Energy-Saving Data Approximation for Data and Queries in Sensor Networks

Ying Li* Seng W. Loke† M. V. Ramakrishna*
*Caulfield School of Information Technology, Monash University, Australia
{Ying.Li, Medahalli.Ramakrishna}@infotech.monash.edu.au
†Department of Computer Science and Computer Engineering, Latrobe University, Australia
s.loke@latrobe.edu.au

Abstract— **One way of conserving the scarce resources in a sensor network is to minimize the amount of data transmitted. This can be accomplished by data compression, aggregation or approximation. The current researches on sensor data compression mainly focus on lossless compression methods, they cannot achieve higher compression ratio than lossy data compression. In-network data aggregation and data approximation can be regarded as lossy data reduction methods. However, in-network data aggregation methods cannot record all the features of sensor data, thus queries referring to the historical data might not be answered. Moreover, the data cached in sensor networks should be used easily for answering queries. Based on the correlation and frequency spectrum analysis results of some types of slowly varying sensor data, we have presented two data approximation methods to reduce data transmission while make queries easy to answer. We have implemented these methods, tested on some real life data sets and compared with related methods. The results indicate that the algorithms are simple and deliver high data reduction ratios, while meeting the user's tolerance of errors.**

I. Introduction

Energy consumption in wireless sensor nodes can be minimized by reducing the amount of data transmitted. This reduction can be accomplished by data compression, in-network data aggregation, and data approximation. Traditional data compression algorithms are not good be used in sensor networks because of the resource constraint in sensor nodes [1]. The focus of the current research of data compression for sensor networks is at the coding level, i.e. how to code sensor data with less bits than normal binary representation [2] [3] [4]. All these are lossless methods of data compression, i.e. no data is lost in the compression process. In-network aggregation and data approximation can be regarded as lossy data reduction methods. With in-network data aggregation methods, the observations from source sensors will be sent to the sensors at upper levels in a hierarchical structure [5] [6]. Along the path to the top level, the data is aggregated and cached at the intermediate levels, which might not be able to answer general queries. This paper investigates data approximation techniques with the following objectives.

Minimise the amount of data transmitted.
Minimise memory space for caching historical data.
Minimise algorithm complexity of query answering.
Provide approximate answers to user queries meeting accuracy requirements.

The data from most sensors change slowly, and has strong autocorrelation and low cutoff frequency in the frequency spectrum. For such data, we present two approximation methods: linear and Bezier curve approximation. The linear approximation uses a one-pass algorithm, and maintains the user accuracy requirement. The Bezier curve approximation algorithm presented achieves a higher data reduction ratio, at the cost of greater computing and memory capacities.

The rest of this paper is organized as follows. We first present the results of correlation and frequency spectrum analysis of temperature data to illustrate the characteristics of sensor data. In section 3, we present our algorithms for sensor data approximation satisfying user accuracy requirement. The performance of these algorithms on data reduction ratio, approximation error, are discussed in section 4. A comparison with the related work is in section 5. Conclusions and a discussion of future work are in the last section.

II. Characteristics of Sensor Data

The slowly varying data, such as temperature, humidity, pressure, stock prices [7], has strong temporal correlation. We used ambient temperature data of two days and analyzed the autocorrelation using SPSS tool. We observe that all the correlation values approximately equal 1 with various lag numbers. This implies that a strong temporal correlation exists among the sensor data. We analyzed the frequency spectrum of the data stream using the Matlab tool and find that it ranges from 0 Hz to 0.067 Hz indicating the data changing period is long. These results suggest that this type of data may be approximated with segments of lines or curves.

III. Approximate Data Representation

A. Linear Approximation

Main Idea: A number of data points can be approximated by a line segment, which only requires transmission of two data points. This compression has to be under the constraint

0-7803-9586-7/06/$20.00 ©2006 IEEE.

that the approximation error of query results should be within the user accuracy requirement.

Algorithm Description: Suppose the user has specified his precision requirement for sensor data as $\pm\sigma$. To meet this requirement, a line segment is constructed between the two farthest data points keeping the error of any data point to be less than σ. This is a one-pass process and is described by Algorithm 1. The first data point is set as the start point of the first line segment. Successive data points are ignored, until a data comes that differs the start data point by more than $\sigma/2$. Suppose the data point i is the first one that differs by more than $\sigma/2$, then the predecessor data point i 1, is used as the end point of the line segment. We set $\sigma/2$ as errorBar in the algorithm. This guarantees the approximation errors of all data points is within σ. The line representation of the temperature data for three hours having 180 data points has 13 line segments and is shown in Figure 1. The end point of a line segment is the start point of the successor. Thus, the 13 line segments need 14 data points to be stored. The data reduction ratio for this example is 92.2%. More experimental results are reported in section 4.

