2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Mixed Model For Prediction Of Bus Arrival Times

Jian Dong
Beihang University
State Key Laboratory of Software Development Environment
BeiJing,China
Ddongjian0000@gmail.com

Lu Zou
Beihang University
State Key Laboratory of Software Development Environment
BeiJing,China
zouluclara@gmail.com

Yan Zhang
Beihang University
State Key Laboratory of Software Development Environment
BeiJing,China
676710535@qq.com

***Abstract*—The public transport information has been focus of social attention, especially bus arrival time (BAT) prediction. Historical data in combination with real-time data may be used to predict the future travel times of vehicles more accurately, thus improving the experience of the users who rely on such information. In this paper, we expound the correspondence among real-time data, history data and BAT. Hence, we propose short distance BAT prediction based on real-time traffic condition and long distance BAT prediction based on K Nearest Neighbors(KNN) respectively. Furthermore, original matching algorithm of KNN is modified for two times to accelerate matching procedure in terms of computationally expensive queries. In empirical studies with real data from buses, the model in this paper outperforms ANN or KNN used alone both in accuracy and efficiency of the algorithm, errors of which is less than 12 percent for a time horizon of 60 minutes.**

Keywords—Bus Arrival Time; Non-parametric; KD Tree;

I. INTRODUCTION

For years, numerous urban public transportation systems deploy Automated Vehicle Location (AVL) to monitor the position of buses in real time. A number of applications in the areas of logistics, transit services, cargo delivery, and collective transport involve the management of fleets of vehicles that are expected to travel along known routes according to fixed schedules. The information of vehicle positions accumulated during this process from vehicle trajectories. A trajectory is a function that returns the position of a vehicle at given time as it traverses its route. Trajectories accumulated are valuable for subsequent research and for predicting travel times more accurately, thus enhancing user experience and reducing users' waiting time. Public transportation service already exists that manage vehicle positions in real time and predict the future arrival times at certain bus stops. In china, the public buses in HangZhou are equipped with PCs, GPRS-based connectivity to a cental server, and GPS devices for positioning. These systems also include on-line displays on electronic screens deployed both at major bus stops and traffic management department. While methods for bus arrival prediction are available today, deviation from actual arrival time and low coverage rate of bus stops for prediction lead to poor usability and bad user experience, which means demand for more accurate and general prediction. Also, faster machine learning algorithms are needed in terms of the big data problems ever-growing number of vehicles cause.

The challenge of predicting bus arrival times has attracted considerable attention in the past few years. [6] give an extensive recent literature survey on technologies emerged in the field of bus arrival prediction. Commonly used prediction methods include Kalman filtering (KF),e.g.,[9], and artificial networks(ANNs),e.g.,[7,10]. Gibbens and Saacti [8] consider predictions based on loop detector data, using linear regression and k-nearest neighbor methods. GPS-based predictions of arrival times are discussed in [5]. Weighted K-nearest neighbor algorithms and various methods to speed up the computations are introduced.

Reference [3] gives analysis on the prediction accuracy through compassion among delay based, Linear Regression, K-nearest Neighbor and Kernal Regression. Kernal Regression outperforms others while the authors adopt basic model and experiment on just one route, ignore running speed of programs as well. Similarity exists between long-term prediction and [1]. But, the main idea of [1] is finding nearest one from historical trajectories, in contrast, it is proposed in this paper that long-term bus arrival prediction performs accurately. Methods to expedite big data computations are introduced as well. KF works well when only short-term prediction is needed, but deteriorates when prediction is needed for more than one time step. ANNs and other learning methods are based on historical data. Their learning is computationally expensive; therefore, ANNs cannot be updated in real-time.

The problem addressed in this paper is two-fold. First, we analyze the law of bus arrival with distance to the target station and time horizon as arguments. The rules denotes that short-distance bus arrival time is correlated with current buses speed which passed the trajectory within 5 minutes. More specifically, after positioning target bus, we calculate travel time of stops in short-distance by live traffic information, and for stops in long-distance we retrieve k nearest neighbor trajectories calculate among the historical trajectories, using the available, partial real-time trajectory from the start of the route until the most recently visited timing point. Second, efficient, real-time access to similar trajectories is needed. kd tree as storage structure of trajectories are applied to accelerate the query speed of k nearest neighbors. We improve the structure design for twice to satisfy computations need.

