國立交通大學

資訊科學與工程研究所

碩 士 論 文

一個在行動感測網路中的品質導向資料蒐集機制

MobiQC: A QoS-aware Data Collection Framework in Mobile Sensor Networks

研 究 生：郭員榕

指導教授：彭文志　教授

中 華 民 國　九 十 七　年 六 月

一個在行動感測網路中的品質導向資料蒐集機制
MobiQC: A QoS-aware Data Collection Framework in Mobile Sensor Networks

研 究 生：郭員榕　　Student：Yuan-Jung Kuo

指導教授：彭文志　　Advisor：Wen-Chih Peng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

摘　　要

行動無線感測網路已經廣泛地使用在許多方面，如動物追蹤和環境監測。有別於傳統靜態的無線感測網路，由於行動無線感測網路受到它本身的行動性和無線存取點的佈署範圍限制，它會有間斷性連結的問題。另一方面，一般在無線感測網路中，對於資料蒐集的架構是採取集中式架構，再加上大量的資料以及有限的無線頻寬，所以在這樣的環境下要蒐集所有的資料是一件很困難的任務。因此，降低資料上傳量也是一個很重要的議題。

在本篇論文中，我們提出了一個在行動感測網路中的品質導向資料蒐集機制，簡稱 MobiQC。我們針對如何從行動感測裝置蒐集資料的問題來做設計，使得蒐集到的資料可以符合給定的服務品質(QoS)規範，並且降低傳輸到伺服器的資料量。在 MobiQC 裡，我們使用了線性回歸模組以及核回歸模組的技術；同時，我們也使用了在網路中的集成法。透過以上的技術，我們可以降低資料傳輸量並保證資料的品質，且我們的實驗顯示 MobiQC 是有效用的。

關鍵字：間斷式連接、行動感測網路、回歸模組

Abstract

Mobile sensor networks have been widely used in many applications, such as animal tracking and environment monitoring. Different with the static one, it suffers from the problem of intermittent connection due to its mobility and the coverage of wireless network. On the other hand, since the data collection performed in a mobile sensor network usually follows a centralized infrastructure, collecting all data from mobile sensor nodes is a hard task due to the large quantities of data and limited wireless bandwidth. Therefore, decreasing the amount of data uploaded is also an important issue. Moreover, in many applications, it is sufficient to support approximate data collection by tolerant errors.

In this paper, we proposed a propose a QoS-aware data collection framework in mobile sensor networks (MobiQC). We focus on the problem of data collection from mobile sensors such that the quality of data satisfies given QoS specifications and the amount of data transmitted to the server is reduced. In the MobiQC, we use both linear and kernel regression model to reduce total transmitted data and in-network aggregation is employed to further reduce the amount of data while guaranteeing the quality of data. Our experiments verify the effectiveness and the efficiency of our proposed MobiQC.

Keywords — intermittent connection, mobile sensor networks, regression

致　　謝

經歷了兩年的碩士，終於走到了今天，這是一段很扎實的學習過程。首先我要感謝我的指導教授 – 彭文志老師。我是中途改變志向從別的實驗室加入他的實驗室，但是由於老師熱血且積極的指導以及照顧我，使我能在短時間內跟上同學的進度，並在此時此刻，完成我的碩士論文。和老師討論時，會感覺沒有很大的壓力，能夠盡情的將自己的想法表達出來；同時，老師也給與了我不少的想法批評以及指教。其次，我還要感謝我的口試委員蔡明哲教授以及黃俊龍教授，在口試的時候提供了許多的意見，讓我可以針對論文做更進一步的改進，從中也學習到一些該注意的事項和細節。另外，我也想感謝我的前指導教授– 鍾崇斌老師，在他那邊我學習到做事情應有的謹慎和態度，讓我能夠重新審視並補足自己不足之處。

在實驗室中，首先我要感謝我的博班學長 – 洪志傑。感謝他對於我的論文提供了許許多多的意見，也幫助我修飾我的論文。其他的學長姐 – 01、Young、Dimension，都給與我不少學習或研究上的幫助。其次，也感謝我的同學們 – Paul、fcamel、sheep、York、P 嫂以及 Jalamorm，和大家一起打牌、打球、討論以及做研究真的很快樂，讓我的碩士生活不會枯燥無聊。真的很難得可以遇到這些好同學好朋友，希望在畢業之後還是能夠常常和大家見面，就像現在一樣。

最後要感謝我的家人，不論是爸媽或是姐姐，都在我的背後支持我。雖然在平日在忙的時候，會覺得他們對我的關心造成我的困擾，但是他們對我始終如一，在我壓力大或是煩惱的時候，總是給與我最大的支持和鼓勵。非常羞愧也非常的感謝他們，有他們在真好。

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent advances in micro-sensing MEMS and wireless communication technologies have motivated the development of wireless sensor networks. Over the past few years, a significant amount of researchers have involved into exploiting wireless sensor networks into variety of applications in real world, such as border detection, environment monitoring(1), smart home, military surveillance, and to name a few. Due to resource constraints in wireless sensor networks, prior works have elaborated on in-network aggregation and data stream techniques in sensor data collections so as to reduce the energy consumption of wireless sensor networks. The idea of treating a sensor network as a streaming data repository over which one can execute data collection, with optimizations such as in-network aggregation, is now well-established(2)(3).

Recently, sensors with the mobility capabilities are widely studied. Such a sensor network is called a mobile sensor network. A mobile sensor network is consists of sensor nodes with mobility and mobile sensor networks, therefore, improve the scale and fidelity of spatiotemporal sensing of a wide range of the important phenomena. Due to the mobility of sensor nodes, we can expect that deploying such a sensor network cannot only be substantially cheaper to cover a wider area than a comparable static one but also perform more efficient data collection from the monitoring regions. In this paper, the data collection scenario for a mobile sensor network is shown in Figure 1.1. Explicitly,

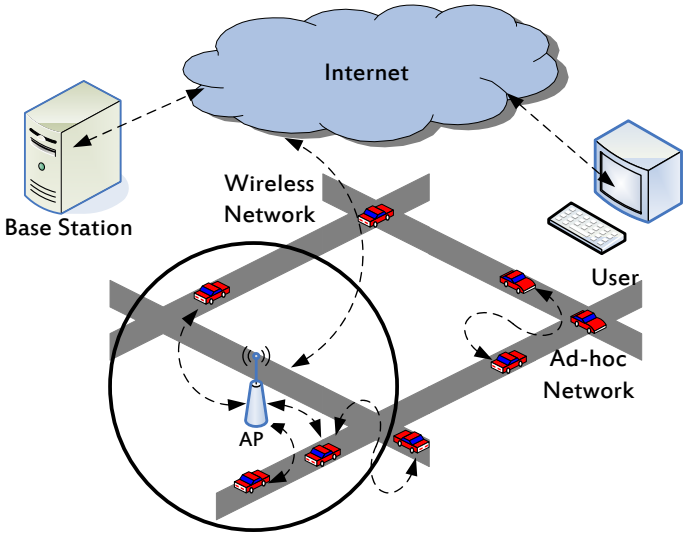


Figure 1.1: Data collection in a mobile sensor network

a mobile sensor is usually able to collect multiple types of data by sensors equipped on it. Mobile sensors form an ad-hoc network such that they can communicate with each other. Data collection in a mobile sensor network can usually connect to *access points* for uploading sensing data. A mobile sensor can deliver data to a *centralized server* by wireless networks(e.g., Wi-Fi, wide-area cellular networks). In light of the benefit of mobile sensor networks, various applications are developed. For example, the CarTel(4) system is a mobile sensor computing system designed to collect, process, deliver, and visualize data from sensors located on mobile units such as automobiles. The CarWeb(5) platform borrows the concept of Web 2.0 to collect sensing data from each car equipped sensors. Once sharing their sensing data, CarWeb can exploit them to analyze real-time traffic information.

Unlike traditional static sensor networks, since each sensor node is installed in a mobile unit which can supply enough energy(4)(5) or is designed to recharge battery automatically(6), energy preservation is not the main issue for data collection in the scenario of CarTel, where sensors are powered by cars. However, there are still several issues for data collection to be addressed in a mobile sensor network. First, a mobile sensor cannot always connect to these access points due to the limited coverage and the

spotty connectivity of wireless networks. For example, in a wide-area cellular wireless infrastructure, cellular ”holes” are common and bandwidth over these networks is low. Thus, the connectivity between a mobile sensor and an access point is possibly intermittent. Second, since the data collection performed in a mobile sensor network follows a centralized infrastructure, it might not be possible to send all the data collected since the quantities of data are relatively large to the bandwidth of the network and the loading of the server. Moreover, the large amount of data transmission also induces the probability of collisions in wireless networks such that mobile sensors are harder to set up connections. Therefore, decreasing data amount uploaded not only benefits the connection quality of the wireless networks but also reduces the network bandwidth and the server load. Third, in many applications of mobile sensor networks, we can observe that it is sufficient to support detailed data collection while guaranteeing tolerant error in sensing readings. For example, a traffic analyzed and monitoring system(7) may analyze the traffic status using history detailed data and response queries from users about the current traffic flow on some roads. In such a scenario, it is not necessary for each mobile sensor to upload all of its raw data. Instead, the server can derive or estimate these information by *models* which can describe the sensed data under given tolerant error.

In this paper, we propose a QoS-aware data collection framework in mobile sensor network (MobiQc). In MobiQC, we focus on the problem of data collection from mobile sensors such that the quality of data satisfies given QoS specifications and the amount of data transmitted to the server is reduced. To realized this concept, MobiQC hybrids both proactive and reactive approaches into data collection. Specifically, mobile sensors builds models, which can guarantee these QoS specifications, for their sensed data. Moreover, mobile sensors can perform in-network aggregation such that the similar data can be aggregated into a mobile sensor which can frequently connect to access points. Through the data modeling and in-network aggregation approaches, the amount of data uploaded from mobile sensors can be efficiently reduced. Moreover, the server

can obtain the approximation of sensed data from mobile sensors via these models. The server will also maintain a statistical structure about collected data which can automatically verify whether it needs to trigger for collecting more data from mobile sensors or not. Once the current data distribution and the past one are similar, it is not necessary to trigger data collection such that the network bandwidth and the server loading can be reduced. Extensive experiments are conducted and experimental results shows the effectiveness and the efficiency of our proposed MobiQC.

