Algorithmica (2007) 49: 357–385
DOI 10.1007/s00453-007-9098-2

Accuracy vs. Lifetime: Linear Sketches for Aggregate Queries in Sensor Networks

Vasundhara Puttagunta · Konstantinos Kalpakis

Received: 5 November 2004 / Accepted: 23 August 2005 / Published online: 13 October 2007

Abstract The in–network aggregation paradigm in sensor networks provides a versatile approach for evaluating aggregate queries. Traditional approaches need a separate aggregate to be computed and communicated for each query and hence do not scale well with the number of queries. Since approximate query results are sufficient for many applications, we use an alternate approach based on summary data–structures. We consider two kinds of aggregate queries: *location range queries* that compute the sum of values reported by sensors in a given location range, and *value range queries* that compute the number of sensors that report values in a given range. We construct summary data–structures called *linear sketches*, over the sensor data using in–network aggregation and use them to answer aggregate queries in an approximate manner at the base–station. There is a trade–off between accuracy of the query results and lifetime of the sensor network that can be exploited to achieve increased lifetimes for a small loss in accuracy. Most commonly occurring sets of range queries are highly correlated and display rich algebraic structure. Our approach takes full advantage of this by constructing linear sketches that depend on queries. Experimental results show that linear sketching achieves significant improvements in lifetime of sensor networks for only a small loss in accuracy of the queries. Further, our approach achieves more accurate query results than the other classical techniques using Discrete Fourier Transform and Discrete Wavelet Transform.

Keywords Sensor networks · In–network aggregation · Linear sketching · Approximate query answering

This work was supported in part by NASA under Cooperative Agreement NCC5–315.

V. Puttagunta (✉) · K. Kalpakis
Dept. of Computer Science and Electrical Engg., University of Maryland Baltimore County,
1000 Hilltop Circle, Baltimore, MD 21250, USA
e-mail: vputta1@csee.umbc.edu

K. Kalpakis
e-mail: kalpakis@csee.umbc.edu

1 Introduction

Wireless sensor networks consist of hundreds of low cost nodes that come with wireless communication and certain processing and storage capabilities in addition to sensing capabilities, and have severe energy constraints. These nodes are deployed in a *sensor field* and can be thought of as distributed streaming data sources. The goal of the sensor network is to collect information from the sensors that is required for various applications at a resource rich base–station. In doing so it should work in an energy efficient manner so that it survives for the longest period of time.

To this end, several energy aware mechanisms for collecting data have been developed. The data–gathering approaches focus on systematically collecting raw data from all the sensors to the base–station. This approach involves communicating large amounts of raw data leading to shorter lifetimes. Fortunately, it is usually not the individual raw data from the sensors that matters to many applications, but certain information in the form of summaries or aggregation of this data that is more relevant [29]. As a result, we have data–fusion or aggregation approaches that make use of the processing power of the sensors to aggregate the data from different nodes as it gets forwarded to the base–station, thereby reducing the communication costs greatly. This mechanism of *in–network aggregation* of data has been shown to dramatically increase lifetimes of sensor networks (Krishnamachari et al. [24]) and has emerged into a core service supported in them (e.g. TAG [30]).

1.1 Query Processing in Sensor Networks

Queries are considered a natural way for users to interact with the sensor network [37]. Further, sensor network applications rely heavily on query processing, particularly of aggregate queries. In this paper we are interested in evaluating aggregate queries over sensor networks in an energy–efficient manner.

From the databases perspective, in–network aggregation provides a mechanism for evaluating aggregate queries over sensor networks. The common approach to (aggregate) query processing in sensor networks has two phases: the *dissemination phase* and the *aggregation phase*. For example consider a query that finds the number of sensors in the sensor field that record temperatures between 10 °F and 30 °F. A user may pose such a query at the base–station using a simple SQL-like language as,

$$\begin{aligned}&\text{select count(*) from sensors}\\&\text{where } 10 \leq \text{temperature} \leq 30.\end{aligned}$$

During the dissemination phase, the query is injected into the sensor network to all the sensors. During the aggregation phase, an aggregation–tree is imposed on the sensor network that is rooted at the base–station, along which in–network aggregation is performed as follows. Every node waits for the counts from each of its children nodes and adds them up. It contributes to the COUNT only if the where–clause is satisfied and forwards the aggregated count to its parent–node. Eventually, the result of the

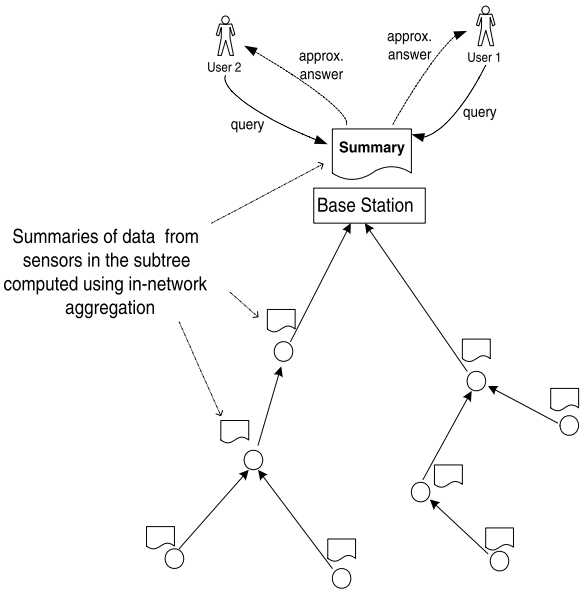

Fig. 1 Summary based query processing in sensor networks

query is computed at the root (base–station). This approach answers queries accurately. However, it has several drawbacks. Every query needs to be communicated to the sensors and this can be an expensive operation. Moreover, when we have multiple aggregate queries (for e.g. queries with different ranges in the where–clause), multiple aggregates need to be maintained and communicated, one for every query. This severely restricts the number of queries that can be answered before the sensor network exhausts its limited battery/energy.

To overcome these drawbacks, we propose an alternate approach that is inspired by *OnLine Analytical Processing* (OLAP) and query processing over streams. Constraints imposed by streams (see Babu and Widom [4]) make techniques for stream computations suitable for sensors. We adopt a summary based approximate query processing approach for sensor networks as illustrated in Fig. 1. During each round, a summary data structure is computed over data from all the sensors using in–network aggregation. At the end of each round, user queries can be answered directly from this summary data structure at the base–station. This way there is no query dissemination phase and any number of queries can be answered at the base–station. Thus the lifetime of the sensor network is not limited by the number of queries. However the kind of summaries that can be computed this way is restricted by the nature of in–network aggregation. Summaries computed in one–pass and over data streams fall into this category. Summaries which can be computed as decomposable aggregate functions are particularly amenable for in–network aggregation paradigm. Such functions can be expressed as an aggregation function f over sets a and b so that $f(a \cup b) = g(f(a), f(b))$, where g is called the *combine* (or merge) function. Further, it is important that such functions are computable in small time and space because

(a) The amount of computation at each sensor to compute the summary determines the number of CPU cycles and the corresponding energy, and
(b) The sensor nodes have limited memory and the space complexity for computing the summary determines the energy required to power memory on the sensor.

But even more critical is

(c) The size of the summary which determines the energy for transmitting and receiving using wireless radio.

Usually there is a direct relationship between the size of the summary and the accuracy of the queries. Since most monitoring applications are interested in approximate query answers, **this approach offers a viable option for increasing the lifetime of the sensor network for a small loss in accuracy of the query results**.

It is important to note that the summary data structure to be computed depends on the kind of queries we wish to answer. We consider the following two important aggregate range queries.

Location Range Queries: What is the aggregate of values recorded by sensors located in a given rectangular region(range)?

$$\begin{array}{l}\text{select sum(reading) from sensors}\\ \text{where } 10 \leq x \leq 20 \text{ and } 10 \leq y \leq 20.\end{array}$$

Value Range Queries: How many sensors recorded values in a particular range?

$$\begin{array}{l}\text{select count(*) from sensors}\\ \text{where } 10 \leq \text{reading} \leq 20.\end{array}$$

Such queries are useful in exploratory data analysis. Techniques for answering such range queries with different where clauses are also useful for finding quantiles in the data [7]. Histograms are popularly used as summary data structures for answering such queries. Recently several sketch based approaches for constructing histograms and answering such queries have been suggested (see for example [15, 36]). Most existing sketch based approaches are designed to capture the Euclidean norm of the data vector. They depend solely on the data and do not consider the queries at all. However, we often have some information about the queries. For example certain regions (ranges) could be more interesting and hence queried more often. Further, in the case of monitoring applications, it is common to have continuous queries, in which case, the same range query is posed in every round. Therefore it is important to design sketches for a specific set of aggregate range queries. In this paper we develop novel query based summaries for the location range queries and the value range queries. These summaries are computed over the sensor network using in–network aggregation and are used to answer multiple queries of the same kind over different ranges at the base–station.

1.2 Road Map

In Sect. 2 we pose the problem of linear sketching given a set of queries and present our main result on query based linear sketches. In Sect. 3 we describe how location

and value range queries can be answered in sensor networks using query based linear sketches. We evaluate our approaches experimentally and present results in Sect. 4. In Sect. 5 we give references to previous work and conclude in Sect. 6.

2 Theory of Linear Sketching

In this section we define *linear sketches*. We consider vector representations of queries, and see how they can be evaluated exactly and approximately using linear sketches. We then pose the problem of finding k representative queries or equivalently a good sketching matrix with k columns. We present some classical sketches used in signal processing, data–reduction, and stream processing. Finally, we design linear sketches that are based on the given set of queries and give error bounds using such sketches.

2.1 Overview of Matrix Analysis

We introduce various basic concepts and notations. For brevity, we provide a quick overview of concepts and facts from matrix analysis that we will be using in subsequent sections. The interested reader is referred to an appropriate reference on matrix analysis, such as Lancaster and Tismenetsky [25]. The proofs of theorems in this section, that are stated without proofs, can be found in [25]. Unless stated otherwise, we are concerned with matrices and vectors over the field of complex numbers $\mathcal{C}$.