The remainder of this paper is organized as follows. Section 2 covers related work in the areas of similarity search and vehicle travel time prediction. Section 3 defines the terms to be used throughout the paper and also the problem to be solved. Section 4 analyzes the distance measures and travel

time prediction using those, while Section 5 describes the data structure and algorithms for similarity search. Section 6 presents experimental results. Finally, Section 7 concludes.

II. DATA SET AND DATA MODEL

A. Data Set

Actually, Bus Arrival Time (BAT) prediction system, based on this paper , has been deployed in Bei Jing municipal committee, providing bus arrival prediction service for citizens. The system is responsible for over 15000 buses' BAT prediction in total and obtain GPS measurement every 20 second for single vehicle in near real-time which is transmit from BaFangDa Bus Company and Bus Company. More information on the collection and the preprocessing can be found in [2].

For analysis, we choose Bus line 944 (Naizifang to west of Laihuying Bridge) as sample. The exact route of bus line 944 is displayed in Fig.1. The dots correspond to the locations of bus stops; overall, there are 59 stops. Typically, this route cover crowed sections both in city center and suburbia and it is one of the longest route, approximately 45 km. Consequently, the sample line can cover algorithm scenario and check algorithm.

Fig. 1. Route of the Beijing Bus line 944

B. Data Model

There were three basic elements in our dataset: buses, stops, links. We define the set of buses as $B=\{b\}$, where b is a tuple(id_b , lat_b , lon_b , $timestamp$) containing a unique bus ID, id_b ;the latitude and longitude of the bus, lat_b , lon_b ; $timestamp$, the time sending back of the GPS item; Set of stops is symbolized as $S=\{s\}$, we use s to denote stop which is a tuple(id_s , lat_s , lon_s , Num_s , dir_s , dis_l). the first three element is similar to definition b ; dir_s denotes direction of the bus line s ; The set of link is symbolized as $L=\{l\}$,and a road is divided into numerous links by intersection and l is a tuple $(lat_{ls}, lon_{ls}, lat_{le}, lon_{le}, num_l, len_l)$, lat_{ls} and lon_{ls} stand for latitude and longitude of start point of l ,as the same way, lat_{le} and lon_{le} denote coordinate of the end point of l .

III. ANALYSIS ON BUS ARRIVAL DISCIPLINARIAN

Li [11] and his team proposed a BAT prediction algorithm based on average speed of road section and instant speed. Let v_{ai} denote the average speed of the road sections, v_r denote instant speed. With sections buses need covered are denoted by n , v is the average speed buses proceeds as predicted.

$$v=\frac{\sum_{i=1}^{n-1} v_{ai}+v_r}{n} \tag{1}$$

The algorithm indicate that v levels off to the average speed of all sections as bus go further apart from target stop; In opposite, v hinges on instantaneous speed while bus get closer to the target stop. The author resolve the problem that v_r can be zero. But, throughout the whole process, the speed of bus frequently changes and the accuracy of GPS instantaneous speed is low, in addition, history average speed cannot reflect traffic flow speed of current section. Therefore, the accuracy of algorithm above is low. Even so, we could also deduce that ,evidently, BAT approach historical law while bus is far away and get close to current traffic flow as bus get closer to the target stop.Meanwhile, in line with traffic flow theory, which contains probability, queuing theory, traffic flow wave theory, all of them reach an consensus that the traffic flow do not fluctuate wildly within short time. Specifically, we analyze the distribution of the bus traffic flow speed in Beijing , clustered in several patterns and displayed in Fig.2. The results are in accordance with theory.