The rest of this paper is organized as follows. In Section 2, related works are presented. Our proposed framework MobiQC is described in Section 3. In Section 4, we present our dual regression model. The in-network aggregation algorithm is proposed in Section 5. Experimental results are shown in Section 6. This paper concludes with Section 7.

Chapter 2

Related Work

For the mobile sensor networks, the authors of (4) and (8) have developed a centralized query processing system in CarTel and ICEDB where sensors are installed on cars and Wi-Fi is used to transmit data to the server. During the transmission, they also judge the transmission order of data by global prioritization which builds up a summary report of data. According to such a report, the base station can determine the transmitted order of data. On the other hand, authors in (9) have proposed future trend and challenge for traffic information system where information sharing is based on a peer-to-peer network instead of the centralized one.

Model-driven data collection is performed by building probabilistic model(10)(11). The authors in (10) explored a model-driven architecture in which a centralized probabilistic model is used to estimate the readings of sensor nodes by generating an observation plan to collect appropriate readings of sensor nodes. Also, the authors of (11) exploit spatial correlation for approximate data collection where a replicated dynamic probabilistic model is build for each clique to minimize the communication from sensor nodes to the sink. Approximation data collection is performed without building probabilistic model(12)(13). In (12), the author exploits spatial correlation for an extension of declarative query in sensor networks, called snapshot queries. The snapshot queries can be answered through a data-driven approach by using a linear regression model to

predict the readings of 1-hop neighbors. The authors in (13) proposed algorithm EEDC that is executed in a centralized server. Based on spatial correlation, EEDC partitions sensor nodes into disjoint cliques such that in the same clique have similar surveillance time series and apply round-robin scheduling to share workload of data collection.

Chapter 3

The MobiQC Framework

The MobiQC framework aims at performing data modeling to guarantee the overall quality of collected data, based on QoS specifications and subject to intermittent connectivity and constraints in network bandwidth and server loading. To achieve this goal, the core idea of MobiQC is that a mobile sensor maintains several models for the sensed data. Rather than transmitting raw data, a mobile sensor uploads the models to the server and the server can apply the models to derive the sensed data and also satisfy QoS specifications. The network bandwidth and the server loading can be preserved. Furthermore, by observing that nearby mobile sensors usually have the similar readings with higher opportunity, in-network aggregation can be performed within mobile sensors to further bring bandwidth reduction.

Our proposed MobiQC framework is shown in Figure 3.1. MobiQC hybrids both push and pull approaches for QoS-aware data collection. A mobile sensor will proactively upload its *linear regression model* which is used to model the trajectories. According to these linear regression models, the server can obtain spatio-temporal data points of mobile sensors. The *statistics manager* is used to evaluate whether it needs to collect more data from mobile sensors in some specific areas or not. The server will require a mobile sensor to upload its *kernel regression model* which is used to model some interested attributes such as speed of a mobile sensor, if the statistics manager discovers that

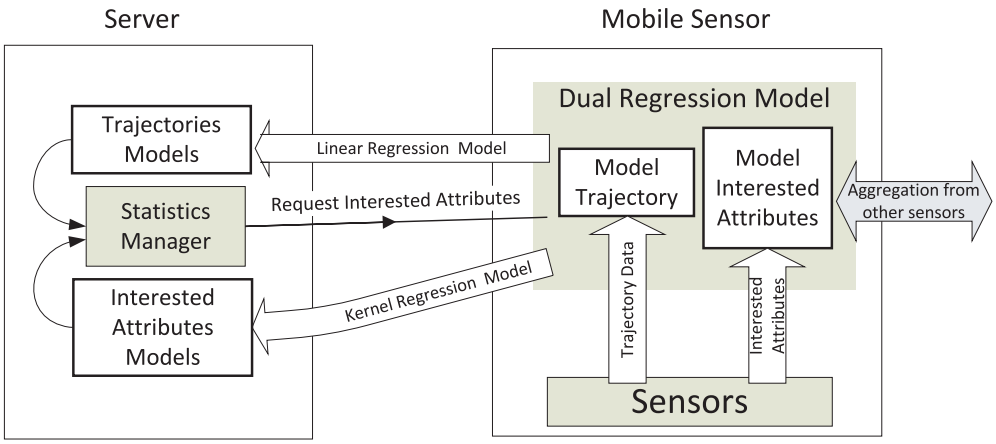


Figure 3.1: The MobiQC framework

the past and current statistical properties, such as the speed histogram, of the interested attributes in some specific area may be different. The detail of the functionality of dual regression model and statistics manager are listed as follows:

Dual Regression Model

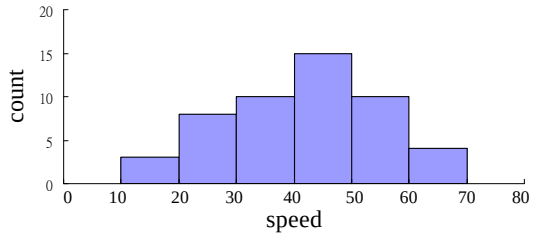
In MobiQC, a mobile sensor keeps sensing the environment via its sensors. According to the type of sensed data, sensed data can be divided into two groups: *trajectory data* and *interested attributes* since they own different properties so that we must deal with them into two cases. For trajectories of a mobile sensor, they can usually be approximated by several straight lines (14). Thus, *linear regression* can be used to model the trajectories of a mobile sensor. On the other hand, *kernel regression* is appropriate to model other interested attributes which might be hard to find the underlying distribution. Moreover, kernel regression allows that the server can obtain a rough view about the interested data when only receiving partial information from a mobile sensor, and incremental refine the model when receiving new information. Thus, it can benefit to those real-time applications under an intermittent-connectivity environment. A mobile sensor will model these data in a *dual regression model* component.

Since responsible to model the sensed data, the dual regression model should gen-

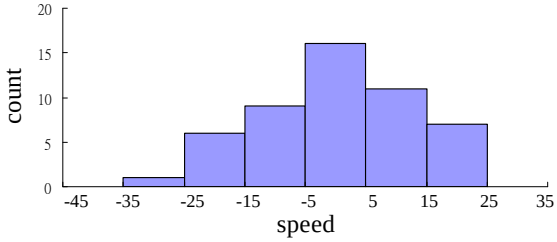
erate curves to fit these sensed data under given QoS specifications. Formally, let a set of data points Ψ can be represented as $\{(X_1, I_1), ...(X_n, I_n)\}$ where a *feature points* $X_i = (X_{i,1}, X_{i,2}, ...X_{i,d})$ is a point with d attributes with an *interested value* I_i. Thus, given a set of data points Ψ and a QoS specification ϵ, a mobile sensor is required to derive a model $\hat{f}$ for each interested value such that the root mean square error (RMSE) is bounded within ϵ, i.e., $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(I_i - \hat{f}(X_i))^2} \leq \epsilon$. For example, suppose that the data points collected by a mobile sensor is a tuple of (t_i, x_i, y_i, s_i) which denotes the speed of mobile sensor is s_i at (x_i, y_i) in time t_i. When modeling trajectories of a mobile sensor, we shall select the feature point as $X_i = t_i$ and the interested value is $I_i = (x_i, y_i)$. When modeling the speed of a mobile sensor, we may choose the feature point to be $X_i = (x_i, y_i, t_i)$ and the interested value $I_i = s_i$.

Statistics Manager

In the server side, a statistic manager is responsible to maintain a statistical structure about the collected data. A statistics manager will trigger data collection every fixed period if necessary. Note that the statistics manager can obtain the location and interested values of mobile sensors in some specific areas via models uploaded from mobile sensors. Therefore, the statistics manager first transform these values to a histogram, such as the one shown in Figure 3.2(a). To simplify computation, the histogram will be shifted to the center being be zero as shown in Figure 3.2(b). The statistic manager will compare whether the history histogram and the current histogram for the current collected data are similar or not. If the two histograms are similar, the sever should not collect data from mobile sensors anymore. Note that there are several proposed approaches to compare the similarity of two histograms(15)(16)(17). In this paper, we will use the Chi-square test for the statistic manager since it is not affected by the amount of total history data. Once the current data points can pass the Chi-square test according to the history data, which means that the distribution of the current data points and the history data points



(a)



(b)

Figure 3.2: Example of representing data into histograms

are the same, the server does not require mobile sensors to upload their models, and vice versa. Through the histograms comparison, the server loading and the bandwidth can be further preserved when the interested values in a monitoring region are stable. On the other hand, while special events happen in an area, the statistic manager can detect it since the current data distribution may differ from history data. For example, the statistic manager would like to collect much more data from sensors if it detects that the current speed distribution is different from the ordinary one if traffic accident happens.

Chapter 4

Dual Regression Model

In this section, we present Dual Regression Model for MobiQC. First, Algorithm LR is used to model trajectories by linear regression model. Then, Algorithm KR is proposed to model other interested attributes by kernel regression model.

4.1 Algorithm LR: Modeling Trajectories

To reduce the bandwidth involved by uploading its location data, a mobile sensor first models its location information, including longitude and latitude, into a *linear regression line* CL and then upload it to the server. The uploaded regression line is called SL. It is trivial that if we model all data of the sensor node into just one line, as shown in Figure 4.1, the RMSE of this line would not satisfy the given QoS specification so that it cannot capture the moving behavior of the sensor node. Thus, we need to separate trajectory data into segment lines. So, as long as the server can use SL to capture the current moving behavior of this mobile sensor, this mobile sensor does not need to upload any data to the server until SL cannot capture its moving behavior, thus reducing the bandwidth for uploading the location data. On the other hand, since the moving behavior of a mobile sensor may change with time passing by, it may be no longer captured by SL. In this case, a mobile sensor will derive a new CL and update the SL in the server side. To

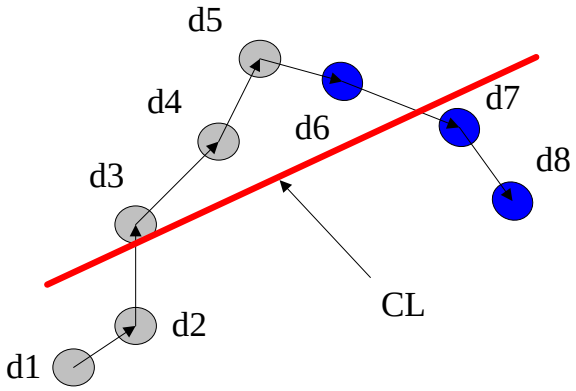


Figure 4.1: Example of modeling all trajectory data into just one line

realize the concept above, the following two steps are executed sequentially when a new data point comes.

