Technical Report

Continuous Probabilistic Sum Queries in Wireless Sensor Networks with Ranges

Nina Hubig[1], Andreas Zuefle[1] Mario A. Nascimento[2], Tobias Emrich[1], Matthias Renz[1], and Hans-Peter Kriegel[1]

[1] Ludwig Maximilians Universität, Muenchen, Germany,
`hubig@ualberta.ca`,{zuefle, renz,kriegel}@dbs.ifi.lmu.de
[2] University of Alberta, Edmonton, Canada
`mn@cs.ualberta.ca`

Abstract. Data measured in wireless sensor networks are inherently imprecise, due to a number of reasons, and aggregate queries are often used to analyze the collected data in order to alleviate the impact of such imprecision. In this paper we will deal with the imprecision in the measured values explicitly by employing a probabilistic approach and we focus on one particular type of aggregate query, namely the SUM query. We consider that sensors in the network may, operate (all collectively at the same time) in two different modes: (1) returning a finite set of discrete values with a probability attached to each value, or (2) a continuous probabilistic density function over a possibly infinite set of possible values. Our foremost concern is to present the first algorithms to efficiently compute the probabilistic SUM according to the possible world semantics, i.e., without any loss of information. Furthermore, we show how this query can be efficiently updated in dynamic environments where sensor values change often and we show techniques to distribute computation over all network nodes. Our experimental results show that processing queries in-network and incrementally as opposed to collecting the measured values from all nodes at the base station and computing the answer centrally, can reduce the total number of messages sent by at least 50%, thus saving energy and extending the network's lifetime, a chief concern regarding wireless sensor networks.

1 Introduction

Recent advances in sensors and wireless communication technologies have enabled the development of small and relatively inexpensive multi-functional wireless sensors. This has lead to the concept of a wireless sensor network (WSN), i.e. a set of spatially distributed autonomous sensors that cooperatively monitor physical or environmental conditions in an area of interest [31], [2]. The communication between the sensors is specified by a network graph where the nodes correspond to the sensor nodes and the edges between two nodes correspond to the ability that the two sensors can communicate with each other wirelessly. The elements of a single sensor node are one (or more) sensor devices, a microprocessor, a small amount of memory, a radio transceiver and a battery [27]. The ability of sensing, processing, and transferring information leverages the idea of sensor networks based on the collaborative effort of a large number of nodes.

The above mentioned features ensure a wide range of scientific and industrial applications for sensor networks, e.g. environmental observation and many other applications in the areas health, military and security [32].

This paper addresses efficient processing of SUM queries in such WSNs. SUM queries are very useful for many applications, for example, if we want to observe the overall amount of traffic in a road network in order to predict potential traffic jams or areas where the risk that an accident will happen due to large volume of incoming traffic is higher. Another example is to estimate the water level of a river by aggregating over all potential inflows observed by a sensor network which is very important to predict or avoid flooding. However, there are a number of challenges sensor network applications have to cope with, including the large number of sensor nodes used in typical sensor networks and the fact that they are prone to failures, as well as limited in power, computational capacities, and memory. Specifically, in this paper we assume that measurements taken by the sensors are imprecise, due to fluctuations in the environment itself or due to hardware limitations [13]. Uncertainty is an inherent problem in sensor networks and may have many reasons, such as:

Device driven uncertainty: Sensors are inherently imprecise, and their precision may be given by the manufacturer. For instance, the precision of tracking devices such as GPS is well documented (e.g. [22]) to be approximately bivariate normal distributed. Manufacturers usually specify the deviation of their devices.

Model driven uncertainty: Imprecision may additionally be imposed if we are not able to make observations by directly measuring the respective value and we are forced to make indirect observations (estimations) based on a given model. For example, in order to measure the total inflow to an underground body of water, parameters like precipitation, evaporation, soil moisture, soil temperature and other environmental variables can be measured, and a geological model can be used to estimate the inflow to a river [8]. The result of such geological models are often given a PDFs due to their high uncertainty ([8]).

Measurement driven uncertainty: Further imprecision arises due to obsoleteness of sensor readings. To preserve energy, new information may only be updated in regular intervals or a sensor node may have died completely. In either case, the current value of the measured parameter has to be estimated stochastically, usually using a model based on empirical data and the last sensor reading. The age of the most recent known sensor reading adds further uncertainty.

In essence, sensor networks are mostly unable to capture exact parameter values and do produce data with some error (or noise). Albeit impossible to derive the exact values from inexact observations, we are often able to make some estimation about the error given in the measurements. Here we assume that we know the probability distribution of the measurements of an observed event. The incorporation of such probabilistic information enables us to achieve more reliable results in data analysis and query processing. Therefore, we need methods that enable us to effectively and efficiently handle as sensor that deliver probability distributions instead of certain values.

Our main goal is to show how to efficiently perform SUM queries on uncertain sensor data while focusing on providing reliable results. We concentrate on the concept of probabilistic query processing, specifically efficient processing of probabilistic SUM queries in WSNs capturing uncertain data. Thereby we assume that the sensor values captured by the sensors are given as probability distributions. The problem to be solved is to answer probabilistic sum queries such as "*Report the probability that the sum of sensor values exceeds a given threshold* τ"

Table 1. Example of probabilistic sensor readings.

Sensor-ID	**Set of (Value,Probability) pairs**
s_1	$\{(12, 0.3), (13, 0.6), (14, 0.1)\}$
s_2	$\{(7, 0.7), (8, 0.3)\}$
s_3	$\{(18, 0.1), (19, 0.3), (20, 0.4), (21, 0.2)\}$

Table 1 shows a sample scenario for the discrete case of the SUM query . Let $S = \{s_1, s_2, s_3\}$ be a WSN with three sensors monitoring three roads leading to a single highway. The task of these sensors is to predict *possible* traffic jams depending on whether the sum of the (probable) values reported by these nodes exceeds a given threshold. For example s_1 may use an inductor coil under the asphalt, s_2 uses visual (camera) data, and s_3 uses GPS data.

Table 2. Example of probabilistic sensor readings.

Sensor-ID	**Set of (Probability,Value Range)**
s_1	$\{(0.5, [0, 0]), (0.3, [0, 10]), (0.2, [10, 20])\}$
s_2	$\{(0.2, [0, 20]), (0.4, [20, 30], 0.3, [30, 40], (0.1, [40, 60])\}$
s_3	$\{(0.5[0, 0]), (0.3, [10, 20]), (0.1, [20, 50])$,(0.1[50,100])$\}$

Another example, related to probabilistic continuous values, is shown in Table 2. Continuity is not possible to measure directly, and thus we use ranges to model it. Analogously to our first example let us assume the sensors s_1, s_2 and s_3 are installed on the tributaries of a river and measure the water level due to rainfall. Each sensor reading contains a set of value ranges, each associated with respective probabilities. A possible query is such a scenario is: "What is the probability that the water level rises above some critical level?".

The main contributions of this paper are the following:

– First, we are able to compute the sum query on uncertain data according to the possible worlds semantics under the assumption of discrete data distributions.
– In the case of continuous distributed data or that an extreme amount of possible results occur we propose a new algorithm on interval based approximations.
– We adapt these algorithms to a WSN environment and propose an incremental in-network algorithm that minimizes communication costs at query processing time.
– Last but not least all these algorithms are evaluated experimentally.

The remainder of this paper is organized as follows. In Section 2 we briefly discuss related work. A formal description of our background theories and the problem definition is given in Section 3. In Section 4 we present two solutions for solving the probabilistic sum query. After that we refine these solutions in adding energy efficiency in Section 6 to make our work valuable for WSNs and add the incremental approach described in Section 6. We experimentally evaluate the efficiency of the proposed approaches in Section 7 and conclude the paper in Section 8.

2 Related Work

2.1 Uncertain Data and Aggregation Query Processing

Uncertainty in databases is a relatively new field that has received a lot of attention in the past few years. The main challenges here are data representation [1, 4, 7, 24] and efficient query processing [18, 30].