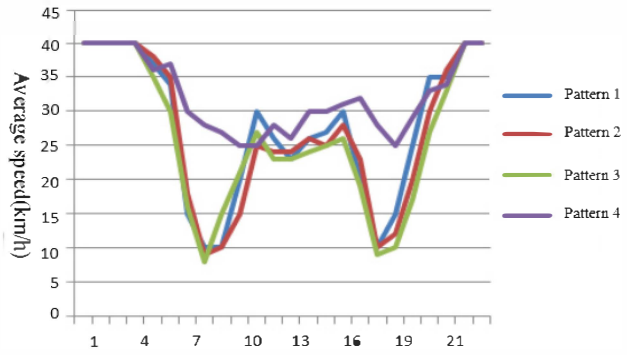

Fig. 2. Patterns of Traffic condition clustered

On the other hand, distribution of the bus arrival time, both at short and long range, illustrate that history law is obvious in long distance prediction but not applicable in short distance prediction since BAT fluctuates wildly as we can infer from Fig.3. With 3km/h as threshold, t denotes time consumed that transformation of traffic flow speed is greater than the threshold, and v denotes the traffic flow speed. Let maximum of $v\times t$, which is 3km approximately, stands for the shortest distance in which traffic flow is stable. Also, we can use it as the boundary to discriminate short distance and long distance BAT prediction.

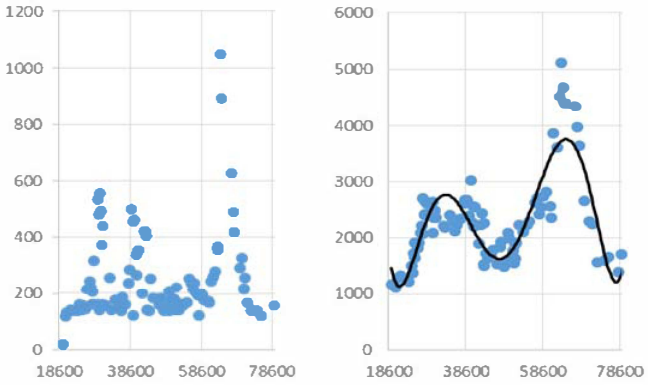

Fig. 3. Figure in left displays travel time between station 12 and 13 in 944 up line and figure in right displays travel time between station 12 and 24 in 944 up. Vertical axis denotes travel time measured in seconds. Horizontal axis denotes time in a day measured in seconds.

IV. SHORT DISTANCE PREDICTION

In this paper, we propose two different algorithms for predicting arrival times in terms of distance from bus to the target stop: a simple method based on traffic flow speed and a non-parametric approach using *K*-Nearest Neighbor. We begin by describing the first one.

The idea of short distance model is to calculate travel time on the basis of traffic flow speed of each link which I could get from Traffic speed computing system and we define it as BAT prediction based on Traffic condition. (BTC) We divide BAT into two parts: Travel time on the link and dwell time.

Traffic speed computing system utilize all the buses traversing the link l to calculate average speed in the last 10 minutes and update every 2 minutes, that is, not just buses affiliated to the bus line bl as input, which amplify real-time input dataset and improve accuracy. Assume that bus b is located on link m. Let vl denotes traffic flow speed of l. Considering that bus is not always at the beginning of the link, let L_{bm} stand for distance from location of bus b to the end of m.There are n links from the target stop s. Time consume on the link t_l can be expressed as :

$$t_l = \frac{L_{bm}}{vl_m} + \sum_{i=1}^{n} \frac{L_i}{vl_i} + \frac{L_{bs}}{vl_n} \tag{2}$$

Time delay t_{delay} on stop s which start on the arrival and ends on the departure from the station. Based on the buses time delay in front within 10 minutes, t_{delay} is calculated as following:

$$t_d = \sum_{i=1}^{n} \frac{T_d - T_a}{n} \tag{3}$$

Therefore , the BAT can be illustrated as :

$$bat_{short} = t_l + t_d \tag{4}$$

V. LONG DISTANCE PREDICTION

When distance from bus to target stop exceed 3km, short distance prediction model based on traffic flow speed fails to conduct an accurate prediction confronting with frequently changed traffic flow. BAT prediction based on Non-parametric regression has higher accuracy when there is abundant history data and can be conducted without priori knowledge and a handful of parameters distinguishing. But, non-parametric regression is computationally expensive. We modify the stucture of kd tree in K-Nearest Neighbor to reduce the amount of data that have to be compared and enhance the speed of the program .

A. Long Distance Prediction Solution

The idea behind prediction with KNN is to identify, in the time series, the past sequences which are the most similar to the current one, and to combine their future values to predict the next value of the current sequence. It contains six parts: high-dimensional database, state vector, distance measurement, choice of k, prediction function and feedback regulation. After that, the distance between a new state vector and historical state vectors can be computed based on the distance measurement method. Then the subsequent values of the k nearest vectors are selected to predict the BAT according to the predefined prediction function.

