R3: A Real-Time Route Recommendation System

Henan Wang[†], Guoliang Li[†], Huiqi Hu[†], Shuo Chen[†], Bingwen Shen[†], Hao Wu[‡], Wen-Syan Li[‡], Kian-Lee Tan[⋆]
[†]Department of Computer Science, Tsinghua University, Beijing, China
[‡]SAP Labs, Shanghai, China
[⋆]Department of Computer Science, National University of Singapore
whn13@mails.thu.edu.cn, liguoliang@tsinghua.edu.cn,
{hhq11,s-chen13,sbw13}@mails.tsinghua.edu.cn,
{michael.wu02,wen-syan.li}@sap.com, tankl@comp.nus.edu.sg

ABSTRACT

Existing route recommendation systems have two main weaknesses. First, they usually recommend the same route for all users and cannot help control traffic jam. Second, they do not take full advantage of real-time traffic to recommend the best routes. To address these two problems, we develop a real-time route recommendation system, called R3, aiming to provide users with the real-time-traffic-aware routes. R3 recommends diverse routes for different users to alleviate the traffic pressure. R3 utilizes historical taxi driving data and real-time traffic data and integrates them together to provide users with real-time route recommendation.

1. INTRODUCTION

With the rapid development of economic, the quantity of vehicles increases rapidly and the traffic jam problem is becoming increasingly serious. Route recommendation systems which provide users with route recommendations have been widely accepted by users. However, existing route recommendation systems have two main weaknesses. First, they usually recommend the same route for all users and cannot help control traffic jam as the recommended route may be congested. Second, they do not take full advantage of real-time traffic to recommend the best routes. It calls for new real-time route recommendation systems.

There are some challenges in designing a real-time route recommendation system. First, the traffic condition is dynamically changing and also unpredictable and it takes time for users traveling from the starting node to her destination. That is to say, the traffic condition will be changed when the user is traveling on the route given at the starting moment. Second, the traffic condition has certain delays. The change of traffic condition is small for short period of time and the difference between current traffic condition and future traffic condition increases with the growth of time. Given two nodes in the road network, if we use the current traffic condition to estimate the future traffic, the estimation become more and more inaccurate with the increase of time. If the two nodes are far and traveling from one to another is time consuming, the estimation has rather low accuracy.

To address these limitations, we demonstrate a real-time route recommendation system, called R3, with the following unique features. (1) R3 integrates rich traffic data from the taxies and utilizes them to provide high-quality route recommendations. (2) R3 can recommend real-time and diverse routes for different users, which can significantly save users driving time and reduce traffic jam. There are two main challenges in designing the system. The first one is how to utilize the taxi driving data to construct the empirical traffic model. The second one is how to seamlessly integrate real-time and empirical traffic to recommend best routes. Obviously the traffic dynamically changes, and it is unreasonable to utilize the current traffic to predict future traffic conditions (e.g., 20 minutes later). For example, if all the routes from a source to a destination take more than 30 minutes, we cannot utilize the current traffic to find the best route. Alternatively, we can first utilize the current traffic to search possible real-time sub-routes near to the source based on the `Dijkstra` algorithm, and then use the historical traffic data to estimate the time from the end points of these sub-routes to the destination. Finally, we use the A^* algorithm to select the best route based on the real-time traffic and historical traffic together to recommend best routes.

We have implemented and deployed a real system, available at `http://tsingnus.cs.tsinghua.edu.cn/map`, which has been commonly used and widely accepted. R3 utilizes the historical taxi driving data to predict the empirical traffic model. We used more than 1 billion driving records generated by 8,000 taxies in a year, and calculated the average driving speed of every road in each second as the empirical traffic model. To efficiently compute the best route based on the empirical traffic model, we also constructed an efficient tree-based index structure. R3 utilizes the empirical model and real-time model to recommend the best route from the source to the destination which took the least time. We utilize the A^* algorithm to recommend real-time routes based on both empirical model and real-time traffic model. We compared our method with Google Maps and Baidu Map on real datasets and the experimental results show that our method outperforms them.