Algorithm 1	Linear Approximation Algorithm
1:	$errorBar = \sigma/2$
2:	**while** $i < n$
3:	$\{\ startP = i$
4:	**while** $\|v[startP] - v[i]\| < errorBar$
5:	$i++$
6:	$drawLine(startP, i-1)$
7:	$i = i-1\}$

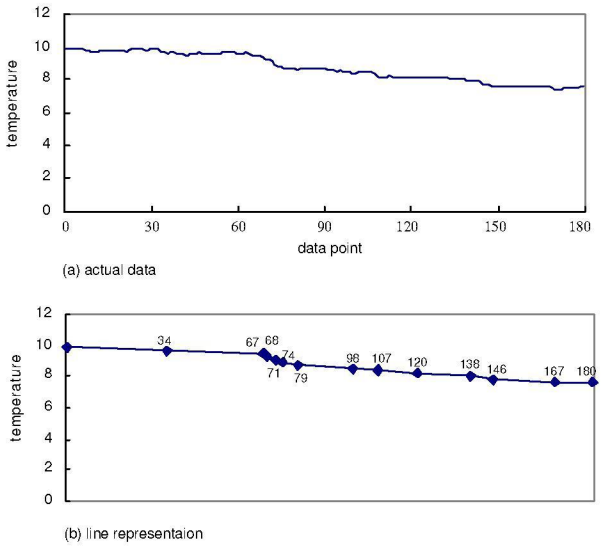

Fig. 1. Line Representation of the Data for Three Hours

Features of the Algorithm: This is a one-pass algorithm well suited for streaming sensor data and resource constraints of sensor nodes. The complexity of this algorithm is $O(n)$, where n is the number of data points in a data stream. It only needs to cache some data points before a line is constructed over them, and hence needs only a small amount of memory. These data points can be used to answer queries referring to them. The queries referring to the data points which have already been processed into a line segment can be answered by line segments approximately.

B. Bezier Curve Approximation

Main Idea: Intuitively, a curve should be able to better match the values of a stream of data points. We investigated the cubic Bezier curve which needs three coefficient points to describe. The Bezier curve changes smoothly and can be evaluated more efficiently than traditional curves [8]. Furthermore, the cubic Bezier curve has greater flexibility of shape than the quadratic Bezier curve. Use of cubic curves instead of that of higher degrees keeps the algorithm complexity lower.

Algorithm Description: Suppose we have K data points to be represented by a Bezier curve. Apart from the start and the end points, two intermediate data points are chosen as control points in position at $\frac{1}{3}K$ and $\frac{2}{3}K$ from the start point. We then determine the corresponding curve and verify if the user accuracy requirement is satisfied between the start and end points. If the error exceeds σ at a particular data point, it is noted as f. After an attempt to determine a new curve by changing the control points to be nearer the start fails once, we construct a new curve from the start point to f. The start point for next curve segment is the end point of the preceding curve, as illustrated in Algorithm 2. The curve representation of the temperature data for seven hours having 420 data points is illustrated in Figure 2, consisting of five curve segments. If we can not find a curve meeting the user error requirements, we say it is incomplete curve. An incomplete Bezier curve requires four points to represent: three coefficients and one end point. The five curves in our case required 20 parameters to store corresponding to a data reduction ratio of 95.2%.

Features of the Algorithm: It is not a one-pass algorithm and requires more computation and memory space than the line segment algorithm, but enables higher data reduction ratio. The algorithm complexity is $O(n)$, where n is the number of data points in a data stream.

IV. Performance Study

Our objectives for approximation are to reduce data transmission, cache data for future queries with minimal memory, meet the user accuracy requirement and answer queries easily. We present experimental results to indicate how the algorithms accomplish these objectives.

