Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 403267, 13 pages
http://dx.doi.org/10.1155/2015/403267

Research Article

A Multimodel Based Range Query Processing Algorithm for Information Collection in CPS

Guilin Li,[1] Xing Gao,[1] Longjiang Guo,[2] JunCong Lin,[1] Ying Gao,[1] and Minghong Liao[1]

[1]*Department of Software, Xiamen University, Xiamen 361005, China*
[2]*School of Computer Science and Technology, Heilongjiang University, Harbin 150080, China*

Correspondence should be addressed to Xing Gao; gaoxing@xmu.edu.cn

Received 31 August 2014; Accepted 2 November 2014

Academic Editor: Xiuzhen Cheng

A multimodel based range query processing algorithm is proposed to solve the information collection task for the CPSs, which utilizes multiple probability models to depict the data distribution of a sensor node. The execution of the multimodel based algorithm consists of two phases, which are the preprocessing phase and the query processing phase. During the preprocessing phase, multiple models are constructed for each node according to their historical data. During the query processing phase, a suitable model is selected from the multiple models with the help of a sampling based algorithm, which is used to process the query. As the multimodel based algorithm needs to sample data from the network, it can waste energy more than that of the single model based algorithm in some cases, which does not sample data from the network. The cost of the multimodel based and single model based algorithm is analyzed. A cost model based algorithm is proposed to select a better algorithm to process a query from the two algorithms. Experimental results show that the cost model based algorithm can save 13.3% energy consumption more than that of the single model based algorithm.

1. Introduction

Cyber-physical systems (CPSs), which consist of computing devices and embedded systems such as distributed sensors and actuators, integrate computation, communication, and control with the physical world [1]. The tasks, running on the CPSs, involve close interactions between the cyber world and the physical world. Extracting knowledge from the physical world is an important task for the CPSs. Some useful information is collected from the physical world firstly and then analyzed to extract knowledge. Wireless sensor networks (WSNs) [2–5] are usually used to fulfill the information collection task, which is transformed into some kinds of queries for the WSNs [6–10].

The range query is one of the most important types of queries to collect information from the WSNs. For instance, a range query is sent to the sensor network distributed in a forest, asking for the places where the temperature lies in $[r_1, r_2]$. The sensors, whose temperature lies in this range, return their locations or IDs to the sink. If the sensors return their IDs, the sink transforms the IDs to locations and returns the locations.

Some existing methods have been proposed to solve the range query in the WSN, which can be classified into two classes. The first class is the data centric storage based algorithms, such as GHT [11], DIM [12], comb-needle [13], double ruling [14], and energy-aware algorithm [15]. The data centric storage algorithms define different types of events for the data collected by sensors. Each type of events is stored in a particular node called event storage node in the WSN. When a node detects an event, it transmits the data of the event to the event storage node. A range query, transformed to a query for an event, is sent directly to the event storage node and answered by the node. The event defined in the data centric storage algorithm is very rigid, which means the users can only ask for the result of a range defined by the event. So these algorithms do not fit for the query asking for the result of any range.

The second class is the local storage based algorithms, such as [16–19]. For the traditional local storage based

algorithms [16, 17], the data collected by a sensor is stored in its local storage. The queries are sent to each node and the nodes satisfying the query return their results to the user. The problem of the traditional algorithm is that all nodes need to return their results to the sink whether they satisfy the query or not, which consumes a lot of energy. In [19], a probability model is used to process the range query. The probability model is used to estimate the probability that each node satisfies the query. Only if the probability of a node, satisfying the query, is above a threshold, the node is considered as a result. With the help of the probability model, nodes do not return any result to the sink for a range query. There are two problems for the algorithm. The first one is the algorithm can only give an approximate answer to a query. The second one is it is hard to determine a threshold balancing the efficiency of energy consumption and the accuracy of the query result.

In this paper, we propose a multimodel based algorithm to solve the range query in the WSN, which is a local storage based algorithm. Compared with the other local storage based algorithms, our algorithm has the following advantages. First, our algorithm constructs multiple probability models. With the help of these probability models, only the most relevant nodes among all nodes need to transmit their results to the sink, which saves more energy than the traditional local storage algorithms. Second, our algorithm can give the precise answer to the range query with minimum energy consumption. The multimodel based range query processing algorithm proposed in this paper is composed of 3 steps:

(1) probability model construction;

(2) sampling based model selection;

(3) model based query processing.

The probability model construction algorithm first constructs multiple probability models for each node in the WSN. By clustering the historical data collected by the nodes, m subclasses are constructed and each node builds a probability model according to the data of its own in a subclass. With the help of the multiple probability models, the query processing algorithm can select the more accurate model to describe the data distribution for the current condition than that of the single model based algorithm.

