Discovering hot routes using license plate number data

Feng Lin[1*], Rongrong Jiang[2], Mingqi Lv[3], Dongxian Shi[1], Wenkang Huang[1]

[1]Department of Information Technology, Zhejiang Institute of Economics and Trade, Hangzhou 310018, China

[2]Department of Information, Zhejiang TV & Radio University, Hangzhou 310012, China

[3]College of Computer Science, Zhejiang University of Technology, Hangzhou 310023, China

Corresponding Author's Email: cnsxlf@163.com

Keywords: hot route; traffic camera; license plate number data; intelligent transportation systems

Abstract: Traffic camera plays an important role in intelligent transportation systems (ITS). A major function of traffic camera is license plate number recognition. This paper focuses on discovering city-wide hot routes using license plate number data recorded by traffic cameras deployed throughout the city. This task is challenging due to the following two reasons: First, a vehicle trajectory could usually contribute to only a small portion of a hot route. Second, the high degree of uncertainty of license plate number data makes the existing mining algorithms in-effective. Aiming at these problems, a two-phase method is proposed: First, it extracts hot routes by aggregating the license plate number data from multiple traffic cameras and vehicles. Second, it compresses the mined hot routes based on a clustering and ranking algorithm. We have evaluated our method based on real-world license plate number data from a city-wide traffic camera system.

1. Introduction

Intelligent transportation systems (ITS) have been widely applied in many cities as an effective way of improving the performance of transportation systems. A significant change of ITS in recent years is that a large amount of traffic data have been collected from a variety of sources, and this can potentially lead to the change of ITS from a technology driven system into a data driven system [1]. Thus, analyzing traffic data and mining traffic patterns have become a hot topic. In this paper, we focus on the problem of discovering city-wide hot routes. A hot route can be informally defined as a sequence of paths, each part of which shares a high amount of common traffic flow. Potential applications of hot routes include route planning [2], traffic flow prediction [3], conges-tion forecast [4], city planning [5], etc.

Most existing works of traffic pattern mining utilized the fine grained GPS trajectory data from floating vehicles for analysis [5-8]. However, floating vehicles are often cho-sen from particular categories (e.g. taxi, bus), so floating vehicle data can usually cover a very limited number of vehicles and road segments. Therefore, the traffic patterns ex-tracted from floating vehicle data are usually insufficient to reflect the city-wide hot routes. On the other hand, as a main component of ITS, traffic cameras have been broadly deployed in most urban areas, and there is a trend that traffic cameras will gain much more market share in ITS [1]. A major function of traffic camera is license plate number recognition. Based on the license plate number data, a vehicle's trajectory could be reconstructed as a time series of traffic cameras, which have detected the vehicle [9]. Hence, license plate number data could be used as a good source for hot route discovery due to the high penetration and wide coverage of traffic cameras.

Discovering hot routes from license plate number data is challenging due to its high degree of uncertainty: First, traffic cameras are usually sparsely distributed over a city due to budget constraints. Second, traffic cameras might often fail to detect passing by vehicles due to technology limitation. The uncertainty makes the existing methods inef-fective: First, most existing methods leverage an underlying road network to facilitate the traffic pattern mining process [2], [3], [10-12]. However, the consecutive two traffic cameras in a trajectory reconstructed from license plate

  DOI: 10.25236/isaicn.2019.015

number data may be several city blocks away from each other, so it is usually impossible to map the trajectories to road segments. Second, a vehicle trajectory often contributes to only a small portion of a hot route since individual vehicles usually do not travel the entirety of the hot route. More often, they join in or leave somewhere in between the origin and destination of the hot route, and these vehicle trajectories in aggregate form the hot route. Thus, the exist-ing mining algorithms designed for trajectory data, such as clustering algorithm [13] and sequential pattern mining algorithm [14], [15], may discover many short hot routes, leading to weak power for interpreting the city-wide traffic flow. Third, traffic pattern mining methods can usually generate a large number of results, with many similar ones. Thus, it is difficult for users to browse all the candidates to find the important ones.

Aiming at these problems, this paper proposes a two-phase method for mining hot routes from license plate number data: First, it extracts short patterns from the recon-structed vehicle trajectories, and splices these patterns to form longer ones as hot routes based on a bidirectional tree (i.e. hot route mining). Second, it discovers exemplary ones from the mined hot routes based on a clustering and ranking algorithm (i.e. hot route compression). The major contributions of this paper are summarized as follows:

(1) Propose a hot route discovery method free of road network based on license plate number data.

