IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 3, Ver. III (May - June 2017), PP 29-32
www.iosrjournals.org

Backward Approach Development for Route Adaptive Mobile Successive Recommendation

Mrs.S.Nithya[1], S.Visakh[2], J.Danny Gollner[3], C.Maria Sylvia Ashley[4]
[1](Assistant Professor, Dept of Computer Science & Engineering, PRIST University, Tamilnadu, India)
[2](Research Scholar, Dept of Computer Science & Engineering, PRIST University, Tamilnadu, India)
[3](Research Scholar, Dept of Computer Science & Engineering, PRIST University, Tamilnadu, India)
[4](Research Scholar, Dept of Computer Science & Engineering, PRIST University, Tamilnadu, India)

***Abstract**: The Backward Approach Development is the pre-planning requisite for the traveller's.So they are more exposed to our taxi service in and out. In addition, accurate query results with up-to-date travel times, price etc.we propose a novel dynamic programming based method to solve the mobile sequential recommendation problem with the new algorithm, named UniBic, outperformed all previous biclustering algorithms in terms of commonly used evaluation scenarios except for BicSPAM on narrow biclusters Simultaneously, the process of sequence generation to reduce the search space of the potential sequences effectively. Moreover, our method can handle the problem of optimal route search with a maximum cruising distance or a destination constraint. Experimental results on real and synthetic data sets show that both the pruning ability and the efficiency of our method surpass the state-of-the-art methods. Our techniques can therefore be effectively employed to address the problem of mobile sequential recommendation with many pick-up points in real-world applications*

I. Introduction

The utilization of wireless sensors such as GPS, Wi-Fi, BlueTooth and RFID, we can easily get access to the location trace data for a large number of moving objects. Finding useful knowledge from these trajectory data will provide strong support for the real-time decision and intelligence services in the related applications . The problem of reducing cruising costs for taxicabs is a typical example . An unloaded taxi cruising on the road not only leads to waste of fuel and time, but also may result in traffic congestion. However, there exist many pick-up hotspots in taxi trajectories of those high-yield taxi drivers, which can be effectively leveraged to guide new drivers to pick up passengers in a more economical and energy-efficient way. Therefore, highly efficient mobile pattern mining and recommendation algorithms can significantly improve business performance and reduce the energy consumption.This problem possesses considerable theoretical significance and practical value . In previous studies, Yu Li and Man Lung Yiu have proposed a novel problem of mobile sequential recommendation , which is to suggest a route connecting a series of pick-up points for an empty cab so that the driver is more likely to get passengers with less travel cost starting from his current position. It is a challenging task, because we need to enumerate and compare all possible routes derived from the given set of pick-up points which involves a rather high computational complexity.

To solve the MSR problem, they proposed a function of potential travel distance (PTD) for evaluating the cost of a driving route. Essentially, the PTD value of a suggested route is the expectation of the cruising distance for an empty cab before it successfully gets new passengers when it travels along the route. To reduce the computational cost, two effective potential sequence Unibic algorithms LCP and SkyRoute, which are based on the monotone property of the PTD function, have been proposed in . However, the time and space complexities of these two algorithms both grow exponentially with the number of pick-up points and the lengths of the suggested driving routes, so they can only be applied to the driving route recommendation with a limited length constraint and a small number of pick-up points. In this project, we propose a novel and efficient solution to the MSR problem.

Our method includes an offline stage and an online stage. The offline stage effectively Unibic the search space and generates a small set of candidate sequences. The online stage is aimed at obtaining the optimal driving route given the current position of an unloaded taxi as the starting point. Specifically, for the offline pre-computation, we have examined the nature of the PTD function and found that it satisfies the iterative calculation property. This property allows us to incrementally construct a potential driving route backward from the terminal point to the starting point. Furthermore, we have also found that a set of potential sequences with the same length and the same source point satisfy the incremental and batch Unibic properties. To this end, we designed a mobile sequential recommendation method taking full advantage of the iterative nature of the PTD function. It incrementally generates potential sequences and removes a lot of unfertile search space, which greatly enhances the efficiency and reduces the memory consumption of our method. Among the

generated potential sequences with the same length, we can further prune a large number of potential sequences that fail to form the optimal route by using a batch Unibic policy. It can dramatically reduce the number of the remaining sequence candidates. In our method, the original MSR problem can be generalized effortlessly to handle the case when a travel distance or a destination constraint is imposed.