There are only few works on uncertain data in the context of aggregation in sensor networks. Approximate count query processing strategies for sensor networks, aiming at producing accurate results with low communication and computation overhead, have been proposed in [9]. In this work the source of uncertainty arises from the network connection between the sensors. Specifically the work assumes node and link failures in the network. Recently, an approach for probabilistic count queries in WSNs has been proposed in [12]. Differently from previous works the source of error is assumed to be the sensor measurement itself. Thus, a sensor returns a reading *along with the probability* that that reading reflects the actual value[3]. Aggregation queries in the context of uncertain and probabilistic databases have been extensively studied recently, e.g., [5, 6, 15, 16, 23, 25]. Murthy et al. proposed in [23] an approach for aggregation in uncertain and probabilistic databases, in particular for the *Trio* system [34]. In order to avoid exponentially-sized results usually produced by probabilistic aggregation (e.g. SUM and AVERAGE) they provide methods for computing bounds of the aggregation value and the expected value. The approach proposed by Ross et.al [25] addresses COUNT query processing in probabilistic databases. Further approaches for the computation of expected aggregates are proposed in [5, 15, 16]. Approximate solutions for query processing on uncertain data based on Monte Carlo sampling is proposed in [14].

2.2 Wireless Sensor Networks

WSNs have been investigated from a variety of angles in recent years, from the development of new energy-efficient devices to the development of new networking protocols. Considering the context of this paper, we focus on work related to query processing within WSNs. If one wants to collect, at the base station, all the actual data from all nodes in a WSN, there is not much alternative to transporting the raw data, hop-by-hop, to the base station. This is energy-wise very expensive. Fortunately, in many applications one is interested in aggregated data, , e.g., what is the average temperature over a given area, what is the maximum (or minimum) pressure exercised on a bridge or what are the top-k more polluted spots in a city. For each one of those types of queries several efficient algorithms have been proposed, and a variety of techniques have been utilized, e.g., based pruning techniques to avoid unnecessary updates [21, 29] or model-based optimization techniques [11, 28]. Nonetheless, most approaches are in some way based on the concept of in-network query processing proposed initially in [20].

Interestingly, not much work has been done regarding the uncertainty aspect of data when processing queries in WSNs. Some efforts have addressed top-k queries, e.g.,

[3] In this paper we extend this assumption to allow to sensor readings to be a set of possible values each associated with a probability

[33] and [35] but not as much for other types of aggregations. In [12] we investigated probabilistic count queries, where the task is to find the probability that a given number of sensors satisfy a query. Note that in that case a sensor either satisfies a query with a given probability or does not with a complementary probability, i.e., there is only one value and one probability involved. It turns out that even though COUNT queries are a special case of the SUM queries we study in this paper, the techniques presented in [12] are not well suited for the latter. In the (probabilistic) SUM queries we consider, each sensor can generate an arbitrary non-binary set of value- or range-probability pairs. In what follows we discuss in detail the probabilistic SUM queries as well as the solutions we propose to process the same in a WSN setting.

3 Problem Definition

We consider a wireless sensor network WSN, consisting of a set $S = \{s_1, s_2, \dots, s_n\}$ of n sensors, each yielding a value at a given point of time t. We assume that the network topology is fixed (i.e. does not change over time) and is a shortest path tree with one single sink node (the tree's root). Instead of a deterministic value, a sensor value of a sensor s_i is a random variable, specified by a probabilistic density function (PDF) denoted as $pdf(s_i)$. The function $pdf(s_i) : V(s_i) \rightarrow (0, 1]$ maps the domain $V(s_i)$ of s_i, i.e. each possible parameter value of s_i, to a non-zero probability value. The measured sensor value s_i, and thus the respective PDF can be discrete (e.g., the number of occurrences of some event, the number of vehicles) or continuous (e.g., temperature, water flow). A sensor s_i producing such probabilistic values, i.e. values given by a PDF, is called a *probabilistic sensor*.

Even though sensor values are typically spatially correlated, our proposed methods do not rely on such assumption. However, it is important to note that we assume that the measurement errors of different sensors are mutually independent, even if the underlying observed events are mutually dependent. Consequently, the sensor value distributions of two different sensors are assumed to be independent.

Furthermore, we note that sensors that measure deterministic values in a traditional sense can be considered as a special case of probabilistic sensors, given by a single discrete value associated with a probability of one. Previous work on uncertain wireless sensor networks [12] assumes that each sensor s measures a single value v associated with a single (existential) probability p. Again, such sensors are a special case of probabilistic sensors as defined above, where the $PDF(s)$ maps the value v to probability p, and maps the value 0 to probability $1 - p$. Note that in the context of SUM queries, if a sensor s reports the value 0, this is equivalent to the case that the sensor s does not exist in the network or does not report any value.

3.1 Probabilistic Query and Possible Worlds Semantic

Based on the above probabilistic sensor model, queries are issued in a probabilistic way by applying the possible worlds semantic model. This model was originally proposed by Kripke [17] for modal logics and is commonly used for representing knowledge with uncertainties. However, there have been different adaptations of the model for probabilistic databases, e.g., [3], [10], [26]. We use the model as proposed in [26], specifically, a possible world w is a set $w = \{s_1^w, \ldots, s_n^w\}$ of instances of sensor values, where each sensor $s \in S$ is assigned to a (certain) value $v \in V(s_i)$. The probability $P(w)$ denotes the probability that the world w is *true*, i.e., that the instances in w coexist in the sensor network. The set of all possible worlds is denoted by $\mathcal{W} = \{w_i, \ldots, w_{|\mathcal{W}|}\}$.

Queries like the SUM query require one to deal with distributions of potential answers. In general, the possible worlds model is the foundation of the concept of probabilistic query processing. From an abstract point of view, the possible worlds are used to generate possible answers to a given query predicate and the probabilities associated with the possible worlds can be used to assign probabilities to the answers.

3.2 Probabilistic Sum Queries

We consider a wireless sensor network *WSN* consisting of a set of probabilistic sensors S and a probability threshold τ. A *probabilistic SUM query* computes the probability, that the random variable corresponding to the sum of all sensor values in the *WSN* is at least τ.

Definition 1. *A* probabilistic sum query *(PSQ) is defined as:*

$$PSQ(WSN, \tau) = \sum_{w \in \mathcal{W}, sum(S^w) \geq \tau} P(w), \tag{1}$$

where $\mathcal{W}$ denotes the set of possible worlds and $sum(S^w)$ denotes the sum of all sensor (value) instances $s \in S$ in world w, as defined by

$$sum(S^w) = \sum_{s_i \in S} s_i^w.$$

In accordance to the above definition, we can answer probabilistic sum queries by materializing all possible worlds with the corresponding probabilities and accumulate the probabilities of all worlds w where the sum of sensor instances in w exceeds τ. However, since the number of possible worlds is exponential in the number of probabilistic sensors, this naive method is not practical. In the following, we will show how to efficiently compute a probabilistic sum query in sensor networks. We consider efficient solutions for the case of discrete, and the case of continuous data distributions. In addition to solutions focusing on reducing the computational overhead of probabilistic sum queries, we show how to reduce the communication overhead, in terms of number of messages, in the sensor network.

4 Probabilistic Sum Queries in Probabilistic Wireless Sensor Networks

We consider two scenarios regarding sensors' operation mode. In the first one, we consider probabilistic sensors with discrete distributions, while the second approach extends the techniques of the first approach in order to cope with continuous distributions that are approximated by *value ranges*. Table 3 summarizes the notation used throughout the paper.

Table 3. Summary of the notation used throughout the paper.

$S = \{s_1, \ldots, s_{\|S\|}\}$	set of all sensors
$S_n \subset S$	subset of size n of sensors in S
$\mathcal{W} = \{w_1, \ldots w_{\|\mathcal{W}\|}\}$	set of possible worlds
$P(w)$	probability of world w
s_i^w	value of s_i in world w
$pdf(v_i^j)$	probability value that $s_i = j$
$ub(\overline{v_i^j}), lb(\overline{v_i^j})$	upper and lower bound of value interval j of sensor s_i
$w = \{s_1^w, \ldots, s_{\|S\|}^w\}$	set of values of all sensors in world w
S_n^w	set of values of sensors in S_n in world w
$sum(S)$	random variable corresponding to the sum of sensors $s \in S$
$GF(S)$	generating function representing $sum(S)$

4.1 Probabilistic Sensors with Discrete Distributions

In the discrete case, the distribution (*pdf*) of the values of a sensor s corresponds to a set of possible values $V_i = \{v_i^1, \ldots v_i^{|V^i|}\}$, each associated with a non-zero probability. As we have seen in the previous section, a probabilistic sum query can be answered in $O(2^{|S|})$ time. The reason for this exponential run-time, is the fact that the number of possible worlds is exponential, and all possible worlds contribute to the result.