(2) Design a bidirectional tree structure to splice short patterns extracted from li-cense plate number data to form longer hot routes.

(3) Propose a clustering and ranking algorithm to discover exemplary hot routes.

(4) Conducted extensive experiments using real license plate number data from a city-wide traffic camera system.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 gives the preliminary of our work. Section 4 and 5 detail the hot route mining method and hot route compression method respectively. Section 6 reports the experiment results. Section 7 concludes the paper.

2. Experiment

Existing works have applied data mining techniques on massive data from various traffic sensors to discover traffic patterns. For example, Ntoutsi et al. [10] and Banaei-Kashani et al. [11] exploited the traffic flow data (measured by loop detectors) to detect traffic flow patterns for each road segments. Yuan et al. extracted driving intel-ligence from floating vehicle data to compute practically fastest and customized routes [6]. Peng et al. discovered three distinct basis traffic flow patterns from floating vehicle data, and combined these basis patterns to approximate the traffic flow between any pair of locations [7]. Janecek et al. leveraged cellular data to infer road traffic status, including vehicle travel time and road congestion [4].

The aggregated license plate number data from traffic cameras are mostly used for traffic flow estimation. For example, Castillo et al. utilized license plate number data and link flow data to reconstruct path flows, and used the path flows to estimate trip matrix [9]. Mínguez et al. proposed methods to optimize the number and location of traffic cameras for OD (i.e. origin-destination) trip matrix estimation [16]. However, traf-fic flow is a statistical value, which indicates the number of vehicles moving on a road or between an OD pair. It cannot interpret how these vehicles choose their paths.

To discover the regularity of how moving objects moves in spatial space, there has been considerable research on mining patterns from trajectory data recently. For exam-ple, Lee et al. proposed a partition-and-group trajectory clustering framework to discover common sub-trajectories from trajectory data [13]. Cao et al. defined pattern elements as spatial regions around frequent line segments, and found patterns based on a sub-string tree structure [14]. However, trajectory patterns represented by clusters or sequen-tial patterns are usually too short to reflect the city-wide hot routes. Actually, they are usually used to interpret the moving behaviors of individual moving objects. In real world, a vehicle trajectory always contributes to only a small portion of a

hot route [12]. For example, a hot route may exist from residential area to working area in the morning, but most vehicles would not travel the entirety of the hot route. More often, they join in or leave somewhere in between the origin and destination of the hot route.

Most relative to our work is the FlowScan algorithm [12] proposed by Li et al. for hot route discovery. A hot route is a sequence of road segments with heavy traffic flow. A sequence of consecutive road segments in a hot route should share some common traffic flow. The FlowScan algorithm extends a hot route to neighboring road segments in a road network using a density-based clustering technique. However, since many road intersections and road segments have not been deployed with traffic cameras, and traf-fic cameras might sometimes fail to detect passing vehicles, the reconstructed trajecto-ries are usually of high degree of uncertainty and could not be mapped to sequences of road segments based on map matching. Besides, the hot route mining algorithm may find a great number of hot routes, and many ones are similar, so it is difficult for decision makers to go through all the hot routes to understand the major traffic behaviors in the city.

3. Preliminary work

A traffic camera can record the license plate number and passing time of the observed vehicles. The originally gathered data is defined as follow.

Definition 1 (License plate number data): The license plate number data consists of the set LD = $\{(I_k, C_k, T_k)\}$, where I_k is the license plate number of the kth observed vehicle, C_k is the identifier of the traffic camera that records I_k, and T_k is the passing time through C_k of I_k.

A cross-search of license plate numbers in all (P_k, I_k, t_k) items and check of the corresponding passing time allows the determination of the path followed by each observed vehicle. The path of a vehicle is defined as trajectory. The hot routes are mined from a large corpus of trajectories, which are segmented on a daily basis.

Definition 2 (Trajectory): The trajectory VT of a vehicle is a sequence VT = $<E_k>$, each element E_k is a pair $E_k = (C_k, T_k)$ ($T_k < T_{k+1}$), where (C_k, T_k) means that the vehicle passes by traffic camera C_k at time T_k.

Definition 3 (Hot route): A hot route R is a sequence R = $<C_k>$ (C_k represents a traffic camera), satisfying that every consecutive K traffic cameras in R share at least minSup common traffic flow during a time span (i.e. at least minSup vehicles pass by the K traffic cameras consecutively in a single trajectory during the time span).