II. Background

In this section, we first introduce the MSR problem and then describe the previous works.In recent years, intelligent transportation systems and trajectory data mining have aroused widespread attentions. Mobile navigation and route recommendation have become a hot topic in this research field. The MSR problem presented by Ge et al is rather different from the traditional problems such as Shortest-Path problem Traveling-Salesman problem and Vehicle-Scheduling problem

Because for the shortest path computation problem, the source and destination nodes of an object are known in advance. However, for MSR problem, both of them are unknown. The traditional Traveling-Salesman Problem (TSP) gets a shortest path that includes all N locations while MSR problem is to find a path that consists of a subset of given N locations. In addition, the traditional Vehicle-Scheduling problem needs to determine a set of duties in advance while the pick-up routes among several locations is uncertain for the MSR problem.

Lu et al. introduced a problem of finding optimal trip route with time constraint. They also proposed an efficient trip planning method considering the current position of a user. However, their method uses the score of attractions to measure the preference of a route.

MSR problem

A set of potential pickup points C = {c1,c2,c3,…..cN};
A probability set P = { P(c1),P(c2),…..P(cN)};
A potential sequence set R={r1,r2,…rM};
The position c0 of cab which needs the service;
Recommending a optimal driving route d= (c0,r),s.t. min(r->R)F(c0,r,P(r)).

Here we set the routing between cluster heads and users and data forwarded to server. The requests of users are sent to the cluster heads. In this process, more data will be generated based on cluster head position. We maintain the request of users properly and set up as data moving to server.

We consider: pickup point's pp1, pp2, pp3…ppN; Sequence r; Sequence length R; Each group of cluster maintains proper communication with pickup point location

III. Proposed System

To address the computational challenge of the generalize MSR problem, we first identify the iterative property of the PTD function,which makes the incremental generation of the potential sequences possible and then propose the UNIBIC algorithm, which uses the iterative property to efficiently reduce the search space.

1. UNIBIC ALGORITHM

In this section, we present our novel biclustering algorithm, which is capable of discovering all the significant trend-preserving biclusters hidden in a data matrix. The basic idea behind the algorithm comes from the following observations:

1) There exists a column permutation of an order-preserving bicluster such that the entries of each permuted row within the bicluster are increasingly arranged.

2) The key to biclustering is the accurate prediction of the columns of each to-be-identified bicluster. Motivated by these two observations, we designed a novel algorithm by applying the LCS algorithm to selected pairs of rows of an index matrix derived from the input data matrix.

The foundation of the algorithm is the fact that if two rows of the input matrix *A* belong to a significant order-preserving bicluster, then the corresponding two rows of the index matrix *Y* will contain a significant common subsequence with a high probability, and vice versa. This elementary observation leads to a novel method to identify a seed for each potential trend-preserving bicluster. To achieve this goal, we could calculate all the significant common subsequences by applying the LCS algorithm to each pair of rows of *Y*. Instead, we identify a number k such that every significant order-preserving bicluster B must contain at least $k+1$ rows. Now assume that B is such a bicluster, if we equally partition the set of rows of A into k subsets of rows, then there must be at least two rows of B falling into one of these k subsets, and the two rows are sufficient to locate a seed for B. Therefore, applying the LCS algorithm to each pair of rows in each of the k subsets of Y would be sufficient to anchor a seed for each significant order-preserving bicluster of more than k rows. This process identifies a seed for each potential bicluster hidden in the data matrix. The algorithm follows the steps below in order.

Algorithm UniBic

Step 1. Index matrix creation

Let $Y = \{yij\}$ be the index matrix derived from input matrix $A = \{aij\}$ by setting:

$$y_{ij} = r \text{ if andonly if } a_{ir} \text{isthe } j\text{'th smallest entry in row } i, \quad (3)$$

Where ties are broken based on the rule that the smaller column index has higher priority to be ranked.

Step 2. Index matrix partition

We calculate an integer *k* based on the significance (default set to 0.05) of the to-be-identified trend-preserving biclusters using the techniques developed. We then equally partition *Y* into *k* subsets of rows.

Step 3. Application of LCS

Apply the LCS algorithm to each pair of rows in each of the *k* subsets of *Y* to find all the significant longest common subsequences. For each pair of rows having a significant longest common subsequence, one such subsequence is chosen as a seed to which steps 4, 5 and 6 are to be applied. They are listed in decreasing order in length with the longest one at the front.