B. Defination of state vector and establisment of database

The essence of non-parameter regression is to extract the most similar k trajectories to predict the future trajectory. State vector is key to describe the similarity of all the trajectories and defined as following:

$$X_n = [R_1, R_2, ...R_{n-1}, D_1, D_2, ...D_{n-1}] \tag{6}$$

n stands for number of stations. $R_1, ...R_{n-1}$ denotes travel time among adjacent stations, and $D_1, ...D_{n-1}$ denotes dwell time at stations. We assume that k denotes the stops bus has passed. $\sum_{i=k}^{n} R_i D_i$ is set as 0. In general, we are trying to find neighbors through matching the current status to historical database.

State vector determines data in historical database. Errors brought by sensors and noise jamming have to be eliminated and more can be found in [2]. With errors filtered, a correct database can be achieved, but not qualified as a sample database for its data redundancy. Even one route hold over 20000 trajectories for one year history data which cost too much search time and occupy amount of storage space. Therefore, tactics for data compaction are needed, as well as to preserve the data diversity.

Data diversity indicates that buses' trajectories in various situations should be included in dataset. In contrast, data compaction is introduced in consideration of real-time

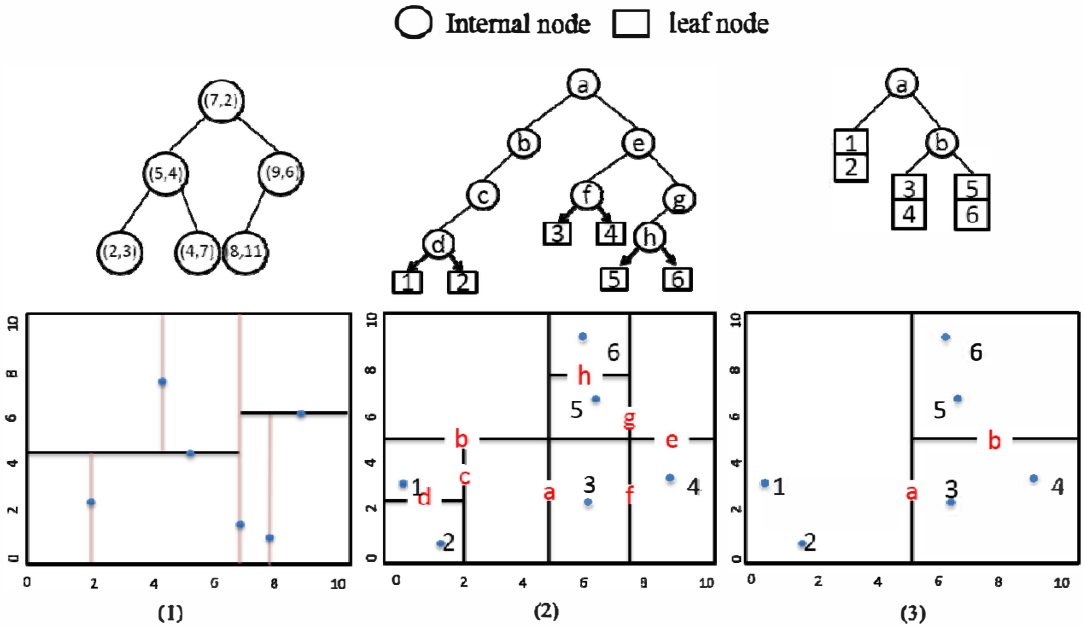

Fig. 4. Figure (1) displays original structure of kd tree. Structure of the first modification version is displayed in figure (2). The final version modified using bucket to store nodes is showed in figure (3)

performance, which imply that database can not be too large. To address the above requirements, we define intensity M

$$M = (\sum_{i=1}^{N} n_i) / N \tag{6}$$

as arithmetic average of all the neighbors ni of all points i in history database within radius R. For edge point, it is $2ni$. The radius R can be assigned in accordance with specific needs. E.g. in this paper, R is assigned 50, and M is intensity of points within 50. Assuming that M greater than 10 satisfy the base line of database, for every point i, the number of i's neighbors within 50 should be greater than 10 , otherwise more data should be joint. In a similar way, when number reach 50 for certain point i, no point is added into its scope.