The proposed approach consists of two main components: (1) mining hot routes from trajectories, and (2) compressing and ranking the extracted hot routes.

4. Hot Route Mining

Due to the uncertainty of the trajectories reconstructed based on traffic cameras, hot routes could not be extracted leveraging an underlying road network. Therefore, we propose a two-step algorithm free of road network to form long hot routes based on short traffic patterns extracted from the trajectories.

First, we extract length-K substring patterns from the trajectories given a time span [T_s, T_e] (the time span is also defined on a daily basis). Using substring patterns rather than sequential patterns is based on the consideration that a moving object should move contiguously in spatial space [14]. We use a hashing based algorithm for extracting substring patterns, where a hash Table is used to map candidate substring patterns to their supports. For each trajectory VT, the algorithm reads in K consecutive elements $<(C_1, T_1),\ldots,(C_K, T_K)>$ from VT. If $T_1 \geq T_s$ and $T_K \leq T_e$, the algorithm takes $<C_1,\ldots,C_K>$ as a candidate substring pattern and updates the hash Table accordingly. Finally, substring patterns with support not less than minSup are kept.

In practice, the parameter K should be set to a small value in order to extract sufficient substring patterns. On the other hand, the parameter minSup is often traffic dependent. An area with heavier traffic should have larger minSup than that with lighter traffic. Thus, it is difficult to specify a good absolute value for minSup without the knowledge about the traffic condition. Hence, we use an

alternative solution to estimate a relative value for minSup by exploring the traffic density distribution of all the extracted length-2 substring patterns (the traffic density of a substring pattern is its support). We plot the CDF (Cumulative Distribution Function) curve of the traffic density of all length-2 substring patterns, and use this curve to estimate a relative value for minSup by using the inverse CDF as Equation 1. With the equation, we can specify a relative parameter rMinSup (0 < rMinSup < 1), and then the absolute value of minSup could be calculated as iCDF(rMinSup).

$$iCDF(p) = \inf\{x \in R : p \le CDF(x)\} \quad (1)$$

Second, we propose an algorithm to splice the extracted short substring patterns to form longer ones as hot routes without an underlying road network. The algorithm is essentially a process of constructing a bidirectional tree based on the concept of forward spliceable and backward spliceable defined as follows. Fig. 1 shows the pseudo-code of the algorithm.

Definition 4 (Forward spliceable & Backward spliceable): Given a length-K substring pattern P_0, a length-K substring pattern P_1 is forward spliceable with P_0 if: the length-(K-1) prefix of P_1 completely matches the length-(K-1) suffix of P_0, while a length-K substring pattern P_2 is backward spliceable with P_0 if: the length-(K-1) suffix of P_2 completely matches the length-(K-1) prefix of P_0.

Algorithm1 **Hot Route Mining**

INPUT: A set of length-K substring patterns PS
OUTPUT: A set of hot routes RS
1. Copy PS to TS
2. **while** TS is not empty **do**
3. Find P with the largest support in TS
4. Remove P from TS
5. Create a tree node fn corresponding to $P.C_K$ and a tree node bn corresponding to $P.C_1$
6. Call Forward_Expand(fn) and Backward_Expand(bn)
7. **for** every branch bb of the backward tree rooted at bn **do**
8. **for** every branch fb of the forward tree rooted at fn **do**
9. Combine the reverse of bb, $<P.C_2,\ldots,P.C_{K-1}>$, and fb as R, and append R to RS

Procedure **Forward_Expand**(tree node n)
1. Let P be the associated substring pattern of n
2. Find FS as the set of substring patterns that are forward spliceable with P in PS
3. **for** every substring pattern fp in FS **do**
4. Create a new tree node nn corresponding to $fp.C_K$, and append it as the child of n
5. Remove fp from TS, and call Forward_Expand(nn)

Procedure **Backward_Expand**(tree node n)
1. Let P be the associated substring pattern of n
2. Find BS as the set of substring patterns that are backward spliceable with P in PS
3. **for** every substring pattern bp in BS **do**
4. Create a new tree node nn corresponding to $bp.C_1$, and append it as the child of n
5. Remove bp from TS, and call Backward_Expand(nn)

Fig. 1. The hot route mining algorithm based on substring pattern splicing.