Definition 1 Let A be a matrix over $\mathcal{C}$. The complex conjugate of A is denoted by $\overline{A}$. The conjugate transpose A^* of A is defined as $A^* = \overline{A}^T$. Matrix A is *Hermitian* if and only if $A^* = A$. Matrix A is *idempotent* if and only if $A^2 = A$. A matrix A is said to have orthonormal columns when $A^*A = I$.

Definition 2 (Standard Inner Product, Norm) For any two complex vectors $x, y \in \mathcal{C}^n$, the standard inner product is the scalar given by $\langle x, y\rangle = y^*x$. The *norm* of x is given by $\|x\|^2 = \langle x, x\rangle = x^*x$.

Definition 3 (Column Subspace and Complementary Orthogonal Subspace) Let A be an $n \times m$ complex matrix. The column space $\mathcal{S}_{(A)}$ of A is the subspace of $\mathcal{C}^n$ spanned by the columns of A. The complementary orthogonal subspace of $\mathcal{S}_{(A)}$ is denoted by $\mathcal{S}^{\perp}_{(A)}$.[1] Each vector in $\mathcal{S}^{\perp}_{(A)}$ is orthogonal to all vectors in $\mathcal{S}_{(A)}$, and any vector $x \in \mathcal{C}^n$ can be written as $x = x_1 + x_2$ where $x_1 \in \mathcal{S}_{(A)}$ and $x_2 \in \mathcal{S}^{\perp}_{(A)}$.

Proposition 1 (Projector Matrices) *A complex matrix P is a projector matrix if and only if it is Hermitian and idempotent. A projector matrix P projects each vector on its column space. Projector matrices commute. If A is an $n \times m$ complex matrix with orthonormal columns, then*

[1] Two subspaces $\mathcal{S}_2$ and $\mathcal{S}_2$ are said to be orthogonal if for any $x \in \mathcal{S}_1$ and $y \in \mathcal{S}_2$, $\langle x, y\rangle = \langle y, x\rangle = 0$, i.e. x and y are perpendicular to each other.

1. *AA^* is a projector matrix that projects any vector in $\mathcal{C}^n$ onto $\mathcal{S}_{(A)}$, and*
2. *$A^\perp = I - AA^*$ is a projector matrix that projects any vector in $\mathcal{C}^n$ onto $\mathcal{S}^\perp_{(A)}$.*

Definition 4 (Matrix Eigenvalues and Eigenvectors) Let A be an $n \times n$ complex matrix. A scalar λ is an eigenvalue of A if and only if there exists a non–trivial vector $v \in \mathcal{C}^n$, called its corresponding eigenvector, such that $Av = \lambda v$. Let $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ denote the eigenvalues of A with maximum and minimum norm respectively.

Theorem 1 (Eigenvalues and Eigenvectors of Hermitian) *If A is an $n \times n$ Hermitian matrix, then eigenvalues of A are real. In that case, matrix A has n eigenvalues, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ with corresponding (orthonormal) eigenvectors $v_1, v_2, \ldots, v_n$. An eigenspace of A is the subspace of $\mathcal{C}^n$ spanned by a subset of A's eigenvectors. The set of $1 \leq k \leq n$ eigenvalues of A with largest norm is called the set of top–k eigenvalues of A, and the set of their corresponding eigenvectors is called a set of top–k eigenvectors of A.*

Definition 5 (Rayleigh Quotient) Let A be an $n \times n$ Hermitian matrix. The Rayleigh quotient of A is the function

$$R_A(x) = \frac{\langle Ax, x\rangle}{\langle x, x\rangle} = \frac{x^*Ax}{x^*x}, \tag{1}$$

over all non–zero vectors $x \in \mathcal{C}^n$.

Theorem 2 (Courant–Fischer) *Let A be an $n \times n$ Hermitian matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, and corresponding orthonormal eigenvectors $v_1, v_2, \ldots, v_n$. Let $\mathcal{S}_j$ be an arbitrary $(n - j + 1)$–dimensional subspace of $\mathcal{C}^n$, $1 \leq j \leq n$. Then,*

$$\lambda_j = \max_{\mathcal{S}_j} \min_{0 \neq x \in \mathcal{S}_j} R_A(x) \tag{2}$$

and

$$\lambda_{n-j+1} = \min_{\mathcal{S}_j} \max_{0 \neq x \in \mathcal{S}_j} R_A(x). \tag{3}$$

The extreme in (2) is attained when S_j coincides with the eigenspace generated by $v_j, v_{j+1}, \ldots, v_n$. The extreme in (3) is attained when S_j coincides with the eigenspace generated by $v_1, v_2, \ldots, v_{n-j+1}$.

2.2 Linear Sketching

Definition 6 (Linear Sketch) Let P be an $n \times k$ complex matrix, with columns $p_1, p_2, \ldots, p_k$ so that $P = [p_1 p_2 \cdots p_k]$. We call P a sketching matrix. Let x be a vector in $\mathcal{C}^n$. The projection of x onto p_i is given by $\langle x, p_i\rangle$, $i = 1, 2, \ldots, k$. The (linear) sketch or projection of x with respect to the sketching matrix P is the k–dimensional vector given by $\widehat{x} = [\langle x, p_1\rangle \langle x, p_2\rangle \cdots \langle x, p_k\rangle]^T$. Equivalently,

$$\widehat{x} = P^*x. \tag{4}$$

The sketch of an $n \times m$ matrix A with respect to the sketching matrix P is a matrix whose columns are sketches of the corresponding columns of A, i.e. $\widehat{A} = P^* A$.

2.2.1 Query Vectors and Query Matrices

Let $q \in \mathcal{C}^n$ be the vector representation of a query so that the answer to the query q evaluated against a data vector $x \in \mathcal{C}^n$ is evaluated using the standard inner product

$$\langle x, q \rangle = q^* x. \tag{5}$$

Example 1 Consider the following query,

select COUNT(*)
from data
where $3 \leq x \leq 5$.

To evaluate this query, we consider the frequency distribution of the data as our data vector x, i.e. the ith component of x is the frequency of i in the data. Assume that the data takes values from the set $\{1, 2, \ldots, 6\}$. Let $x = [3, 5, 8, 5, 3, 1]^T$.

Then the SQL–query can be represented by the vector $q = [0, 0, 1, 1, 1, 0]^T$ and the answer to the query $\langle h, q \rangle = q^* h = 0 + 0 + 8 + 5 + 3 + 0 = 16$.

When q is a vector with 0–1 entries, $q^* x$ simply sums the components of x where there are 1's in q. Therefore we refer to such queries as **sum–queries**. Further, sum–queries that have all the 1's together are referred to as **range–sum queries** and the number of 1's in the query vector is referred to as the **extent** of the range–sum query. The query vector in Example 1 is a range–sum query whose extent is 3.

Consider a query matrix Q which is a matrix whose columns are query vectors. The exact answers to queries in Q is a vector,

$$a = \langle x, Q \rangle = Q^* x. \tag{6}$$

Let Q be an $n \times m$ matrix whose columns correspond to m queries. Then from (4) and (6), we can see that $\widehat{x} = Q^* x$ yields a vector of length m, whose components are the exact answers to the m queries in the query matrix Q. In other words, when we use the query matrix Q as the sketching matrix, the sketch $\widehat{x}$ of the data vector gives the exact answers to all the queries. In general, the number of queries m can be very large making the sketch too big. It is desirable to have small sketches ($k \ll m$) in many applications including query processing over sensor networks as we shall see in Sect. 4.

2.2.2 Estimating Query Results Using Sketches

The data vector x can be very large. Therefore our approach is to use a sketch $\widehat{x}$ that is much smaller in size for answering queries. An approximate answer to the query q with respect to vector x can be estimated by the inner product between their sketches

$\widehat{q}$ and $\widehat{x}$ respectively as $\langle \widehat{x}, \widehat{q} \rangle = \widehat{q}^* \widehat{x}$. Therefore approximate answers to queries in matrix Q with respect to x,

$$\widetilde{a} = \langle \widehat{x}, \widehat{Q} \rangle = (\widehat{Q})^* \widehat{x}. \tag{7}$$

Further, the error in estimating the answer to the queries is given by the error vector,

$$e = a - \widetilde{a} = Q^* x - Q^* P P^* x = Q^* (I - P P^*) x, \tag{8}$$

where I is the identity matrix.

The sum of squared errors (SSE) over all the queries in the query matrix is given by,

$$\text{SSE} = \|e\|^2 = e^* e. \tag{9}$$

Ideally, the sketching matrix used to generate the sketches of x and q should be such that the answer estimated using the sketches is the same as the exact answer. This is true for example when we use the identity matrix as the sketching matrix. However this does not reduce the size of the sketch. Therefore, we want to have a sketching matrix with a small number (k) of columns that minimizes the SSE.

Since a sketch of x with respect to a sketching matrix P contains exact answers to queries that are represented by columns of P, the problem is equivalently that of *finding a small number* (k) *of representative queries whose answers can be maintained accurately, so that we can use their answers to estimate answers to queries in* Q *effectively?*

We consider sketching matrices with orthonormal columns. Then we have $e = Q^*(I - P P^*)x = Q^* P^\perp x$, where $P^\perp$ is a projector matrix that projects any vector in C^n onto the complementary orthogonal subspace of the column space of P. Therefore we have

$$\text{SSE} = e^* e = (a - \widetilde{a})^* (a - \widetilde{a}) = (Q^* P^\perp x)^* Q^* P^\perp x = (P^\perp x)^* Q Q^* P^\perp x,$$

QQ^* is a Hermitian matrix and we have from the definition of Rayleigh quotient of QQ^*, $R_{QQ^*}(P^\perp x) = \frac{(P^\perp x)^* QQ^* P^\perp x}{(P^\perp x)^* (P^\perp x)}$. Therefore we have,

$$\begin{aligned}
\text{SSE} &= R_{QQ^*}(P^\perp x)[(P^\perp x)^* (P^\perp x)] = R_{QQ^*}(P^\perp x)[x^* P^{\perp *} P^\perp x] \\
&= R_{QQ^*}(P^\perp x)[x^* P^\perp x] = R_{QQ^*}(P^\perp x)[x^* (I - P P^*) x] \\
&= R_{QQ^*}(P^\perp x)[x^* x - x^* P P^* x] = R_{QQ^*}(P^\perp x)[x^* x - (P^* x)^* (P^* x)] \\
&= R_{QQ^*}(P^\perp x)[x^* x - \widehat{x}^* \widehat{x}] = R_{QQ^*}(P^\perp x)[\|x\|^2 - \|\widehat{x}\|^2].
\end{aligned}$$

2.2.3 Problem Statement

Given a data vector $x \in \mathcal{C}^n$, an $n \times m$ query matrix Q,

– The exact answers to queries in Q are given by,

$$a = Q^* x. \tag{10}$$

– The estimated answers to queries in the query matrix Q using an $n \times k$ sketching matrix P with k orthonormal columns ($k \ll \min(m, n)$) are given by,

$$\widetilde{a} = \widehat{Q}^* \widehat{x} = (P^* Q)^* (P^* x) = Q^* P P^* x. \tag{11}$$

– The sum of squared errors over all the queries in the Q matrix is given by,

$$\text{SSE} = e^* e = R_{QQ^*}(P^{\perp} x)[\|x\|^2 - \|\widehat{x}\|^2], \tag{12}$$

where $P^{\perp} = I - P P^*$.