In the first step, a mobile sensor derives CL according to the current data points. Once CL can represent the current moving behavior, a mobile sensor will upload to the server. In the second step, a mobile sensor detects the variation of moving behaviors and derive the new regression function if necessary.

Part 1: Deriving and Uploading the Current Regression Line

Since the locations of a mobile sensor are relevant to time, we can represent the locations with respect to time into a sequence of data points $(t_1, x_1, y_1), ...(t_m, x_m, y_m)$ where x_i and y_i represent the longitude and latitude of the mobile sensor at time t_i. Suppose that $(X(t), Y(t))$ be the coordinate of a mobile sensor in time t. We intend to derive two vectors (a_x, a_y) and (b_x, b_y) such that CL can be represented into $(X(t), Y(t)) = (a_x, a_y) \times t + (b_x, b_y)$ when t is in the valid time interval $[t_1, t_m]$.

Given a sequence of data points $(t_1, x_1, y_1), ...(t_k, x_k, y_m)$, the following terms are defined:

$$H = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix}, F = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_m & y_m \end{bmatrix}, A = \begin{bmatrix} a_x & a_y \\ b_x & b_y \end{bmatrix}$$

By solving the equation $A = (H^T H)^{-1} H^T F$, we can guarantee the *RMSE* is minimized.

Once obtaining the coefficient matrix A, the server can derive a regression line SL with valid time interval $[t_1, t_m]$ for a mobile sensor. To simplify our discussion, uploading the coefficient matrix of a regression line L to the server is abbreviated as uploading L to the server for the rest of this paper. However, we can observe that if there are a few data points collected in a mobile sensor, a new coming data point may affect the direction of CL significantly. For example, from Figure 4.2(a) and 4.2(b), we can see that the direction of CL changes a lot. In this situation, keeping uploading CL cannot benefit the server for modeling the moving behavior of the mobile sensor. To prevent the waste of bandwidth, a mobile sensor should keep collecting data points until the direction of CL does not change dramatically, i.e., CL can represent the longer-term moving behavior. To achieve this goal, a mobile sensor maintains the regression line LL derived in the last time point to ensure that the direction of CL is going to be stable. Explicitly, the direction of CL is considered to be stable if the direction of CL and LL are similar. To indicate the direction of CL and LL, we first define the direction vectors as follows:

Definition 1. Direction Vector:

Suppose that the coefficient matrices a linear regression line L are $A = \begin{bmatrix} a_x & a_y \\ b_x & b_y \end{bmatrix}$. The *direction vectors* of L, denoted as $\overrightarrow{v_L}$, is the vectors parallel to L. That is, $\overrightarrow{v_L} = (a_x, a_y)$.

To distinguish whether the direction of CL and LL are similar or not, we can com-

pute the cosine similarity between their direction vectors $\overrightarrow{v_{CL}}$ and $\overrightarrow{v_{LL}}$ as follows:

$$sim(\overrightarrow{v_{CL}}, \overrightarrow{v_{LL}}) = \frac{\overrightarrow{v_{CL}} \cdot \overrightarrow{v_{LL}}}{|\overrightarrow{v_{CL}}||\overrightarrow{v_{LL}}|}$$

Given a threshold α, if $sim(\overrightarrow{v_{CL}}, \overrightarrow{v_{LL}}) > \alpha$, a mobile sensor will upload CL and the corresponding starting time t_1 to the server. The server can derive a regression line SL which can capture the moving behavior by CL uploaded by a mobile sensor.

Step2: Detecting Variation

With time passing by, a mobile sensor collects more and more data points. However, the trajectory of a mobile sensor may become complicated and thus need be described by several linear regression lines rather than one. For example, in Figure 4.2(c), it can be seen that this mobile sensor makes a turn between time 5 and 6, and the original SL cannot represent its moving behavior after time 6. Thus, a mobile sensor should detect when its moving behavior changes and select a *break point*. According to such a break point, a mobile sensor can decide which data points can be used to derive a new CL representing its current moving behavior precisely. Thus, the server can capture new behavior by deriving SL according to the new regression line uploaded by a mobile sensor.

However, deciding a break point is not a trivial task. A break point should make a mobile sensor able to derive a precise CL. One may detect the variation by checking the cosine similarity between the direction vectors of SL and CL when a new data point are generated. When the cosine similarity between them are smaller than a given threshold, which represents the direction of SL and CL changes, the new incoming data point are set to be the break point. However, when there are a lot of data points involving in CL, a new incoming data point may not affect $\overrightarrow{v_{CL}}$ significantly until there are enough incoming data points for new moving behavior. Thus, this approach cannot select a

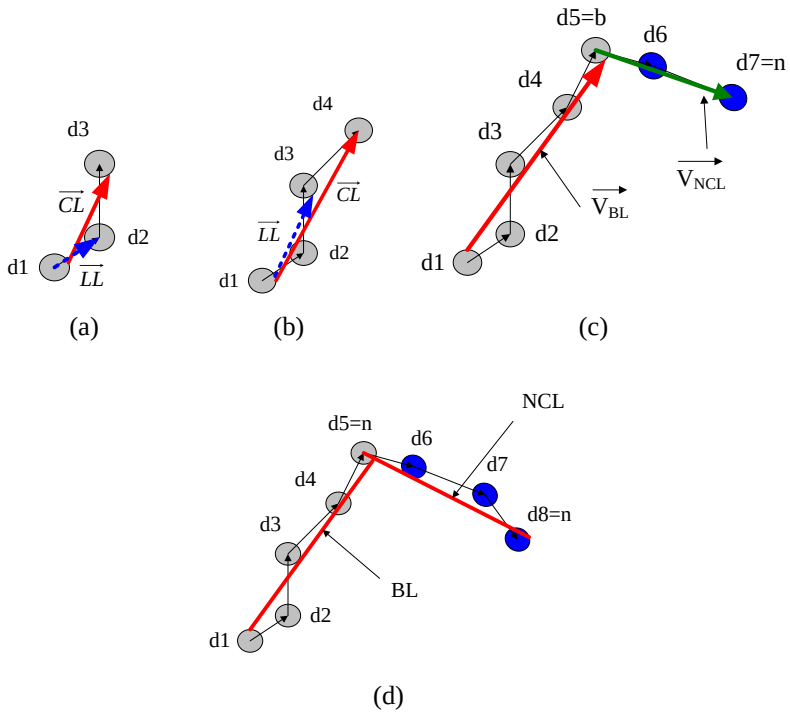


Figure 4.2: An illustrative for Algorithm LR

New Point(NP)	T	X	Y	Speed
d1	1	14.6	7.6	9.01
d2	2	22.1	12.6	12.24
d3	3	22.1	24.9	14.14
d4	4	32.1	34.9	11.18
d5	5	37.1	44.9	10.31
d6	6	47.1	42.1	15.17
d7	7	61.2	36.6	10.96
d8	8	67.6	27.7	15.81

Table 4.1: The data points of a mobile sensor

proper break point which can reflect when moving behavior changes, incurring that the server can hardly capture the location of a mobile sensor.

To select a break point to adapt the variation of moving behavior, a mobile sensor incrementally maintains a potential break point b to verify whether the moving behavior changes or not. Specifically, suppose that the current data points involving in CL are $(d_1, d_2, ..., d_k)$ where d_k is a new incoming data point. According to a potential break point b, we can derive a backward regression line BL for $(d_1, ..., b = d_j)$ and a potential

regression line NCL for $(d_j, ..., d_k)$ with their direction vectors $\overrightarrow{v_{BL}}$ and $\overrightarrow{v_{NCL}}$. Given a threshold β, if $sim(\overrightarrow{v_{BL}}, \overrightarrow{v_{BL}}) > \beta$, the moving behavior described by data points before and after b does not change tremendously so that b is set to be the the new coming data point d_k. On the contrary, if $sim(\overrightarrow{v_{BL}}, \overrightarrow{v_{BL}}) \leq \beta$ where β is a given threshold, the position of b does not change since data points before b and that after b may represent different moving behavior. Given a QoS threshold for trajectories ϵ_T, if the RMSE between SL and data points $(d_1, ..., d_k)$ exceeds ϵ_T, these data points represent different moving behavior than SL does. In this case, a mobile sensor should upload the potential regression line NCL to the server to set the CL to be NCL to guarantee QoS of data collection. Based on the two steps above, Algorithm LR is shown in Algorithm 1.

Algorithm 1: Algorithm LR

Input: Thresholds: α, β and ϵ_T

$flag \leftarrow 0$;
$b \leftarrow d_1$;
$LL \leftarrow$ regression line by $\{d_1, d_2\}$;
while *there is a new incoming data point* d_k **do**
  $CL \leftarrow$ regression line by $\{d_1, ..., d_k\}$;
  $BL \leftarrow$ regression line by $\{d_1, ..., d_j\}$;
  $NCL \leftarrow$ regression line by $\{d_j, ..., d_k\}$;
  if $sim(\overrightarrow{v_{LL}}, \overrightarrow{v_{CL}}) > \alpha$ *and* $flag = 0$ **then**
    Upload CL;
    $SL \leftarrow CL$;
    $flag \leftarrow 1$;
  if $RMSE(SL, \{d_1, ..., d_k\}) > \epsilon_T$ **then**
    $CL \leftarrow NCL$;
    $flag \leftarrow 0$;
    $\{d_1, ..., d_k\} \leftarrow \{d_j, ..., d_k\}$;
    $b \leftarrow d_k$;
  if $sim(\overrightarrow{v_{BL}}, \overrightarrow{v_{NCL}}) > \beta$ **then**
    $b \leftarrow d_k$;
  $LL \leftarrow CL$;

Consider an illustrative example in Table 4.1 where each data point contains four attributes, time (T), coordinates (X and Y) and speed (S). Given the thresholds $\alpha = 0.85$,