We use an example to illustrate the hot route mining algorithm. Given 7 length-3 substring patterns (P_1 = <1, 2, 3>, P_2 = <2, 3, 4>, P_3 = <2, 3, 5>, P_4 = <3, 4, 6>, P_5 = <3, 4, 7>, P_6 = <8, 1, 2> and P_7 = <9, 1, 2>), and P_1 is the one with largest support, the constructed bidirectional tree is shown in Fig. 2 (the identifier in the square denotes the associated substring pattern of the tree node). Then, we can find 6 hot routes by combining every branch of the backward tree and the forward tree, i.e. R_1 = <8, 1, 2, 3, 5>, R_2 = <9, 1, 2, 3, 5>, R_3 = <8, 1, 2, 3, 4, 6>, R_4 = <8, 1, 2, 3, 4, 7>, R_5 = <9, 1, 2, 3, 4, 6>, R_6 = <9, 1, 2, 3, 4, 7>.

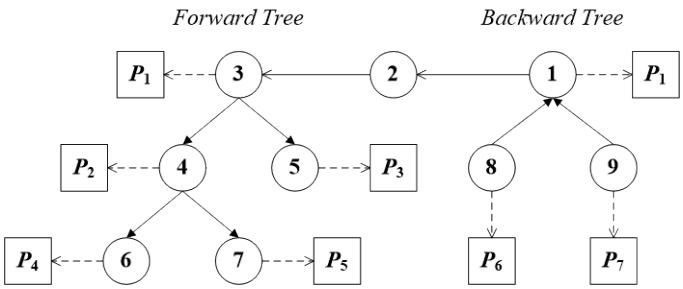

Fig. 2. The bidirectional tree for hot route mining.

5. Hot Route Compression

The hot route mining phase would generate a large number of hot routes, with many similar ones. Besides, some hot routes might be of higher importance than others. As shown in Fig. 2, six hot routes are extracted from seven substring patterns, so a large corpus of hot routes could be extracted from real-world trajectory dataset. Besides, sev-eral pairs of hot routes (e.g. R1 and R2, R3 and R4, R5 and R6) are almost the same. Hence, we compress the hot route mining results based on a clustering and ranking algorithm: First, we diversify the hot route mining results by dividing them into multiple clusters. Second, we find the most exemplary hot route for each cluster.

In order to cluster hot routes, we define the similarity between hot route Ri and Rj based on the longest common subsequence as Equation 2, where LCSS(Ri, Rj) is the longest common subsequence of Ri and Rj. Using the maximum of S(Ri□ Rj) and S(Rj□ Ri) as the similarity between Ri and Rj favors longer hot routes, because a longer hot route is more likely to absorb shorter ones which are similar with its subsequence into the same cluster. Based on the similarity measure, we could divide the extracted hot routes into multiple clusters by using clustering techniques.

$$S\left(R_i,R_j\right)=\max\left\{S\left(R_i\rightarrow R_j\right),S\left(R_j\rightarrow R_i\right)\right\} \tag{2}$$

$$S\left(R_i\rightarrow R_j\right)=\frac{\left|LCSS\left(R_i,R_j\right)\right|}{\left|R_i\right|} \tag{3}$$

A cluster may comprise many similar hot routes, so we would like to detect the most exemplary one for each cluster. We consider two factors to compute the exemplary score for the hot routes: representativeness and importance.

To measure representativeness, since $S(R_i \rightarrow R_j)$ can be viewed as how well hot route R_i could be represented by hot route R_j, we quantize the representativeness of hot route R_k in cluster RC based on Equation 4.

$$RScore_{RC}\left(R_k\right)=\frac{\sum_{R_i\in RC}S\left(R_i\rightarrow R_k\right)}{\left|RC\right|} \tag{4}$$

To measure importance, since a hot route can be viewed as a sequence of traffic cameras, we take into account the following assumptions for quantizing the importance: (1) A hot route is important if it contains more important traffic cameras. (2) A traffic camera is important if it is contained in more important hot routes. (3) The importance of a traffic camera can be quantized by the traffic volume detected by it.

Assumptions 1 and 2 imply a mutual reinforcement relationship between hot routes and traffic cameras. It is similar with that between authorities and hubs of the HITS algorithm [17]. Thus, we treat a traffic camera as a hub and a hot route as an authority. Given n traffic cameras and m hot routes, we build an m×n matrix M_{RC}, where $M_{RC}[i, j]$ indicates whether the jth traffic camera is involved in the ith hot route. The hub scores and authority scores are denoted as P_C and P_R. Then, a power iteration algorithm can be used to calculate the final P_C and P_R. In order to incorporate

assumption 3, at every iteration of the algorithm, the scores are redistributed from authorities (i.e. hot routes) to hubs (i.e. traffic cameras) in a way that is proportional to the traffic volume detected by the traffic cameras. This means that traffic volume becomes a multiplicative factor for score propagation. We summarize the hot route ranking algorithm in Fig. 3, where V_T is an n-dimensional column vector ($V_T[k]$ indicates the average traffic volume detected by the kth traffic camera during the time span over multiple days).