Algorithm 2	Bezier Curve Approximation Algorithm
1:	**while** $i < (n - K)$
2:	{ $cp1 = i$;
3:	**if** (flag == 0)
4:	{$cp2 = i + \frac{1}{3}K$;
5:	$cp3 = i + \frac{2}{3}K$;
6:	$cp4 = i + K$;
7:	curve = constructCurve($cp1, cp2, cp3, cp4$);
8:	f = errorCheck(curve);
9:	**if** $f < cp4$
10:	{ **if** (flag == 1)
11:	{curve[j] = drawCurve($cp1, f - 1$);
12:	$i = f - 1$;
13:	flag = 0;}
14:	**else**
15:	{change $cp2$ and $cp3$;
16:	flag = 1; } }
17:	**else**
18:	{ curve[j] = drawCurve($cp1, cp4$);
19:	flag = 0:
20:	$i = cp4$; }
21:	$j++$ }

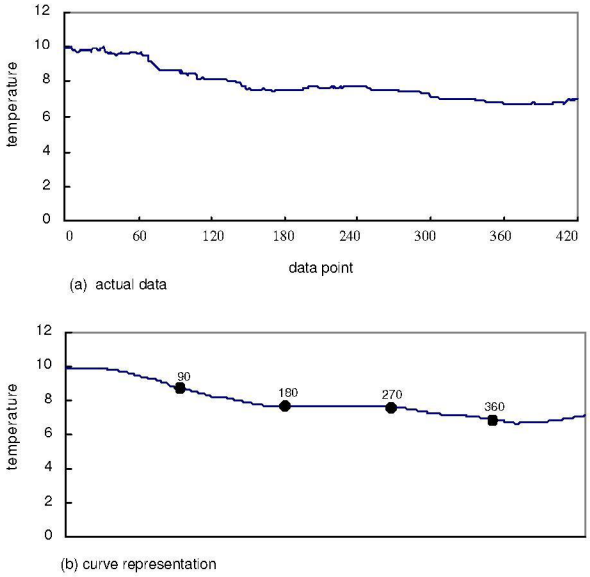

Fig. 2. Bezier Curve Representation of the Temperature Data for Seven Hours

Data Reduction Ratios

We use the real temperature data from Australian Bureau Of Meteorology to test data reduction ratio (denoted by η).

$$\eta = \frac{N_{original} \quad N_{reduced}}{N_{original}}$$

where $N_{original}$ is the the number of actual data points, $N_{reduced}$ is the number of data points in the approximation. The linear and Bezier curve approximation algorithms are applied to the temperature data from different periods of time, and the results are shown in Table 1. We observe that the line segment approximation gives a data reduction of about 88% and the Bezier curve approximation upto around 92%.

TABLE I
DATA REDUCTION RATIOS ($\sigma = 0.5$)

Number of data points	Line Approx. No.	Line Approx. ratio	Bezier Approx. No.	Bezier Approx. ratio
2880 (2days)	335	88.4%	200	93.1%
5760 (4days)	650	88.7%	452	92.2%
8640 (6days)	993	88.5%	720	91.7%

Approximation Errors

Our algorithms are designed for keeping approximation error of each value within σ. We use two days temperature data to tabulate the probability distribution of approximation error, and are shown in Table 2. We find that with the linear approximation method, 94% approximate data values have difference less than 0.2 compared with actual data values. Similarly 81.6% approximate values obtained by Bezier curve have a difference less than 0.2.

TABLE II
PROBABILITY DISTRIBUTION OF APPROXIMATION ERROR (σ = 0.5)

Algorithm	[0,0.1)	[0.1,0.2)	[0.2,0.3)	[0.3,0.4)	[0.4,0.5)
Linear	73%	21%	5%	1%	0%
Bezier curve	55.2%	26.4%	12.1%	4.7%	1.6%

Sensor networks produce large amounts of continuous data, and users are generally interested in the changing tendency or summary character of the situation monitored by sensors, rather than a particular data item. Thus, aggregation queries are the most common queries in sensor networks. Here, we report on the accuracy of the results when queries refer to AVG, MAX, and MIN values. The errors of query results are shown in Table 3, where AvgErr column indicates the approximation errors of query results when a query requests an average value over 24 hours (1440 data points) or 48 hours (2880 data points), MaxErr column indicates the errors for a query with MAX operation and MinErr column is the errors for a query with MIN operation.