In order to reduce such time cost, we need to identify classes of possible worlds that we can treat as equal, i.e. possible worlds having the same sum, and then consider this (smaller) set of equivalent classes of worlds. We propose a technique, that is based on the probabilistic sum of n sensors $S_n \subset S$ iteratively computes the probabilistic sum of subset $S_{n+1} = S_n \cup s \in S \setminus S_n$. Since the set S has no particular order, we assume without loss of generality that $s = s_{n+1}$. In each iteration, worlds of sensors in S_n are grouped into a (smaller) set of equivalent classes. Only equivalent classes are used in the subsequent iteration.

For this approach, we require the following observation.

Lemma 1. *Let $S_n \subset S$ and let $W_k^n = \{w \in \mathcal{W} | \sum_{s \in S_n} s^w = k\}$, then*

$$\forall w_1, w_2 \in W_k^n, j : P(\sum_{s \in S_{n+1}} s = j | s_1^{w_1}, ..., s_n^{w_1}) = P(\sum_{s \in S_{n+1}} s = j | s_1^{w_2}, ..., s_n^{w_2})$$

where $P(\sum_{s \in S_{n+1}} s = j | s_{i_1}^w, ..., s_{i_n}^w)$ denotes the probability that the sum of all sensors in $s \in S_{n+1}$ equals j, assuming that the sensor values of the sensors $s \in S_n$ are given by world w.

Proof. Let $w \in W_k^n$, and let the value of sensor s_{n+1} in world w be s_{n+1}^w. By exploiting that the sum of sensors in S_n is k, we can rewrite $P(\sum_{s \in S_{n+1}} s = j | s_{i_1}^w, ..., s_{i_n}^w)$ as the probability that sensor s_{n+1} has the value $k - j$:

$$P(\sum_{s \in S_{n+1}} s = j | s_1^w, ..., s_n^w) = P(s_{n+1} = j - k | s_1^w, ..., s_n^w)$$

exploiting independence between all sensors, this can be rewritten as

$$P(s_{n+1} = j - k)$$

which is independent of w.

Lemma 1 allows us, to treat all worlds in W_k^n as one single world with probability $\sum_{w \in W_k^n} P(w)$. The reason is that due to independence of sensor errors, sensor s_{n+1} is not affected by the fact of how k is distributed among sensors in S_n. Furthermore, we can use the following Lemma to prune possible worlds from computation.

Lemma 2. *Let $PSQ(WSN, \tau)$ be a probabilistic sum query on a probabilistic WSN having nodes S. For any $S_n \subset S$, it is sufficient to only consider worlds where the sum of sensor values in S_n is less than τ.*

Proof. Equation 10 can be rewritten as

$$PSQ(WSN, \tau) = \sum_{w \in \mathcal{W}, \sum_{s \in S} s^w \geq \tau} P(w) =$$

$$1 - \sum_{w \in \mathcal{W}, \sum_{s \in S} s^w < \tau} P(w)$$

Assume a possible world w where the sum of sensor values in S_n is at least τ. Since the values of sensors in $S \setminus S_n$ in each world w must return non-negative values, we obtain $\tau \leq \sum_{s \in S_n} s^w \leq \sum_{s \in S} s^w$, and thus, w is not considered in the rewritten equation above.

Lemma 2 allows us to discard, for each intermediate set of sensors S_n, any world in which the sum of sensor values in S_n already returns a value greater than k. Then, $PSQ(WSN, \tau)$ can still be computed as one minus the probability of the remaining worlds.

Assume that the values of each sensor node $s_i \in S$ is given by a finite set of values $V_i = \{v_i^1, ..., v_i^{|V_i|}\}$ having a non-zero probability. The probabilities of each value v_i^j are given by probabilistic density function $pdf(v_i^j)$.

To efficiently compute the probabilistic sum of the sensor network $S_n = \{s_1, ..., s_n\}$, consider the following generating function.

$$GF(S_n) = \prod_{i=1}^{n} \sum_{j=1}^{|V_i|} pdf(v_i^j) x^{v_i^j} \tag{2}$$

The following lemma allows us to compute the probabilistic sum of S_n

Lemma 3. *Let $GF(S_n) = \sum_j c_j x^j$ be the expansion of the right-hand-side of Equation 2. Each coefficient c_j corresponds to the probability, that the sum of all sensors S_n equals j.*

Proof. Let $n = 1$, then $GF(S_1) = \sum_{j=1}^{|V_1|} pdf(v_1^j)x^{v_1^j}$. Here, each coefficient c_j equals $pdf(v_j^1)$. Thus, Lemma 3 states that $pdf(v_1^j)$ corresponds to the probability that the probabilistic sum of S_1 equals j. Trivially, this is correct since for $n = 1$ the probabilistic sum of S_1 equals the distribution of s_1.

Now let $n = k + 1$ and assume that Lemma 3 applies for $n = k$.

$$GF(S_{k+1}) = \prod_{i=1}^{k+1} \sum_{j=1}^{|V_i|} pdf(v_i^j)x^{v_i^j} = (\prod_{i=1}^{k} \sum_{j=1}^{|V_i|} pdf(v_i^j)x^{v_i^j}) \times \sum_{j=1}^{|V_{k+1}|} pdf(v_{k+1}^j)x^{v_{k+1}^j} =$$

$$GF(S_k) \times \sum_{j=1}^{|V_{k+1}|} pdf(v_{k+1}^j)x^{v_{k+1}^j} =$$

Expansion of the left side of the product yields

$$\sum_j c_j x^j \times \sum_{j=1}^{|V_{k+1}|} pdf(v_{k+1}^j)x^{v_{k+1}^j}$$

According to the induction hypothesis, each coefficient c_j of $\sum_j c_j x^j$ corresponds to the probability that the sum of sensors S_k equals j. Further expansion yields

$$\sum_{j,m} c_j \times pdf(v_{k+1}^m) \times x^{j+v_{k+1}^m}$$

Due to assumption of independent sensor errors, the event of having a sum of j in sensors S_k is independent of the sensor value of sensor s_{k+1}. Thus, the probability that the sum of sensors in S_k equals j can be multiplied with the probability of s_{k+1} having a value of v_{k+1}^m in order to obtain the probability of the event that the sum of S_k equals j *and* s_{k+1} has value v_{k+1}^m. This multiplication is performed in the above term, for each combination of possible sums of S_k and each value of s_{k+1}, and the respective sum of S_{k+1} is the respective exponent of x.

Now, Corollary 1 allows us to combine worlds having the same sum, into a single world, such that we get

$$\sum_{j,m} c_j \times pdf(v_{k+1}^m)x^{j+v_{k+1}^m} = \sum_{j'} c_{j'} x^{j'}$$

where j' iterates over all possible values of the probabilistic sum of sensors S_{k+1}.

Lemma 3 allows us to translate the semantic of the probability $P(sum(S_n) = j)$ of the random event that the sum of all sensors in S_n equals j, to the syntactical representation $c_j x^j$, i.e., $c_j = P(sum(S_n) = j)$. In this representation, a single monomial

$c_j x^j$ can be interpreted as a equivalent class of possible worlds. The coefficient c_j corresponds to the total probability of this class of worlds, i.e. it corresponds to the sum of probabilities of all possible worlds in this class. The exponent j of the anonymous variable x can be interpreted as the sum of sensor values in S_n in this class of worlds. This representation allows for efficient computation, as we will see next, and also allows for translation back into semantic meaning.

To compute $GF(S_n)$ efficiently, we iteratively compute $GF(S_{k+1})$ from $GF(S_k)$ $(0 < k < n)$ by rewriting Equation 3 as

$$GF(S_{k+1}) = GF(S_k) \times \sum_{j=1}^{|V_{k+1}|} pdf(v_{k+1}^j) x^{v_k^j} \tag{3}$$

and exploiting that the sum of an empty sensor network is always zero, i.e. $GF(S_0) = 1 \times x^0 = 1$. In each iteration, we can now apply Lemmas 1 and 2 to prune terms (corresponding to worlds) that cannot have any influence on the probabilistic sum query with threshold τ.

To illustrate the proposed technique, consider the following example:

Example 1. Assume a set of sensors S in Table 1 measuring the (directed) traffic at three different road segments, e.g., the measured traffic densities in terms of number of vehicles per minute for each sensor. All these road segments lead to one highway, with a maximum capacity of 40 vehicles per minute. If the probability that this capacity is exceeded with a probability of at least 50%, we want to flash traffic warning signals. To predict the probability that this maximum capacity is exceeded, we perform a probabilistic τ sum query on S with $\tau = 40$.