Multiple probability models have been constructed for each node in the WSN. For a particular range query, there must be a method to select a suitable probability model for each node to process the query. In this paper, a sampling based algorithm is proposed to fulfill this task. Some typical nodes are selected by a preprocessing algorithm. Then the sampling based algorithm collects the data from these typical nodes to determine a suitable model for a query.

Combining the model selected by the sampling based model selection algorithm with the real data sensed by a node, the model based query processing algorithm can minimize the energy consumption of processing a range query. While the performance of the multimodel based algorithm is not the best in all cases, we analyze the cost of the multimodel based algorithm and propose a range query processing algorithm augmented with the cost model, which selects a suitable query processing algorithm for a range query according to the cost model. Experimental results show that the cost model based algorithm can provide the accurate answer with 13.3% energy consumption less than that of the single model based algorithm.

The contributions of this paper are as follows. First, a multimodel based algorithm is proposed to solve the range query in the WSN, which utilizes multiple probability models to improve the accuracy of the data distribution function and saves the energy consumption of the algorithm. Second, the energy consumption of the multimodel based algorithm is analyzed and a query processing algorithm augmented with the cost model is proposed to save energy in most cases. Third, extensive experiments were done to verify the efficiency of the proposed algorithms.

The rest of the paper is organized as follows. Section 2 introduces the multimodel based range query processing algorithm. Section 3 analyzes the cost model of the multimodel based algorithm and proposes the query processing algorithm augmented with the cost model. Section 4 evaluates the performance of the proposed algorithms on real dataset. Section 5 briefly discusses the related work and in Section 6 we draw the conclusion.

2. The Model Based Range Query Processing Algorithm

2.1. Probability Model Construction. First, multiple probability models are constructed for each node in the WSN based on the historical data collected from each node. Let N be the set of all nodes in the WSN and let $|N|$ be the number of nodes in the WSN. Each node collects data, such as the temperature and humidity, from the environment at a certain rate. The data from all nodes at a given timestamp t consists of a vector $V_t = (v_t^1, v_t^2, \ldots, v_t^{|N|})$, where v_t^j represents the data collected by the jth node n^j at the timestamp t. The historical data set H is composed of a set of vectors V_t at different timestamps. Given the historical data set H, the vectors in H can be clustered into many subclasses. If the vectors in H are clustered into m subclasses, represented as $H_1, H_2, \ldots, H_m$, a probability model can be constructed for each node based on the vectors contained in a subclass. For convenience, we list the notations used throughout this paper in the Notations section shown at the end of the paper.

The data collected by a node is a random variable, which can be described by a probability distribution function (PDF). The PDF of a random variable is usually hard to calculate, but it can be estimated by a histogram. A histogram is a representation of tabulated frequencies, erected over discrete intervals (bins), with an area equal to the frequency of the observations in the interval. The total area of the histogram is equal to the amount of data.

The vectors belonging to the ith subclass H_i can be used to construct a histogram for each node in the WSN. For H_i, all data belonging to the node n^j forms a data set represented as $D_i^j = \{v_t^j \mid \forall V_t \in H_i, v_t^j \in V_t\}$. Let $v_{t1}^j = \max\{D_i^j\}$ and

$v_{t2}^j = \min\{D_i^j\}$, which means v_{t1}^j and v_{t2}^j are the maximum and minimum value of node n^j in D_i^j. The range $[\lfloor v_{t2}^j \rfloor, \lceil v_{t1}^j \rceil)$ can be equally divided into $(\lceil v_{t1}^j \rceil - \lfloor v_{t2}^j \rfloor + 1)$ bins, whose length is 1. f_x^j represents the number of data falling into the interval $[x, x+1)$. If the range $[\lfloor v_{t2}^j \rfloor, \lceil v_{t1}^j \rceil)$ is smaller than 1, we can enlarge the range multiple times to make it larger than 1. If the range is enlarged, the data sampled from a sensor must be enlarged by the same times.

In the same way, a probability model can be constructed for each node based on the vectors in each subclass. If the historical data set H is divided into m subclasses, m probability models are constructed for each node. The ith probability model of a node n^j is represented by M_i^j. There is one special probability model for each node, called general model, which is constructed based on all the historical data collected by a node instead of a subset of data. The construction method of the general model is the same as that of the other models. The general model of a node n^j is represented as M_g^j. There are altogether $m+1$ probability models for each node, which can be classified into two classes $M^j = \{M_1^j, M_2^j, \ldots M_m^j\}$ and M_g^j.

