted index is built. The key is the POI and the values are all itineraries involving the POI. By scanning the index, we can retrieve all the itineraries. we create an index file for each POI in the DFS. The file includes all single itineraries involving the POI, which are sorted based on their weights. In this way, split the itineraries of a POI into n groups and each group can be efficiently sorted in the memory. However, it is not necessary to merge the files, as the files are partitioned based on the weights.

Algorithm 2: Mapping Function

```
map(Object key, Text value, Context context).

//value: single-day itinerary
1: Itinerary it = parse(value)
2: for i = 0 to it.POISize() do
3: intnextPOI = it.getNext(i)
4: Key key=new CompKey(nextPOI, it.weight/ bucketSize )
5: context.collect(key, it)
```

Algorithm 3 : Reducing Function

```
reduce(Key key, Iterablevalues,Context context).

1: CompositeKeyck = key, Set s = Ø;
2: for Itinerary it: values do
3: s.add(it)
4: sort(s)
5: DFSFile f = new DFSFile(ck:first + "-"+ck:second)
6: f.write(s)
```

Algorithms 2 and 3 show the process. The mappers load the single-day itinerary and generate key-value pairs for each involved POI. The reducers collect all itineraries for a specific POI and sort them based on the weights before creating the index file. In our system, the size of the index file may vary a lot. Some POI may have an extremely large index file, due to its popularity and short visit time. In reducers, those POIs may result in the exception of memory overflow in the sorting process. To address this problem, in the map phase, instead of using the POI as the key, we generate the composite key by combining the POI and the itinerary weight.

Greedy-Based Approximation Algorithm

After the itinerary indexes are constructed, the user request (Sp; k) can be processed by selecting k best itineraries from the indexes. Namely, the problem of generating optimal k-day itinerary is transformed into a weighted set-packing problem. In these module having two phases. There are,

- Initialization phase
- Adjustment phase

Hotel Selection

In fact, hotels can be considered as a special type of POIs. It must appear as the last POI in the itinerary. We need to calculate the traveling time from other POIs to the hotel POIs. Hotel POIs do not incur access cost and their weights are set as users rankings for the hotels. Based on the user's preference, they have two processing strategies.

- Multiple Hotels
- sSingle Hotel

5. Conclusion

Present an automatic itinerary generation service for the backpack traveler. The service creates a customized multiday itinerary based on the multiple user's preference. To search for the optimal solution, a two-stage scheme is adopted. In the preprocessing stage, iterate and index the candidate single-day itineraries using mapreduce framework. After the preprocessing stage, the TOP is transformed into the weighted set-packing problem, which has efficient approximate algorithms. In the next stage, simulate the approximate algorithm for the set-packing problem. The algorithm follows the initialization- adjustment model and can generate a result.

6. References

1. Archetti C., Hertz A. and Speranza M.G., (2007), 'Metaheuristics for the Team Orienteering Problem', J. Heuristics, Vol. 13, pp. 49-76.

2. Chao I.M., Golden B.L. and Wasil E.A., (1996), 'The Team Orienteering Problem', European J. Operational Research, Vol. 88, no. 3, pp. 464-474.

3. Choudhury M.D., Feldman M., Amer-Yahia S., Golbandi N., Lempel R. and Yu C., (2010), 'Automatic Construction of Travel Itineraries Using Social Breadcrumbs', Proc. 21st ACM Conf. Hypertext and Hypermedia (HT), pp. 35-44.

4. Clements M., Serdyukov P., de Vries A.P. and Reinders M.J., (2010), 'Using Flickr Geotags to Predict User Travel Behaviour', Proc.33rd Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR).

5. Dean J. and Ghemawat S., (2010), 'MapReduce: A Flexible Data Processing Tool', Comm. ACM, Vol. 53, pp. 72-77.

6. Halldo´rsson M.M. and Chandra B., (2001), 'Greedy Local Improvement and Weighted Set Packing Approximation', J. Algorithms, Vol. 39, pp. 223-240.

7. Laporte G., (1992), 'The Travelling Salesman Problem: An Overview of Exact and Approximate Algorithms', European J. Operational Research, Vol. 59, no. 2, pp. 231-247.

8. Roy S.B., Das G., Amer-Yahia S. and Yu C., (2011), 'Interactive Itinerary Planning', Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE), pp. 15-26.