Proceedings of the VLDB Endowment, Vol. 7, No. 13

2. SYSTEM OVERVIEW

Given a routing query from a source to a destination, we want to find the best route with the least traveling time. To recommend high-quality routes, we utilize various heterogeneous data to support the routing query. (1) Road networks: there are many nodes and edges between nodes in the map. The weights of edges reflect the traffic conditions. (2) Real-time traffic data: the traffic dynamically (i.e., the weights of edges) changes every 1 minute. We take the traveling time on an edge as the edge weight. (3) Taxi driving data: we have 1 billion taxi driving records generated by 8000 taxies and each taxi reports a record with longitude, latitude, and speed in each second. We utilize these information to recommend real-time route. The architecture is illustrated in Figure 1 which includes the following main components.

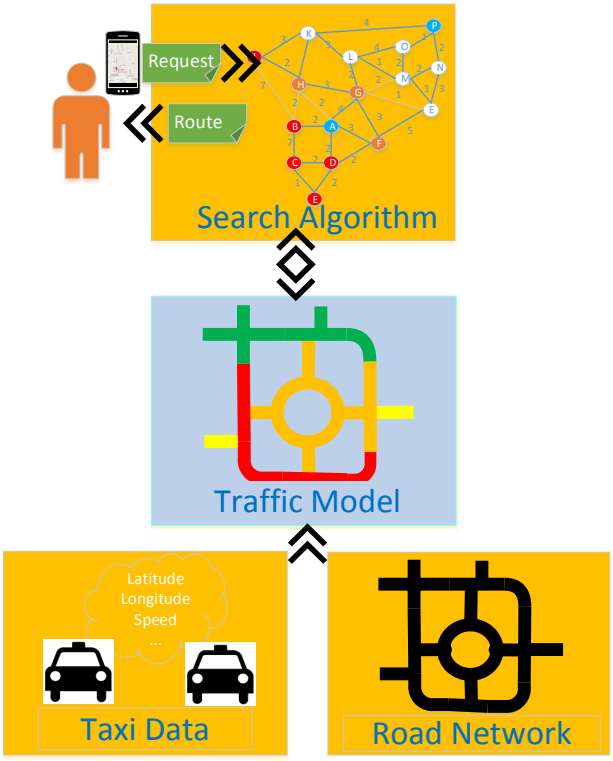


Figure 1: System Architecture

Empirical Traffic Model Construction. We first partition the traffic data based on working dates, weekends and public holidays. For each day, we split the 24 hours traffic data into 48 intervals (half an hour for each interval), and for each interval we compute the average driving speed in each road based on the taxi driving data in this particular interval. Obviously, to model the empirical traffic condition, if there is a single driving taxi on a road, R3 utilizes its speed as the empirical model on this road. If there are multiple taxies, we calculate the average speed of the taxies running on the road in the interval. If there is no car runs on the road, we treat the traffic condition of its nearest road as its traffic condition. We also build index to facilitate computing the shortest paths between two nodes based on the empirical traffic model (see Section 3.1).

Real-time Traffic Model Construction. In the real-time model, we collect the real-time traffic from the background data system published by the government or the third parties. Notice that these traffic information has 5-10 minutes delay. We treat these information as the approximate real-time traffic condition.

Search Algorithm. Real-time route recommendation problem can be reduced to the problem that finding the shortest path between two nodes on a connected and directed graph while the weights of all edges are dynamically changed all the time. We consider both real-time traffic and the empirical model to recommend routes. We first expand some possible candidates and sub-routes from the starting node and based on the real-time model through an A^* resembled algorithm. Then we estimate the cost of routes from the candidates to the destination. Finally we combine the two sub-routes to select the best one with the least total time.

User Interface. To facilitate users type in the source and destination, we enable location-aware instant keyword search which provides users with a user-friendly way for typing in the source and destination [3, 4].

3. RECOMMENATION ALGORITHM

3.1 Empirical Model

We begin with introducing the empirical model of our real-time route recommendation method. There are two challenges in empirical model. The first is to effectively estimate the speed on each road. The second is to efficiently compute the traveling time between two nodes.