Algorithm2 **Hot Route Ranking**

INPUT: The relationship matrix M_{RC}, and the traffic volume vector V_T

OUTPUT: The importance score vector of the hot routes

1. Initialize $P_C^{(0)} = [\underbrace{1,1,...,1}_{n}]^T$, $P_R^{(0)} = [\underbrace{1,1,...,1}_{m}]^T$
2. **while** algorithm has not converged **do**
3. $P_C^{(t+1)} = M_{RC}^T \cdot P_R^{(t)} \cdot V_T$, $P_R^{(t+1)} = M_{RC} \cdot P_C^{(t+1)}$ // the tth iteration
4. Normalize $P_R^{(t+1)}$ to the range [0, 1]
5. Output the final P_R as the importance score vector

Fig. 3. The hot route ranking algorithm based on power iteration.

We use an example with three hot routes (R_1, R_2 and R_3) and four traffic cameras (C_1, C_2, C_3 and C_4) as shown in Fig. 4 to illustrate the procedure of calculating representativeness and importance scores. Assume that these hot routes are within the same cluster, the representativeness score of R_1, R_2 and R_3 are 0.833, 0.5 and 0.583 respectively. After applying the hot route ranking algorithm, the importance score of R_1, R_2 and R_3 are 0.363, 0.248 and 0.389 respectively. Thus, R_1 has the highest representativeness (since it has the largest amount of overlapping with other hot routes) and R_3 has the highest importance (since it passes through traffic camera C_4 with the heaviest traffic).

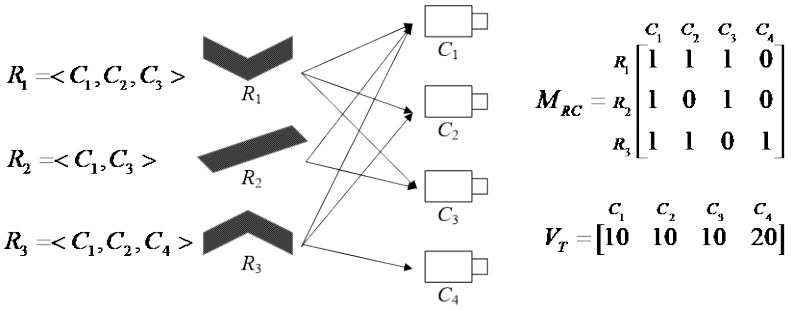

Fig. 4. An example to illustrate the procedure of representativeness and importance calculation.

After obtaining both representativeness and importance scores of all hot routes in a cluster, we could find the most exemplary one by taking both scores into account (e.g. using a weight parameter to control the tradeoff). Then, the exemplars of all clusters are outputted as the final hot routes. These exemplars are outputted and ranked by the size of the corresponding clusters (i.e. an exemplar would be ranked higher if more hot routes belong to its corresponding cluster).

6. Experiment

6.1 Dataset

In the experiment, the data are collected from 821 traffic cameras deployed within the urban area of Hangzhou, China for over a month (from 1, June 2012 to 4, July 2012). The experiment was conducted based on the data of all the weekdays (totally 24 days) during the period. The final dataset contains 105594340 license plate number logs. The average numbers of logs and the detected vehicles per day are 4399764 (SD = 459417) and 704856 (SD = 74048) respectively.

The dataset is of high degree of uncertainty. First, as shown in Fig. 5, when we zooming in the map, we can find that a large proportion of road intersections have not been deployed with traffic cameras.

Second, there are 12744855 logs which have not been assigned with a valid license plate number. This means that these traffic cameras can fail to recognize passing vehicles at about 12% times. Besides, the traffic cameras can miss to detect passing vehicles, so the failure rate may be even higher.