Using Equation 2 for $S_1 = \{s_1\}$ we obtain its generating polynomial:

$$GF(S_1) = 0.3x^{12} + 0.6x^{13} + 0.1x^{14}$$

Lemma 3 states that the (value, probability) pairs of the sum of all sensors in S_1 corresponds to the (coefficient, exponent) pairs of the generating function, which is trivial to confirm in this case[4]. Using $GF(S_1)$ we can compute $GF(S_2)$ using Equation 3:

$$GF(S_2) = (0.3x^{12} + 0.6x^{13} + 0.1x^{14}) \times (0.7x^7, 0.3x^8)$$

Which, after expansion, yields:

$$GF(S_2) = 0.21x^{19} + 0.09x^{20} + 0.42x^{20} + 0.18x^{21} + 0.07x^{21} + 0.03x^{22}$$

Here, each monomial corresponds to exactly one possible world; one world for each possible combination of sensor values in $S_2 = \{s_1, s_2\}$. Due to Lemma 3, the coefficient of the monomial corresponds to the probability of the respective world, and the exponent corresponds to the sum of that world. Unifying monomials having the same exponent yields

$$GF(S_2) = 0.21x^{19} + 0.51x^{20} + 0.25x^{21} + 0.03x^{22}$$

[4] Since s_1 is the only sensor, the distribution of the sum equals to the distribution of s_1, e.g. the sum equals 12 with a probability of 0.3, etc.

Now, each monomial corresponds to a class of worlds with an equivalent sum value. For example, the probability of all worlds of S_2 having a sum of 20 equals 0.51.

Next, Lemma 1 allows us to compute S_3 using Equation 3:

$$GF(S_3) = (0.21x^{19}+0.51x^{20}+0.25x^{21}+0.03x^{22}) \times (0.1x^{18}+0.3x^{19}+0.4x^{20}+0.2x^{21}) =$$

$$GF(S) = 0.021x^{37}+0.114x^{38}+0.262x^{39}+0.324x^{40}+0.211x^{41}+0.062x^{42}+0.006x^{43}$$

Again, Lemma 3 allows us to interpret (coefficient,exponent) pairs as (sum,probability) pairs of the probabilistic sum of S, i.e., the probability that the sum of S_3 equals 37 is 2.1%, etc. Thus, the probability of having a sum of more than 40 can now be aggregated to $0.211 + 0.062 + 0.006 = 0.218 = 21.8\% < 50\%$. That is, the probability of a potential traffic jam, i.e., more than 40 incoming vehicles at the highway would would not warrant the issue of any warning.

If further sensors, such as s_4 are considered, then we only need to consider monomials having an exponent of less than 40 using Corollary 2, since these worlds already satisfy the predicate of $sum > 40$ and due to non-negative sensor values, this predicate cannot change in these worlds.

If we assume that the possible values of each sensor are integers, then due to Lemma 2, we note that at each iterations of the algorithm, the length of each polynomial cannot exceed τ. Then, at each iteration of the algorithm, the current polynomial is multiplied with the generating polynomial of one sensor, which can also be cut using Lemma 2 to a maximum size of τ monomials. In this case, the total runtime of the algorithm equals $O(n \times \tau^2)$. However, if sensor values can be real-valued, then each polynomial may have infinite length, since the number of real values of at most τ is infinite. Still, this approach will work well if the number of possible real values is finite (e.g., rounded to a precision of 0.01). If the possible number of real values is too large, in addition to large polynomials, the probability of being able to combine sets of possible worlds becomes very small, since it becomes unlikely that two different worlds have the same sum.

For the above cases, where the real number of possible sensor values becomes very large, or even infinite for the case of continuous distributions, we propose an approximate solution in the next Section. This approach, while not generally able to return the exact probability that the sum of S_n exceeds τ, will return a probability interval, with a guarantee that the exact probability (according to possible world semantics) must be in this interval. Therefore, we propose a solution which discretizes the possibly large (infinite) space of possible values of a sensor into a (smaller) set of value intervals. We will show how correct bounds of the probabilistic sum can be computed, and how probabilistic sum queries can be answered efficiently and effectively.

4.2 Probabilistic Sensors with Continuous Distributions

In the following, we assume that each sensor measures a large, or infinite number of possible values. For instance, sensors measuring traffic per minute on a large road segment, may yield a non-zero probability for rather large range of number of vehicles. Models for the impact of precipitation on water-level of a river may return continuous distributions, i.e. a continuous distribution over any real value in some interval [8]. In the later case, the distribution may be very complex and not representable in a parametric form. We propose to approximate both cases by a probabilistic histogram, i.e. the pdf of a sensor s_i is partitioned into a (finite and relatively small) set of value intervals $\overline{V_i} = \{\overline{v_i^1}, ..., \overline{v_i^{|V_i|}}\}$. and each interval $\overline{v_i^j}$ is associated with the probability $pdf(\overline{v_i^j}) = cdf(ub(\overline{v_i^j})) - cdf(lb(\overline{v_i^j}))$, where $cdf(a) = \int_{-\infty}^{a} pdf(x)dx$ and $ub(\overline{v_i^j})$ and $lb(\overline{v_i^j})$ correspond to the upper and lower bound of interval $\overline{v_i^j}$, respectively.

In summary, a sensor node is now assumed to be characterized by a set of value intervals, each associated with the probability that the actual value falls into the corresponding interval. While information is absent regarding the probabilistic distribution of the sensor within the interval, we aim at bounding, efficiently, the result of a probabilistic SUM query based on these intervals.

To achieve this, consider the following generating function:

$$GF(S_n) = \prod_{i=1}^{n} \sum_{j=1}^{|\overline{V_i}|} pdf(\overline{v_i^j}) x^{lb(\overline{v_i^j})} y^{ub(\overline{v_i^j}) - lb(\overline{v_i^j})} = \sum_{i,j} c_{i,j} x^i y^j \tag{4}$$

Analogously to Section 4.1, each monomial represents a class of possible worlds, having a total probability of $c_{i,j}$ and a sum of at least i. Additionally, the exponent j of y corresponds to a possible additional value of j, which may or may not exist in the worlds corresponding to $c_{i,j}$, leading to the following lemma:

Lemma 4. *Let $S_n = \{s_1, ..., s_n\}$ be a set of probabilistic sensors. Each coefficient $c_{i,j}$ of the expansion of $GF(S_n)$ corresponds to the probability of all worlds, in which the sum of all sensors S_n must be between i and $i + j$.*

Proof. For $n = 1$, we obtain $S_1 = \sum_{j=1}^{|\overline{V_1}|} pdf(\overline{v_1^j}) x^{lb(\overline{v_1^j})} y^{ub(\overline{v_1^j}) - lb(\overline{v_1^j})}$. By definition of Equation 4, the exponent of x corresponds to the lower bound of the value of s_1, and the sum of exponents of x and y corresponds to the upper bound of the value of s_1. Since for s_1 the probabilistic sum of S_n equals the pdf of s_1, these bounds also apply for the sum of S_n.

Let $n = k + 1$, and assume that Lemma 4 holds for k. Expansion of the first k factors of Equation 4 yields

$$\sum_{i,j} c_{i,j} x^i y^j \cdot \sum_{j=1}^{|\overline{V_{k+1}}|} pdf(\overline{v_{k+1}^j}) x^{lb(\overline{v_{k+1}^j})} y^{ub(\overline{v_{k+1}^j}) - lb(\overline{v_{k+1}^j})}$$

According to the induction hypothesis, each monomial $c_{i,j} x^i y^j$ corresponds to one equivalent class of worlds of S_k in which it holds that the sum is at least i and at most $i + j$. Expansion yields

$$\sum_{i,j,x} c_{i,j} \cdot pdf(\overline{v_{k+1}^x}) x^{i + lb(\overline{v_{k+1}^x})} y^{j + ub(\overline{v_{k+1}^x})}$$

which consists of exactly one monomial for each combination of equivalent classes of worlds in S_k and each possible value interval of s_{k+1}. Since we assume independent sensor errors, the probabilities of pairs of such events can be multiplied, which is performed here. The result is the probability of the equivalent class of worlds of S_{k+1}, where the lower bound sum of S_k is i, and the lower bound value of s_{k+1} is $lb(\overline{v^m_{k+1}})$. The total lower bound of S_{k+1} thus is $i + lb(\overline{v^m_{k+1}})$ which is equal to the exponent of x. Analogously, the exponent of y corresponds to the sum of upper bounds $j + ub(\overline{v^m_{k+1}})$.