C. Similarity mechanism and Predict Function

In this paper, weighting coefficients Euclidean distance is applied as similarity mechanism. From section above, we can discover that the variables exert different influences on the arrival time. Arrival time at previous station weighs most, travel time secondly and dwell time has the minimal impact relatively. Consequently, Ordinary Euclidean distance can not reflect differences that different parameter impacts on BAT prediction. We adopt weighting coefficients Euclidean distance:

$$d = \sqrt{\frac{r_R \sum_{k=1}^{n-1}\sum_{j=1}^{N}[R_{n-1} - R_{j,n-1}]^2 + r_D \sum_{k=1}^{n-1}\sum_{j=1}^{N}[D_{n-1} - D_{j,n-1}]^2}{r_R + r_D}} \tag{7}$$

To utilize neighbor points in various distance, inverse weighted average method is introduced. $T_{i,k+n}$ denotes the arrival time at $k+n$ station of the ith neighbor. β_i represent the proportion that various neighbors account.

$$T_{k+n} = \sum_{i=1}^{K} \beta_i T_{i,k+n}, \beta_i = \frac{d_i^{-1}}{\sum_{i=1}^{K} d_i^{-1}} \tag{8}$$

D. Improvement on real-time performance

Although non-parameter regression method construct with less variables and can be transplantable, computational expense linear match algorithm leads to make little sense for a real-time application. In this paper, we introduce kd tree to improve efficiency of matching. Specifically, each bus line, either in up or down, correspond to a prediction entity of which the priority of initialization is to construct kd tree in main-memory according to the database file. And then travel time, dwell time both extracted from database and record real-timely compose state vector.

In contrast with complexity of BBD tree's implementation, refactor when vp tree confront data changes, kd tree is more suitable for real-time system. From Fig.4 (1), we could obtain the original structure of kd tree. However, original structure which use both internal node and external node to record path not only increase the complexity of each node and make node cumbersome but also reduce the efficiency of traverse. Based on this, consider actual demand, we refactor kd tree: first, internal node do not store record, only for keywords and layers that splice the records; second, all the record are deposited in the leaf node, which simplify and accelerate the query procedure by comparing on the internal node and calculating on the leaf node. Structure after refactoring is displayed in Fig.4 (2)

However, confronting with a huge mount of history data,

nodes of kd tree are tremendous, which means numerous layers. More specifically, the tactics adopt in the procedure of creating kd tree is that Euclidean space is split continuously until only one point exist in split space. Thus, it spilt the space into the minimal and can obtain the accurate matching point from a query. But in another way, with depth of the tree increasing and space of tree node reduce exponentially, layers that space split balloons exponentially, which is bound to increase the comparison times and reduce search efficiency. To tackle the problem, we refactor the tree above and transform leaf node into bucket node as it is displayed in Fig.4 (3). Through the melioration, layers of tree in Fig.4 (3) are less than the previous than original.

VI. EXPERIMENT

The data we used in our experiments are 3982 trajectories of the Beijing Bus line 944. See Fig.2 for some sample trajectories and section 2 for more background information.

A. Evaluation of accuracy

To evaluate the performance of the prediction method, the differences between the actual arrival time and the predicted one need to be measured. The absolute percentage error (APE) and the absolute error(AE) are introduced to evaluate our prediction algorithm.

$$APE = \frac{\left|\sum_{i=1}^{N} E(i) - \sum_{i=1}^{N} T(i)\right|}{\sum_{i=1}^{N} T(i)} \tag{9}$$

$$AE = \frac{\left|\sum_{i=1}^{N} E(i) - \sum_{i=1}^{N} T(i)\right|}{N} \tag{10}$$

APE is used to evaluate long distance prediction, in which the proportion difference count for value most. AP is to evaluate short distance prediction for that, occasionally, high percentage error account for does not correspond to large error, passenger cannot perceive in practice.

Moreover , To verify algorithm in this paper, we introduce Dr.Ran's doctoral dissertation[4] that concerns machine learning prediction algorithm based on neural network for that the application scenario of Ran's dissertation is same with ours. The model of Dr.Ran adopt a 3 layer $18 \times 37 \times 15$ neural network and Levenberg-Marquardt optimization algorithm as training function.

