International Journal of Emerging Engineering Research and Technology
Volume 2, Issue 4, July 2014, PP 369-373
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Enhanced Driving Based on Local Smoothing Algorithm

G. Tharakeswar Rao[1], K. Narayana[2]

[1] M.Tech Scholor CSE DEPT, SIT, Puttur, India
tharakeswarr@gmail.com
[2] Assistant professorHead of The Department of CSE, SIT, Puttur, India
karnatham.narayana@gmail.com

Abstract: *GPS-equipped taxis can be regarded as mobile sensors probing traffic flows on road surfaces, and taxi drivers are usually experienced in finding the fastest (quickest) route to a destination based on their knowledge. In this paper, we mine smart driving directions from the historical GPS trajectories of a large number of taxis, and provide a user with the practically fastest route to a given destination at a given departure time. In our approach, we propose a time-dependent*

Landmark graph, where a node (landmark) is a road segment frequently traversed by taxis, to model the intelligence of taxi drivers and the properties of dynamic road networks. Then, a Variance-Entropy-Based Clustering approach is devised to estimate the distribution of travel time between two landmarks in different time slots. Based on this graph, we design a two-stage routing algorithm to compute the practically fastest route. We build our system based on a real-world trajectory dataset generated by over 33,000 taxis in a period of 3 months, and evaluate the system by conducting both synthetic experiments and in the field evaluations.

Keywords: *Driving directions, time-dependent fast route, taxi trajectories, Local Smoothing Algorithm*

1. Introduction

Finding efficient driving direction has become a daily activity and been implemented as a key feature in many maps in Google map and Bing maps. A fast driving route saves not only the time of a driver but also energy consumption (as most gas is wasted in traffic jams). Therefore, this service is important for both end users and governments aiming to ease traffic problems and protect environment.

Essentially, the time that a driver traverses a route depends on the following three aspects:

1) The physical feature of a route, such as distance, capacity (lanes), and the number of traffic lights as well as direction turns;

2) The time-dependent traffic flow on the route; and

3) A user's driving behavior. Given the same route, cautious drivers will likely drive relatively slower than those preferring driving very fast and aggressively. Also, users' driving behaviors usually vary in their progressing driving experiences. For example, traveling on an unfamiliar route, a user has to pay attention to the road signs, hence drive relatively slowly. Thus, a good routing service should consider these three aspects (routes, traffic, and drivers), which are far beyond the scope of the shortest/fastest path computing. Usually, big cities have a large number of taxicabs traversing in urban areas. For efficient taxi dispatching and monitoring, taxis are usually equipped with a GPS sensor, which enables them to report their locations to a server at regular intervals, e.g.,2-3minutes.Thatis,alotofGPS- equipped tax is already exist in major cities, generating a huge number of GPS every day. Intuitively, taxi drivers & auto rixas drivers are experienced drivers who can usually find out the fastest route to send passengers to a destination based on their knowledge (we believe most taxi drivers are honest although a few of them might give passengers around about

trip).When selecting driving directions, besides the distance of a route, they also consider other factors, such as the time- variant traffic flows on road surfaces, traffic signals and direction changes contained in a route. These factors can be learned by experienced drivers but are too subtle and difficult to incorporate into existing routing engines. Therefore, these historical taxis, which imply the intelligence of experienced drivers, provide us with a valuable resource to learn practically fast driving directions. The public work or nature disasters may divert the normal direction. But we can avoid them by proper updating from the particular area representatives through corporation help. Here we are going to use BACK TRACKING algorithm to achieve this solution.

2. LOCALSMOOTHING ALGORITHM

Local smoothing. This step aims to find the longest subsequence from the resulting sequence of the global smoothing so as to satisfy the next-nearest principle. It's clear that the brute-force algorithm which checks all the subsequences (whether satisfy Principle 3) takes exponential time. We propose an polynomial time algorithm as shown in Algorithm 1.

Input: a sequence $L = l_1 \to l_2 \to l_3 \to \ldots \to l_{n-1} \to l_n$, dist($l_i$,$l_j$), $i, j = 1, 2, \ldots, n$
Output: a subsequence (of L) $L' = l'_1 \to l'_2 \to l'_3 \to \ldots \to l'_{m-1} \to l'_m$ that satisfies: $\forall i = 1, 2, \ldots m-1, \text{dist}(l_i, l_{i+1}) = \min_{j>i}\{\text{dist}(l_i, l_j)\}$