Step 4. Strict order-preserving bicluster development

We start with a longest seed at the front of the seed list obtained from step 3. The LCS algorithm is then repeatedly applied to find a $3 \times C$ order-preserving submatrix of *A*, where two of the rows are from the seed and the value of C is as large as possible. We continue to add rows one at a time in a greedy fashion until the order-preserving submatrix has more rows than columns, at which point the submatrix from the previous stage is passed on to step 5.

Step 5. Extension to an approximately trend-preserving bicluster

From the strict order-preserving bicluster obtained in step 4, we extend it by first repeatedly adding new columns one at a time with an error rate $r \leq 0.3$ until none is available. Up to now, the bicluster obtained is order-preserved. To identify a significant trend-preserving bicluster, we have to get those remaining original rows and their negative ones involved in the row extension process by repeatedly adding new rows (original or negative) one at a time with an error rate ≤ 0.15 until none is available. The row extension would be achieved by applying the LCS algorithm between the common (consensus) sequence of the column extended order-preserving bicluster and the corresponding index row in *Y* or its reverse row when we consider negative rows to be added. Then remove from the current seed list those with two corresponding rows belonging to discovered biclusters. Repeat step 4 for the next potential trend-preserving bicluster until the list is exhausted.

Step 6. Output as many trend-preserving biclusters as the user needs

We calculate the significance value for those trend-preserving biclusters obtained in step 5. Those with *p*-values less than 0.05 are decreasingly ordered in their significance. Then UniBic outputs first 0 trend-preserving biclusters, where 0 is a parameter which can be pre-specified by users with a default set to 100.

Example 3: Illustrates how to locate an initial seed of a trend-preserving bicluster in the input matrix *A*.

Example 3: Illustration of locating an initial seed.

Example 3a: Input matrix *A*: with entries of two rows and eight columns

row\column	$a_{\cdot 1}$	$a_{\cdot 2}$	$a_{\cdot 3}$	$a_{\cdot 4}$	$a_{\cdot 5}$	$a_{\cdot 6}$	$a_{\cdot 7}$	$a_{\cdot 8}$
i	15	3	10	11	12	10	1	7
j	10	2	6	11	8	6	16	9

Example 3b: The index matrix Y of A: with entries being obtained based on eq. (3)

row\column	$a_{\cdot 2}$	$a_{\cdot 3}$	$a_{\cdot 6}$	$a_{\cdot 5}$	$a_{\cdot 1}$
i	3	10	10	12	15
j	2	6	6	8	10

Example 3c: Initial seed: obtained by the longest common subsequence (2, 3, 6, 5, 1) through applying the LCS algorithm between rows i and j in Y

row\column	$a_{\cdot 2}$	$a_{\cdot 3}$	$a_{\cdot 6}$	$a_{\cdot 5}$	$a_{\cdot 1}$
i	3	10	10	12	15
j	2	6	6	8	10

2. DISTANCE BETWEEN TWO PLACES

Using the google map api keys which is used to fetch the latitude and longitude of the desired place.Haversine formula gives great circle distance between two points on a sphere or ellipsoid. It calculates distance with known latitude and longitude between two points.

Here is the formula.
$a=\sin^2(\Delta latDifference/2)+\cos(lat1).\cos(lat2).\sin^2(\Delta lonDifference/2)$
$c=2.atan2(\sqrt{a},\sqrt{(1-a)})$
$d = R.c$
Where,ΔlatDifference=lat1–lat2(differenceoflatitude)
ΔlonDifference=lon1–lon2(differenceoflongitude)
and R is radius of earth i.e 6373 KM or 3961 miles.

IV. Conclusion

This project presents a dynamic programming based method to solve the problem of mobile sequential recommendation. The proposed method utilizes the iterative nature of the cost function and multiple Unibic policies which greatly improve the Unibic effect. The overall time complexity for handling mobile sequential recommendation problem without length constraint has been reduced from m O(N !) to O(N 2 · 2N). Experimental results show that the Unibic effect and the online search time are better than those of other existing methods. In the future, it will be interesting to use parallel techniques for sequence generation and recommendation.

References

[1] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory Pattern Mining," Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data (KDD '07), pp. 330-339, 2007.

[2] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, "An Energy-Efficient Mobile Recommender System," Proc. 16th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '10), pp. 899-908, 2010.

[3] J. Powell, Y. Huang, F. Bastani, and M. Ji, "Towards Reducing Taxicab Cruising Time Using Spatio-Temporal Profitability Maps," Proc. 12th Int'l Conf. Advances in Spatial and Temporal Databases (SSTD '11), pp. 242-260, 2011. [4] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to Find My Next Passenger," Proc. 13th Int'l Conf. Ubiquitous Computing (UbiComp '11), pp. 109-118, 2011.