NP	Lines	F	BP	SL	$\overrightarrow{V_{LL}}$	$\overrightarrow{V_{CL}}$	S_{LC}	$\overrightarrow{V_{BL}}$	$\overrightarrow{V_{NCL}}$	S_{BN}	RMSE
d1	(d1,d1)	0	-	-	-	-	-	-	-	-	-
d2	(d1,d2)	0	d2	-	-	(7.5, 5)	-	-	(7.5, 5)	-	0
d3	(d1,d3)	0	d2	-	(7.5, 5)	(3.8, 8.7)	0.84	(7.5, 5)	(0, 12.3)	0.55	2.45
d4	(d1,d4)	1	d4	(d1,d4)	(3.8, 8.7)	(5.3, 9.4)	1.00	(3.8, 8.7)	(10, 10)	0.93	2.31
d5	(d1,d5)	1	d5	(d1,d4)	(5.3, 9.4)	(5.5, 9.7)	1.00	(5.3, 9.4)	(5, 10)	1.00	1.84
d6	(d1,d6)	1	d5	(d1,d4)	(5.5, 9.7)	(6.2, 8.0)	0.99	(5.5, 9.7)	(10, -2.8)	0.24	3.58
d7	(d1,d7)	1	d5	(d1,d4)	(6.2, 8.0)	(7.3, 5.9)	0.97	(5.5, 9.7)	(12.1, -4.2)	0.18	7.24
d8	(d1 d5) (d6 d8)	0	d8	(d1,d4)	(7.3, 5.9)	(7.7, 3.8)	0.98	(5.5, 9.7)	(10.6, -5.7)	0.02	11.27

$*S_{LC} = Sim(\overrightarrow{V_{LL}}, \overrightarrow{V_{CL}}), S_{BN} = Sim(\overrightarrow{V_{BL}}, \overrightarrow{V_{NCL}})$, F = Flag

Table 4.2: Execution Scenario of algorithm LR

$\beta = 0.85$ and $\epsilon_T = 20$, the execution scenario of algorithm LR is listed in Table 4.2. When d_3 is generated (i.e., T=3), we can see that the CL is not stable since $sim(\overrightarrow{V_{LL}}, \overrightarrow{V_{CL}}) = 0.84 < 0.85$ which indicates that the directions of LL and CL are not similar, as shown in Figure 4.2(a). While T=4 shown in Figure 4.2(b), it can be verified that $sim(\overrightarrow{V_{LL}}, \overrightarrow{V_{CL}}) = 0.99 > 0.84$ which represents that the direction of CL is stable. Thus, this mobile sensor uploads its CL to the server in T=4 and sets the flag to be 1. Note that from T=4 to T=5, the potential break point b keeps going forward to be d_4 and d_5 since the similarity between $\overrightarrow{V_{BL}}$ and $\overrightarrow{V_{NCL}}$ are larger than 0.85. With the new data points d_6 and d_7 coming, as shown in Figure 4.2(c), the potential break point b keep being d_5 since $sim(\overrightarrow{V_{BL}}, \overrightarrow{V_{NCL}}) < 0.85$. Until d_8 comes, the RMSE between d_6, d_7, d_8 and SL is 18.03 larger than the threshold 18. Thus, d_5 is used to update the current sequence of data points to be $\{d_6, d_7, d_8\}$ and the new current regression line CL is derived by it with valid time interval [6,8] as shown in Figure 4.2(d).

4.2 Algorithm KR: Modeling Interested Attributes

As long as modeling the trajectory of a mobile sensor, in this section, we exploit *kernel regression* to model other interested attributes to further decrease the bandwidth of the network and the server. Different from trajectories, the underlying distributions of the interested attributes are dependent to their features. For example, the distribution of the

movement speed may be very different to that of air pollution. Therefore, it is hard to use a pre-defined model to model all of these attributes. Without a priori knowledge of the underlying distributions of these attributes, kernel regression provides a non-parametric data-driven technique to determine the shape of the curve that fits the given data. Also, under the intermittent connective network, a kernel regression model can be incrementally refined which benefit for real-time applications. However, as mentioned above, the issues of bandwidth and intermittent connectivity should also be addressed. Therefore, in this section, we first introduce the background of kernel regression and then propose a kernel-regression-based approach, algorithm KR, to make the RMSE between the curve and the data points bounded within the QoS specification.

4.2.1 Kernel Regression

Let the collected data points for an interested attribute from a mobile sensor be Ψ and each data point in S can be represented as (X_i, I_i) where a *feature points* $X_i = (X_{i,1}, X_{i,2}, ...X_{i,d})$ is a point with d attributes with an *interested value* I_i. For example, the speed of a mobile sensor is related to its location and the time. Thus, the feature point $X_i = (x_i, y_i, t_i)$ where this mobile sensor is in (x_i, y_i) at time t_i and the interested value I_i denotes the corresponding speed.

Given Ψ from a mobile sensor, the idea of kernel regression is to estimate the interested value at x according to that at nearby feature points in Ψ. Conceptually, for each feature point X_i, since a mobile sensor "observed" that the corresponding interested value to be I_i, we have the most "confidence" to estimate the interested value at X_i (i.e., I_i). When a point is far from a feature point X_i, we have fewer "confidence" to estimate the interested value at this point by that of X_i. That is, the interested values with most information about I_i should be those at points closet to X_i, and vice versa. Therefore, based on this concept, when given an arbitrary point x, we can estimate its interested value by evaluating amount of information which x contains about each ob-

served feature points.

Formally, suppose that there are n data points $\{(X_1, I_1), ..., (X_n, I_n)\}$. the interested value at an arbitrary point x is estimated by

$$\hat{I} = \frac{\sum_{j=1}^{n} Y_j \times K_\alpha(x, X_j)}{\sum_{j=1}^{n} K_\alpha(x, X_j)} \tag{4.1}$$

, where $K_\alpha(x, X_j)$ is the kernel with its kernel radius α.

In kernel regression, the kernel $K_\alpha(x, X_i)$ is responsible to evaluate how much information x contains about the interested values I_i, i.e., weight x, according to the distance between x and X_i. The kernel radius α is represented as the width of a window center at the X_i and give weighing value to x if it locates in the window. With the increasing of distance from x to X_i, the value of $K_\alpha(x, X_i)$ keeps decreasing in a window decided by α. For example, suppose that x and X_i are in one dimension. The Gaussian kernel is formulated as $K_\alpha(x, X_i) = e^{-\frac{(x-X_i)^2}{2\alpha^2}}$. Let α be 3 and X_i be 16. It can be verified that when $x = 10$, $K_3(10, 16) = e^{-\frac{(10-16)^2}{2\times 0.5^2}} = 0.135$. When $x = 0.9$ which is more closer to 1 than previous one, we can obtain a larger $K_3(14.16) = e^{-\frac{(14-16)^2}{2\times 0.5^2}} = 0.801$. The weighting behavior of Gaussian kernel with $\alpha = 3$ is illustrated in Figure 4.3. Once given several data points, we can estimate the interested values at every feature point by their kernels. Suppose that there are three feature points $X_1 = 16$, $X_2 = 20$ and $X_3 = 8$. The bold line shows the weighting effect by the kernels of X_1, X_2 and X_3 in Figure 4.4. Usually, a kernel is designed for weighting a one-dimensional point. However, we can extend a kernel for a multiple dimension data point. Specifically, given a feature point $X_i = (X_{i,1}, X_{i,2}, ...X_{i,d})$ with its interested value I_i, we can derive a Gaussian kernel for a multiple dimension data point $x = (x_1, ...x_d)$ as $K_\alpha(x, X_i) = \prod_{j=1}^{d} e^{-\frac{(x_j - X_{i,j})^2}{2\alpha^2}}$ (18). For example, let $\alpha = 3$ and $X_i = (16, 16)$. Given $x = (10, 14)$, we can obtain $K_3(x, X_i) = e^{-\frac{(10-16)^2}{2\times 0.5^2}} \times e^{-\frac{(14-16)^2}{2\times 0.5^2}} = 0.108$.

Although there have been several kernels proposed, it is worth mentioning that the accuracy of kernel regression is highly dependent on the selection of radius instead of

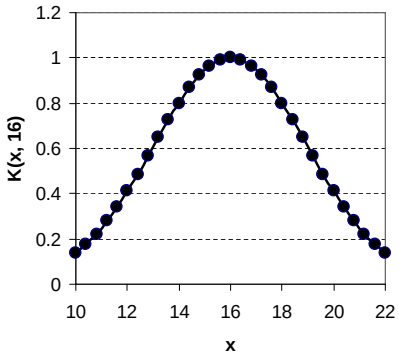


Figure 4.3: Example of the kernel regression

the selection of the kernel(19)(18).

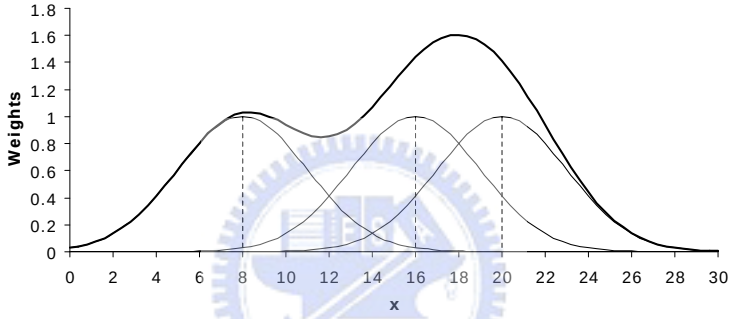


Figure 4.4: Example of the kernel regression

4.2.2 Algorithm KR

Kernel regression can be used to model data points with unknown underlying distribution. Note that the accuracy of the curve build by kernel regression is dependent on both the number of data points involving in building the model and the selected kernel radius. For preserving the bandwidth, several data points are sampled from the set of collected data points Ψ and uploaded to the server. Through selecting a proper kernel radius, the server can use kernel regression to build a curve by fewer data points and the QoS specification can be satisfied.