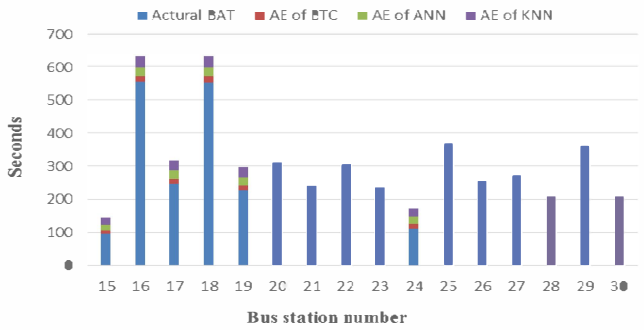

Fig. 5. BAT time and errors of BTS,ANN and KNN during travel in city at peak time

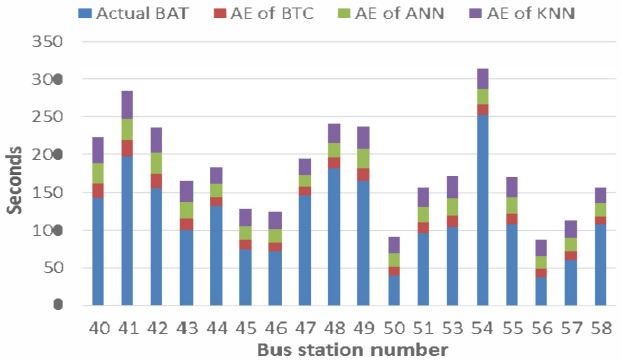

Fig. 6. BAT time and AE of BTS,ANN and KNN during travel in suburb

As displayed in Fig.5 and Fig.6, short distance prediction conducts on certain stops, respectively, AE of BTC in red, ANN in green and KNN in purple. It can be inferred apparently from Fig 10 that BTC outperforms in accuracy that reach 17 seconds in average and has the characteristics of less computation, easy implementation relative to the KNN and ANN methods. Focus on KNN method, the accuracy of the prediction based KNN improved slightly with accumulation of the path when bus travel to the stops away from the beginning. The results further prove the necessity of the introduction of the prediction method appropriate for short distance and that prediction based on traffic condition can both enhance the accuracy and reduce computation.

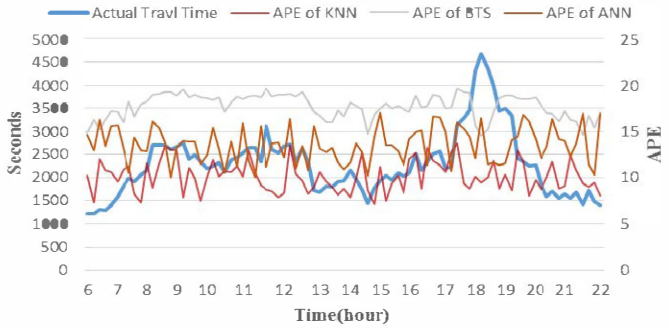

Fig. 7. APE of three algorithms. Actual travel time displayed in contrast

The experiment in Fig.7 is set as the bus travel time prediction between the 12th stop and 26th stop, aiming at evaluating the performance of the three algorithms in long distance prediction. From the results, KNN outperforms in accuracy among the three algorithms, ANN next and BTC worst. The errors of BTC increase at the switching time from flat to the peak or in reverse, which further proof limit of the algorithm and necessity of the long distance prediction. The error of KNN model is 7% in average and not greater than 12% in worst case which has better performance than ANN. In addition, ANN is not appropriate for real-time application for its complex and time-consuming training.