Given a range $[a, b]$, the probability of the data falling in $[a, b]$ calculated by M_i^j is represented as $\mathrm{pr}_i^j\{[a, b]\}$, which can be estimated by formula (1) based on the histogram. The ratio of the area of the histogram of the range $[\lfloor a \rfloor, \lceil b \rceil)$ to the total area of the histogram of the range $[\lfloor v_{t2}^j \rfloor, \lceil v_{t1}^j \rceil)$ is used to estimate $\mathrm{pr}_i^j\{[a, b]\}$ in

$$\mathrm{pr}_i^j\{[a, b]\} = \frac{\sum_{x=\lfloor a \rfloor}^{\lceil b \rceil - 1} f_x^j}{\sum_{y=\lfloor v_{t2}^j \rfloor}^{\lceil v_{t1}^j \rceil - 1} f_y^j}. \quad (1)$$

2.2. The Typical Node Selection Algorithm. Multiple probability models have been constructed for each node in the WSN. For a particular range query, there must be a method to select a probability model for each node to process the query. In this paper, a sampling based method is proposed to fulfill this task.

Firstly, some typical nodes are selected from the WSN. Before sending a range query to the WSN, the sink samples data from the typical nodes. Based on the data sampled from the typical nodes, a suitable probability model $M_i^j \in M^j$ is selected to make the query processing algorithm be carried out efficiently. Before giving the typical node selection algorithm, we present some definitions.

Let μ_i^j represent the mean of D_i^j of the node n^j. μ_i^j can be calculated by formula (2), where $|H_i|$ represents the number of vectors in the subclass H_i:

$$\mu_i^j = \frac{\sum_{\forall v_t^j \in D_i^j} v_t^j}{|H_i|}. \quad (2)$$

Definition 1 (data range). The data range of a node n^j, whose data belongs to D_i^j, is defined as $R_i^j = [\lfloor \mu_i^j \rfloor - x, \lceil \mu_i^j \rceil + y]$ and $\mathrm{pr}_i^j\{R_i^j\} \geq \sigma$, where σ $(0 < \sigma < 1)$ is called the coverage threshold.

The range R_i^j is a subrange of the total range of the model M_i^j. The ratio of the area of R_i^j of the histogram and the total area of the histogram is not less than a threshold σ.

Theorem 2. *The x and y of a data range are integers.*

Proof. R_i^j of a node n^j is constructed as follows. μ_i^j is calculated according to formula (2), which falls into the bin $[\lfloor \mu_i^j \rfloor, \lceil \mu_i^j \rceil]$ of the model M_i^j of the node n^j. Initially, we set $R_i^j = [\lfloor \mu_i^j \rfloor, \lceil \mu_i^j \rceil]$ and check whether the ratio of the area of current R_i^j of the model M_i^j and the total area of the model M_i^j is not less than σ. If not, we add all adjacent bins of the current R_i^j to it, which means one or two bins are added to the current R_i^j. The R_i^j is expanded until the coverage threshold σ is reached. As the interval of the histogram is 1, the final R_i^j is $[\lfloor \mu_i^j \rfloor - x, \lceil \mu_i^j \rceil + y]$ where x and y are integers. □

As there are m subclasses, the node n^j has m data ranges, represented by $R_1^j, R_2^j, \ldots, R_m^j$. The intersection between the lth and the kth data range of the node n^j is represented as $R_{lk}^j = R_l^j \cap R_k^j$. There are altogether $C(m, 2)$ intersections between any two data ranges of a node, where $C(m, 2)$ represents the number of the 2 combinations from integer 1 to m.

Definition 3 (candidate node). If an intersection R_{lk}^j of the node n^j satisfies the condition $\mathrm{length}(R_{lk}^j) = \min\{\mathrm{length}(R_{lk}^x) \mid x = 1, 2, \ldots, |N|\}$, the node n^j is called a candidate node. The pair (l, k) is called distinguishable by the candidate node n^j. lengh$(\cdot)$ is a function, representing the length of the intersection.

For example, in Figure 1, there are two nodes n^1 and n^2 in the WSN. The data collected by them is clustered into two subclasses. The data ranges of n^1 are $R_1^1 = [v_1^1, v_2^1]$ for subclass 1 and $R_2^1 = [v_3^1, v_4^1]$ for subclass 2. The data ranges of n^2 are $R_1^2 = [v_1^2, v_2^2]$ for subclass 1 and $R_2^2 = [v_3^2, v_4^2]$ for subclass 2.

In Figure 1(a), the intersection $R_{12}^1 = [v_1^1, v_2^1] \cap [v_3^1, v_4^1] = \emptyset$ and the intersection $R_{12}^2 = [v_1^2, v_2^2] \cap [v_3^2, v_4^2] = [v_3^2, v_2^2]$. As $\mathrm{length}(R_{12}^1) = 0 < \mathrm{length}(R_{12}^2) = v_2^2 - v_3^2$, according to the definition of the candidate node, n^1 is the candidate node and it can distinguish the subclass 1 from the subclass 2. If we sample data v_t^1 from the node n^1 and find $v_t^1 \in R_1^1$, at this time, the models for every node (M_1^1 and M_1^2) constructed from the subclass 1 are better than those (M_2^1 and M_2^2) constructed from the subclass 2 to process the query.