To achieve this goal, we are starting with sampling a certain number of data points

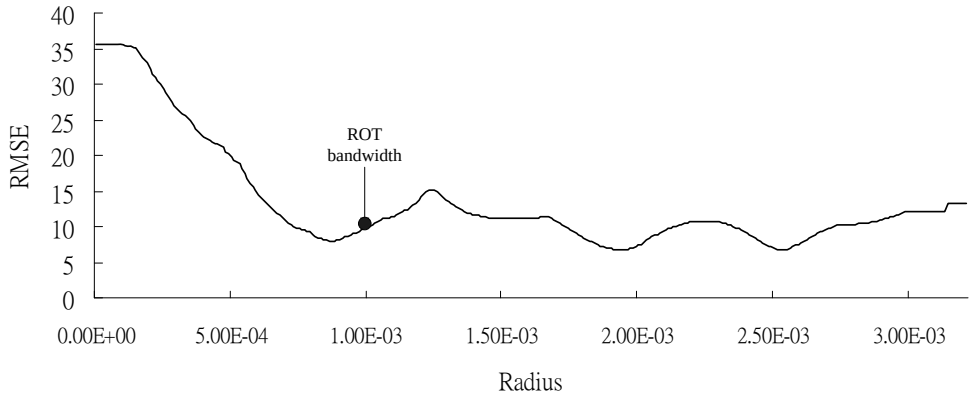


Figure 4.5: Effect of kernel radius on the RMSE

according to an initial sampling rate $rate_{init}$. Given these sampled data points, we can apply rule-of-thumb(ROT) method to selection the initial kernel radius of each dimension (18). Base on the ROT, the bandwidth $\hat{\alpha}$ for the kernel function K_i could be computed using the following equation:

$$\hat{\alpha_i} = \left(\frac{4}{d+2}\right)^{-1/(d+4)} |S|^{-1/(d+4)} \sigma_i$$

where each feature point is d-dimensional and σ_i is the standard deviation of the sampled data points in the i-th dimension. From Figure 4.5, we can see that the RMSE under the kernel radius chosen by ROT method, which is about 9.52, can be improved to the local minimal RMSE which is 7.9. For preserving the bandwidth, we first use (1+1)-ES, an efficient local search method, for searching a kernel radius which can reduce RMSE to local minimum under the same number of sampled points. According to these sampled data points and the derived kernel radius, we can exploit kernel regression to generate a curve fitted them. To guarantee this curve can represent all data points in Ψ, we shall examine RMSE between the curve and data points in Ψ. Once satisfying the QoS specification, we can upload the sampled data points to the server such that the same curve can be derived by them in the server side. Otherwise, the sampling rate will increase and more data points will be sampled to involve the computation of kernel

regression. At last, if the sampling rate is higher than $rate_{max}$ and the derived curve still cannot achieve QoS specification, a mobile sensor will upload all data points in Ψ to the server. To realize this concept, we proposed Algorithm KR listed in Algorithm 2.

Algorithm 2: Algorithm KR

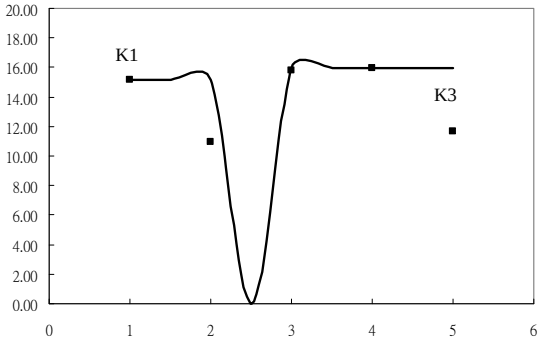
input: QoS Threholds: ϵ_I; Parameters: $rate_{min}, rate_{max}, rate_{inc}, g$

```
rate ← rate_min;
rmse ← ε_I;
while rmse > ε_I AND rate ≤ rate_max do
    n ← rate × |Ψ|;
    S ← Sample n data from Ψ;
    α ← kernel radius by ROT according to Ψ;
    α ← apply α to (1+1)-ES for g generations;
    C ← the curve derived by kernel regression with α, S;
    rmse ← RMSE(C,Ψ);
    rate ← rate + rate_inc;
if rmse ≤ ε_I then
    Upload (α, S);
else
    Upload Ψ;
```

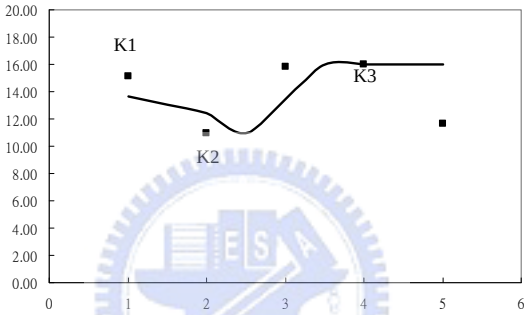
Consider $\Psi = d_1, d_2, ..., d_5$ in Table 4.1 and focus on deriving the relation about time and speed as our example. Suppose that the QoS specification ϵ_I is 2.5, $rate_{min}$ is 40% and $rate_{inc}$ is 20%. Initially, we sample $5 \times 40\% = 2$ points which are d_1 and d_4. Figure 4.6(a) shows the results by two kernels centered at d_1 and d_4. Since the RMSE of this curve is 2.71 which is larger than ϵ_I, then the sampling rate is increasing to 40% + 20% = 60% such that we sample $5 \times 60\% = 3$ points, d_1, d_2 and d_4, to apply kernel regression. It can be verified that the RMSE of the derived curve being 2.41 does not exceed ϵ_I and thus these three points are uploaded to the server.

From another view, this example also shows the ability of kernel regression to deal with intermittent connectivity. Suppose that the data points we intend to upload are d_1, d_2 and d_4. When the server only receive two data points d_1 and d_4, the server can first derived a rough view in time about the whole data points as shown in Figure 4.6(a). While receiving more and more data points, the curve built by the server will be refined to

the one shown in 4.6(b), and much fits the data points. As such, this born feature of kernel regression can benefits many real-time applications, for example, traffic monitoring.



(a)



(b)

Figure 4.6: The kernel regression model based on Epanechnikov kernels by (a) two and (b) three data points

Chapter 5

In-network Aggregation Mechanism

In this section, we introduce an in-network aggregation mechanism allowing mobile sensors with similar data to aggregate their models. Obviously, in-network aggregation between mobile sensors can reduce the amount of data needed to be uploaded so that the bandwidth of the server and the network can be saved.

However, there are several issues to be addressed. First, grouping mobile sensors into clusters and selecting a cluster head for aggregation, the state-of-the-art approach used in a static sensor network, may not be applicable in a mobile sensor network. In a mobile sensor networks, the connectivity between mobile sensors are usually highly dynamic and may vary drastically due to the unpredictable mobility of mobile sensors. Therefore, the proposed in-network aggregation mechanism is required not to maintain extra information to organize cluster between mobile sensors. Second, due to the deployment of access points, mobile sensors may have different probabilities to connect access points. For example, a mobile sensor often moving in an urban area usually has higher probability than that in a countryside area since the density of access points deployed in an urban area is more higher than that in a countryside area. Therefore, the proposed in-network aggregation mechanism should be aware of the probability of connecting to access points such that the aggregated data can be sent to the server more quickly.

To tackle the issues above, we proposed a random-coupling approach for in-network

aggregation. Our approach is composed into three phases, randomly coupling, aggregator selection and aggregation. The details of each phase are described as follows:

Phase 1: Randomly Coupling

In this phase, each mobile sensor which can communicate with each other will randomly match to one mobile sensor and form a *couple*. For two mobile sensors in a couple, one of them will be the *aggregator*, which is responsible to aggregate data between mobile sensors, if their data can be aggregated. Moreover, the data of aggregators can be aggregated again if they are matched to be a couple. We can observe that the closer the mobile sensors are, the higher probability the aggregators can further aggregate their aggregated data. Thus, based on this observation, our approach can aggregate data for mobile sensors as the similar result as the clustering approaches, i.e., grouping nearby mobile sensors and selecting a cluster head to aggregate.

Phase 2: Aggregator Selection

To select the proper aggregator for each couple, the number of successful connection have to be taken into account. One may choose a mobile sensor with higher the probability of successful connection P_S as the aggregator, which P_S is defined as $\frac{N_S}{N_S+NF}$, where N_S and N_F denotes the number of successful and failure connection to an access point. However, P_S cannot reflect the occurring time of successful connections. For example, suppose that mobile sensor A and B is a couple. If A can connect in the current time t_5, and time t_4. then we can derive $P_S = \frac{2}{5}$. On the other hand, suppose that B can connect in time t_3, time t_2 and t_1. Then we can derive its $P_S = \frac{3}{5}$. However, it is more proper to select A as the aggregator because A have higher probability to successfully connect to an access point recently.

To consider the occurring time of successful connections, each mobile sensor will maintain a variable $P_{connect} = I_k \times \frac{1}{2} + I_{k-1} \times \frac{1}{2^2} + ...$ where I_{k-i} is 1 (or 0) if the i-th connection before the current connection is (or not) successful. $P_{connect}$ provides a metric to evaluate how often the recent successful connection happens. It can be seen

that $P_{connect}$ gives higher weight to the current successful connection and the weight of the successful connection will exponential decay according to their occurring time. That is, $P_{connect}$ will be larger if successful connections occur frequently, and vice versa. Consider the example above, we can derive that A owns $P_{connect} = \frac{1}{2} + \frac{1}{2^2} = \frac{3}{4}$ and B owns $P_{connect} = \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} = \frac{7}{32}$. Thus, we can imply that A has more successful connection recently than B does. For two mobile sensors in the same match, the mobile sensor with larger $P_{connect}$ will be the aggregator.

Phase 3: Aggregation

For two mobile sensors A and B in a couple, the aggregator A will first try to aggregate the regression lines of A and B, say L_A and L_B. If L_A can *represent* L_B, B will further transmit the corresponding kernel regression model to A. Thus, A is responsible to upload its L_A and corresponding kernel regression models. B does not upload its data to the server. Through such an aggregation, not only the bandwidth of the server and the network can be saved but also the server can have higher probability to receive the data of A and B in time since A has higher probability to connect to an access point than B does.