The problem is to find a sketching matrix P that minimizes SSE.

In this paper we consider the above problem when the data vector x is not know in advance and is given incrementally.

2.2.4 Classical Sketches

Definition 7 (Fourier Matrix) A Fourier matrix F of order n is an $n \times n$ matrix for which

$$F^*_{(j,k)} = \frac{1}{\sqrt{n}} \omega^{(j-1)(k-1)}, \tag{13}$$

where ω^i for $i = 0, \ldots, n-1$ are the n distinct complex roots of unity. Hence,

$$F^* = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{n-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \omega^{n-2} & \cdots & \omega \end{bmatrix}. \tag{14}$$

Every Fourier matrix F is unitary. We refer to the columns of F^* as Fourier vectors.

Definition 8 (Discrete Fourier Transform) The Discrete Fourier Transform (DFT) of a vector $x \in \mathcal{C}^n$ is given by

$$\widehat{x} = F x, \tag{15}$$

The Fourier matrix is a classical sketching matrix. From the signal processing perspective, when we use the Fourier matrix as the sketching matrix, the sketch $\widehat{x}$ is simply the Discrete Fourier Transform (DFT) of the data vector x. The DFT of a vector of length n is usually computed using the celebrated Fast Fourier Transform (FFT) algorithm in $O(n \lg n)$ time and not by the direct computation as implied by (15), which takes $O(n^2)$ time. However, if we need to maintain only certain k Fourier coefficients, this direct computation approach requires only $O(nk)$ time. Similarly any Wavelet matrix can be used as the sketching matrix to get the corresponding Discrete Wavelet transform. Therefore when P contains the wavelet vectors, $\widehat{x}$ gives the respective wavelet coefficients of x.

The standard practice is to maintain the top–k coefficients of the data vector which implicitly means having only vectors corresponding to the top–k coefficients as the sketching matrix. By maintaining the top–k coefficients, the difference in norms of the data vector and its sketch (i.e. the $[\|x\|^2 - \|\widehat{x}\|^2]$ factor in (12)) is minimized. Thus these approaches rely on the ability to capture the norm (energy) of the data vector using the top–k coefficients and do not make use of the queries at all. (The sketching matrix depends only on the data vector.) Further, in the scenario where the data vector is not known in advance and is presented in an incremental fashion, we do not know which coefficients would belong to the top–k coefficients and hence have to be estimated as the data is presented. The errors in estimating query results escalate with the error in estimating the top–k coefficients.

Linear sketches using random projections have been recently used for streaming data applications (see for example [36]). In this case, the sketching matrix, simply contains random vectors. Due to the Johnson–Lindenstrauss lemma [8], the linear sketch of a vector using random projections has the nice property that the norm of the vector can be estimated well from the norm of the sketch. Again, such sketches try to minimize the $[\|x\|^2 - \|\widehat{x}\|^2]$ factor in (12) but do not consider the Q matrix at all.

2.2.5 Query Based Sketches

As observed earlier, classical sketches do not consider the queries at all. However, queries often have certain patterns and structures that can be exploited to get better results. For this reason, we propose to use an alternate approach of using sketching matrices that depend on the queries (we try to find a sketching matrix that minimizes the Rayleigh quotient factor in (12)). We have the following theorem on sketching matrices

Theorem 3 *Let Q be an $n \times m$ query matrix, x a non–zero data vector, and P be an $n \times k$ sketching matrix with orthonormal columns. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of the matrix QQ^*, and let $v_1, v_2, \ldots, v_n$ be its corresponding orthonormal eigenvectors. Then we can achieve the following bound on the SSE by choosing P to have as columns the top–k eigenvectors of the matrix QQ^*, i.e. when $P = [v_{n-k+1} v_{n-k+2} \cdots v_n]$.*

$$\|e\|^2 \leq \lambda_{n-k}(\|x\|^2 - \|\widehat{x}\|^2). \tag{16}$$

Proof The sum of squared errors (SSE) over all the queries in the query matrix is given by,

$$\text{SSE} = e^*e = R_{QQ^*}(P^{\perp}x)[\|x\|^2 - \|\widehat{x}\|^2].$$

Since for any vector x, since $P^{\perp}x \in \mathcal{S}^{\perp}_{(P)}$ we have,

$$R_{QQ^*}(P^{\perp}x) \leq \max_{y \in \mathcal{S}^{\perp}_{(P)}} R_{QQ^*}(y).$$

Thus, we want to find P so that the maximum value of $R_{QQ^*}(y)$ for any $y \in \mathcal{S}^{\perp}_{(P)}$ is minimized.

The rank of the sketching matrix P is equal to k, since its columns are linearly independent. Thus, P's column space and the complementary orthonormal space of its column space are of dimension k and $n-k$ respectively. From the Courant–Fischer (Theorem 2) we know that

$$\min_{S} \max_{y \in S} R_{QQ^*}(y) = \lambda_{n-k},$$

where the minimization is over all subspaces of $S \subseteq \mathcal{C}^n$ of dimension $n-k$, and that the minimum is achieved for the eigenspace spanned by the eigenvectors $v_1, v_2, \ldots, v_{n-k}$. Note that in that case, the complementary orthonormal subspace of S is the eigenspace spanned by the eigenvectors $v_{n-k+1}, \ldots, v_n$. Therefore, we want $\mathcal{S}_{(P)}^{\perp}$ to be equal to the eigenspace spanned by the eigenvectors $v_1, v_2, \ldots, v_{n-k}$, and thus the space $\mathcal{S}_{(P)}$ to equal to the eigenspace spanned by the eigenvectors $v_{n-k+1}, \ldots, v_n$.

Therefore, when $P = [v_{n-k+1} v_{n-k+2} \cdots v_n]$, we have that

$$\begin{aligned} \text{SSE} &= R_{QQ^*}(P^{\perp}x)[\|x\|^2 - \|\widehat{x}\|^2] \leq \min_{S} \max_{y \in S} R_{QQ^*}(y)[\|x\|^2 - \|\widehat{x}\|^2] \\ &\leq \lambda_{n-k} \ [\|x\|^2 - \|\widehat{x}\|^2]. \end{aligned}$$

□

Therefore we use the sketching matrix whose vectors are the top–k eigenvectors of the matrix QQ^*.

Further, there are ways to extend an existing sketching matrix by adding a few columns to improve the accuracy of a given query matrix (see [33]). For example, if the original sketch depends only on the data, our approach provides a mechanism to extend it so that it depends on the queries as well.

3 Answering Queries in Sensor Networks

In this section, we show how the above results on query based sketching can be applied to answer location range queries and value range queries over sensor networks.

3.1 Answering Location Range Queries

Location range queries are the queries that ask for the sum of values reported by sensors located in a particular (spatial) range. Let n be the number of sensors. Let the data vector h be an n–dimensional vector containing data generated by all the sensors that is indexed by sensor–id. Then a location range query gets translated into a sensor-query q (a 0–1 vector with 1's at the indices corresponding sensors located within the query range and 0's otherwise), so that the exact answer to the range query can be answered using $\langle h, q \rangle = q^* h$. Location range queries are sum–queries but need not be range–sum queries since they may not contain all the 1's together. Figure 2 illustrates an example location range query. In order to obtain the data vector h at the base–station using in–network aggregation, we can consider each sensor to be generating an n–dimensional vector as follows. Say the value recorded by sensor–i

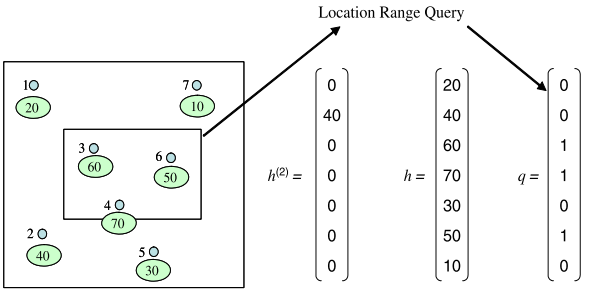

Fig. 2 Location Range Query: The sensor field consists of 7 sensors whose readings are within the ellipses. $h^{(2)}$ is a vector at sensor–2. h is the data–vector with sensor data from all the sensors. q is the vector representation of the location range query, the exact answer to which is $\langle q, h\rangle = q^*h = 60 + 70 + 50 = 160$

is α. Then it generates a n–dimensional vector $h^{(i)}$ with α as the ith element and 0's everywhere else. Then $h = \sum_{i=1}^{n} h^{(i)}$. In other words, h is a simple *SUM* aggregate of the vectors generated at each sensor and can therefore be computed using in–network data–aggregation scheme in sensor–networks. However, this approach will need packets of size $O(n)$ to be communicated and aggregated. Since the number of sensors n can be very large, it may be infeasible to compute the data vector using in–network aggregation due to energy constraints.

We propose to apply the linear sketching approach mentioned previously for this problem. Because sketching is a linear operator, the sketch $\widehat{h}$ using the sketching matrix P is simply the sum of sketches of the individual vectors generated at each sensor. i.e. $\widehat{h} = P^*h = P^* \sum_{i=1}^{n} h^{(i)} = \sum_{i=1}^{n} P^*h^{(i)} = \sum_{i=1}^{n} \widehat{h}^{(i)}$. Therefore, just as h is a sum–aggregate over the vectors generated at each sensor, $\widehat{h}$ is also a sum–aggregate over the sketches of these vectors using the sketching matrix P and can therefore be carried out over the sensor–network using any data–aggregation service.