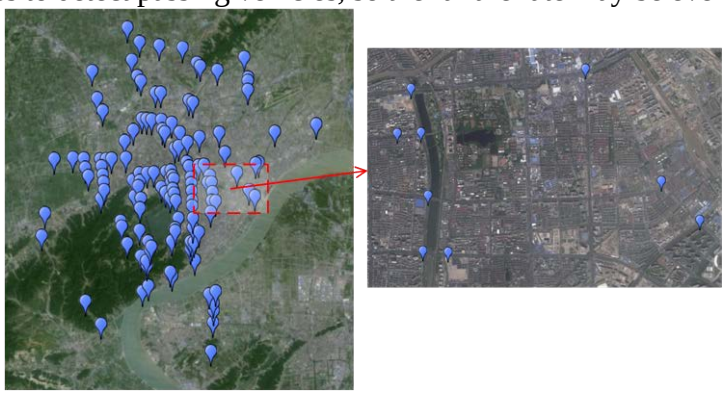

Fig. 5. The sparsity of the traffic cameras.

6.2 Parameter Tuning

There are two key parameters of the proposed hot route mining algorithm: K and minSup. The parameter K controls to what extent the involved trajectories in a hot route should overlap with each other. The value of K should not be less than 3. This is because that substring patterns with length less than 3 could not be spliced with each other (a path segment is formed by 2 traffic cameras, and two substring patterns should share at least one path segment to be spliced). The parameter minSup is often traffic dependent.

We set the time span as morning rush-hour (i.e. "08:00-09:00") to conduct the experiment. We expect to extract hot routes that can better interpret the city-wide traffic flow, i.e. with longer length and heavier involved traffic volume. Fig. 6 plots the average length, average traffic volume, minimum traffic volume and number of the extracted hot routes by adjusting K and rMinSup. The traffic volume is calculated based on the average support of the length-2 substring patterns per hour involved in a hot route. As shown in Fig. 6(a) and (b), by increasing K and rMinSup, the average length decreases significantly, while the average traffic volume increases slightly. It motivates us to set K and rMinSup with lower values. However, setting both K and rMinSup too low would cause unexpected problems: involving substring patterns with extremely low traffic volume (K = 3 and rMinSup < 0.955 as shown in Fig. 6(c)) or increasing the number of hot routes explosively (K = 3 and rMinSup < 0.95 as shown in Fig. 6(d)). In the rest of the experiment, we set K = 3 and rMinSup = 0.955 (the corresponding minSup is about 172 per hour).

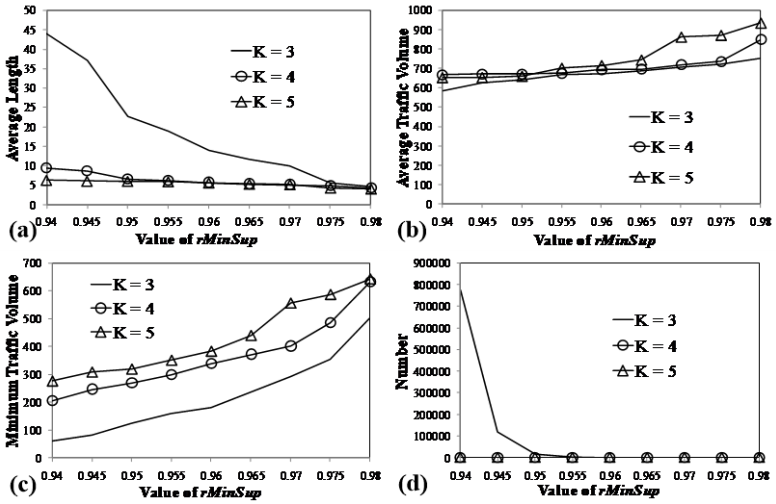

Fig. 6. The effect of adjusting K and rMinSup.

6.3 Performance Evaluation

We evaluate our hot route discovery method (abbreviated as OurMining) by comparing it with the following competitors:

(1) CloSpan: It extracts sequential patterns using the CloSpan algorithm [18], and then discovers exemplary ones as hot routes using the method in Section 4.

(2) FlowScan: It extracts hot routes using the FlowScan algorithm [12], and then compresses the extracted hot routes using the method in Section 4. FlowScan creates hot routes by expanding a starting road segment to its neighboring road segments using a density-based clustering algorithm. The neighbors of a road segment r is defined as a set of road segments RS, the minimum number of hops of road segments between r and each one in RS is less than a threshold Eps. However, we cannot map a trajectory reconstructed from license plate number data to road segments. Instead, we define the neighbors of a traffic camera c as a set of traffic cameras CS, the Euclidean distance between c and each one in CS is less than a threshold dEps, and create hot routes by expanding a starting traffic camera to its neighboring traffic cameras.