In Figure 1(b), $\mathrm{length}(R_{12}^1) = v_2^1 - v_3^1 < \mathrm{length}(R_{12}^2) = v_2^2 - v_3^2$, which means n^1 is the candidate node. When data sampled from n^1 falls in $[v_1^1, v_3^1]$ or $[v_2^1, v_4^1]$, the probability models constructed by subclass 1 or subclass 2 are selected. But when the data from n^1 falls in $[v_3^1, v_2^1]$, the models cannot be determined directly. We propose a Euclidean distance based method to solve this problem in the next section. Compared with the length of $[v_3^2, v_2^2]$ of n^2, the length of $[v_3^1, v_2^1]$ of n^1

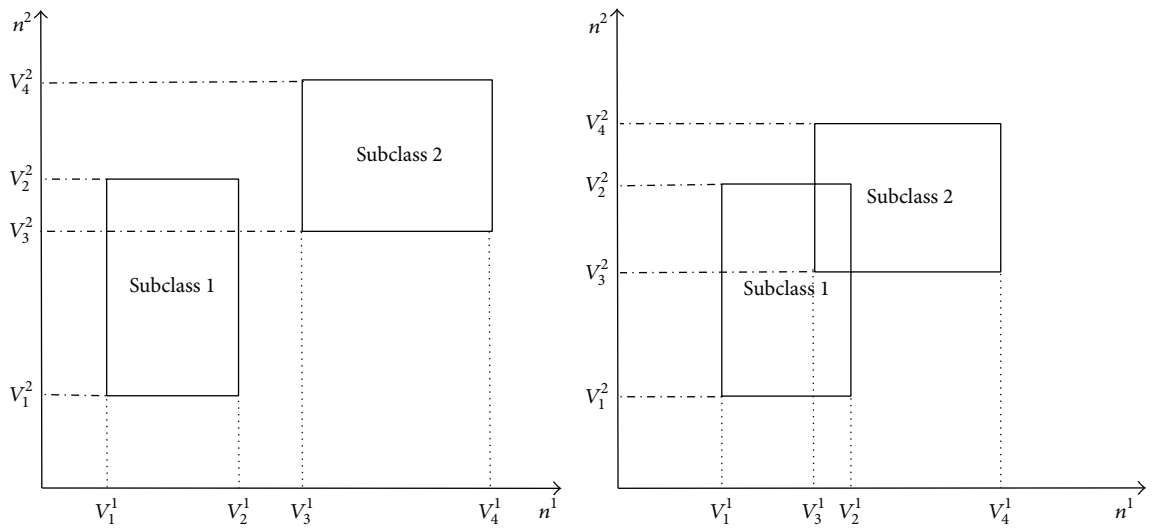


Figure 1: Illustration to data range and candidate node.

Input: $N, H_i\ (1 \le i \le m), I$
Output: MCN
(1) **for** each node $n^j \in N$ **do**
(2) calculate the m Data Ranges R_i^j for a node n^j
(3) **end for**
(4) **for** each node $n^j \in N$ **do**
(5) calculate all the intersections $R_{lk}^j = R_l^j \cap R_k^j$ for a node n^j
(6) **end for**
(7) **for** each pair of $(l, k) \in I$ **do**
(8) find the node n^j with the minimum R_{lk}^j among all nodes
(9) put n^j into the candidate node set CN_{lk}
(10) **end for**
(11) construct the MCN by selecting the unique nodes from all CN_{lk}
(12) **for** each node n^j in the MCN **do**
(13) $n^j \rightarrow$ counter = the number of CN_{lk} containing n^j
(14) $n^j \rightarrow$ list = all the pairs (l, k) of CN_{lk} containing n^j
(15) **end for**
(16) **return** MCN

Algorithm 1: The preprocessing algorithm.

is much smaller, which means the probability, that a model cannot be determined, is minimized. That is why n^1 is selected as the candidate node.

The typical node selection algorithm needs to prepare some parameters with the help of a preprocessing algorithm. The preprocessing algorithm first calculates the m data ranges R_i^j $(i = 1, \ldots, m)$ for each node. Then the preprocessing algorithm calculates the $C(m, 2)$ intersections R_{lk}^j for each node. As there can be multiple nodes satisfying the definition of candidate node for an intersection R_{lk}^j, the preprocessing algorithm calculates a candidate node set CN_{lk} for each intersection, which is composed of all the candidate nodes of the intersection. Finally, the preprocessing algorithm merges the candidate nodes in the candidate node sets of all nodes into a merged candidate node (MCN) set. Each element in MCN has two attributes, which are a counter and a list. If multiple intersections have the same candidate node in their candidate node sets, these candidate nodes are merged into a unique one, called merged candidate node. The counter of the merged candidate node is the number of CN_{lk} containing the merged candidate node. The list of the merged candidate node is all the pairs (l, k) of CN_{lk} containing the merged candidate node. Let I be a set of the pairs (l, k), which contains the $C(m, 2)$ 2 combinations of the m subclasses. The preprocessing algorithm is given in Algorithm 1.