Note that location range queries are not any 0–1 vectors. When a certain sensor is queried, the probability that sensors located in the vicinity are queried as well is high where as the probability that sensors that are far away are queried is low. For example consider 200 sensors that are randomly placed in a 100 m $\times$ 100 m sensor field. Let Q contain query vectors that are representations of all location range queries of extent 30 m $\times$ 30 m over the sensor field. Figure 3 shows the eigenvalues of QQ^*. It is clear that the eigenvalues drop very quickly. Therefore using only the top few eigenvectors as the sketching matrix can reduce the SSE greatly (from Theorem 16). Given a certain ordering of the sensors, let Q be a query matrix with some location range queries. Further let QQ^* be diagonalized as $QQ^* = V\Lambda V^*$, i.e. Λ is a diagonal matrix with eigenvalues on the diagonal and V has eigenvectors of QQ^* as eigenvectors. With a different ordering of the sensors, the query matrix is a row permuted version of Q, i.e. the new query matrix is $Q' = EQ$, where E is an $n \times n$ permutation matrix. Then $Q'(Q')^* = EQQ^*E^* = E(V\Lambda V^*)E^* = (EV)\Lambda(EV)^*$. Therefore the eigenvalues (Λ) are the same and the eigenvectors of $Q'Q'^*$ are the columns of (EV), i.e. they are simply the corresponding row permuted versions of V. The actual ordering of the sensors only permutes (row–wise) the sketching matrix but does not change the eigenvalues and consequently it does not change the error bound. Thus it does not matter what ordering of the sensors we consider.

Further since the vector generated at each sensor contains only a single non–zero element at the index corresponding to its sensor–id, sensor–i requires only the ith row of the sketching matrix P to compute the sketch of the vector it generates. In

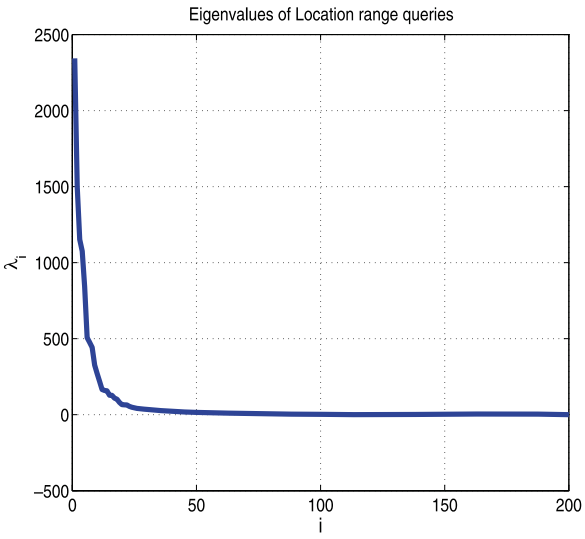

Fig. 3 Eigenvalues of the matrix QQ^* in ascending order

fact, $\widehat{h}^{(i)} = \alpha r_i$, where r_i is the ith row–vector of P. Given a query matrix Q, the sketching matrix P consisting of the top–k eigenvectors of QQ^* is computed (at the base–station). Let $r_i, i = 1, 2, \ldots, n$ be the rows of P. The sketch of the data vector $\widehat{h}$ is computed using in–network aggregation as follows:

1. Each sensor–i stores the ith row of the sketching matrix r_i. Using finite precision, this takes $O(k)$ space to store.
2. During every round, each sensor–i computes the sketch of the data generated by it during that round as follows $\widehat{h^{(i)}} = \alpha r_i{}^*$, where α is the value it recorded in that round. This needs k multiplications.
3. Each sensor waits until it receives sketches from all the child–sensors in the aggregation tree and computes the sum of all the sketches including its own.
4. It transmits this added sketch to its parent–sensor in the aggregation tree.

Since the sketch size is k, the transmit and receive costs will only be $O(k)$. Using a b–byte representation for each component of the sketch, we need to communicate and aggregate packets of size $(bk + p)$ bytes where p is the header size. Typically $k \ll n$. We consider $p = 8$ and $b = 2$. The actual queries are evaluated at the base–station using the sketch $\widehat{h}$ by computing the sketch of the query vector q as $\widehat{q} = P^*q$ and taking the inner product $\langle \widehat{q}, \widehat{h} \rangle$.

3.2 Answering Value Range Queries

Value range queries ask for the number of sensors that record values in a given range. The frequency distribution of the sensor data gives the number of sensors recording a particular value. Maintaining the frequency distribution of values from all the sensors allows us to answer value range queries accurately by simply adding the frequencies reported at the values in the query range.

We assume that the values recorded by sensors take integer values between 1 and M. Therefore, the frequency distribution is an $M \times 1$ column vector that we consider as our data vector h. This is also referred to as the complete histogram of the dataset. A value range query can then be represented as a 0–1 column vector q of the same size, with 1's in the range of the query and 0's everywhere else. Note that value range queries are range–sum queries because they contain consecutive 1's and can be answered accurately using $\langle h, q \rangle = q^* h$. To use this approach we need to be able to compute the frequency distribution, h over the sensor network. We observe that, if we consider each of the sensors generating an $M \times 1$ column vector with a 1 at the index corresponding to the value it measures, then h is again simply the sum of such vectors from all the sensors. Let $h^{(i)}$ be the vector generated by the i^{th} sensor, then $h = \sum_{i=1}^{n} h^{(i)}$. Therefore, as in the case of location range queries, h is a simple *SUM* aggregate of the vectors generated at each sensor and can therefore be computed using any in–network data–aggregation scheme in sensor–networks. However, this approach will need packets of size $O(M)$ to be communicated and aggregated which may be infeasible due to energy constraints.

Therefore, we use the linear sketching approach. Again, because of the linearity of sketching, the sketch $\widehat{h}$ using the sketching matrix P is simply the sum of sketches of the individual vectors generated at each sensor. i.e. $\widehat{h} = P^* h = P^* \sum_{i=1}^{n} h^{(i)} = \sum_{i=1}^{n} P^* h^{(i)} = \sum_{i=1}^{n} \widehat{h}^{(i)}$. Therefore, just as h is a sum–aggregate over the vectors generated at each sensor, $\widehat{h}$ is also a sum–aggregate over the sketches of these vectors using the sketching matrix P and can therefore be carried out over the sensor–network using any data–aggregation service.

The only difference is that the entire sketching matrix P needs to be available at each sensor in order to compute the sketch of the data it generates. Storing it explicitly takes $O(Mk)$ space which can be too big because M can be very large in general. However, computing the sketch involves only retrieving the appropriate row of the P matrix. And since the sketch size is k, the energy consumed for transmitting and receiving is still $O(k)$.

Fortunately, value range queries display rich structure. In the next subsection, we consider certain sets of value range queries that arise naturally. We characterize the eigenvectors of QQ^* in such cases and show that the eigenvectors and hence the sketching matrix has a succinct representation. Therefore, the sketching matrix need not be stored explicitly at each sensor and any element of the matrix can be generated on the fly using a fast small space program.

3.2.1 Special Sets of Value Range Queries

Fixed Extent Value Range Queries: In the case of range queries, the query vectors contain blocks of consecutive ones. Let Q contain all range queries (with wrap–around) of a fixed extent (i.e. query vectors with fixed number of consecutive ones) then Q and QQ^* are circulant matrices.

Definition 9 (Circulant Matrix) A circulant (matrix) of order n, is a square matrix of the form

$$C = \text{circ}(c_0, c_1, \ldots, c_{n-1}) = \begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-2} \\ \vdots & \vdots & \vdots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{bmatrix}. \tag{17}$$

Each row of C is a circular right shift of its preceding row.

C is a circulant iff C^* is a circulant. The product of two circulant matrices is also a circulant.

For example, the following query matrix Q has as columns, all range queries of fixed extent 2 with $M = 4$.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Note that we consider the domain values M and 1 to be adjacent. The last column of Q is a query that wraps around the boundary. We shall refer to such queries as wrap–around queries. Although wrap–around queries may be uncommon, we include them here so that the query matrix is circulant and for the algebraic properties that follow.

We have the following theorem on circulant matrices.

Theorem 4 (See Davis [9]; Universal Eigenvectors of Circulants) *Let* $C = \text{circ}(c_0, c_1, \ldots, c_{n-1})$ *which is a circulant matrix of order* n. *Matrix* C *is diagonalized by the Fourier matrix* F *of order* n,

$$C = F^* \Lambda F, \tag{18}$$

where Λ *is a diagonal matrix with the eigenvalues of* C *as its diagonal elements. Thus, each column of* F^* *is a* (*right*) *eigenvector of every circulant matrix, i.e. the columns of* F^* *is a universal set of eigenvectors for all the circulant matrices. Moreover, for each root of unity* $\rho \in \{1, \omega, \omega^2, \ldots, \omega^{n-1}\}$,

$$\lambda(\rho) = \sum_{k=0}^{n-1} c_k \rho^k \tag{19}$$

is an eigenvalue of C, *and its corresponding eigenvector is*

$$\frac{1}{\sqrt{n}} [\, 1 \quad \rho \quad \rho^2 \quad \cdots \quad \rho^{n-1} \,]^T. \tag{20}$$

Further, any matrix that is diagonalized by the Fourier matrix is a circulant matrix.

Therefore by Theorem 4, the eigenvectors of QQ^* are in fact the Fourier vectors. Further, if $Q = F^* \Lambda F$, then $QQ^* = F^* \Lambda F F^* \Lambda^* F = F^* \Lambda \Lambda^* F$. This means

that the if λ is an eigenvalues of Q corresponding the eigenvector v then $\|\lambda\|^2$ is the eigenvalue of QQ^* corresponding to the same eigenvector. i.e. if $Qv = \lambda v$ then $QQ^*v = \|\lambda\|^2 v$. Thus we can have the sketching matrix P to be the certain Fourier vectors corresponding to the top–k eigenvalues of QQ^* and then the sketch $\widehat{h}$ is simply the corresponding Fourier coefficients of h. Note that the choice of the Fourier coefficients depends solely on the query matrix Q and may not be the same as the top–k Fourier coefficients of the data vector h. The advantage of Fourier vectors is that they have a succinct representation and do not have to be explicitly stored. Any element of the sketching matrix can be generated on the fly using (13).