[4] S. B€orzs€onyi, K. Stocker, and D. Kossmann, "The Skyline Operator," Proc. 17th Int'l Conf. Data Eng. (ICDE '01), pp. 421-430, 2001.

[5] [6] H. Kargupta, J. Gama, and W. Fan, "The Next Generation of Transportation Systems, Greenhouse Emissions, and Data Mining," Proc. 16th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '10), pp. 1209-1212, 2010.

[6] J.G. Lee, J. Han, X. Li, and H. Cheng, "Mining Discriminative Patterns for Classifying Trajectories on Road Networks," IEEE Trans.

[7] Knowledge and Data Eng., vol. 23, no. 5, May 2011.

[8] N. Pelekis, I. Kopanakis, E. Ramasso, and Y. Theodoridis, "Segmentation and Sampling of Moving Object Trajectories Based on Representativeness," IEEE Trans. Knowledge and Data Eng., vol. 24, no. 7, pp. 1328-1343, July 2012.

[9] E.H.C. Lu, C.Y. Lin, and V.S. Tseng, "Trip-Mine: An Efficient Trip Planning Approach with Travel Time Constraints," Proc. IEEE

[10] 12th Int'l Conf. Mobile Data Management (MDM '11), pp. 152-161, 2011.

[11] H. Wang, "The Strategy of Utilizing Taxi Empty Cruise Time to Solve the Short Distance Trip Problem," master's thesis, the Univ. of Melbourne, 2009. [11] K. Yamamoto, K. Uesugi, and T. Watanabe, "Adaptive Routing of Cruising Taxis by Mutual Exchange of Pathways," Proc. 12th Int'l Conf. Knowledge-Based Intelligent Information and Eng. Systems (KES '08), pp. 559-566, 2008.

[12] Q. Li, Z. Zeng, B. Yang, and T. Zhang, "Hierarchical Route Planning Based on Taxi GPS-trajectories," Proc. 17th Int'l Conf. Geoinformatics (Geoinformatics '09), pp. 1-5, 2009.

[13] S.F. Cheng and X. Qu, "A Service Choice Model for OptimizingTaxi Service Delivery," Proc. 12th Int'l IEEE Conf. Intelligent Transportation Systems (ITSC '09), pp. 1-6, 2009.

[14] G. Hong-Cheng, Y. Xin, and W. Qing, "Investigating the Effect of Travel Time Variability on Drivers' Route Choice Decisions in Shanghai, China," Transportation Planning and Technology, vol. 33, no. 8, pp. 657-669, 2010.

[15] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numerische Mathe, vol. 1, no. 1, pp. 269-271, 1959.

[16] E.W. Dijkstra, "A note on two problems in connexion with graphs," Numerische mathematik, vol. 1, no. 1, pp. 269–271, 1959.

[17] E.V. Denardo and B.L. Fox, "Shortest-route methods: 1. reaching, pruning, and buckets," Operations research, pp. 161–186, 1979.

[18] D.L. Applegatea, "The traveling salesman problem: a computational study," Princeton Univ Pr, 2006.

[19] M. Dell'Amico, M. Fischetti, and P. Toth, "Heuristic algorithms for the multiple depot vehicle scheduling problem," Management Science, pp. 115–125, 1993.

[20] R. Portugal, H.R. Louren¸co, and J.P. Paix˜ao, "Driver scheduling problem modellin," Public transport, vol. 1, no. 2, pp. 103–120, 2009. 27

[21] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in: GIS'10, pp. 99–108, 2010.

[22] J. Yuan, Y. Zheng, X. Xie, and G.Sun, "Driving with knowledge from the physical world," in: KDD'11, pp. 316–324, 2011.

[23] Y. Ge, H. Xiong, C. Liu and Z. Zhou, "A taxi driving fraud detection system," in: ICDM'11, pp. 181–190, 2011.

[24] D. Grosu and A.T. Chronopoulos, "Algorithmic mechanism design for load balancing in distributed systems," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 34, no. 1, pp. 77-84, 2004.

[25] Z. Xu and R. Huang, "Performance study of load balancing algorithms in distributed web server systems," in: International Conference on Electrical Engineering and Informatics, Malaysia, 2004. [26] Y. Xun and G. Xue, "An online fastest-path recommender system," Knowledge Engineering and Management, AISC, vol. 123, pp.341–348, 2012.