For example, Figure 2 shows that there are two nodes n^1 and n^2 in the WSN. The data collected by the two nodes are

Input: MCN
output: T
(1) **while** $I \neq \emptyset$ **do**
(2) sort the nodes in the MCN according to their counters in descending order
(3) select the node with maximum counter from MCN
(4) add the selected node n^j to the typical node set T
(5) set $n^j \rightarrow counter^j$ to the current counter of n^j in MCN
(6) remove the selected node from MCN
(7) remove all the pairs in the list of the selected node from I
(8) remove all the pairs in the list of the selected node from the list of the other candidate nodes
(9) subtract the counter of a candidate node in MCN by 1 when a pair is removed from the node's list
(10) **end while**
(11) **return** T

Algorithm 2: The typical node selection algorithm.

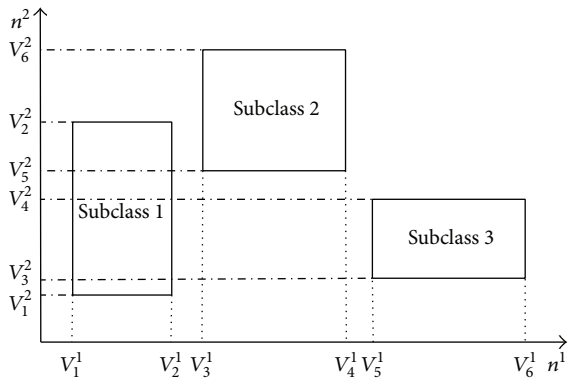


Figure 2: Illustration to the preprocessing algorithm.

clustered into three subclasses. The data ranges of n^1 for the three subclasses are $R_1^1 = [v_1^1, v_2^1]$ for subclass 1, $R_2^1 = [v_3^1, v_4^1]$ for subclass 2, and $R_3^1 = [v_5^1, v_6^1]$ for subclass 3. The data ranges of n^2 for the three subclasses are $R_1^2 = [v_1^2, v_2^2]$ for subclass 1, $R_2^2 = [v_3^2, v_4^2]$ for subclass 2, and $R_3^2 = [v_5^2, v_6^2]$ for subclass 3. Also $R_{12}^1 = R_1^1 \cap R_2^1 = \emptyset$. In the same way, we can calculate $R_{13}^1 = \emptyset$, $R_{23}^1 = \emptyset$, $R_{12}^2 = [v_5^2, v_2^2]$, $R_{13}^2 = [v_3^2, v_4^2]$, and $R_{23}^2 = \emptyset$. Based on the results, the candidate node set $\mathrm{CN}_{12} = \{n^1\}$, because $\mathrm{length}(R_{12}^1) = 0 < \mathrm{length}(R_{12}^2) = v_5^2 - v_2^2$. Moreover $\mathrm{CN}_{13} = \{n^1\}$ and $\mathrm{CN}_{23} = \{n^1, n^2\}$. The MCN contains two nodes $\{n^1, n^2\}$. After merging, the counter of n^1 is 3, because all of the candidate node sets contain n^1. The list of n^1 contains $\{(1,2),(1,3),(2,3)\}$. The counter of n^2 is 1. The list of n^2 only contains $\{(2,3)\}$.

Based on the definitions and the MCN calculated by the preprocessing algorithm, we propose a greedy based algorithm to select the typical nodes from the WSN. The greedy algorithm sorts all the merged candidate nodes in the MCN according to their counters in descending order. Then it selects the candidate node with the largest counter as the typical node and removes the selected node from the MCN. Then the entries in the list of the selected node are removed from I and the list of the other merged candidate nodes is left in the MCN. When an entry is removed from the list of a merged candidate node left, the counter of the node is subtracted by 1. The greedy based algorithm repeats this process until the set I is empty, which means any two models can be distinguished by a candidate node selected. All the candidate nodes are selected from the typical node set T. The jth typical node n^j in T has an attribute represented as $\mathrm{counter}^j$, which is the number of intersections n^j can distinguish. The greedy based typical node selection algorithm is given in Algorithm 2.

For example, after sorting, n^1 is selected into the typical node set, because its counter is 3, which is bigger than the counter of n^2. The pairs in the list of n^1 are removed from $I = \{(1,2),(1,3),(2,3)\}$ and the list of n^2. After removing, $n^2 \rightarrow \mathrm{list} = \emptyset$, $n^2 \rightarrow \mathrm{counter} = 0$, and $I = \emptyset$. As $I = \emptyset$, Algorithm 2 finished. Otherwise, the candidate nodes left in MCN are sorted and the process is repeated again. In our example, the typical node set T is $\{n^1\}$. It is obvious in Figure 2 that the data collected from n^1 can distinguish the models constructed by the three subclasses.