To address the first challenge, we utilize the taxi driving records to estimate the driving speed of each road. To capture the dynamic feature of the traffic information, we split the data based on different time intervals as discussed in Section 2 and utilize the average driving speed on each road at different intervals to estimate the empirical traffic. We use the driving speed to estimate the traveling time on each edge and set the traveling time as the edge weight.

To address the second challenge and meet the real-time route recommendation requirement, we extend our G-Tree index structure [5] to take into consideration the dynamic traffic. G-Tree is then utilized to efficiently compute the traveling time between any two nodes.

G-Tree Overview. G-Tree is an efficient tree index for KNN search on road networks. To utilize G-Tree, we need to define the distance metrics on the road network. Based on historical taxi driving records, the weight between two nodes on road network depends on not only the actual physical distance but also the driving speed, hence the weight is traffic specific. Thus the shortest path between two nodes in this road network has real-world meaning of least traveling time cost route in the current traffic.

G-Tree is constructed by recursively partitioning the road network. Each G-Tree node is a sub-network and each G-Tree leaf node is a set of nodes on the road network. A leaf node is considered on the border of its parent if it has direct edge to other outside leaf nodes. The shortest path between two border leaf nodes in each partition is calculated offline. For the G-Tree nodes with the same parent, we keep the shortest path distance between the border nodes of these G-Tree nodes. To search a shortest path between two leaf nodes p_1, p_2 with parent P_1 and P_2, it utilizes the offline calculated shortest paths between P_1 and P_2's border nodes and p_1/p_2's shortest path to P_1/P_2's border. The search

complexity is positively correlated to the height of the G-Tree and for more details, refer to [5].

A single G-Tree cannot meet our real-time route recommendation requirement, because it is constructed with only a snapshot of the traffic information. So, we have to extend G-Tree to be aware of traffic changes. We build one G-Tree instance at each snapshot of the traffic information within one interval as discussed in Section 2, based on the historical traffic model. By integrating these G-Tree together, we can support routes recommendation under different traffic.

Algorithm Implementation. The extended G-Tree is implemented by integrating multiple G-Tree for different intervals. With one instance of G-Tree for one interval, we can easily answer a real-time route recommendation occurred within a particular interval if the shortest path we find can be finished before the beginning of next interval. However, in many cases, the real-time route recommendation may occur at different intervals, and it takes several intervals before user finishes her route. In this situation, we have to update our real-time route recommendation based on user's current location and current interval using another G-Tree instance built with this interval's traffic information. Thus this is a history based greedy local-optimization algorithm.

3.2 Search Algorithm

Next we introduce the philosophy of our real-time route recommendation algorithm.

Real-time Algorithm. We organize the real-time route recommendation algorithm as two phrases: *expansion* phrase and *estimation* phrase. During the first phrase, we expand routes from the starting node of road network with current traffic condition. We only utilize the current traffic condition to expand the node within a limited distance (otherwise, the expansion loses accuracy). We call the expanded nodes *candidates*. In the second phrase, we first use the empirical model to estimate the rest traveling time from candidates to the destination, then combine the first phrase with the empirical model together to recommend best routes. It is worth noting that the weights of all edges in our problem is estimated, therefore the distance between two nodes is exactly the traveling time between the two nodes.

In particular, in the expansion phrase, we expand routes from the starting node within a limited time. Formally, given a time limitation UB(s), and a starting node s, we expand all the nodes c as candidates, where c satisfies

$$\texttt{Dist}(s, c) \leq \text{UB}(s).$$

After expansion, there are two possible cases for the destination (1) If the destination is in the candidate set, then we directly use the shortest route expanded as our recommendation since $\texttt{Dist}(s, e) \leq \text{UB}(s)$ and the destination is very close to the starting point. (2) Otherwise we consider the estimation phrase and use the empirical model to estimate the traveling time from candidates to destination (denoted by $\hat{\texttt{Dist}}(c, e)$). For each candidate, to use the empirical model, we set the time interval with the expanded time computed in the first phrase, i.e the current querying time with the addition of the obtained expansion time($\texttt{Dist}(s, c)$). Then we rank and select the top k routes by the order of

$$\texttt{Dist}(s, c) + \hat{\texttt{Dist}}(c, e). \quad (1)$$

Figure 2 illustrates our idea. Consider the starting node A and destination P. The weights of edges are corresponding traveling time. In the first phrase (suppose we set UB(A) =