To evaluate whether two models can be aggregated, we define the sampling RMSE between two models as follows:

Definition 2. Sampling RMSE: Let two models be M_A and M_B with valid time interval $[t_s^A, t_e^A]$ and $[t_s^B, t_e^B]$. Let $T = [t_s^A, t_e^A] \bigcup [t_s^B, t_e^B]$, the sampling RMSE between M_A and M_B is

$$SR(M_A, M_B) = \sqrt{\frac{1}{|T|} \sum_{t_i \in T} |M_A(t_i) - M_B(t_i)|}$$

When an aggregator A receives the model M_B from other sensor B in the couple, an aggregator A will evaluate whether the sampling RMSE of their models M_A and M_B is bounded with QoS specification or not. If so, an aggregator A will derive an aggregated model $M_{A,aggr}$ for data points sampling from M_A and M_B, i.e., $\{d | d \in M_A(t_i) \bigcup M_B(t_i), t_i \in T\}$. Figure 5.1 shows an example of the aggregation

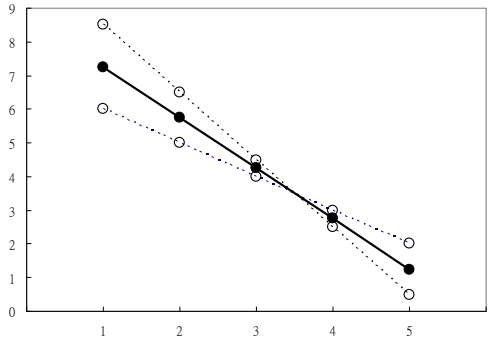


Figure 5.1: Example of the aggregation of two line models

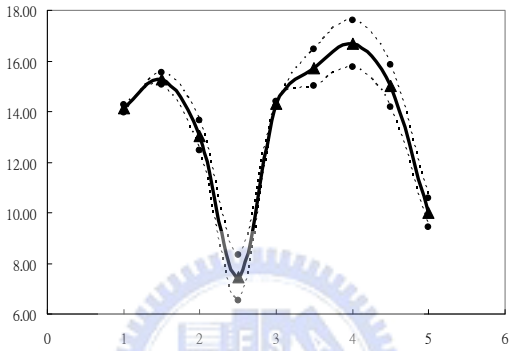


Figure 5.2: Example of the aggregation of two kernel models

of two linear regression model which are dash line with circle on lines. When finished aggregation, the new line would be the solid line with solid circle. Figure 5.2 shows a similar example of kernel regression models.

To put it all together, consider an illustrative example in Figure 5.3 where there are four mobile sensors. Suppose that their data can be fully aggregated with each other. In Figure 5.3(a), {A,B} and {C,D} become couples by random coupling. Since $P_{connect}$ of A is higher than that of B, A becomes an aggregator such that A aggregates the models of A and that of B. By the similar reason, C is also an aggregator which is responsible to aggregate the models of C and D. In Figure 5.3(b), when A and C become a couple, A will be the aggregator. Note that A and C both keep aggregated models. Thus, through aggregation of the aggregated models from A and C, the models kept by A is the aggregated model from A, B, C, and D, which can achieve the same result as

the clustering A, B, C and D.

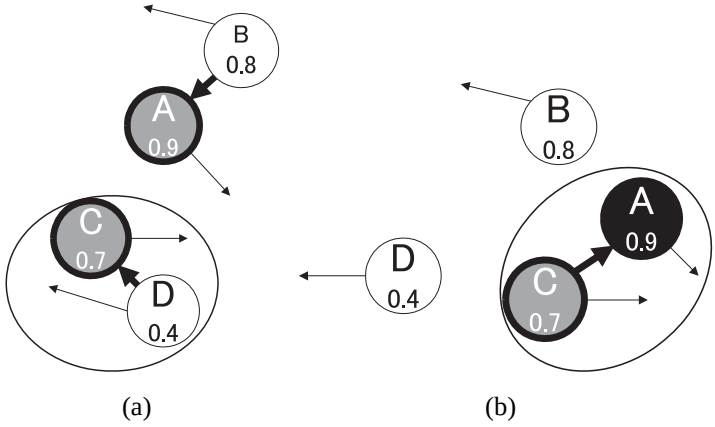


Figure 5.3: An illustrative example for random coupling

Chapter 6

Performance Evaluation

In this section, we present our experimental results. Both synthesis dataset and real dataset are used to verify the proposed framework MobiQC.

Each data point in the following experiments is composed of four attributes, say longitude, latitude, collected time stamp and the speed of a mobile sensor. Each attributes will take 4 bytes to represent it. Thus, a data point needs totally 16 bytes. For a regression line, it need 6 parameters, say a_x, b_x, a_y, b_y and the corresponding valid time interval t_s and t_e, to describe a regression line. Thus, each regression line costs 24 bytes. Finally, every kernel regression model is organized from the set of sample points S and its bandwidth value. Note that for the sampled data points, a mobile sensor just needs to transmit the attributes of time and speed since the spatial data can be generated in the server by the linear regression model. As such, the total bytes of a kernel model is |S|*12+4.

As mentioned above, MobiQC not only reduces the network bandwidth and server loading but also guarantees the QoS specification. In the following experiments, *RMSE* is used to measure the error between the derived model and the data points. Once the value of RMSE does not exceed a given QoS specification, we can claim that MobiQC can provide a QoS-aware data collection mechanism. We use *the number of bytes* to evaluate the network bandwidth and the server loading. The fewer the number of bytes

need to be transmitted, the more the network bandwidth and the server loading can be preserved. To eliminate the effect of random generated data set by our simulator, we define the following two metrics, said *total cost ratio* and *trajectory cost ratio*. Specifically, the *total cost ratio* denotes the proportion between the number of transmitted bytes and the effective data which are received or derived from models by the server. A lower *total cost ratio* represents that modeling can bring more benefit for preserving the network bandwidth and the server loading. On the other hand, the *trajectory cost ratio* represents the proportion between the number of transmitted trajectory bytes of models or data and the effective data which are received or derived from models by the server. By this two metrics, we can see the performance of our framework from different aspects.

6.1 Real Dataset

In the real dataset, we extract real trajectories from CarWeb platform (5). In CarWeb platform, each car obtains its location (in terms of longitude and latitude), and speed for every five seconds via equipped GPS sensor. In the following experiments for real dataset, there are totally 90 trajectories and around 25500 points. We can verify the effectiveness of MobiQC in a small-scale mobile sensor system. The following subsections we will evaluate the effect of various QoS specification ϵ_T and similar threshold β on the precision of linear regression model. Note that since the α threshold decide if the modeled line should be transmitted to the server, it will not affect the result of experiments. Thus, in the following all experiment, we will set its value the same with β. For kernel regression model, we also discuss the precision of different sample methods and the effectiveness of local search of radius under different sampling rates.

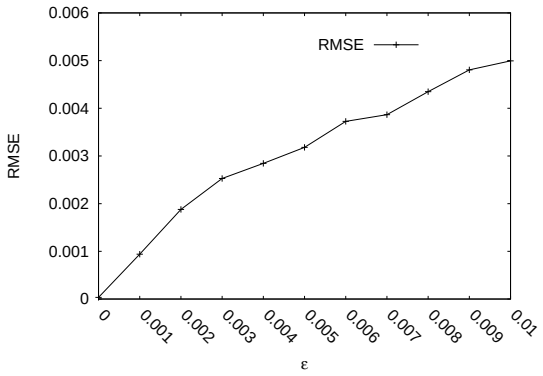


Figure 6.1: The RMSE of linear regression for real data under different ε.

6.1.1 QoS Specification for Linear Regression Model

In this experiment, we set the sampling rate of kernel regression to be 0.33 and the similar threshold α and β to be 0.8 as the default settings.

We first discuss the impact of the total number of bytes and average RMSE with QoS specification ϵ_T varied from 0.00001 to 0.01. From Figure6.1, we can observe that the smaller ϵ_T will cause higher precision, i.e., lower RMSE) and more total bytes are needed since a regression line is easily divided into two ones, and vice versa.

For the aspect of the reducing transmitted packages, Figure6.2 shows that MobiCQ can reduce much more transmitted packages than transmitting all raw data. Since the spatial and temporal data of mobile sensors can be modeled into linear regression lines, the number of bytes decreases significantly. Also, kernel regression model also contributes for reducing the number of bytes. On the other hand, we can also see that the better selection for the parameter ϵ_T is about 0.001 since it can lead both less number of bytes and higher precision.

6.2 The Impact of β

In this experiment, we evaluate the impact of variant β. The value of ϵ_T is 0.001 and the sampling rate for kernel regression is 0.33 as default. Figure 6.3 shows that the β

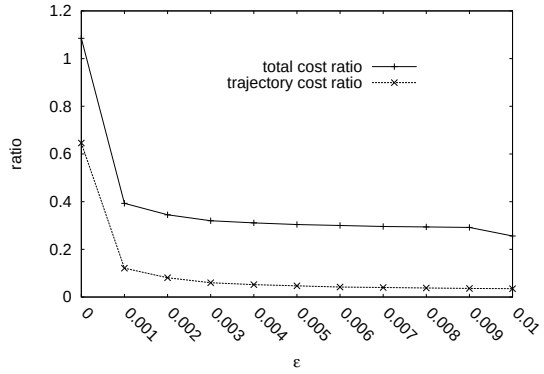


Figure 6.2: The ratio of modeled bytes versus total data bytes for real data under different ε.

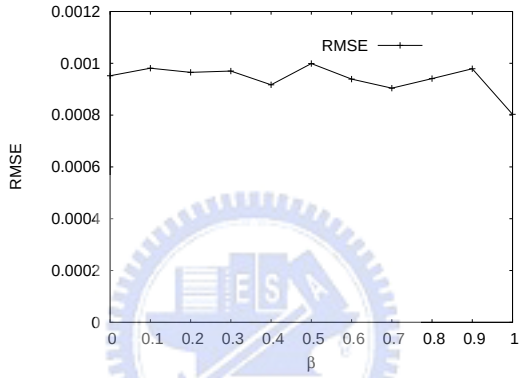


Figure 6.3: The RMSE of linear regression for real data under different β.

does not affect the RMSE significantly. Although when it is larger than 0.9, the RMSE can be achived the minimum value, its number of bytes also increases significantly, as shown in Figure 6.4. Thus, the better selection of $beta$ is between 0.8 and 0.9 since we can achieve both less number of bytes and higher precision at the same time.

6.2.1 Impact of Local Search for Radius Selection

As mentioned above, we can select a proper kernel radius via local search. In this experiment, we vary the sampling rate and the number of generation for (1+1)-ES local search. Suppose that the sampling rate is r, sampled data points are those at position $\frac{1}{r}$-th, $\frac{2}{r}$-th,..., and so on. In Figure6.5, LS-0 shows the case without local search. On the

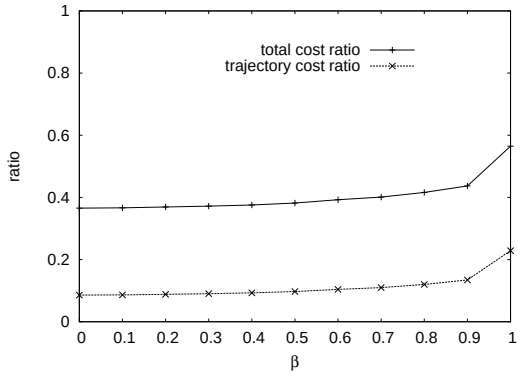


Figure 6.4: The reduction rate for real data under different β.