TABLE III
ERROR OF QUERY RESULT BY LINE OR CURVE REPRESENTATION (σ = 0.5)

	AvgErr		MaxErr		MinErr	
Approx.	1440	2880	1440	2880	1440	2880
Line	0.003	0.01	0.2	0.2	0.1	0.1
Curve	0.002	0.007	0.2	0.2	0.0	0.0

Query Answering Analysis

As the lines and curves can be represented by equations, the data points for answering queries can be obtained through the line equations or curve equations. The algorithm complexity for getting a data point is $O(1)$.

V. Comparison with Related Methods

Deligiannakes, Kotidis and Roussopoulos presented a Self-Based Regression(SBR) method, for representing multiple data streams in a compressed format by exploiting spatial correlation [7]. Similar to ours, their method is also lossy. However, their algorithm complexity is $O(n^{1.5})$, where n is the number of data in a data set, as against the complexities each of our algorithms is $O(n)$. Their algorithm provides lower Mean Squared Error (MSE) compared with the methods of wavelet transform, DCT, and Histogram. However, the error bound for a single query cannot be guaranteed as the user accuracy requirement is not a parameter in that algorithm. We calculate the MSE with our algorithms and roughly compare with the result from [7] which is also applied on weather data. We tested our algorithms on different data sets, and the lowest reduction ratios are 88% for line representation and 91% for curve representation, as against the 90% achieved from SBR. The comparison is shown in Table 4. Thus we conclude that for slowly varying data, our algorithm can achieve superior performance. Moreover, in SBR method, queries referring to a data stream need use of base signal, which may not convenient for answering queries.

TABLE IV
Comparison with SBR method

	Line	Curve	SBR
Reduction ratio	88%	91%	90%
Mean sq. err.	0.011	0.027	0.403

Some researchers have investigated compression of sensor data on coding level [2] [3] [4]. As the coding methods investigate data value representation at bit level, and hence our algorithms can be used after their compression. Several techniques are used for data stream approximation such as sampling, histogram, and wavelet transforms [9]. Sampling data can be used to answer queries within a known error bound. However, it may not reflect real features of the data in some cases [9]. Data approximation by histogram and wavelet transforms can guarantee a bound on the overall error for answering queries (e.g. L2). However, the answers for individual queries might be heavily biased [10]. Our data approximation method can keep both the errors of individual queries and overall errors within user accuracy requirement. Compared with in-network data aggregation methods, our methods keep the approximate values for all the data points and hence the features such as rate of data change are maintained.

VI. Conclusions and Future Work

We presented two methods for sensor data approximation. Both are suitable for streams in which data changes slowly. They greatly reduce amount of data transmitted and readily answer queries with approximation errors within the user requirements. The algorithms are simple and efficient, and hence suitable for running on sensor nodes which have limited computational resources.

To obtain a better data reduction rate we are working on the following. We are working on improving the line representation algorithm by selecting an optimal error bar in Algorithm 1. Finding optimal control points for curves is being investigated to improve the curve representation algorithm. Further we will address how to deal with sensor data with errors.

References

[1] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *Proceedings of the International conference on Information Technology: Coding and Computing (ITCC'05)*, 2005.

[2] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distirbuted compression in a dense microsensor network," *IEEE Signal Processing Magazine*, March 2002.

[3] T. Arici, B. Gedik, Y. Altunbasak, and L. Liu, "Pinco: a pipelined in-network compression scheme for data collection in wireless sensor networks," in *International Conference on Computer Communications and Networks*, October 2003.

[4] D. Petrovic, R. Shah, K. Ramchandran, and J. Rabaey, "Data funneling: Routing with aggregation and compression for wireless sensor networks," in *IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *Proceedings of Operating Systems Design and Implementation*, December 2002.

[6] A. Cuzzocrea, F. Furfaro, E. Masciari, and D. Sacca, *Approximate Query Answering on Sensor Network Data Streams*. A. Stefanidis, S. Nittel (eds.) CRC Press, September 2004.

[7] A. Deligiannakes, Y. Kotidis, and N. Roussopoulos, "Compressing historical infromation in sensor networks," in *ACM SIGMOD conference*, June 2004.

[8] http://www.damtp.cam.ac.uk/user/na/partiii/cagd2002/parcdef.htm.

[9] L. Golab and M. T. Ozsu, "Issues in data stream management," *ACM SIGMOD Record*, vol. 32, no. 2, June 2003.

[10] M. Garofalakis and P. Gibbons, "Approximate query processing: Taming the terabytes," in *VLDB tutorial*, 2001.