Let $f_\rho = [\,1 \quad \rho \quad \rho^2 \quad \cdots \quad \rho^M\,]^T$, then given a value range query q over a value range $[l, u]$, $1 \le l \le u \le M$, the inner-product $\langle f_\rho, q\rangle$ is the sum of $(u - l + 1)$ terms in a geometric series: $\langle f_\rho, q\rangle = \sum_{i=l-1}^{u-1} \rho^i$, and can be computed as

$$\langle f_\rho, q\rangle = \begin{cases} u - l + 1, & \rho = 1, \\ \frac{\rho^{u-l+1}-1}{\rho-1}, & \rho \neq 1. \end{cases} \tag{21}$$

Since in this case, the sketching matrix contains columns of the form f_ρ where ρ belongs to the Mth roots of unity, the sketch of the query vector can be computed in $O(k)$ time using (21) at the base station.

However, we are more likely to have value range queries whose extent follows a certain known distribution (e.g. a normal distribution with known mean extent) rather than a fixed extent. In that case we can consider the sketching matrix that is designed for fixed extent value range queries (as above) whose extent is the mean extent. Note that even if the query that needs to be evaluated is different from those in the query matrix Q, we can still evaluate it approximately using the sketches with reasonable errors (see the next section for experimental results).

All Value Range Queries: Next, we consider the case when Q contains all possible range–sum queries (of all extents) but without the wrap around queries. In this case we show that QQ^* is a special matrix whose eigenvectors are vectors from the Discrete Sine Transform.

A value range query is an $M \times 1$ vector, that has ones in the range $[i, j]$ and zeroes everywhere else. We consider all possible range queries, i.e., $1 \le i \le j \le M$. Therefore the query matrix has $M(M+1)/2$ range–sum queries. We shall refer to this matrix as Q_M. The sketching matrix consists of the top–k eigenvectors of the matrix $Q_M Q_M{}^*$. Let $D_M = Q_M Q_M{}^*$.

For example Q_4 has as columns, all possible value range queries with $M = 4$ and $D_4 = Q_4 Q_4{}^*$.

$$Q_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \qquad D_4 = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 6 & 4 & 2 \\ 2 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}.$$

D_M is a special matrix with interesting algebraic properties.

Lemma 1 *D_M is a symmetric matrix whose (i, j)th element is given by,*

$$(D_M)_{i,j} = \begin{cases} i(M - j + 1), & i \leq j, \\ (D_M)_{j,i}, & i > j. \end{cases} \tag{22}$$

Proof The (i, j)th element of D_M gives the number of queries that have 1's at both index i and index j. Thus $(D_M)_{i,j} = (D_M)_{j,i}$. Further, $(D_M)_{i,j}$ contains all the range queries $[l, u]$, where $1 \leq l \leq i \leq j \leq u \leq M$. Therefore l can take i values from $\{1, 2, \ldots, i\}$ and u can take $M - j + 1$ values from $j, j + 1, \ldots, M$. Therefore there are $i(M - j + 1)$ such queries and hence $(D_M)_{i,j} = i(M - j + 1)$ for $i \leq j$. □

Definition 10 (K_M) K_M is an $M \times M$ matrix with 2's on its principal diagonal, -1's on the diagonals above and below the principal diagonal. For, e.g.

$$K_4 = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}. \tag{23}$$

K_M is a symmetric tridiagonal Toeplitz matrix that appears commonly in many engineering applications (see [34] and Example 2 in [35] where it is referred to as the *stiffness matrix*).

Lemma 2 (See [34]) *Eigenvectors of K_M are the Discrete Sine vectors given by,*

$$v_k = \left[\sin \frac{k\pi}{M+1}, \sin \frac{2k\pi}{M+1}, \ldots, \sin \frac{nk\pi}{M+1} \right]^T, \tag{24}$$

and the eigenvalues are given by,

$$\lambda_k = 2 - 2 \cos \frac{k\pi}{M+1}. \tag{25}$$

Lemma 3 (See [34]) *K_M and D_M are related by*

$$K_M^{-1} = \frac{1}{M+1} D_M. \tag{26}$$

Therefore, if $K_M v = \lambda v$, i.e. v is an eigenvector of K_M and λ is the corresponding eigenvalue, then $D_M v = \frac{1}{\lambda} v$, i.e. v is an eigenvector of D_M with eigenvalue $\frac{1}{\lambda}$. Thus eigenvectors of D_M and hence the sketching matrix P has a succinct representation given by the Discrete Sine vectors and any element can be generated on the fly using (24).

4 Experimental Evaluation

We perform experiments to observe the trade–off between the gain in lifetime of a sensor network and loss in accuracy in estimating aggregate queries over sensor net-

works using in–network aggregation. The results presented in this section are based on the following experimental setup.

We have a synthetically generated field in which sensors are randomly placed and there is a single base–station. The readings of the sensors vary over time. In every round, data from the sensors is gathered or aggregated (depending on the approach) and the queries are evaluated at the base–station.

4.1 Sensor Fields

We consider a 100 m × 100 m field with 200 sensors that are randomly placed and the base station located at (50 m, 300 m). Sensor fields are usually spatially correlated and exhibit periodicity. We generate synthetic data for the sensor fields that exhibit these properties using intuitive techniques. Photographs exhibit strong spatial correlation among their pixel values. So, we consider the pixel values of a picture to be the initial field distribution to account for the spatial correlation of the data. More specifically we consider a 100 × 100 subimage of the second band from the Landsat image of Miami [26] to be the initial field value distribution for our synthetic sensor field. Therefore, the initial value sensed by a sensor is the value of pixel corresponding to the cell it is located. Further, the value sensed by the i^th sensor at time t is varied according to a noisy sine–curve that is given by $X_i(t) = \mu_i + a_i \sin(\frac{2\pi t}{\tau_i} + \phi_i) + \varepsilon_i$. Here $\mu_i = X_i(0)$ is the initial value of the sensor i; a_i, τ_i and ϕ_i are amplitude, frequency and phase for the change in values sensed by sensor i. These are time invariant for all the sensors and are chosen as follows. $a_i = \mu_i/3$. $\tau_i = 60$. ϕ_i is chosen randomly according to a uniform distribution in $(0, \pi)$. ε_i is a random variable that corresponds to white noise with variance $(\mu_i/10)^2$.

4.2 Approaches

We compare results from several approaches for evaluating queries. The approaches can be classified into two groups: data–gathering approach with no aggregation and summary structure based approach using in–network aggregation.

For the data–gathering approach, values recorded by each sensor are forwarded without any aggregation. We use the algorithm due to Chang and Tassiulas [5] for routing of data packets and estimating lifetimes. For the summary data–structure based approaches using in–network aggregation for estimating queries, we use the data–aggregation technique by Kalpakis et al. [23] for generating aggregation trees and evaluating the lifetimes. Given a set of candidate aggregation trees, this approach picks the best trees and determines the number of rounds each tree should be used. We use the A-Randomized-LRS approach suggested by the authors where the candidate tree set consists of 200 aggregation trees from the LRS protocol [27] along with 100 variants (mutated trees) of each.

All sensors are assumed to have equal initial energy of 1 Joule. Both of these methods assume that the sensor locations are fixed and are known at the base–station where route or aggregation tree computation is performed. Further, lifetime is defined to be the number of rounds that sensor network lasts before the first sensor is drained out of energy. While this definition may be too restrictive and it may be possible to

have more rounds before a significant fraction of the sensors die, it is observed that the residual energy left in the sensors by the time the first sensor dies is very low and does not contribute to a significant number of rounds. We also assume that the energy for communication dominates the energy used for processing, and ignore the computation cost.

No Aggregation (NoA): This is the data–gathering approach. The data vector h is constructed at the base–station from the gathered data, which used to answer the queries. This approach gives exact answers to all the queries.

Naive Aggregation (NA): In this approach, each packet has n slots, one for each sensor. Each sensor generates a packet with its own value in the reserved slot and zeros elsewhere. Data aggregation is performed by addition of the corresponding $n \times 1$ vectors. We can construct the exact data vector h at the base station using this. Thus queries will be answered exactly. We assume 2 bytes for each sensor value and 8 bytes for the header. Therefore, the packet size is $2n + 8$ bytes.

We use the following sketches for summary based query processing. We measure the performance for various sketch sizes. With header size of 8 bytes and 2 byte representation for each component of the sketch, the packet size is $2k + 8$ bytes, where k is the size of the sketch.

Haar Wavelets (DWT): We maintain the top–k Haar wavelet coefficients. Normally we do not know which coefficients are the top–k coefficients. Thus they must be estimated using in–network aggregation [18, 21] and the errors in answering queries escalate with the errors in estimating the top–k coefficients. We present results using the true top–k DWT coefficients.

Discrete Fourier Transform (DFT): Here we use the top–k DFT coefficients of the data vector.

Linear Sketching (LS): Here, we use the query based linear sketching approach described in Sect. 3 to compute the sketch

4.3 Quantitative Performance Measures

We use the following quantities to measure the performance of different techniques:

MLT: Mean LifeTime of the sensor network where the mean is computed over 20 different placements of sensors in the field.

MSE: If α and $\widetilde{\alpha}$ are the exact answer and estimated answers to a query, then the squared error for the query is defined as $\|\alpha - \widetilde{\alpha}\|^2$. MSE is the mean of squared errors of results to all the queries over all the rounds and over all the placements.

RLE: If α and $\widetilde{\alpha}$ are the exact answer and estimated answers to a query, then the relative error for the query is defined as $|\alpha - \widetilde{\alpha}|/\alpha$ when $\alpha \neq 0$ and as $|\alpha - \widetilde{\alpha}|$ when $\alpha = 0$. RLE is the mean of relative errors of results to all the queries over all the rounds and over all the placements.