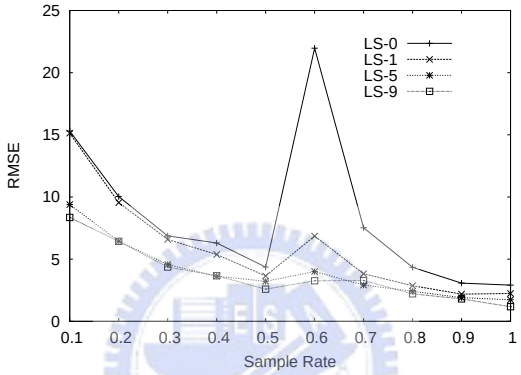


Figure 6.5: The RMSE of kernel regression with various local search times for real data under different sample rates.

contrary, LS-1, LS-5, and LS-9 represent that the (1+1)-ES will execute for 1, 5 and 9 generation respectively. It can be seen that RMSE LS-0 is larger than that of performing local search. It supports the effectiveness of local search for selecting radius. Furthermore, the values of RMSE in LS-5 and LS-9 are almost the same. It represents that we can only execute five rounds of local search is sufficient so that the RMSE can be significantly reduced. On the other hand, Figure 6.6 shows a tradeoff between the precision and the total cost rate with respect of sampling rate. From the experiments above, we can obtain that sampling rate 0.3 can take a good balance for the tradeoff.

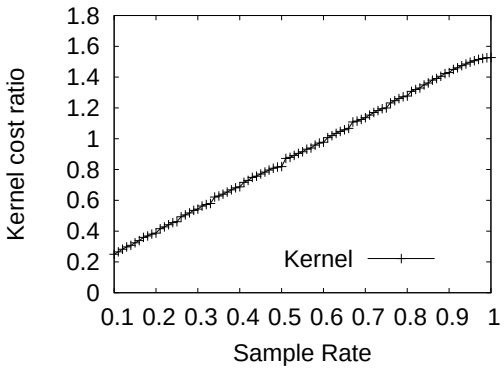


Figure 6.6: The rate of modeled bytes of kernel regression versus total data bytes for real data under different sample rates.

6.2.2 Impact of Sampling Methods for Kernel Regression

In MobiQC, we uniformly sample data points according to the sampling rate. In this experiment, we evaluate the impact of sampling methods for the precision of kernel regression.

Here, we use a sampling method from ”The art of Computer programming volume 2” of Knuth to see if the sample method will affect the RMSE of kernel regression. Given the sampling rate, this method aimed at uniform random sampling from original data set. From Figure 6.7, we can see that uniform random sample will let the RMSE of kernel regression decrease compared to the original sample method although it is more stable than original one. We think that the original sampling method is over uniform compared to this uniform random sampling method. However, it is more suitable for kernel regression since random sampling method is just let the selected probability of each data point be the same. It is possible that there are some data in the feature space without any kernel cover it. Thus, the original method. which is over uniform, is more suitable than unform random sampling method.

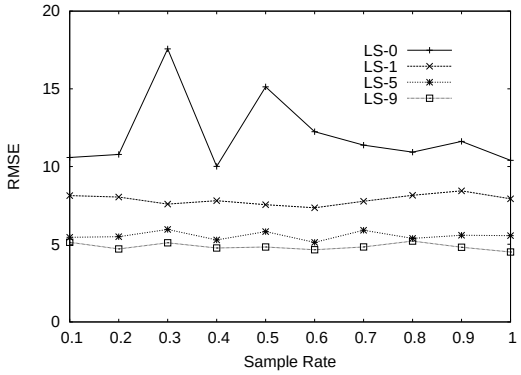


Figure 6.7: The RMSE of kernel regression with various local search times for real data under different sample rates using the random sampling method.

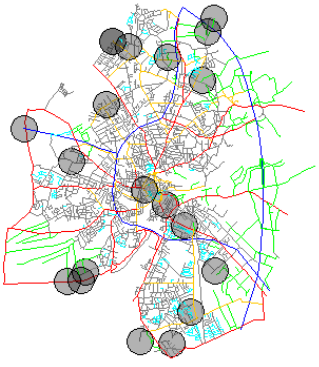


Figure 6.8: The map of Network-based generator of moving objects

6.3 Synthetic Dataset

On the other hand, we implement the synthetic dataset on the well-known traffic simulator(20). The synthetic dataset is used to test the performance of MobiQC in a large-scale mobile sensor system. For the comparison purpose, the GROUP method of CarTel(4) is implemented to compare the effect of server loading with MobiQC. To simulate the intermittent connectivity, we add several access points into the map. A circle is represent the range of an access point as shown in Figure6.8. To simulate the movement of mobile sensors in real world, we use the map of San Francisco and the coordinates of mobile sensors are represented in terms of longitude and latitude.

In the following experiments, there are 20 access points on the map and 250 cars on roads. Each experiment executes for 2000 time units. There are three classes of speed of cars, say slow, medium and fast, and the assignment of the car speed is uniformly random. We generate a trajectory for each car and the movement of this car will follow this trajectory. During its movement, each car will record its coordinates and speed for each time unit. In the server side, the map will be divided into several grids according to the parameter *cell length*. The server will build a histogram for each grid by the data upload from mobile sensors in it. The statistic manager adopts Chi-square test to verify the current and the past histogram are similar or not. Besides the group method in CarTel, we will implement the proposed in-network aggregation method and evaluate how much benefit it can bring for reducing the server loading. The following experiments use $\alpha = 0.8$, $\beta = 0.8$, $\epsilon_I = 6$, wireless bandwidth as 5000 bytes and cell length as 0.01 as our default settings.

6.3.1 Data Collection Without Chi-Square Test

In this section, each mobile sensor will upload its model when connecting to an access point. That is, the statistic manager does not use Chi-Square test to verify whether the

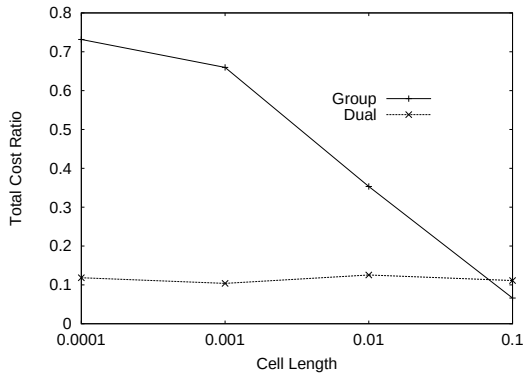


Figure 6.9: The impact of varied cell length

server should collect more data or not. Under different setting of wireless bandwidth and cell length, we evaluate the reduction of server loading. Note that cell length will affect the number of cells which is used as the summary report of data in the group method. As such, it also affects the total number of transmitted packages.

The Impact of Varied Cell Length

In this experiment, we observe the impact of varied cell length. In Figure 6.9, we can see that the group method will generate more summary report when the cell length is small. However, MobiQC is not effected by the cell length. The reason why is that the mobile sensors notifies the server information of the cell it owns. Thus, with increasing of the number of cell, the transmitted data will increase at the meantime.

The Impact of Varied Wireless Bandwidth

In Figure 6.10, we compare three methods, say group method in CarTel, MobiQC without and with in-network aggregation, in terms of total cost rate. It can be seen that MobiQC outperforms group method in each wireless bandwidth setting. When the bandwidth is 100 bytes, group method needs more number of bytes to transmit its data to the server. It is because a mobile sensor using the group method cannot successfully trans-

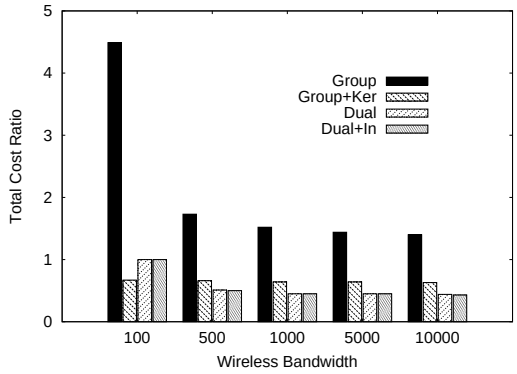


Figure 6.10: The rate of total transmitted packages versus total data bytes under various wireless bandwidth.

mit its summary report to the server due to a lack of wireless bandwidth. It can be seen that MobiQC can adapt the environment with low wireless bandwidth.

6.3.2 Data Collection With Chi-Square Test

In this section, we will consider the condition of judging if the collected data is enough or not by server. The server will collect data for query ranges of random generated queries. We will see if we can reduce server loading under this condition. On the other hand, we will also gather statistics about average time difference between the data sensing time and the time of data arriving to server in order to see if our methods can achieve the approximate real-time data delivering since the group method of CarTel has good performance on it.

The Impact of Varied Wireless Bandwidth

In Figure 6.11, we can see that our proposed methods are both better than the group method. However, comparing to previous experiment which is without Chi-Square Test, the total cost rate is more. There are two reasons. The first is that the server does not collect data of all areas but the sensor still transmit its line models to server. On the other hand, the amount of the total data is less since the server will not ask sensor nodes to

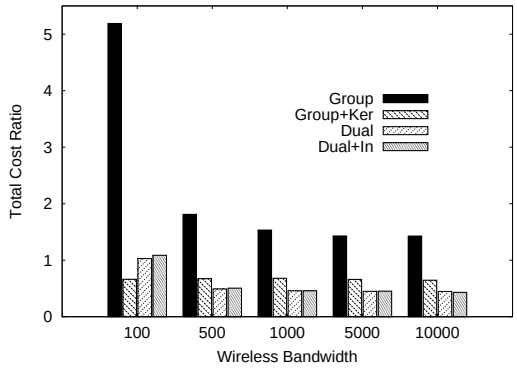


Figure 6.11: The rate of total transmitted packages versus total data bytes under various wireless bandwidth.

transmitted its kernel models back to server if the collected data is enough. By these two reasons, the total rate arise.