(3) DirectMining: It extracts hot routes using method in Section 3 without the hot route compression phase. These hot routes are ranked by their involved traffic volume (i.e. the sum of the support of all the involved length-2 substring patterns).

We still use the data from morning rush-hour (i.e. “08:00-09:00“) for evaluation. We use Affinity Propagation [19] as the clustering algorithm for the hot route compression phase. The weights of representativeness and importance scores are both set to 0.5. We use the average and maximum length of all hot routes, and the city traffic flow coverage of the top-N hot routes as metrics to evaluate the mining results. The city traffic flow coverage is the ratio of the traffic volume of the top-N hot routes to the traffic volume of the whole city. The traffic volume of the top-N hot routes or the whole city is calculated by summing up the support of all the distinct length-2 substring patterns involved in the top-N hot routes or all the extracted length-2 substring patterns. The parameters are set as follows: K = 3 (for OurMining, FlowScan and DirectMining), rMinSup = 0.955 (for OurMining, CloSpan, FlowScan and DirectMining), dEps = 1000 to 5000 meters (for FlowScan, corresponding to FlowScan_1000 to FlowScan_5000 in Fig. 7). Besides, in order to keep the vehicle movement continuity, we limit the maximum gap between two items in the extracted patterns to 3 for the CloSpan algorithm.

The experimental results are shown in Fig. 7. CloSpan can only discover very short hot routes. This is because that sequential pattern mining algorithms require a trajectory completely covers the supported hot route, but a trajectory usually contributes to only a small portion of a hot route in real world. DirectMining achieves the longest hot routes, but its top-N city traffic flow coverage is low even with a large N. The reason is that a great number of long hot routes overlap with each other due to the lack of hot route compression, especially in the top results (since longer hot routes that focus on several paths with the most heavy traffic volume are more likely to be ranked higher). Thus, the top-N city traffic flow coverage tends not to grow that much by increasing N. The performance of FlowScan is bad when the value of dEps is low, and it is generally improved by increasing dEps. However, its top-N city traffic flow coverage is lower than that of OurMining, especially when N is large. By analyzing the experiment results, we find that although FlowScan can discover hot routes with high quality, it usually has a low recall. This is mainly caused by the unavailability of the road network: First, it is inappropriate to define the neighborhood based on Euclidean distance, while it is defined based on the number of hops of the road segments in [12]. For example, if two traffic cameras were deployed at two ends of a highway, it might be a few kilometers away. But if it were a city block, it might be just a hundred meters. Besides, we cannot choose a very large dEps considering the continuity of vehicle movement. Second, it is difficult to initialize the density-based clustering process. The FlowScan algorithm begins at a road segment with at least minSup vehicles start their trajectory at it or converge at it from the neighboring road segments. However, we cannot test the latter condition without the road network. Overall, OurMining could generate longer hot routes which cover larger proportion of the city traffic flow.

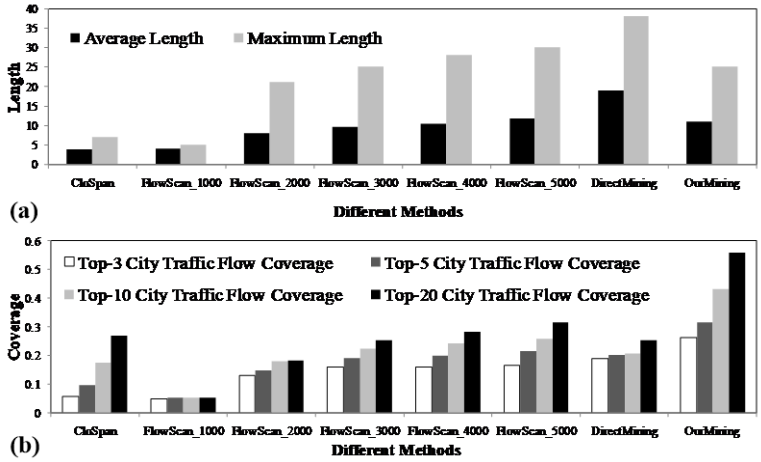

Fig. 7. The performance of hot route discovery.

Fig. 8 visualizes two hot routes extracted from the morning rush-hour ("08:00-09:00" shown in Fig. 9(a)) and evening rush-hour ("17:00-18:00" shown in Fig. 9(b)) respectively. Each hot route is drawn in red polyline with a dot indicating the start and an arrow indicating the end. The gap between two consecutive traffic cameras is estimated as the shortest path in the city road network. The hot route in Fig. 9(a) is from the north to the center of the city near the West Lake. The hot route in Fig. 9(b) is from the industrial area to the residential area of the city, across the city downtown. It can be found from the figure that the traffic tends to flow through city highways (especially the Zhonghe highway across the center of the city).