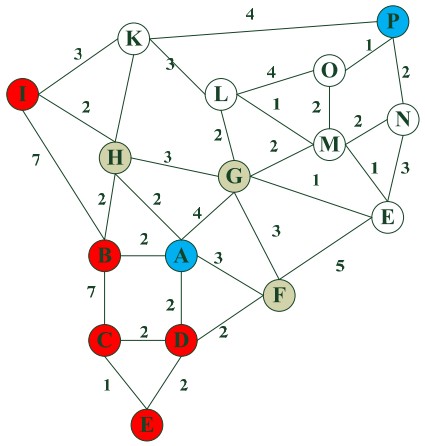


Figure 2: An Example of Real-time Routing

4), we will expand $\{B, C, D, F, G, H, I\}$ which has the shortest distance less than 4 from A as our candidates. In the second phase, we calculate the traveling time from those nodes to P with empirical model. At last, we rank the time (also routes) according to the combination of the two parts. For instance, suppose $k = 1$ and the empirical model computes route $\{G, M, O, P\}$ with time cost 5 and route $\{H, K, P\} = 8$ (also route from B, C, D, F, I), then we will connect and rank route $\{A, G\}$,$\{G, M, O, P\}$ as our top selection with the smallest time cost 9.

Expanding Candidates. A simple method to implement the first phase is using the `Dijkstra` algorithm [1] to expand the route from the starting node. However, the candidate set generated is very large and it is time consuming if we estimate rest traveling time for all of them. To address this issues, we propose a more elegant algorithm to select top k' candidates. Then we rank the top k candidates as results using equation 1. We give the pseudo code of generating candidates in Figure 3. The algorithm expands the route in a similar way of A^* algorithm [2]. The algorithm has two salient properties. First, it only adds the bordered nodes into the candidate set (lines 7-8, i.e., if node t is within UB (s) and node n expanded from t is also within UB (s), we do not need to keep t as a candidate). Second, it selects the possible candidates which is toward the destination instead of nodes back toward destination within the Euclidean distance (time) estimation(line 10), where $E\texttt{Dist}(t, e)$ denotes the traveling time by considering the Euclidean distance between node t and destination) dividing the average speed of vehicles.

For example, in Figure 2, the candidate set expanded from A is very large. However, we obtain a quite smaller candidate set(F, G, H) by utilizing our expansion algorithm and set k' =3. This is because our algorithm can judiciously select nodes toward the direction of destination before those nodes(B, C, D, E, I) back toward the destination.

4. EXPERIMENT

We compared our proposed method with Google Maps and Baidu Map on real datasets.

Datasets. The system is based on the basic traffic data of Beijing. (1) Road network data. The dataset of Beijing consists of 1,285,215 nodes and 2,690,296 directed edges, which forms a extremely large connected and directed graph. (2) Taxi data. We collected the driving data every second

Algorithm 1: Real Time Expansion

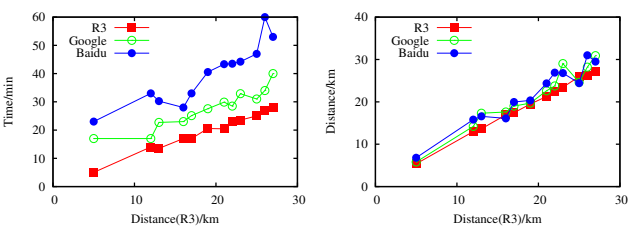
Input: s:starting node; e:destination; k':candidate number
Output: $\mathcal{C}$: candidate set

```
1  begin
2      PriorityQueue Q = C = ∅; visit(s) = true
3      while Q ≠ ∅ & |C| ≤ k' do
4          Queue Node t=Q.pop();
5          for Neighbor n of t do
6              Dist(s,n) = Dist(s,t) + Dist(t,n);
7              if Dist(s,n) > UB then
8                  C.Add(t);
9              else if !visit(n) then
10                 Q.Add(n, Dist(s,n) + EDist(t,e));
11                 visit(n) = true;
12     return C;
13 end
```