2.3. Sampling Based Model Selection and Model Based Query Processing Algorithms. The model based query processing algorithm for a range query works as follows. A node n^j in the WSN stores the m models M^j and a general model M_g^j in its local storage. After receiving a range query from a user, the sink samples data from the typical nodes in T selected by Algorithm 2. According to the data sampled, a probability model is selected. The sink sends the range query together with the index of the model selected to all nodes in the WSN. A node processes the query with the help of the probability model selected. In this section, we will introduce the algorithm selecting a suitable probability model based on the sampled data and the model based query processing algorithm.

To save energy, the model selection algorithm does not sample data from all typical nodes at the same time. We sort the nodes in T according to their counters in descending order. The model selection algorithm samples data from one node in the typical node set at a time from the beginning. Only when the sampled data cannot determine a model,

Input: $T, R_i^j (1 \le i \le m, 1 \le j \le |N|)$
Output: index of the selected model
(1) set the candidate model set CM to $\{1, 2, \ldots, m\}$
(2) sort the nodes in T according to their counters in descending order
(3) **for** each node $n^j \in T$ **do**
(4) sample data v_t^j from n^j
(5) **for** each R_i^j of n^j **do**
(6) **if** $v_t^j \notin R_i^j$ **then**
(7) remove i from CM
(8) **end if**
(9) **end for**
(10) **if** |CM| == 0 *or* |CM| == 1 **then**
(11) break
(12) **end if**
(13) **end for**
(14) **if** |CM| == 1 **then**
(15) *index* = the element in CM
(16) **else**
(17) construct the vector V_s containing data sampled from the typical nodes
(18) construct a partial center vector V_i for each subclass
(19) set the $d_{\min} = \infty$
(20) **for** $1 \le i \le m$ **do**
(21) d = the Euclid distance between V_s and V_i
(22) **if** $d \le d_{\min}$ **then**
(23) $d_{\min} = d$
(24) *index* = i
(25) **end if**
(26) **end for**
(27) **end if**
(28) **return** *index*

Algorithm 3: The model selection algorithm.

the data of the next typical node is sampled. Initially, the candidate model set, represented as CM, is set to $\{1, 2, \ldots, m\}$. For data v_t^j, sampled from a typical node n^j, the model selection algorithm checks whether $v_t^j \in R_i^j$ for all m data ranges of n^j. If the $v_t^j \notin R_i^j$, the model selection algorithm removes the number i from CM. If there is only one number left in CM, the corresponding model is selected as the suitable model. Otherwise, there will be no number or multiple numbers in CM. In this case, the model selection algorithm uses a distance based method to select the suitable model. The data sampled from the typical nodes forms a vector. As each subclass is composed of a lot of vectors, the center of each subclass also forms a vector. The vector formed by the data of the typical nodes is part of the center vectors of the subclasses. The data corresponding to the typical nodes is drawn from each center vector of the subclasses, which forms a partial center vector for each subclass. The model selection algorithm calculates the Euclidean distance d between the typical node vector and the partial center vector of each subclass and selects the subclass with the minimum distance as the suitable model. The model selection algorithm is given in Algorithm 3.

After the sink selects a suitable probability model for a range query, it sends the index of the model together with the range query to all the nodes in the WSN. When a node n^j receives the query, it calculates two probabilities. Firstly, it gets the index of the probability model and calculates the probability that the data of the node satisfies the range query according to the selected probability model. Secondly, it calculates the probability according to the general probability model. We represent the first probability as $\text{pr}_{\text{index}}^j$ and the second probability as pr_g^j. The larger one of the two probabilities is chosen as the final probability represented as $\text{pr}_{\text{final}}^j$. If $\text{pr}_{\text{final}}^j$ is larger than a threshold ρ, which is called the probability threshold, but the data really collected by the node does not satisfy the range query, the node returns a negative answer to the sink. If $\text{pr}_{\text{final}}^j \le \rho$, while the data really collected by the node satisfies the range query, the node returns a positive answer to the sink. The node does not return any answer to the sink in other cases.

We calculate two probabilities that the data of the node satisfies the range query and use the larger one as the final probability, because the model constructed by a subclass of data of a node is not more accurate than the general model. For example, the multimodel based range query processing algorithm uses two models and a general model to depict the temperature of a room. One model depicts the distribution of the temperature from 7:00 A.M. to 9:00 A.M. and another

Input: query range $[r_1, r_2]$, index of the selected model *index*
Output: the result node set Q
(1) *index* = value returned by Algorithm 3
(2) **for** $\forall n^j \in N$ **do**
(3) pr^j_{index} = the probability of the node satisfying the received range query calculated by the *index* model
(4) $\text{pr}^j_{general}$ = the probability of the node satisfying the received range query calculated by the general model
(5) **if** $\max\{\text{pr}^j_{index}, \text{pr}^j_g\} > \rho$ **then**
(6) put n^j into the result set Q
(7) **end if**
(8) **end for**
(9) broadcast the query and the *index* throughout the sensor network
(10) collect answers from the nodes in the network
(11) **if** receive a positive answer from a node n^j **then**
(12) put ID of node n^j into the result set Q
(13) **else if** receive a negative answer from a node n^j **then**
(14) remove the ID of node n^j from the result set Q
(15) **end if**
(16) **return** Q

Algorithm 4: The model based query processing algorithm for the sink.