REN: Relative Error in the norm of the data vector is defined as $|\|h\|^2 - \|\widehat{h}\|^2|/\|h\|^2$, where h is the data vector under consideration. In the case of location range queries, h is the data–vector with sensor data from all the sensors and in the case of value range queries, h is the frequency distribution of the values recorded by the different sensors. REN is the relative error in norm of the data vector over all the rounds and over all the placements.

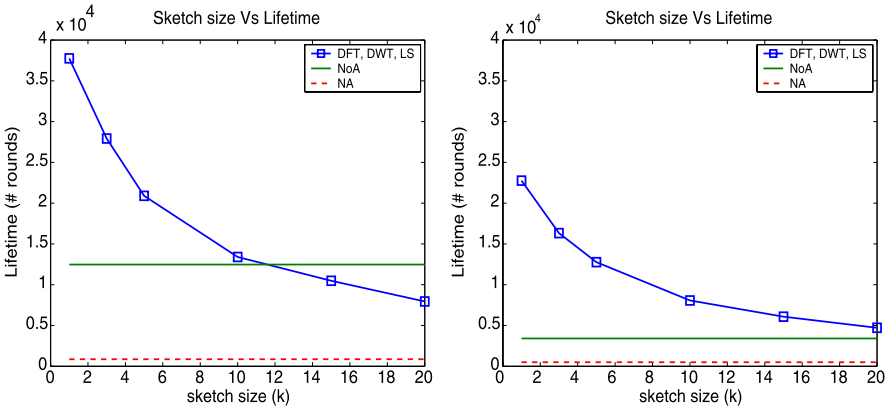

Fig. 4 Sketch size vs. Lifetime: **a** MLT's using different approaches. The sketch size (k) is varied for the sketch based approaches (DWT, DFT and LS). **b** MLT's with the distances doubled

4.4 Experimental Results

Lifetime: We consider 20 different sensor placements and report the mean lifetime (MLT) using the different approaches. The lifetimes of the sensor networks depend only on the packet sizes. For NoA and NA we use packet sizes of 10 bytes and 408 bytes respectively as discussed earlier. For the sketch based approaches we vary the size of the sketch to observe how the lifetimes decrease. For a given sketch size, all the sketch based approaches (DWT, DFT, LS) have the same lifetimes because the packet size is fixed and is $2k + 8$ bytes, where k is the size of the sketch. It is clear that it is not beneficial to use a sketch whose size is greater that the number of sensors because NA gives better lifetimes as well as exact query answers. As we increase the k for the sketch based approaches, at some point, the lifetimes reported would be worse than the NoA approach, in which case we use the NoA approach rather than the NA or sketch based approaches. We refer to the sketch sizes for which, sketch based approaches give better lifetimes as the *sketch–zone*. Within the sketch–zone it is important to see how well sketch based approaches perform with respect to the queries. Beyond this zone, it is beneficial to use the NoA or NA approaches (which ever gives better lifetimes) since they gives exact answers.

The mean lifetimes of the different approaches are plotted in Fig. 4a. The NA aggregation approach gives the worst lifetimes. The MLT for the sketch based approaches decreases as we increase the sketch size; they report higher lifetimes than NoA when the sketch size is less than or equal to 10. Therefore the sketch–zone extends up to $k = 10$.

Note that as the distances between the sensors increase, it becomes more beneficial to perform aggregation rather than just routing the data and the sketch–zone could potentially get extended up to the number of sensors n when the distances between the sensors is large. To illustrate this, we compute the lifetimes when we double all the distances. The MLT's are plotted in Fig. 4b. The sketch zone extends up to more

than $k = 20$. The performance of the sketch based approaches with respect to the query estimates does not depend on the distances. Thus the errors will remain the same as earlier and we can use larger sketches to get better query estimates with the extended sketch–zone.

The performance of the sketch based approaches with respect to the query estimates for different sets of queries is presented next.

Experiment with Fixed Extent Value Range Queries: For each of the 20 placements we consider a set of fixed extent value range queries for 60 rounds. We chose an extent of 35 that is 10% of the total range (354). We report MLT, MSE, RLE and REN for the different approaches in Table 1. Plots of MSE and REN are shown in Fig. 5.

NoA and NA approaches give exact results, therefore, the MSE, REL and REN are always zero. For the remaining aggregate based approaches we present the results with different sketch sizes. For the LS approach, the sketching matrix contains the Fourier vectors in this case, therefore we do not store it explicitly at the sensors. The LS method outperforms DWT in estimating query results. LS gives lesser errors than DFT does, and although we can not claim it does significantly better, we should remember that these errors for DFT and DWT are by using the true top–k coefficients and they would be worse if we were to estimate the top–k coefficients incrementally (using in–network aggregation). We also observe that although the DFT approach is better at capturing the norm of the data, the accuracy of query answers is worse than LS. Further, with a sketch size of 3, LS achieved a MLT of 27, 925 which is 124% improvement over NoA and 6.48% RLE. Therefore it is evident that the LS method improves lifetime of the sensor network while incurring only a small loss in accuracy.

Experiment with Random Extent Value Range Queries: The set up for this experiment is the same as above, except that we use a different query set in each round. During each round we use 100 value range queries whose extent is randomly chosen according to a normal distribution with a mean of 35 (10% of the total range) and a variance of 16. For the LS method we use the sketching matrix used in Experiment 1 where it contains certain Fourier vectors. The results are reported in Fig. 6 and Table 1. Note that although the sketching matrix for LS is designed for a different set of queries, it is still able to estimate answers to the original queries very well. LS gives significantly more accurate query results than DFT or DWT. We observe LS with sketch size of 3 gives 6.93% RLE and significant improvement in MLT. Therefore, it is clear that the LS sketching method works even for queries that were not used to prepare the sketching matrix.

Experiment with Location Range Queries: Finally we perform experiments with location range queries. For each placement over the sensor field we consider the set of all location range queries of a fixed 30 m $\times$ 30 m spatial extent with at least 1 sensor. The average number of sensors in the location queries is 5.12. For the LS method, we use the top–k eigenvectors of the matrix QQ^* as the sketching matrix. The results are summarized in Table 1. The MSE and REN are plotted in Fig. 7. Again, LS achieves better accuracy than DWT and DFT. Using a sketch size of 10, the LS method achieves a MLT of 13,398 that is 7.4% better than NoA, with a RLE of 11.85%. Therefore, from this experiment, we observe that LS achieve better estimates for the location queries than what DWT or DFT achieve and that too using small sketch sizes that gives improved lifetimes.

Table 1 Experimental results

Notation	
p	Packet size in bytes
k	Sketch size
MLT	Mean life time of the sensor network
MSE	Mean squared error over all the queries
RLE	Mean relative error over all the queries
REN	Relative error in the norm of frequency distribution

Experiment 1: Results with fixed size value range queries

$M = 354$; Query set: ValQFixed(35); Mean exact answer: 56.03

Method	p	MLT	MSE	RLE	REN
NoA	10	12,472	0	0	0
NA	408	865	0	0	0

Other Aggregation methods (DWT/DFT/LS)

k	p	MLT	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	54.57	10.68	10.68	97.98	20.03	20.03	96.44	26.59	26.59
3	14	27,925	52.02	6.21	3.57	94.32	11.53	6.48	90.87	22.31	23.14
5	18	20,901	49.68	3.60	2.81	90.91	6.55	4.99	86.20	19.94	22.93
10	28	13,398	44.39	3.39	1.84	83.00	6.16	3.28	76.45	17.72	22.63
15	38	10,487	39.73	3.18	1.21	75.81	5.78	2.12	68.41	16.46	22.38
20	48	7,945	35.48	3.05	1.10	69.18	5.54	1.93	61.45	15.47	22.14

Experiment 2: Results with random size value range queries

$M = 354$; Query Set: ValQRand(35,16); Mean exact answer: 53.80

Method	p	MLT	MSE	RLE	REN
NoA	10	12,472	0	0	0
NA	408	865	0	0	0

Other Aggregation methods (DWT/DFT/LS)

k	p	MLT	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	52.48	10.53	10.53	98.17	21.06	21.06	96.44	26.59	26.59
3	14	27,925	50.16	6.15	3.53	94.84	12.24	6.93	90.87	22.31	22.14
5	18	20,901	48.03	3.57	2.74	91.71	7.01	5.20	86.20	19.94	22.93
10	28	13,398	43.15	3.35	1.80	84.38	6.58	3.44	76.45	17.72	22.63
15	38	10,487	38.82	3.14	1.20	77.64	6.16	2.26	68.41	16.46	22.38
20	48	7,945	34.85	3.01	1.10	71.39	5.89	2.07	61.45	15.47	22.14

Table 1 (*Continued*)

Experiment 3: Results with location range queries of fixed extent

Query Set: LocQFixed(30); Mean exact answer: 1446

Method	p	MLT	MSE	RLE	REN
NoA	10	12,472	0	0	0
NA	408	865	0	0	0

Other Aggregation methods (DWT/DFT/LS)

k	p	MLT	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	1250	294	919	86.23	21.17	67.18	85.90	14.72	85.38
3	14	27,925	826	277	503	55.44	20.30	37.01	57.83	13.95	71.16
5	18	20,901	475	266	329	31.11	19.47	25.33	33.27	13.34	62.17
10	28	13,398	264	244	159	19.27	17.82	11.85	12.76	12.06	49.55
15	38	10,487	235	223	100	17.25	16.31	7.46	10.95	10.98	43.34
20	48	7,945	212	207	75	15.62	15.13	5.67	9.55	10.05	40.32

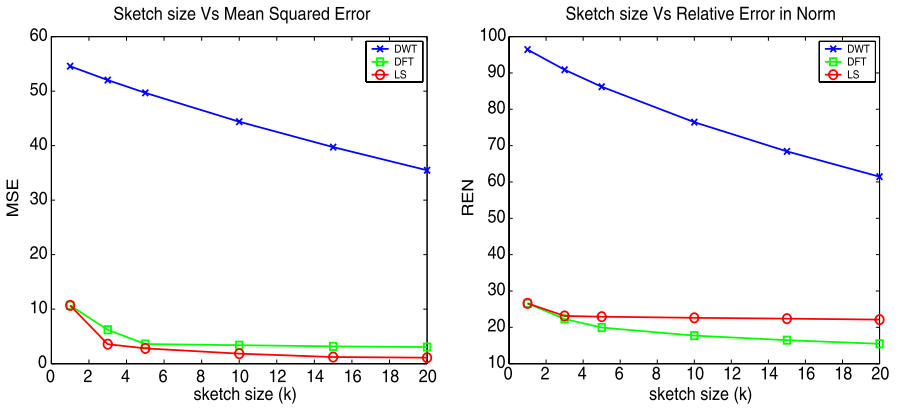

Fig. 5 Sketch size vs. Mean Squared Error in **Experiment 1**. $M = 354$; Fixed extent value range queries with extent = 35; Mean exact answer: 56.03