Now, Corollary 1 allows to combine resulting worlds having identical lower and upper bound sums, resulting in

$$\sum_{i,j,x} c_{i,j} \cdot pdf(\overline{v^m_{k+1}}) x^{i+lb(\overline{v^m_{k+1}})} y^{j+ub(\overline{v^m_{k+1}})} = \sum_{i',j'} c_{i',j'} x^{i'} y^{j'}$$

where i', j' iterate over all possible combination of possible lower and upper bounds of the sum of S_k.

Lemma 4 allows us to compute a lower bound of the probability that the sum of S_n exceeds τ by adding up the probabilities $c_{i,j}$ of all worlds where the lower bound i (i.e., the exponent of x) exceeds τ, as

$$LB(PSQ(\mathcal{W}, \tau)) = \sum_{i \geq \tau, j} c_{i,j} \tag{5}$$

and the corresponding upper bound can be computed by

$$UB(PSQ(\mathcal{W}, \tau)) = \sum_{i+j \geq \tau} c_{i,j} \tag{6}$$

To allow pruning similar to Lemma 2, we can use the following observation: if the sum of S_n is greater than or equal to τ with a probability of at least (at most) p, then the probability that the sum is less than τ can be at most (must be at least) $1 - p$. This permits to rewrite Equations 5 and 6 as:

$$UB(PSQ(\mathcal{W}, \tau)) = 1 - \sum_{i+j < \tau} c_{i,j} \tag{7}$$

$$LB(PSQ(\mathcal{W}, \tau)) = 1 - \sum_{i < \tau, j} c_{i,j} \tag{8}$$

Now, we note that in this representation, for any monomial $c_{i,j}x^i y^j$ it holds that where $i \leq \tau$. This directly leads to the following corollary

Corollary 1. *Any monomial $c_{i,j}x^i y^j$ where $i > \tau$ can be pruned without loss of information.*

Furthermore, we can combine monomials $c_{i,j}x^i, y^j$, $c_{m,n}x^m, y^n$ where $i = m$, $i+j > \tau$ and $m+n\tau$. The rationale of this is, that for both classes of worlds represented by $c_{i,j}x^i, y^j$ and $c_{m,n}x^m, y^n$ the only difference is a different upper bound sum, since the lower bounds i and n are identical. Since both upper bounds $i + j$ and $m + n$ are greater than τ, we can treat these worlds equivalently, since we do not care how much τ can be exceeded, all we need to know is whether τ can be exceeded at all in the possible worlds. Thus This leads to the following corollary.

Corollary 2. *Any two monomial $c_{i,j}x^i, y^j$, $c_{m,n}x^m, y^n$ where $i = m$, $i + j > \tau$ represent worlds of an equivalent class of worlds. We represent this class of possible worlds by the monomial $c_{i+m,\infty}x^{i+m}y^{\infty}$*

Proof. In Equation 8, the aggregation is independent of j, thus for this equation we may aggregate any monomials having the same i-value. In Equation 7, the aggregation selects all monomials having $i + j \leq \tau$, thus this computation is independent of the above aggregation.

In the iterative computation of $S_n, 1 \leq n \leq |S|$, Corollaries 1 allows to prune, in each iteration, monomials of $GF(S_n)$ which cannot have influence on the sum of S. Furthermore, Corollary 2 allows to combine multiple monomials of $GF(S_n)$ into a single one. Both these pruning criteria reduce the number of monomials, thus decreasing the cost of computing S_{n+1}.

We note that the generating function in Equation 2 in Section 4.1 is a special case of Equation 4, where for all coefficients $c_{i,j}x^iy^j$ is holds that $y = 0$. Thus, using Equation 4 we can handle both cases of discrete and continuous data.

Example 2. Assume a set of sensors S that estimate the expected influx of water to a common river. Let us assume that if this influx exceeds a value of $\tau = 40$ with a probability of at least 10%, a flood warning should be issued. Each sensor s_i computes a continuous probabilistic density function of the water influx, using appropriate water-flow and precipitation models (e.g. [8]). Discretization of these continuous functions yields the value ranges illustrated in the table 2, associated with respective probabilities that the true water influx is within the interval[5].

Applying Equation 4 yields the following polynomial:

$$G(S_1) = 0.5x^0y^0 + 0.3x^0y^{10} + 0.2x^{10}y^{10}$$

By definition, each monomial px^iy^j corresponds to a world having a probability of p and having a sum that is lower bounded by i, and upper bounded by $i + j$. For S_2 we obtain:

$$G(S_2) = G(S_1) \times (0.2x^0y^{20} + 0.4x^{20}y^{10} + 0.3x^{30}y^{10} + 0.1x^{40}y^{20})$$

Here, monomials in $G(S_1)$ correspond to all possible worlds of S_1, while the remaining monomials correspond to worlds of S_2. Due to the observation that the sum of two values between $[lb_1, ub_1]$ and $[lb_2, ub_2]$ must be in the interval $[lb_1 + lb_2, ub_1 + ub_2]$, we now expand the above term to get all possible intervals of the sum of S_1 and S_2:

$$G(S_2) = 0.1x^0y^{20} + 0.2x^{20}y^{10} + 0.15x^{30}y^{10} + 0.05x^{40}y^{20} + 0.06x^0y^{30} + 0.12x^{20}y^{20} +$$

$$0.09x^{30}y^{20} + 0.03x^{40}y^{30} + 0.04x^{10}y^{30} + 0.08x^{30}y^{20} + 0.06x^{40}y^{20} + 0.02x^{50}y^{30}$$

Each coefficient p of each monomial px^ix^j and is the product of the probabilities of two sensor values. Due to the assumption of independent sensor errors, this product is

[5] For illustration purpose we use a very coarse discretization. In practice, a much larger set of value ranges per sensor could be used.

the probability of observing both values. Exploiting Lemma 1, we can combine worlds of S_2 having the same lower and upper bounds

$$G(S_2) = 0.1x^0y^{20} + 0.06x^0y^{30} + 0.04x^{10}y^{30} + 0.2x^{20}y^{10} + 0.12x^{20}y^{20}+$$

$$0.15x^{30}y^{10} + 0.17x^{30}y^{20} + 0.11x^{40}y^{20} + 0.03x^{40}y^{30} + 0.02^{50}y^{30}$$

Now, recall that ultimately, we want to compute the probability that $P(sum(S)) > 40$, which equals $1 - P(sum(S) \leq 40)$. To compute the latter probability, we can now prune any monomial having an x-exponent of $i > 40$, since for the equivalent class of worlds represented by this monomial, we can already conclude that the sum cannot be less than 40 –and due to non-negative sensor values, considering further sensors cannot change this conclusion. We obtain:

$$G(S_2) = 0.1x^0y^{20} + 0.06x^0y^{30} + 0.04x^{10}y^{30} + 0.2x^{20}y^{10}+$$

$$0.12x^{20}y^{20} + 0.15x^{30}y^{10} + 0.17x^{30}y^{20}$$

which will then be used in the computation of $G(S_3)$. Furthermore, we can combine monomials $c_{i,j}x^i, y^j$, $c_{m,n}x^m, y^n$ where $i = m$, $i + j > \tau$ and $m + n\tau$ as discussed above, yielding:

$$G(S_2) = 0.1x^0y^{20} + 0.06x^0y^{30} + 0.04x^{10}y^{30} + 0.2x^{20}y^{10}+$$

$$0.12x^{20}y^{20} + 0.15x^{30}y^{10} + 0.17x^{30}y^{\infty}$$

Due to space limitations, let us now assume that there is only two sensors, i.e. that $S = S_2$. Now, in order to lower (upper) bound the probability that the sum of S is at most 40, we simply sum up all worlds where the sum must be (can be) lower than 40. Clearly, a world must have a sum of at most 40, if its corresponding upper bound is at most 40, i.e. if for the corresponding monomial px^iy^j it holds that $i + j$ is at most 40. Thus we obtain

$$\sum_{i+j<\tau=40} c_{i,j} = 0.1 + 0.06 + 0.04 + 0.2 + 0.12 + 0.15 = 0.67$$

A world can have a sum of 40 or less, if its corresponding lower bound is 40 or less, independent of its upper bound, i.e. if for the corresponding monomial px^iy^j it holds that $i \leq 40$

$$\sum_{i<\tau=40} c_{i,j} = 0.1 + 0.06 + 0.04 + 0.2 + 0.12 + 0.15 + 0.17 = 0.84$$

Note, that this sum equals the sum of coefficients of all remaining monomials, since any monomial which cannot have a sum of 40 or less has already been pruned.