Figure 3: Real Time Routing Algorithm

for a year from about 8000 taxies in Beijing, and the total amount of the records reaches more than 1 billion. Each item of the records is consisted of taxi id, latitude, longitude, speed, etc., and we calculate the average speed of the vehicles on the same road in the same time as the historical traffic condition. Here, the fusion of road network data and traffic condition data enables us to recommend real-time route to drivers. (3) Real-time Traffic Data. We crawled the real-time traffic data from the government website. (4) GIS data. In order to obtain a better user experience, we support location-aware instant search based on more than 10 million POIs in China and the algorithm was proposed in [3, 4].

Comparison. We compare our route recommendation system R3 with Google Maps and Baidu Map. We randomly selected 1,000 pairs of sources and destinations, and compared the performance in Figure 4. To utilize the same traffic, we compared the three systems on the same time. The metrics of performance consists of time and distance. We got the estimate time and distance from the API of Google Maps and Baidu Map. To compare the performance, we group the results by the distance of R3 at 1km intervals. In each point, we show the traveling distance and time of different methods. As shown in Figure 4, although R3 had little advantages over the other two maps on distance, R3 took much less time than Google and Baidu. This is because we take the real-time traffic condition into consideration to reduce the average time of recommended routes. For example, on 20km distance, R3 took about 20 minutes while Google took 30 minutes and Baidu took 40 minutes. In average, our system can save 30-50% time compared with exiting systems.



(a) Time Comparison (b) Distance Comparison

Figure 4: Performance Comparison

5. SCENARIOS

Our system has many real-world applications and can be utilized in the following scenarios.

- GPS Navigation Devices – Our system can be obviously used in GPS navigation devices to help drivers select the best routes. Using our system, drives can not only save lots of time but also fuel.
- Intelligence Driving System – In the future, smart cars will be equipped with multiple sensors to capture driving information, e.g., traffic, and we must consider the real-time traffic condition for smart cars. In this way, our system can be utilized to analyze the huge amounts of data and provides the best route recommendation.
- Polices Pursue Prisoners – Our system can be used for polices to pursue prisoners who are escaping by cars. As our system supports traffic-aware route recommendation, it has higher probability to catch the prisoners and avoids accidentally injuring bystanders.

6. DEMONSTRATION

Our system has the following salient features.

- High Recommendation Quality – Our R3 system can recommend a route based on real-time traffic condition between the source and destination. Our system can recommend diverse routes for different users and thus can alleviate traffic pressure. We compare our system R3 with Google Maps and Baidu Map. The results show that our recommended routes are much better than those of Google and Baidu.
- High Recommendation Performance – We use the empirical model, the real-time model, effective G-Tree index, and effective pruning strategy to speed up the performance of the route recommendation algorithm.

7. ACKNOWLEDGEMENTS

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61272090 and 61373024, National Grand Fundamental Research 973 Program of China under Grant No. 2011CB302206, Beijing Higher Education Young Elite Teacher Project under grant No. YETP0105, Tsinghua-Samsung Joint Laboratory, Tsinghua-Tencent Joint Laboratory, the "NExT Research Center" funded by MDA, Singapore, under Grant No. WBS:R-252-300-001-490, and the FDCT/106/2012/A3.

8. REFERENCES

[1] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, 4(2):100–107, 1968.

[3] G. Li, N. Zhang, R. Zhong, S. Liu, W. Huang, J. Fan, K.-L. Tan, L. Zhou, and J. Feng. Tsingnus: a location-based service system towards live city. In *SIGMOD Conference*, pages 957–960, 2013.

[4] R. Zhong, J. Fan, G. Li, K.-L. Tan, and L. Zhou. Location-aware instant search. In *CIKM*, pages 385–394, 2012.

[5] R. Zhong, G. Li, K.-L. Tan, and L. Zhou. G-tree: an efficient index for knn search on road networks. In *CIKM*, pages 39–48, 2013.