B. Evaluation of accuracy

The BAT prediction system based on the model in this paper support over 15000 buses' BAT prediction, which of algorithm efficiency is vital. As mentioned in Section II, the rate for Bei Jing buses is typically one update per 20 seconds. Thus, BAT prediction system is confronted with 15000

TABLE I. EVALUTION RESULTS OF PERFORMANCE

Algorithms /Structures	*Conditions*			*Time (millisecond)*
	Node number	*Dimensions*	*Neighbors Num(K)*	
Liner	30,000	10	10	45
	300,000	15	10	550
Dynaimic cluster	30,000	15	10	57
	300,000	30	10	634
Kd tree (First Version)	30,000	15	10	21
	300,000	30	10	1039
Kd tree (Second version)	30,000	15	10	2
	300,000	30	10	11

predictions within 20 seconds, which leads to bottleneck for nearest search. With algorithms described above, we add linear search and dynamic cluster algorithm into our experiment as comparisons. Dynamic cluster algorithm is a method to accelerate the procedure of the KNN. It is implemented by clustering the history data in k-means, then do multiple linear regression analysis on the parameter and establish hash function to allocate input to the address of the center of the cluster, eventually accelerate the matching procedure. The computer in experiment is equipped with core Duo 2.5GHz and 4g RAM.

As Table.1 displays, it is demonstrated that linear search is not available as the dimensions rise. It consumes about half second to find the 10 nearest neighbors as dimension reach 15 and the total sample points are 300000.Secondly, dynamic cluster has remarkable promotion on the search speed as it only take half second to finish finding 10 neighbors when dimension comes to 30. Finally, the model modified first time accelerates the procedure as we expect, but not outstanding, worse than dynamic cluster. And the efficiency decrease a lot when dimensions get higher and iteration times increase. The model after second improvement outperforms all above with 1 millisecond to get the 10 nearest as shown in Table.1, which illustrate second modification is necessary. In practice, the system consumes 5 seconds to conduct prediction for 15000 buses.

VII. CONCLUSION

The experimental results of this paper demonstrate the necessity that divide BAT prediction to the short distance and long distance. The strong performance of the algorithm in this paper clearly outperform ANN and KNN alone in both accuracy and efficiency of the algorithm. Furthermore, the paper elaborate reasons to apply short distance and long distance prediction, and ways to accelerate nearest neighbor matching of the non-parameter regression when KNN is applied as the matching mechanism. Also, the system is deployed in Bei Jing transportation committee to predict BAT based on real-time updates. Note that computing KNN requires a quantity of memory space to store the sample points. We plan to investigate further strategies to reduce the computational burden, both in terms of memory and computational time. Otherwise, further problems are how to integrate other regression variables and how to cope abnormal conditions (e.g. feedback mechanism needed).

REFERENCES

[1] Tiesyte, Dalia, and Christian S. Jensen. "Similarity-based prediction of travel times for vehicles traveling on known routes." Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems. ACM, 2008.

[2] Zhu, Tongyu, et al. "The bus arrival time service based on dynamic traffic information." Application of Information and Communication Technologies (AICT), 2012 6th International Conference on. IEEE, 2012.

[3] Sinn, Mathieu, et al. Predicting arrival times of buses using real-time GPS measurements. Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on. IEEE, 2012.

[4] Ran J. The prediction of bus arrival time using automatic vehicle location systems data. Texas A&M University, 2004

[5] D. Tiesyte and C.S. Jensen: Similarity-based prediction of travel times for vehicles traveling on known routes. Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems, Irvine, California, 2008

[6] B. Yu, W.H.K. Lam and M.L. Tam: Bus arrival time prediction at bus stop with multiple routes. Transportation Research Part C: Emerging Technologies, 2011.

[7] S. I.-J. Chien, Y. Ding, and C. Wei. Dynamic bus arrival time prediction with artificial neural networks. Trans. Engrg., 128: 429–438, 2002.

[8] R.J. Gibbens and Y. Saacti: Road traffic analysis using MIDAS data: journey time prediction. Technical Report UCAM-CL-TR-676, University of Cambridge, Computer Laboratory, 2006.

[9] D. Dailey, S. Maclean, F. Cathey, and Z. Wall. Transit vehicle arrivalprediction: An algorithm and a large scale implementation. Transp.Res. Rec., Transportation Network Modeling, pp. 46–51, 2001.

[10] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. Journal of Comp. and Syst. Sciences, 66(4): 614–656, 2003.

[11] Lin, Wei-Hua, and Jian Zeng. "Experimental study of real-time bus arrival time prediction with GPS data." Transportation Research Record: Journal of the Transportation Research Board 1666.-1 (1999): 101-109.