Using Equations 7 and 8, we can now bound the probability that the sum of S exceeds τ:

$$UB(PSQ(\mathcal{W}, \tau)) = 1 - \sum_{i+j<\tau} c_{i,j} = 1 - 0.67 = 0.33$$

and

$$LB(PSQ(\mathcal{W},\tau)) = 1 - \sum_{i<\tau} c_{i,j} = 1 - 0.84 = 0.16$$

Thus we can conclude that the event that the water influx exceeds 40 must have a probability of at least 16%, and in particular must be greater than 10%, thus we will broadcast a flood warning. Note that we were able to answer this query despite a very coarse approximation of the sensor pdfs. In the general case, the probability threshold p may be between the lower and the upper bound of $PSQ(\mathcal{W},\tau)$. In this case we cannot say for certain whether the probability is greater than p or not. However, we may re-initiate the whole query, asking each sensor for a pdf having a more refined granularity. This will reduce the uncertainty of the query result but will come at an additional network traffic, since for the new query, sensors will be forced to send a much more refined pdf, which corresponds to more information, which corresponds to more data.

This filter refinement approach can be iterated until a definite answer can be given, or until the resulting approximation is good enough. In the example, an approximation that the probability of a flood must be between 9% and 11% may be good enough to decide whether to broadcast a warning or not.

5 Energy Efficient Computation of Probabilistic Sum Queries

As mentioned in Section 1, it is crucial for applications on WSNs to reduce energy cost, and this is mainly achieved through reducing CPU cost and communication. In the previous section we showed how to reduce the CPU cost for computing the count distribution. In this section, we focus on reducing the communication costs. For that purpose, we must consider the typical underlying characteristics of WSNs such as network topology, routing and scheduling. We will propose two algorithms which solve the problem of answering continuous count queries in a WSN. Thus, we now take the local distribution of data into consideration. We assume that the nodes in S are connected together via a logical tree where the sink node (or base-station) is the tree's root. The choice of the tree's topology does matter, but is outside the scope of this paper. For the sake of simplicity, we assume it to be a hop-based shortest path tree commonly used in other works, e.g., [20].

In the previous Section 4, we assumed that for each sensor $s_i \in S$, the distribution $pdf(s_i)$ is readily available at the sink node. This requires each $pdf(s_i)$ to be iteratively propagated along the branch of the logical tree to the root. However, we can benefit from computing intermediate results at intermediate nodes, in order to send these condensed results to their parent node. In this Section, we will show how an intermediate node can compute the partial probabilistic sum of all nodes of its subtree, based on the partial probabilistic sum of its individual child nodes. On the one hand, we can decrease the number of messages sent. On the other hand, intermediate results could be used to query subtrees or apply early stopping conditions if a subtree satisfies the query [13].

Without loss of generality, we assume that sensors send intervals as proposed in Section 4. The case of sending discrete values can be seen as a special case where the lower bound of each interval equals its upper bound - and thus the exponents of all y variables are zero.

Lemma 5. *Let s_0 be a sensor node having (direct) children $s_1, ..., s_n$. Let S_c denote the set of sensors in the subtree rooted at s_c, including s_c itself. Then*

$$GF(S_0) = GF(s_0) \cdot \prod_{c=1}^{n} GF(S_c) \tag{9}$$

Proof. Follows directly from the associative property of polynomial multiplications.

Now, at each intermediate node, pruning of monomials as proposed by Corollaries 1 and 2 can be performed, to reduce the amount of information that s is required to send to its parent node.

Our **in-network algorithm** now works as follows: Each leaf node s_{leaf} of the logical connection tree sends its sensor readings[6] to its parent node after using Corollary 1 to remove worlds of s_{leaf} which are not required to compute $PSQ(WSN, \tau)$. Intermediate nodes s_{dir}, upon receiving data from their children, compute the probabilistic sum of their respective subtree using Lemma 9 using polynomial multipliction. To facilitate the expansion of two possibly large polynomials, we propose to use FFT (Fast-Fourier-Transformation). It is well known that the multiplication of two polynomials of degree O(n) can be done in $O(n \log n)$ time using FFT. Furthermore, the approach by [19], using divide-and-conquer and Fast Fourier Transformation allows us to further reduce this time complexity to $O(n \log^2 n)$. Unless s_{dir} is the root, s_{dir} uses Corollaries 1 and 2 to reduce the size of its polynomial, and sends the resulting polynomial to its parent node. If s_{dir} is the root, then $PSQ(S, \tau)$ is bounded using Equations 7 and Equation 8. This bound is returned and the algorithm terminates.

For sensor networks where in-network computation is not viable, and for comparison, we furthermore propose a **central algorithm**, which simply propagates all sensor readings to the root node, where all the computation is performed using FFT.

6 Incremental Sum Queries

The solutions proposed in Section 4 work well for the snapshot case, that is the case where the state of the wireless sensor network only matters at a single point of time t. In most practical applications however, we are interested in a monitoring of the sum value of the network. Thus, we want to have the current sum at any point of time.

Definition 2 (Continuous Probabilistic Sum Query). *Let WSN be a wireless sensor network consisting of a set of probabilistic sensors S and τ be a real value. Let $T = \{t_1, t_2, ...\}$ be a discrete set of points of time, so called* rounds *and let pdf$_{t \in T}(s)$ denote the probability distribution of sensor s at time t.*

A continuous probabilistic sum query *(cPSQ) returns the probabilistic sum of WSN at each point of time $t \in T$.*

$$cPSQ(WSN, \tau, t) = \sum_{w \in \mathcal{W}_t, sum(S^w) \geq \tau} P(w), \tag{10}$$

[6] Sensor readings can be (value, probability) pairs (c.f. Section 4.1), (interval, probability) pairs (c.f. Section 4.2) or a mix of both.

where $\mathcal{W}_t$ denotes the set of possible worlds at time t.

Naively, we can apply the solutions of Section 4 for each individual point of time. This works well if the probability distributions $pdf(s)$ of all sensors s change frequently from one round t to the next round $t+1$. However, in practice, only a small set of sensors will update their pdf in one round, since changes may not happen too frequently, and many sensors may be sleeping to preserve energy, and will only send a new pdf once in a while.

Clearly, at the initial point of time $t = 0$ we must use the techniques of Section 4. In the following, we propose how $cPSQ(WSN, \tau, t+1)$ can be computed incrementally, based on the result of $cPSQ(WSN, \tau, t)$.

Therefore, we propose to adapt the two-phase approach for incremental algorithms proposed in [12]. For each sensor s that changes its pdf at time $t+1$ we proceed as follows:

- (Phase 1) The effect of $pdf_t(s)$ is removed from $sum(S)$ to derive $sum(S \setminus \{s\})$
- (Phase 2) The effect of $pdf_{t+1}(s)$ is then incorporated into $sum(S \setminus \{s\})$

For phase 1, recall that the probabilistic sum of $S_n = \{s_1, ..., s_n\}$ can be computed using the product of generating functions of each sensor:

$$GF(S_n) = GF(S_{n-1}) \cdot GF(\{s_n\}) = \prod_{1 \le i \le n} GF(\{s_i\})$$

Due to the associativity of polynomial multiplication, this can be rewritten to

$$GF(S_n) = GF(\{s_k\}) \cdot \prod_{1 \le i \ne k \le n} GF(\{s_i\}),$$

where $s_k \in S$ is the sensor whose effect we want to remove from $GF(S)$. To remove the effect of one sensor $s_k \in S$, we can now perform a polynomial division of $GF(s_k)$.

$$GF(S \setminus \{s_k\}) = GF(S) : GF(\{s_k\})$$

However, this approach of polynomial division is **not** viable, since in the polynomial $GF(S)$, many monomials $c_{i,j}x^i y^j$ have already been pruned (cf. Section) and are no longer available, e.g. polynomials having $i > \tau$. Yet, these monomials of $GF(S)$ having $i > \tau$, may have an influence on monomials of $GF(S \setminus \{s_k\})$ having $i < \tau$. Semantically, this means that we no longer have any access to any worlds of S, where the sum certainly exceeds k. However, in some of these worlds, the sum of sensors $S \setminus \{s_k\}$ may still be less than τ. These worlds (and their corresponding probabilities) would be missing in the interpretation of $GF(S \setminus \{s_k\})$, thus the result may be incorrect.