For average time difference between data gathering time and arriving time to the serve, as shown in Figure 6.12, we can see that when the wireless bandwidth is enough, the proposed two methods can achieve almost the same performance with the group method. On the other hand, our methods outperformed the group method since our line model can represents several cells instead of just one which is represented by the group method. Thus, the server can judge if a sensor node have required data more quickly. On the other hand, since a sensor node will transmit its real data back to server by using group method, it will also waste a lot of wireless bandwidth. So it will affected by wireless bandwidth very much. Hence, we can see that our dual method can reducing server loading because of reducing transmitted packages while we can also achieve approximate real-time data collection under designated quality, as shown in Figure 6.13 and 6.14.

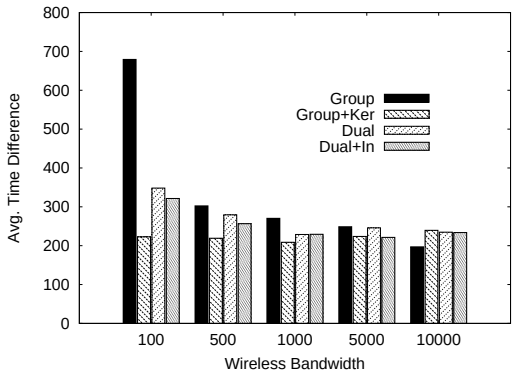


Figure 6.12: Average time difference between data gathering time and arriving time to the serve under various wireless bandwidth.

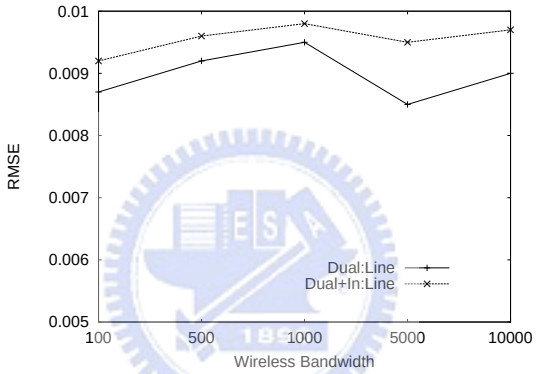


Figure 6.13: The RMSE of Linear Regression for synthetic data under various wireless bandwidth.

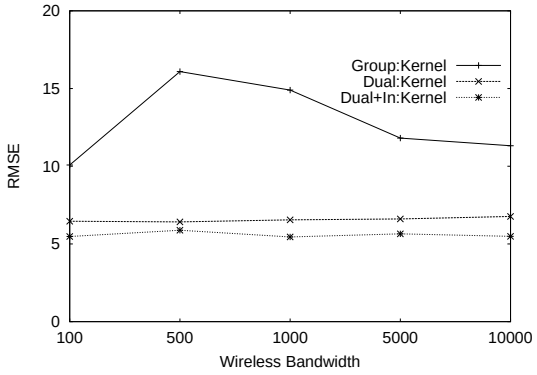


Figure 6.14: The RMSE of Kernel Regression for synthetic data under various wireless bandwidth.

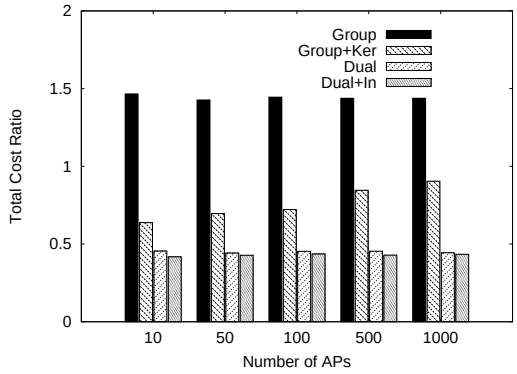


Figure 6.15: The rate of total transmitted packages versus total data bytes under different condition of intermittent connection.

The Impact of Varied Coverage of Wireless Access Points

In this experiment, we want to test the effect to the three methods under different intermittent connection condition. Thus, we vary the number of APs to simulate different intermittent connection condition. From the Figure 6.15, we can see that, in the different coverage condition, our methods can reduce total cost rate and outperform the group method. On the other hand, when number of APs is lower, the Dual-In method can reduce more data packages than the Dual method. However, they have almost the same performance when number of APs is more since the probability of transmitted data back to server is more in this situation. So the chance of aggregation is less. Also, For the average time difference of data gathering time and arriving time to the server, as shown in Figure 6.16, we can see that our methods is slower than the group method. But it is just a little difference. Thus, we can see that in this condition, our methods can still reduce server loading while guaranteeing the quality of data, as shown in Figure 6.17 and 6.18.

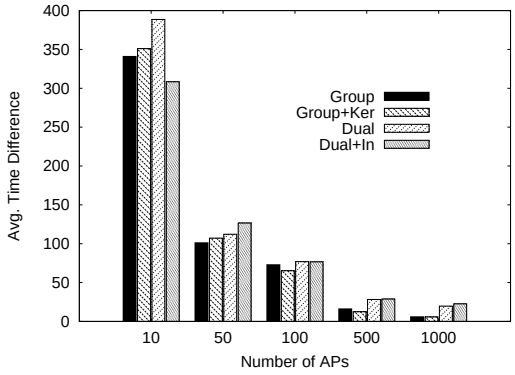


Figure 6.16: Average time difference between data gathering time and arriving time to the server under different condition of intermittent connection.

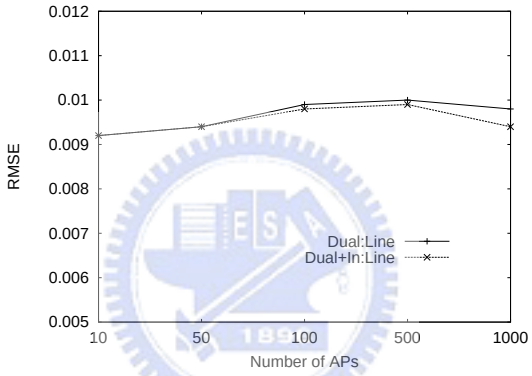


Figure 6.17: The RMSE of Linear Regression under different condition of intermittent connection.

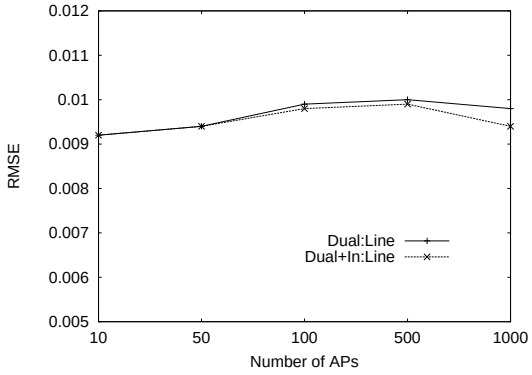


Figure 6.18: The RMSE of Kernel Regression under different condition of intermittent connection.

Chapter 7

Conclusion

This paper shows that how data can be collected in mobile sensor networks while considering reducing server loading and guaranteeing the quality of data at the same time by using our proposed MobiQC framework. In our experimental evaluation, the dual regression models can achieve reducing a lot of server loading while keeping the quality of data under given requirement. Furthermore, our proposed method can also collect data in the intermittent-connected environment with approximate real-time performance. As mobile sensor networks have applied to more and more applications due to its low cost and high efficiency, the data collection insensitive to the real accurate data will also show up its benefits and importance while server loading is relative arised. In this situation, MobiQC can give these applications a good guideline to develop their system.

Bibliography

[1] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks," in *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 145–156, 2006.

[2] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," in *Proc. of Symposium on Operating System Design and Implementation (OSDI)*, 2002.

[3] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Record*, vol. 31, no. 3, pp. 9–18, 2002.

[4] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "Cartel: a distributed mobile sensor computing system," in *Proc. of International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 125–138, 2006.

[5] C. H. Lo, C. W. Chen, T. Y. Lin, C. S. Lin, and W. C. Peng, "Carweb: A traffic data collection platform," in *Proc. of International Conference on Mobile Data Management (MDM)*, pp. 221–222, 2008.

[6] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *Proc. of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, pp. 96–107, 2002.

[7] J. Yoon, B. Noble, and M. Liu, "Surface street traffic estimation," in *Proc. of International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 220–232, 2007.

[8] Y. Zhang, B. Hull, H. Balakrishnan, and S. Madden, "ICEDB: Intermittently-Connected Continuous Query Processing," in *Proc. of International Conference on Data Engineering (ICDE)*, pp. 166–175, April 2007.

[9] J. Rybicki, B. Scheuermann, W. Kiess, C. Lochert, P. Fallahi, and M. Mauve, "Challenge: peers on wheels - a road to new traffic information systems," in *Proc. of the International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 215–221, 2007.

[10] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proc. of International Conference on Very Large Data Bases (VLDB)*, pp. 588–599, 2004.

[11] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proc. of International Conference on Data Engineering (ICDE)*, pp. 48–59, 2006.

[12] Y. Kotidis, "Snapshot queries: Towards data-centric sensor networks," in *Proc. of International Conference on Data Engineering (ICDE)*, pp. 131–142, 2005.

[13] C. Liu, K. Wu, and J. Pei, "A dynamic clustering and scheduling approach to energy saving in data collection from wireless sensor networks," in *Proc. of Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pp. 374–385, 2005.

[14] J. G. Lee, J. Han, and K. Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 593–604, 2007.

[15] T. Ahonen, A. Hadid, and M. Pietikajnen, “Face recognition with local binary patterns,” in *Proc. of European Conference on Computer Vision (ECCV)*, pp. 469–481, 2004.

[16] T. Ahonen, M. Pietikainen, A. Hadid, and T. Maenpaa, “Face recognition based on the appearance of local regions,” in *Proc. of International Conference on Pattern Recognition (ICPR)*, pp. 153–156, 2004.

[17] D. Zhong and I. Defee, “Performance of similarity measures based on histograms of local image feature vectors,” *Pattern Recognition Letters*, vol. 28, no. 15, pp. 2003–2010, 2007.

[18] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, “Online outlier detection in sensor data using non-parametric models,” in *Proc. of International Conference on Very Large Data Bases (VLDB)*, pp. 187–198, 2006.

[19] C. Guestrin, P. Bodík, R. Thibaux, M. A. Paskin, and S. Madden, “Distributed regression: an efficient framework for modeling sensor network data,” in *Proc. of International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 1–10, 2004.

[20] T. Brinkho, *Network-based Generator of Moving Objects*. Technical Report of the Institut fur Angewandte Photogrammetrie und Geoinformatik (IAPG), [available] http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/.