for $i \leftarrow 2$ **to** n **do**
  for $j \leftarrow i-1$ **downto** 1 **do**
    if $\mathsf{SL}(j) == \emptyset$ **then**
      Insert the sequence $l_j \to l_i$ to $\mathsf{SL}(j)$
    else
      Binary search in $\mathsf{SL}(j)$ for the largest integer p such that dist(l_j,l_{j_p})≤dist(l_j,l_i), if no such value exists, $p := 0$;
      /* where $l_j \to l_{j_p} \to \cdots$ is the p-th sequence in $\mathsf{SL}(j)$ */
      Insert $l_j \to l_i$ after the p-th sequence of $\mathsf{SL}(j)$;
      /* if $p == 0$, insert $l_j \to l_i$ as the first element of $\mathsf{SL}(j)$ */
      for $w \leftarrow 1$ **to** p **do**
        if $\mathsf{SL}(j)^{(w)} \overset{l}{\ominus} l_j \overset{r}{\oplus} l_i \in \mathsf{SL}(j_w)$ **then**
          /* $\mathsf{SL}(j)^{(w)} == l_j \to l_{j_w} \to \cdots$ is the w-th sequence of $\mathsf{SL}(j)$
          $\mathsf{SL}(j)^{(w)} \overset{l}{\ominus} l_j \overset{r}{\oplus} l_i$ represents that the $\mathsf{SL}(j)^{(w)}$ removes the first landmark l_j from the beginning(left) and adds l_i to the end(right), i.e., $\mathsf{SL}(j)^{(w)} \overset{l}{\ominus} l_j \overset{r}{\oplus} l_i == l_{j_w} \to \cdots \to l_i$ */
          $\mathsf{SL}(j)^{(w)} := \mathsf{SL}(j)^{(w)} \overset{r}{\oplus} l_i$;
          /* add l_i after the sequence $\mathsf{SL}(j)^{(w)}$, i.e., $\mathsf{SL}(j)^{(w)} := l_j \to l_{j_w} \to \cdots \to l_i$ */
  return The longest sequence L' in $\{\mathsf{SL}(i) | i = 1, 2, \ldots, n\}$

Algorithm1-Local Smoothing algorithm

Learning Custom Factor

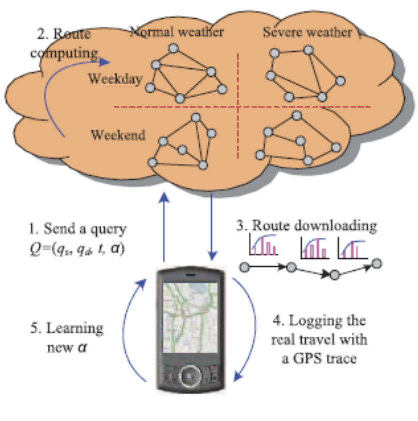

Fig. 1. *Framework of self-adapted routing service.*

This section describes the process for learning the user's custom factor and providing self-adapted fastest route, which contains five steps as illustrated in Fig. 3:

1. Query sending. First, the user sends her query tuple to the cloud, where qs and qd are start point and destination and td is the departure time. The parameter, is the custom factor

2. Route computing. According to the departure time, start and destination point, the cloud chooses a proper landmark graph considering the weather information and whether it's a holiday or a workday. Based on the landmark graph, a two-stage routing algorithm is performed to obtain a time-dependent fastest route based.

3. Route downloading and

4. Path logging. The cloud sends the computed driving routes along with the travel time distributions of the landmark edges contained in the driving route to the phone. Later, the mobile phone logs the user's driving path with a GPS trajectory, which will be used for Recalculate the user's custom factor. The more a driver uses this system, the deeper this system understands the driver; hence, a better driving direction services can be provided.

5. Adapting the custom factor. The custom factor of a given user can be learned in an self-adaptive way.

3. EVALUATION ON ROUTING

For evaluating the effectiveness of the routes suggested by different methods (say methods A and B), we use the following two criteria: Fast Rate 1 (FR1) and Fast Rate 2 (FR2) where method B is used as a baseline.

$$\left\{ FR1 = \frac{number(A^{'}s\ travel\ time < B^{'}s\ travel\ time}{number(queries)} \right.$$

$$\left\{ \mathrm{FR2} = \frac{B^{'}s\ travel\ time - A^{'}s\ travel\ time}{B^{'}s\ travel\ time} \right.$$

FR1 represents how many routes suggested by method A are faster than that of baseline method B, and FR2 reflects to what extent the routes suggested by A are faster than the baseline's. Meanwhile, we use SR to represent the ratio of method A's routes being equivalent to the baseline's.We generate 1,200 queries with different geo-distances of origin-destination pairs and departure times. The geo distances range from 3 to 23 km and follow a uniform distribution. The departure times range from 6 am to 10 pm and are generated randomly in different time slots. The results are shown in figure.