Yet, we can compute $GF(S \setminus \{s_k\})$ correctly, in a different way. For simplicity, now assume that each sensor measures discrete values as proposed in Section 4.1, i.e. that for each monomial $c_{i,j}x^i y^j$, it holds that $y = 0$. The following technique can be extended to $y > 0$ in a straightforward way.

Rather than performing polynomial division, we first consider each monomial $c_i x^i$ in $GF(S)$, and invert the step of combining coefficients having the same exponent, which has been performed during the expansion of $GF(S)$. We recall, that c_i is the probability of all worlds of S having a total sum of i. This is equivalent to the probability of all worlds in $GF(S \setminus \{s_k\})$ having a sum of zero, multiplied with the probability of s_k having a value of i, plus the probability of all worlds in $GF(S \setminus \{s_k\})$ having a sum of one, multiplied with the probability of s_k having a value of $i-1$, and so on. Formally:

$$P(sum(S)=i) = \sum_{j=0,\dots,i} P(sum(S \setminus \{s_k\})=j) \cdot P(pdf(v_k^{i-j}))$$

Writing this in terms of $P(sum(S \setminus \{s_k\})=i)$ yields

$$P(sum(S\setminus\{s_k\})=i) = \frac{P(sum(S)=i) - \sum_{j=0}^{i-1} P(sum(S \setminus \{s_k\})=j) \cdot P(pdf(v_k^{i-j}))}{pdf(v_k^0)} \tag{11}$$

In this formula, we note that in order to compute the value $P(sum(S \setminus \{s_k\})=i)$, we only require the probability $P(sum(S)=i)$, which is not affected by the pruning of monomials, since if the monomial $c_i x^i$ had been pruned, then we had $i > \tau$, and then we would not need to compute $P(sum(S \setminus \{s_k\})=i)$ in the first place. Furthermore, we need all $P(sum(S \setminus \{s_k\})=j)$, for $j < i$. For $i=0$ this is the empty set, thus no more information is required. Then, for $i=a$, we inductively obtain the required values from previous computations of $i < a$.

Example 3. Assume a set of sensors $S=\{s_1, s_2\}$. Assume that sensor s_1 has a value of two with a probability of 0.5, a value of one with a probability of 0.4, and a value of zero with a probability of 0.1. Further assume that s_2 has a value of one with a probability of 0.8 and a probability of zero with a probability of 0.2, i.e.

$$GF(\{s_1\}) = (0.5x^2 + 0.4x + 0.1), GF(\{s_2\}) = 0.8x + 0.2$$

According to Section 4.1 we obtain the following generating function:

$$GF(S) = (0.5x^2 + 0.4x + 0.1) \cdot (0.8x + 0.2) = 0.4x^3 + 0.42x^2 + 0.16x + 0.02$$

Now, assume that s_2 changes its value. And we want to remove the effect of s_2 on $G(S)$. Clearly, this can be performed by dividing of $GF(S)$ by the generating polynomial of s_2. And the correct result will be the generating polynomial of s_1. However, we now assume that $\tau = 1$, so that we have already pruned monomials $c_i x^i$ where $i > 1$ and get

$$GF(S) = 0.16x + 0.02$$

Due to the missing monomials, polynomial division can no longer be applied. However, using Equation 11, we get:

$$P(sum(\{s_1\})=0) =$$

$$\frac{P(sum(S)=0) - \sum_{j=0}^{-1} P(sum(\{s_1\})=j) \cdot P(pdf(v_2^{0-j}))}{pdf(v_2^0)} = \frac{0.02}{0.2} = 0.1$$

based on this we can compute

$$P(sum(\{s_1\}) = 1) =$$

$$\frac{P(sum(S) = 1) - \sum_{j=0}^{0} P(sum(\{s_1\}) = j) \cdot P(pdf(v_2^{1-j}))}{pdf(v_2^1)} = \frac{0.16 - (0.1 \cdot 0.8)}{0.2} = 0.4$$

now, we have completely reconstructed $GF(S \backslash \{s_2\}) = 0.4x + 0.1$, which, as expected, corresponds to $GF(\{s_1\})$. Note that we cannot reconstruct any value $P(sum(\{s_1\}) = i)$ where $i > 1$, since the values $P(sum(S) = i)$ required for the computation have been pruned. Gladly, we do are not required to compute these probabilities, since we could prune them anyways, since $i > \tau$.

For phase 2, we can simply use polynomial multiplication to incorporate the new value of sensor s_k

$$GF(S') = GF(S \setminus \{s_k\}) \cdot GF(s'_k),$$

where s'_k corresponds to the sensor s_k having its new pdf_{t+1}, and S' corresponds to the new sensor network where the value of s_k has been updated.

7 Performance Evaluation

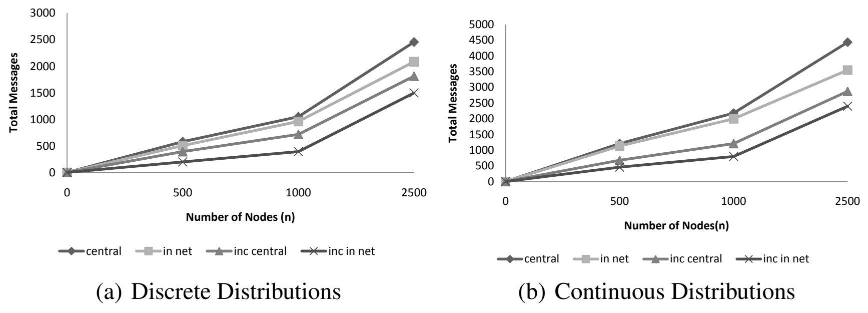
For the experimental evaluations we used the following setup: We used a simulation of a wireless sensor network containing between 100 and 2500 sensors. The locations of the sensors were randomly chosen within a $100m \times 100m$ area and each sensor node was assumed to have a fixed wireless radio range of 30m. All generated sensor instances of the WSNs used a hop-wise shortest-path tree as the routing topology. We assume in all experiments that messages are delivered using a multi-hop setup. Since the query is only sent once from the root to all child nodes and will be amortized over time, we only measure nodes-to-root messages.

The uncertain sensor values are simulated as follows: The values of probabilistic sensors with discrete distributions consists of a probability distribution with three possible values randomly selected from the interval [0,50]. The corresponding probabilities assigned to these values are randomly selected such that they sum up to 1. A similar value generator has been used for the probabilistic sensors with continuous distributions whereas intervals instead of single values are randomly selected within the interval [0,50].

The experimental results are based on an average of 10 simulation runs whereas each run consists of a series of 100 measurements per sensor. Thereby, the result of the probabilistic sum query is updated after each measurement. The performance of the methods proposed in this paper are evaluated by taking the average of the number of messages that have been sent for each update of the query result. Here we compare four different approaches: ***central*** sends all sensor signals to the central node and performs the query centrally at that node, ***in net*** performs in-network query processing as described in Sec. 5, ***inc central*** performs incremental query processing as described in Sec. 6 at the central node, and ***inc in net*** combines the in-network query processing approach with the incremental query processing approach.

Table 4. Parameter Values Used in the Performance Evaluation (Default Values Printed in Bold Type)

Parameter	Values
n (Number of Sensors)	100, 500, **1000**, 2500
δ (Probability of Changing Values)	0%, **50%**, 75%, 100%



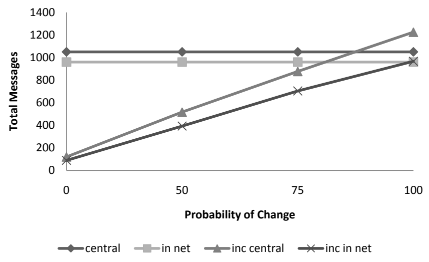
(a) Discrete Distributions (b) Continuous Distributions

Fig. 1. Scalability experiments.

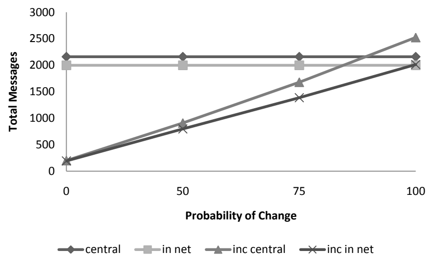
In the experiments we vary the following parameters: the number of sensors within the network (n) and the percentage of changing measurements of a sensor, i.e. where the new value (value distribution) differs from the previous one (δ).If the size of the information that has to be sent from one node to another node exceeds m, then the information has to be distributed to multiple messages. The setting of these parameters are summarized in Table 4, where default values are marked bold. In every experiment all parameters but the one in focus are fixed to their default values.