Fig. 8. A visualization of the discovered hot routes.

7. Conclusion

In this paper, we have proposed a method for discovering city-wide hot routes from license plate number data from traffic cameras. It firstly mines hot routes by splicing substring patterns extracted from license plate number data based on a bidirectional tree, and then finds the exemplary ones based on a clustering and ranking algorithm. The experiment based on real license plate number data has demonstrated that the proposed method could discover hot routes with stronger interpretation power (i.e. the discovered hot routes are longer and could involve more traffic volume with fewer instances) as compared to state-of-the-art methods.

Acknowledgement

Key projects of Zhejiang Province Natural Science Foundation: Research on Key Technologies of Verifiable Secure Cloud Storage (LZ18F020003).

References

[1] J. Zhang, F.Y. Wang, K. Wang, W.H. Lin, X. Xu, and C. Chen. Data-driven intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems, 2011, 12 (4) : 1624-1639.

[2] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J.P. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach, Proceedings of VLDB, 2007: 794-805.

[3] P.S. Castro, D. Zhang, and S. Li. Urban traffic modelling and prediction using large scale taxi GPS traces, Proceedings of Pervasive, 2012: 57-72.

[4] A. Janecek, K.A. Hummel, D. Valerio, F. Ricciato, and H. Hlavacs. Cellular data meet vehicular traffic theory: Location area updates and cell transitions for travel time estimation, Proceedings of UbiComp, 2012: 361-370.

[5] C. Chen, D. Zhang, Z.H. Zhou, N. Li, T. Atmaca, and S. Li. B-Planner: Night bus route planning using large-scale taxi GPS traces, Proceedings of PerCom, 2013: 225-233.

[6] J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-Drive: Enhancing driving directions with taxi drivers' intelligence, IEEE Transactions on Knowledge and Data Engineering, 2013, 25 (4) : 220-232.

[7] C. Peng, X. Jin, K.C. Wong, M. Shi, and P. Liò. Collective human mobility pattern from taxi trips in urban area, Plos One, 2012, 7 (4) : 1-7.

[8] C. Fabritiis, R. Ragona, and G. Valenti. Traffic estimation and prediction based on real time floating car data, Proceedings of ITSC, 2008:197-203.

[9] E. Castillo, J.M. Menéndez, and P. Jiménez. Trip matrix and path flow reconstruction and estimation based on plate scanning and link observations, Transportation Research Part B: Methodological, 2008, 42 (5) : 455-481.

[10] I. Ntoutsi, N. Mitsou, and G. Marketos. Traffic mining in a road-network: How does the traffic flow?, International Journal of Business Intelligence and Data Mining, 2008, 3 (1) : 82-98.

[11] F. Banaei-Kashani, C. Shahabi, and B. Pan. Discovering patterns in traffic sensor data, Proceedings of IWGS, 2011:10-16.

[12] X. Li, J. Han, J.G. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks, Proceedings of SSTD, 2007: 441-459.

[13] J.G. Lee, J. Han, and K.Y. Whang. Trajectory clustering: A partition-and-group framework, Proceedings of SIGMOD, 2007: 593-604.

[14] H. Cao, N. Mamoulis, and D.W. Cheung. Discovery of periodic patterns in spatiotemporal sequences, IEEE Transactions on Knowledge and Data Engineering, 2007, 19 (4): 453-467.

[15] F. Lipan and A. Groza. Mining traffic patterns from public transportation GPS data, Proceedings of ICCP, 2013: 123-126.

[16] R. Mínguez, S. Sánchez-Cambronero, E. Castillo, and P. Jiménez. Optimal traffic plate scanning location for OD trip matrix and route estimation in road networks, Transportation Research Part B: Methodological, 2010, 44(2) : 282-298.

[17] J.M. Kleinberg. Authoritative sources in a hyperlinked environment, Journal of the ACM, 1999, 46(5) : 604-632.

[18] X. Yan, J. Han, and R. Afshar. CloSpan: mining closed sequential patterns in large datasets, Proceedings of SDM, 2003: 166-177.

[19] B.J. Frey and D. Dueck. Clustering by passing messages between data points, Science, 2007, 315(5814): 972-976.