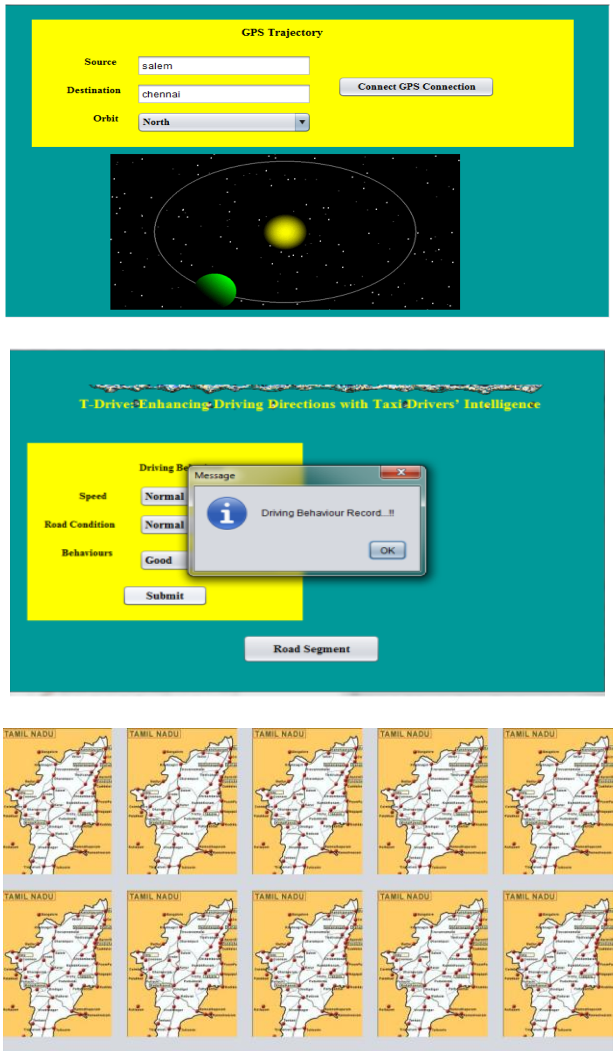

Figure4. *Verified Results.*

4. Conclusion

This paper presents an approach that finds out the practically fastest route to a destination at a given departure time in terms of taxi drivers' intelligence learned from a large number of historical taxi trajectories. In our method, we first construct a time-dependent landmark graph, and then perform a two-stage routing algorithm based on this graphto find the fastest route. We build a real system with real world GPS trajectories generated by over 33,000 taxis in a period of 3 months, and evaluate the system with extensive experiments and in-the-field evaluations. The results show that our method significantly outperforms both the speed- constraint-based and the real-time-traffic-based method in the aspects of effectiveness and efficiency.

References

[1] R. Chhikara and L. Folks. The inverse Gaussian distribution: theory, methodology, and applications. 1989.

[2] K. Cooke and E. Halsey. The shortest route through a network with time-dependent internodal transit times. J.Math. Anal. Appl, 14(492-498):78.

[3] B. C. Dean. Continuous-time dynamic shortest path algorithms. Master's thesis, Massachusetts Institute of Technology, 1999.

[4] B. Ding, J. Yu, and L. Qin. Finding time-dependent shortest paths over large graphs. In Proc. EDBT, pages 205{216. ACM, 2008.

[5] S. Dreyfus. An appraisal of some shortest-path algorithms. Operations Research, 17(3).

[6] Fayyad and Irani. Multi-interval discretization of continuous-valued attributes for classification learning. Proc. IJCAI, pages 1022{1027, 1993.

[7] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach. In Proc. VLDB, 2007.

[8] A. Guhnemann, R. Schafer, K. Thiessenhusen, and P. Wagner. Monitoring traffic and emissions by outing car data. Institute of transport studies Australia, 2004.

[9] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding fastest paths on a road network with speed patterns. In Proc. ICDE, 2006.

[10] J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. LNCS, 4206:243{260.

[11] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In Proc. GIS. ACM, 2009.

[12] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. JACM, 37(3):625, 1990.

[13] D. Pfoser, S. Brakatsoulas, P. Brosch, M. Umlauft, N. Tryfona, and G. Tsironis. Dynamic travel time provision for road networks. In Proc. GIS. ACM, 2008.

[14] J. Yuan, Y. Zheng, C. Zhang, and X. Xie. An interactive-voting based map matching algorithm. In Proc MDM, 2010.

[15] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W. Ma. Understanding transportation modes based on GPS data for Web applications. ACM Transactions on the Web, 4(1):1{36, 2010.

[16] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W. Ma. Understanding mobility based on GPS data. In Proc. Ubicomp, pages 312{321, 2008.

[17] Y. Zheng, L. Liu, L. Wang, and X. Xie. Learning transportation mode from raw gps data for geographic applications on the web. In Proc.