Input: $[r_1, r_2]$, *index*, v^j_t
Output: *NULL*
(1) extract the *index* and query range $[r_1, r_2]$ from the packet received
(2) pr^j_{index} = the probability of the node satisfying the received range query calculated by the *index* model
(3) $\text{pr}^j_{general}$ = the probability of the node satisfying the received range query calculated by the general model
(4) **if** $(\max\{\text{pr}^j_{index}, \text{pr}^j_{general}\} > \rho)$ & $(v^j_t \notin [r_1, r_2])$ **then**
(5) send a negative answer containing the ID of the current node to the sink
(6) **else if** $(\max\{\text{pr}^j_{index}, \text{pr}^j_{general}\} \leq \rho)$ & $(v^j_t \in [r_1, r_2])$ **then**
(7) send a positive answer containing the ID of the current node to the sink
(8) **end if**

Algorithm 5: The model based query processing algorithm for an ordinary node.

model depicts the distribution of temperature from 9:00 A.M. to 11:00 A.M. The general model depicts the distribution of the temperature from 7:00 A.M. to 11:00 A.M. If the query range is from 8:00 A.M. to 10:00 A.M., neither of the two models can depict the distribution of the temperature more accurately than the general model. By comparing the probabilities calculated by the selected model and the general model, we guarantee that the final probability is not worse than that of the general model.

When $\text{pr}^j_{\text{final}} > \rho$, it means that the data sampled by the node has a large probability to satisfy the query. The event, that the real data collected by the node does not satisfy the query, is a small probability event. Only if the small probability event happens, the node needs to send the answer to the sink, which minimizes the number of messages transmitted between the node and the sink. The same reason is for the other two cases. The model based range query processing algorithm is a distributed algorithm. The algorithm executed by the sink is given in Algorithm 4. The algorithm executed by an ordinary node is given in Algorithm 5.

3. Cost Analysis of the Multimodel Based Range Query Processing Algorithm

The multimodel based query processing algorithm is not always efficient for all range queries, because it must sample data from the typical nodes first and then collects the results from these nodes. While the single model based algorithm directly collects data from the nodes, the multimodel based algorithm may consume more energy than that of the single model based algorithm. We construct the cost model to estimate the energy consumed by the multimodel based algorithm and the single model based algorithm. By comparing the cost consumed by the two algorithms, we can select a better one to process a query. In this paper, we use the number of messages transmitted by an algorithm to represent the energy consumption.

3.1. Cost Analysis of the Single Model Based Algorithm. Before analyzing the cost of the multimodel based algorithm, we present a single model based algorithm. In the single model based algorithm, a node n^j in the WSN only stores its general

model M_g^j in its local storage. After receiving a query from a user, the sink directly broadcasts the query throughout the WSN. A node calculates the probability pr_g^j that its data satisfies the query with M_g^j. If $\mathrm{pr}_g^j > \rho$, but the data really collected by the node does not satisfy the range query, the node returns a negative answer to the sink. If $\mathrm{pr}_g^j \leq \rho$, but the data really collected by the node satisfies the range query, the node returns a positive answer to the sink. The node does not return any answer to the sink in other cases.

The energy cost for the single model based query processing algorithm C_s can be estimated by formula (3), in which pr_g^j is the probability that a node n^j satisfies a query. The first part of formula (3) is the energy cost of broadcasting the query throughout the network. The second part of formula (3) is the expectation of the energy cost that the sink receives answers from the nodes in the WSN:

$$C_s = |N| + \sum_{\substack{n^j \in N \\ \mathrm{pr}_g^j > \rho}} \left(1 - \mathrm{pr}_g^j\right) * \mathrm{pl}^j + \sum_{\substack{n^j \in N \\ \mathrm{pr}_g^j \leq \rho}} \mathrm{pr}_g^j * \mathrm{pl}^j. \tag{3}$$

3.2. Cost Analysis of the Multimodel Based Algorithm. We analyze the cost consumed by the multimodel based algorithm. The cost of the multimodel based algorithm is composed of two parts. The first part is the energy cost by the sampling phase, represented by E_s. The second part is the energy cost by the query processing, represented by E_q.