5 Previous Work

Several studies have been carried out towards performing in–network aggregation in sensor networks [20, 22, 23]. The main focus here is to come up with good data aggregation trees along which in–network aggregation can be carried out. Kalpakis et al. [22] proposes the Maximum Lifetime Data Aggregation algorithm for data aggregation that gives a near optimal data–aggregation schedule consisting of aggregation trees. A computationally efficient approximate scheme for maximizing lifetime by intelligent selection of a data aggregation trees is presented in [23]. Chang et al. [5]

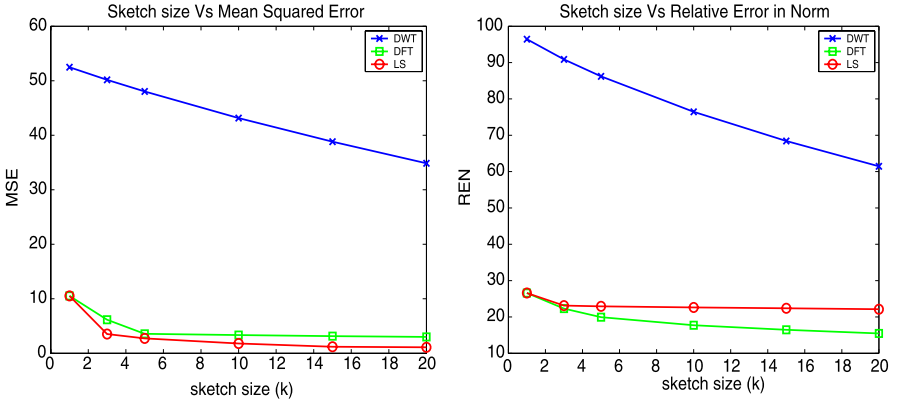

Fig. 6 Sketch size vs. Mean Squared Error in **Experiment 2**. $M = 354$; Random extent value range queries with mean extent = 35; Mean exact answer: 53.80

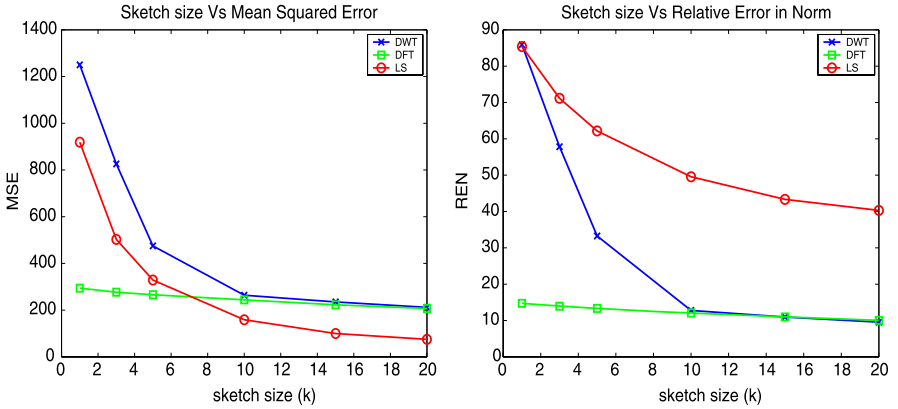

Fig. 7 Sketch size vs. Mean Squared Error in **Experiment 3**. $M = 354$; 30 m × 30 m Location range queries; Mean exact answer: 1446

gives a flow–based approach to gather data (without aggregation) in an optimal manner.

Aggregate query processing in sensor networks has received a lot of attention recently [28, 30, 37]. The common approach to answering aggregate range queries in sensor networks, is to maintain a separate aggregate for each range query with a different where clause. Therefore, this approach is not scalable with respect to the number of range queries. Our approach works by maintaining summary data–structures using in–network aggregation. Several such data structures have been proposed for answering a variety of queries over streaming data (see for e.g. Gibbons et al. [12, 13]). Small space samples [1, 11, 12, 19], histograms [14, 18, 32, 36], and data reduction techniques such as wavelets [16, 18], are popularly used for answering selectivity and aggregate range queries and to compute the size of joins and the number of distinct elements. Summary based query processing has been pro-

posed for sensors as well. Hellerstein et al. [21] argue that monitoring applications demand more sophisticated aggregate query processing over sensor networks. They compute wavelets over the sensor data that can be used to answer approximate aggregate range queries. However, these summary data–structures are not optimized for aggregate range queries. Recently, several sketch based summary data structures have been proposed [2, 6, 7, 16, 31, 36].

We compare and contrast some of these sketching techniques with our linear sketches. These methods are based on randomized sketches where the sketching matrix is a random matrix with certain properties. Therefore the sketches in these techniques are also certain linear sketches. The main differences lie usually in the way queries are estimated. Thaper et al. [36] use random matrices as the sketching matrix, where each entry is chosen from a certain distribution such as Gaussian distribution or uniform distribution over $\{+1, -1\}$. Selectivity queries (i.e. count queries over different ranges) are answered by extracting a histogram from the sketch. Alon et al. [2, 3] use 4–wise independent vectors with elements in $\{+1, -1\}$ as the sketching matrix to estimate the size of the self–join (which is the norm of the data vector). The square of each element in the sketch is a random variable whose expected value is the size of the self–join. Alon et al. [2, 3] use the median–of–means algorithm that provides a *probably approximately correct* (PAC) estimate of the size of the self–join, i.e. the estimate is a (good) approximate answer with some (high) probability. The algorithm works as follows: each element of the sketch is squared and treated as an element of a 2 dimensional array. It computes the means of the columns and outputs the median of these means. Similarly, Gilbert et al. [16] uses 7–wise independent random variables in the sketching matrix. The top–k wavelet coefficients are extracted from the sketch. Queries are estimated by considering the approximate data vector computed from the estimated top–k wavelet coefficients. Such an approach is geared towards preserving the norm of the data vector and does not take queries into account. Our experimental results indicate that preserving the norm is not necessarily beneficial when it comes to answering range queries.

Count-Min (CM) sketches Cormode and Muthukrishnan [7] presented several sketches for estimating point queries (or range queries of extent 1), range queries and inner–product queries over streaming data that are useful in the in–network aggregation setting as well. CM sketches are primarily designed for estimating answers to point queries. They are extended to develop new sketches for answering a variety of other queries including range queries and inner–product queries.

The basic CM sketch maintains $d \times w$ counters where d and w are determined by the desired error and the probability that the estimate is approximately correct. These are labeled by CM$[i, j]$, $1 \leq i \leq d$, $1 \leq j \leq w$. It uses d pairwise independent hash functions $f_1, f_2, \ldots, f_d$ that are mappings from $[1, \ldots, n]$ to $[1, \ldots, w]$. When a data point $p \in \{1, \ldots, n\}$ appears in the stream, the counters CM$[i, f_i(p)]$ for all $1 \leq i \leq d$ are updated (by adding 1 or the number associated with the point in the stream). Therefore, a CM sketch can be viewed as a linear sketch, where the sketching matrix has $d \times w$ column vectors of length n with entries in $\{0, 1\}$ as follows. The vector of the sketching matrix corresponding to CM$[i, j]$ contains 1 as the pth component if $f_i(p) = j$ and 0's everywhere else. Let $q(p)$ be the point query at

point p. Then the exact answer to $q(p)$ is given by the pth component of the data vector. The estimate for $q(p)$ using CM sketches is given by either $\min_j \text{CM}[i, h_i(p)]$ or $\text{median}_j \text{CM}[i, h_i(p)]$, each giving a different probabilistic bound on the error of estimation.

CM sketches are used together with dyadic ranges [17] to answer range queries. Dyadic range are ranges of the form $[i2^j + 1, (i + 1)2^j]$. Each point belongs in exactly $\lg n$ dyadic ranges. The answer to any range query can be computed as a sum of answers to at most $2 \lg n$ dyadic range queries. Therefore if we assume that the data vector contains the counts within each of the dyadic ranges as its elements, any range query over the original data can be translated into a sum–query over such a data vector. We can consider the dyadic ranges to be the representative queries for range queries. However the number of dyadic ranges is $2n$, which is large. Therefore the approach in [7] is to maintain sketches for estimating answers to the dyadic ranges rather than maintaining exact answers to all the (representative) dyadic range queries. This is achieved by maintaining one CM-sketch for each set of dyadic ranges of size $2^j, 0 \leq j \leq \lg n - 1$. If $a_{[l,u]}$ is the exact answer to the range query $[l, u]$ over the (integer) data vector x, and $\widetilde{a_{[l,u]}}$ is the estimate using CM sketches, then (by Theorem 4 in [7]) $a_{[l,u]} \leq \widetilde{a_{[l,u]}}$ and with probability $1 - \delta$, $\widetilde{a_{[l,u]}} \leq a_{[l,u]} + 2\epsilon \lg n ||x||_1$, where $||x||_1 = \sum_{i=1}^{n} |x_i|$ is referred to as the L_1 norm of the data vector. $||x||_1$ for non-negative real vector x is simply $a_{[1,n]}$. The time to estimate an answer to a range query or to make an update is $O(\lg n \lg \frac{1}{\delta})$ and the space used is $O(\frac{\lg n}{\epsilon} \lg \frac{1}{\delta})$.