7.1 Scalability Experiments

In the first experiment, we evaluate the performance of the four approaches by varying the size of the network. The results are shown in Figure 1 for sensor values with discrete and continuous distributions. As expected, the number of messages increases with the number n sensors in the network. It is more interesting to see that the communication cost grow super linear to the number of sensors for all four approaches. The reason is, that the number of sensors not only influences the number of nodes that have to send messages, but also the number of messages that have to be transmitted through the network. Both approaches, the incremental approach and the in-network approach, each improves significantly the communication cost compared to the simple centralized approach. However, the incremental in-network query processing approach is the overall best solution saving up to 50% of the communication cost compared to the simple centralized approach.

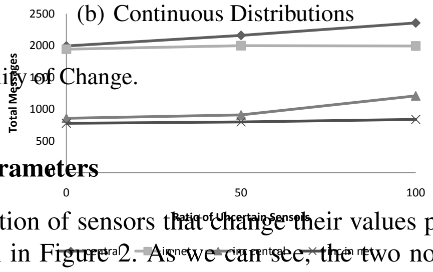


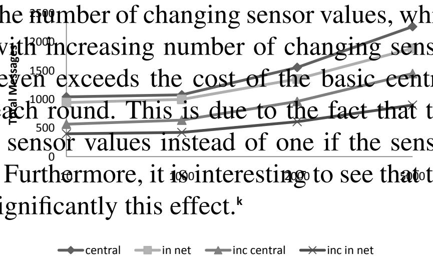
(a) Discrete Distributions



(b) Continuous Distributions

Fig. 2. Probability of Change.





7.2 Experiments on Various Dataset Parameters

In the next experiment we evaluate the fraction of sensors that change their values per measurement round. The results are shown in Figure 2. As we can see, the two non-incremental approaches are not affected by the number of changing sensor values, while the two incremental approaches degrades with increasing number of changing sensor values. The incremental central approach even exceeds the cost of the basic central approach if all sensors values changes at each round. This is due to the fact that the incremental approach requires to send two sensor values instead of one if the sensor value changes, the old one and the new one. Furthermore, it is interesting to see that the in-network processing approach mitigates significantly this effect.

8 Conclusions

Summarizing this work we introduced the first solution to efficiently answer probabilistic count queries on discrete uncertain data. The naive approach to compute the result to such a query would have exponential runtime. We showed how to utilize the concept of generating functions to build a polynomial algorithm. Afterwards we extended such that also continuous distributions can be handled. We adapted our solutions to wireless sensor networks which have the additional constraint of limited energy. For this purpose we proposed an in-network algorithm which distributes the computation over all network nodes and thus results in much lower cpu and communication cost. As future work, we plan to develop an approach which is able to incrementally update the result of a probabilistic sum query when a small subset of the sensors changes, without recomputing the result from scratch.

References

1. Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: A system for data, uncertainty, and lineage. In: Proc. VLDB (2006)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38(4), 393–422 (2002)

3. Antova, L., Koch, C., Olteanu, D.: 10 worlds and beyond: efficient representation and processing of incomplete information. The VLDB Journal 18, 1021–1040 (2009)
4. Beskales, G., Soliman, M.A., IIyas, I.F.: Efficient search for the top-k probable nearest neighbors in uncertain databases. Proc. VLDB Endow. 1(1), 326–339 (2008)
5. Burdick, D., Deshpande, P.M., Jayram, T.S., Ramakrishnan, R., Vaithyanathan, S.: Olap over uncertain and imprecise data. VLDB J. 16(1), 123–144 (2007)
6. Chen, A.L.P., Chiu, J.S., Tseng, F.S.C.: Evaluating aggregate operations over imprecise data. IEEE Trans. Knowl. Data Eng. 8(2), 273–284 (1996)
7. Cheng, R., Chen, J., Mokbel, M., Chow, C.: Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In: Proc. ICDE (2008)
8. Clark, M.P., Slater, A.G.: Probabilistic quantitative precipitation estimation in complex terrain. Journal of Hydrometeorology (2006)
9. 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 2004, 30 March - 2 April 2004, Boston, MA, USA. pp. 449–460 (2004)
10. Dalvi, N., Suciu, D.: Efficient query evaluation on probabilistic databases. The VLDB Journal 16, 523–544 (October 2007)
11. Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J.M., Hong, W.: Model-driven data acquisition in sensor networks. In: Proc. of VLDB. pp. 588–599 (2004)
12. Follmann, A., A.Nascimento, M., Züfle, A., Renz, M., Kriegel, H.P.: Continuous probabilistic count queries in wireless sensor networks. 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings pp. 279–296 (2011)
13. Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking queries on uncertain data: a probabilistic threshold approach. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 673–686. SIGMOD '08 (2008)
14. Jampani, R., Xu, F., Wu, M., Perez, L.L., Jermaine, C.M., Haas, P.J.: Mcdb: a monte carlo approach to managing uncertain data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008. pp. 687–700 (2008)
15. Jayram, T.S., Kale, S., Vee, E.: Efficient aggregation algorithms for probabilistic data. In: SODA. pp. 346–355. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2007)
16. Jayram, T.S., McGregor, A., Muthukrishnan, S., Vee, E.: Estimating statistical aggregates on probabilistic data streams. ACM Trans. Database Syst. 33(4) (2008)
17. Kripke, S.A.: Semantical analysis of modal logic i normal modal propositional calculi. Mathematical Logic Quaterly 9, 67–96 (1963)
18. Li, J., Saha, B., Deshpande, A.: A unified approach to ranking in probabilistic databases. PVLDB 2(1), 502–513 (2009)
19. Li, J., Saha, B., Deshpande, A.: A unified approach to ranking in probabilistic databases. VLDB J. 20(2), 249–275 (2011)
20. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tag: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Operating Systems Review 36, 131–146 (December 2002)
21. Malhotra, B., Nascimento, M.A., Nikolaidis, I.: Exact top-k queries in wireless sensor networks. IEEE Transactions on Knowledge and Data Engineering 99(PrePrints) (2010)
22. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: Big data: The next frontier for innovation, competition, and productivity (2011)
23. Murthy, R., Ikeda, R., Widom, J.: Making aggregation work in uncertain and probabilistic databases. IEEE Trans. Knowl. Data Eng. 23(8), 1261–1273 (2011)
24. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: Proc. VLDB. pp. 15–26 (2007)

25. Ross, R., Subrahmanian, V.S., Grant, J.: Aggregate operators in probabilistic databases. J. ACM 52, 54–101 (January 2005)
26. Sarma, A., Benjelloun, O., Halevy, A., Widom, J.: Working models for uncertain data. In: Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on. pp. 7–7 (2006)
27. Schurgers, C., Tsiatsis, V., Ganeriwal, S., Srivastava, M.: Optimizing sensor networks in the energy-latency-density design space. IEEE Transactions on Mobile Computing 1, 70–80 (2002)
28. Silberstein, A., Braynard, R., Ellis, C., Munagala, K., Yang, J.: A sampling-based approach to optimizing top-k queries in sensor networks. Proc. of the 22nd IEEE Conf. on Data Engineering (ICDE'06) pp. 68–78 (2006)
29. Silberstein, A., Munagala, K., Yang, J.: Energy-efficient monitoring of extreme values in sensor networks. Proc. of the ACM Conf. on Management of Data (SIGMOD'06) pp. 169–180 (2006)
30. Soliman, M.A., Ilyas, I.F., Chang, K.C.C.: Top-k query processing in uncertain databases. In: Proc. ICDE. pp. 896–905 (2007)
31. Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., Mainwaring, A., Estrin, D.: Habitat monitoring with sensor networks. Commun. ACM 47, 34–40 (June 2004)
32. Waltenegus Dargie, Poellabauer, C.: Fundamentals of Wireless Sensor Networks: Theory and Practice. John Wiley & Sons (2010)
33. Wang, S., Wang, G., Gao, X., Tan, Z.: Frequent items computation over uncertain wireless sensor network. Hybrid Intelligent Systems, International Conference on 2, 223–228 (2009)
34. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: CIDR. pp. 262–276 (2005)
35. Ye, M., Liu, X., Lee, W.C., Lee, D.L.: Probabilistic top-k query processing in distributed sensor networkss. In: Proc. of the ICDE. pp. 585–588 (2010)