The time and space complexity of maintaining and answering range queries using linear sketches that we proposed is $O(k)$, where k is the number representative queries or the eigenvectors of the QQ^* matrix that are used in the sketching matrix. Therefore, the time and space complexity of CM-sketch based summary maintenance and range query estimation is comparable to that of query based linear sketches that we propose when the number of representative queries k is of the order of $\lg n$. We derive the error bound for range queries with CM-sketches in terms of SSE over all range queries in order to compare it with query based linear sketching. From the above bound on error, we can arrive at the following bound on the SSE for different range queries over a non-negative real data vector x.

$$\text{SSE} = \sum_{[l,u]} (\widetilde{a_{[l,u]}} - a_{[l,u]})^2 \leq \sum_{[l,u]} (2\epsilon \lg n ||x||_1)^2 \leq O(n^2) 4\epsilon^2 \lg^2 n \|x\|^2.$$

The error bound using query based linear sketching is given by $\text{SSE} \leq \lambda_{n-k}[\|x\|^2 - \|\widehat{x}\|^2] \leq \lambda_{n-k}\|x\|^2$. It is clear that the bound on the SSE in both approaches has the norm of the data vector $\|x\|^2$ as a multiplicative factor. In the case of CM-sketches, there is a factor of $O(n^2)4\epsilon^2 \lg^2 n$ while in our case we have a factor of λ_{n-k}.

Inner–product queries described in [7] can be answered using linear sketches in exactly the same manner as we answer range queries as follows. Let $\widehat{x_1}$ and $\widehat{x_2}$ be the linear sketches of data vectors x_1 and x_2 of the same length respectively (if they are not of equal length we can pad the shorter one with 0's). Then the estimate for $\langle x_1, x_2 \rangle$ is given by $\langle \widehat{x_1}, \widehat{x_2} \rangle$. In fact we treat range queries as inner product queries (see (5)).

While the sketches in [7] and linear sketches presented in this paper share a lot of similarities, there are clear distinctions. Techniques in [7] are randomized in nature

and require pairwise independent hash functions, where as our approaches are deterministic. While CM sketches use a randomized sketching matrix, we use a sketching matrix that is constructed based on the queries. The important distinction that linear sketches that we proposed have is that they take full benefit of the correlation among range queries as well as their rich algebraic structure. To the best of our knowledge this is the first effort in this direction.

Order and Duplicate Insensitive (ODI) Aggregates Aggregation over trees is very sensitive to node and transmission failures because one such failure implies loss of the aggregate over the entire subtree. This motivated development of order and duplicate insensitive (ODI) aggregates that can be computed in a more robust manner using energy efficient multi-path routing schemes. Considine et al. [6] presented ODI sketches for estimating COUNT, SUM and AVG. Nath et al. [31] presented a more general framework called *synopsis diffusion* to compute robust aggregates with ODI synopsis in an energy efficient manner using multi-path routing protocols. They provide ODI synopsis for count queries based on [10] as well as to compute uniform samples over sensor values. Both [6] as well as [31] rely on certain randomized sketches that estimate answers in a PAC manner where larger sketches typically mean lesser errors in estimation with higher probabilities. As a result these approaches also have a trade–off between accuracy and lifetime. While the synopsis in [6] as well as the count synopsis in [31] are ODI aggregates, separate aggregates need to be maintained in order to compute them over different ranges. In general any approach that computes an aggregate over the entire data (from all the sensors) can be used to answer the aggregate over a specified range by disregarding the sensor value if it does not satisfy the where-clause in the aggregate range query. Such techniques do not scale well with the number of queries. While the uniform samples in [31] could potentially be used for answering multiple range–sum queries, they are not optimized for answering them.

6 Conclusion

Techniques based on summary data–structures for approximate query answering fit well with the in–network aggregation paradigm in sensor networks. In this paper we propose summary data–structures called *linear sketches* for answering aggregate range queries over sensor networks. While the accuracy of the query results increases with the size of the sketch, the lifetime of the sensor network decreases. Therefore there is a trade–off between accuracy of the query results and lifetime of the sensor network that we exploit to achieve significant increase in lifetimes while incurring a small loss in accuracy of query results. Linear sketches that we proposed take full benefit of the rich correlation among range queries as well as their algebraic structure. To the best of our knowledge this is the first effort in this direction. We performed several experiments to compare the accuracy of queries as well as lifetime of the sensor networks. The proposed method of linear sketching achieves much better accuracy compared to results using classical techniques such as DFT and DWT and significant gains in lifetimes compared to the naive aggregation and no aggregation schemes, in the case of both value range queries as well as location range queries.

References

1. Acharya, S., Gibbons, P.B., Poosala, V., Ramaswamy, S.: Join synopses for approximate query answering. In: Proceedings ACM SIGMOD International Conference on Management of Data, pp. 275–286, 1999
2. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. In: STOC '96: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, New York, NY, USA, pp. 20–29, 1996
3. Alon, N., Gibbons, P.B., Matias, Y., Szegedy, M.: Tracking join and self-join sizes in limited storage. In: PODS '99: Proceedings of the Eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, New York, NY, USA, pp. 10–20, 1999
4. Babu, S., Widom, J.: Continuous queries over data streams. Technical report, Stanford University (2001)
5. Chang, J.-H., Tassiulas, L.: Energy conserving routing in wireless ad-hoc networks. In: Proceedings of IEEE INFOCOM, vol. 1, pp. 22–31, 2000
6. Considine, J., Li, F., Kollios, G., Byers, J.W.: Approximate aggregation techniques for sensor databases. In: Proceedings of the 20th International Conference on Data Engineering (ICDE), pp. 449–460, 2004
7. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. J. Algorithms **55**(1), 58–75 (2005)
8. Dasgupta, S., Gupta, A.: An elementary proof of the Johnson-Lindenstrauss Lemma. Technical Report TR-99-006, University of California, Berkeley, CA, 1999
9. Davis, P.J.: Circulant Matrices. Wiley, New York (1979)
10. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. J. Comput. Syst. Sci. **31**(2), 182–209 (1985)
11. Gibbons, P.B., Matias, Y.: New sampling-based summary statistics for improving approximate query answers. In: SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, pp. 331–342, 1998
12. Gibbons, P.B., Matias, Y.: Synopsis data structures for massive data sets. In: DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization, vol. A, 1999
13. Gibbons, P.B., Matias, Y., Poosala, V.: Aqua project white paper. Technical report, Information Sciences Research Center, Bell Laboratories (1997)
14. Gibbons, P.B., Matias, Y., Poosala, V.: Fast incremental maintenance of approximate histograms. In: Proc. 23rd Int. Conf. Very Large Data Bases, VLDB, pp. 466–475, 1997
15. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: QuickSAND: quick summary and analysis of network data. Technical report, DIMACS (2001)
16. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: Surfing wavelets on streams: one-pass summaries for approximate aggregate queries. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, San Francisco, CA, USA, pp. 79–88, 2001
17. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: How to summarize the universe: dynamic maintenance of quantiles. In: VLDB, pp. 454–465, 2002
18. Guha, S., Indyk, P., Muthukrishnan, S., Strauss, M.: Histogramming data streams with fast per-item processing. In: ICALP 2002, pp. 681–692, 2002
19. Haas, P.J., Naughton, J.F., Seshadri, S., Stokes, L.: Sampling-based estimation of the number of distinct values of an attribute. In: VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, Zurich, Switzerland, pp. 311–322, 1995
20. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Hawaii International Conference on System Sciences, vol. 8, 2000
21. Hellerstein, J.M., Hong, W., Madden, S., Stanek, K.: Beyond average: toward sophisticated sensing with queries. In: Proceedings of 2nd International Workshop on Information Processing in Sensor Networks (IPSN), pp. 63–79, 2003
22. Kalpakis, K., Dasgupta, K., Namjoshi, P.: Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. Comput. Netw. **42**(6), 697–716 (2003)
23. Kalpakis, K., Dasgupta, K., Namjoshi, P.: Improving the lifetime of sensor networks via intelligent selection of data aggregation trees. In: Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'03), pp. 139–147, 2003

24. Krishnamachari, B., Estrin, D., Wicker, S.B.: The impact of data aggregation in wireless sensor networks. In: Proceedings of International Workshop on Distributed Event-Based Systems, pp. 575–578, 2002
25. Lancaster, P., Tismenetsky, M.: The Theory of Matrices with Applications, 2nd edn. A Series of Monographs and Textbooks, vol. Computer Science and Applied Mathematics. Academic Press, Orlando (1985)
26. LANDSAT Image Gallery: http://landsat.gsfc.nasa.gov/images/gallery.html. Previously at http://www.nnic.noaa.gov/SOCC/gallery.htm
27. Lindsay, S., Raghavendra, C.S., Sivalingam, K.M.: Data gathering in sensor networks using the energy*delay metric. In: IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium, Washington, DC, USA, p. 188, 2001
28. Madden, S., Franklin, M.J.: Fjording the stream: an architecture for queries over streaming sensor data. In: Proceedings of the 18th International Conference on Data Engineering (ICDE'02), p. 555, 2002
29. Madden, S., Szewczyk, R., Franklin, M.J., Culler, D.: Supporting aggregate queries over ad-hoc wireless sensor networks. In: WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, Washington, DC, USA, p. 49, 2002
30. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a tiny aggregation service for ad-hoc sensor networks. In: Proceedings of the ACM Symposium on Operating System Design and Implementation (OSDI), 2002
31. Nath, S., Gibbons, P.B., Seshan, S., Anderson, Z.R.: Synopsis diffusion for robust aggregation in sensor networks. In: SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, New York, NY, USA, pp. 250–262, 2004
32. Poosala, V.: Histogram-based estimation techniques in database systems. Ph.D. thesis, University of Wisconsin, Madison, Wisconsin, USA (1997)
33. Puttagunta, V., Kalpakis, K.: Answering approximate aggregate queries using linear sketches. Technical Report TR-CS-03-29, University of Maryland Baltimore County (2003)
34. Strang, G.: Mathematical methods for engineers I: supplementary material. http://www-math.mit.edu/18085. Four Special Matrices (applmath1_1.pdf), Solutions (applmath1_1sols.pdf), Eigenvalues and Eigenvectors (applmath1_5.pdf), The Stiffness Matrix (applmath2_1.pdf)
35. Strang, G.: Introduction to Applied Mathematics. Wellesley–Cambridge Press, Wellesley (1986)
36. Thaper, N., Guha, S., Indyk, P., Koudas, N.: Dynamic multidimensional histograms. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 428–439, 2002
37. Yao, Y., Gehrke, J.E.: Query processing in sensor networks. In: Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR), 2003