E_s can be estimated as follows. Define an event as follows: "after sampling data from the first j typical nodes, the model selection algorithm can choose the probability model used by the query processing algorithm." The probability of the event is $(1 - \mathrm{pr}\,t^1)(1 - \mathrm{pr}\,t^2)\cdots(1 - \mathrm{pr}\,t^{(j-1)})\mathrm{pr}\,t^j$, where $\mathrm{pr}\,t^j$ is the probability that the jth typical node can determine the probability model. $\mathrm{pr}\,t^j$ can be estimated by

$$\mathrm{pr}\,t^j = \frac{\mathrm{counter}^j}{C(m,2)}. \tag{4}$$

If there are m models in the multimodel based range query processing algorithm, there are altogether $C(m,2)$ pairs of models to be distinguished by the typical nodes, where $C(m,2)$ represents the number of 2 combinations of $\{1,2,\ldots,m\}$. In the typical node selection algorithm (Algorithm 2), $\mathrm{counter}^j$ is used to record the number of pairs of models that node n^j can distinguish. Formula (4), which divides $\mathrm{counter}^j$ by $C(m,2)$, is the probability that the jth typical node can distinguish a model from other models, which is used to estimate the probability that the jth typical node can determine a probability model.

The probability that a model is determined by the jth typical node is $\prod_{i=1}^{j}(1 - \mathrm{pr}\,t^{i-1})\mathrm{pr}\,t^i$ and the energy cost is $(\sum_{i=1}^{j}\mathrm{pl}^i)$, where pl^j represents the path length between the sink and the jth typical node. As there are $|T|$ typical nodes altogether and the sampling process is composed of sending and collecting data, the expectation of the energy consumption of the model selection algorithm E_s is given by

$$E_s = 2\sum_{j=1}^{|T|}\left(\left(\prod_{i=1}^{j}\left(1 - \mathrm{pr}\,t^{i-1}\right)\mathrm{pr}\,t^i\right)\left(\sum_{i=1}^{j}\mathrm{pl}^i\right)\right). \tag{5}$$

Let $\mathrm{pr}\,m_i$ be the probability that the ith model is chosen by the model selection algorithm and let E_i be the energy consumption of the query processing algorithm of the chosen model. The expectation of the energy cost of the query processing algorithm E_q is given by

$$E_q = |N| + \left(\sum_{i=1}^{m}\mathrm{pr}\,m_i * E_i - \frac{E_s}{2}\right). \tag{6}$$

The first part of formula (6) is the number of messages by which the sink broadcasts the range query throughout the network. The second part of formula (6) is the expectation of the energy cost by which the sink receives answers from the nodes in the WSN. As the sink has received the data from the typical nodes at the sampling phase, the typical nodes do not transmit their data to sink at the query processing phase and $E_s/2$ is subtracted from $\sum_{i=1}^{m}\mathrm{pr}\,m_i * E_i$. We analyze the estimations for $\mathrm{pr}\,m_i$ and E_i next.

As each model is constructed by the vectors contained by a subclass, the number of vectors belonging to a subclass can be used to estimate the probability that the corresponding model is chosen. The larger the number of vectors contained by a subclass is, the higher the probability, that data sampled from the typical nodes belongs to the data range of the subclass, is. Let $|H|$ be the total number of vectors contained by all subclasses and let $|H_i|$ be the number of vectors contained by the subclass H_i. The $\mathrm{pr}\,m_i$ can be estimated by

$$\mathrm{pr}\,m_i = \frac{|H_i|}{|H|}. \tag{7}$$

Let A^j be the random variable that a node n^j returns answer to the sink. If the answer is yes, $A^j = 1$. Otherwise, $A^j = 0$. The expectation of A^j is $(1 - \max\{\mathrm{pr}_i^j, \mathrm{pr}_g^j\})$ when $\max\{\mathrm{pr}_i^j, \mathrm{pr}_g^j\} > \rho$, where pr_i^j and pr_g^j are the probabilities calculated by the ith model M_i^j and the general model M_g^j, because the node n^j returns answer only when its real data does not satisfy the query at this time. The expectation of A^j is $\max\{\mathrm{pr}_i^j, \mathrm{pr}_g^j\}$ when $\max\{\mathrm{pr}_i^j, \mathrm{pr}_g^j\} \leq \rho$. The number of messages transmitted by the ith model can be calculated in formula (8), where N represents all node sets in the sensor network:

$$\begin{aligned} E_i = &\sum_{\substack{n^j \in N \\ \max\{\mathrm{pr}_i^j, \mathrm{pr}_g^j\} > \rho}} \left(1 - \max\left\{\mathrm{pr}_i^j, \mathrm{pr}_g^j\right\}\right) * \mathrm{pl}^j \\ &+ \sum_{\substack{n^j \in N \\ \max\{\mathrm{pr}_i^j, \mathrm{pr}_g^j\} \leq \rho}} \max\left\{\mathrm{pr}_i^j, \mathrm{pr}_g^j\right\} * \mathrm{pl}^j. \end{aligned} \tag{8}$$

Input: $[r_1, r_2]$
Output: Q returned by the called algorithm
(1) estimate the cost C_m for the multi-model based query processing algorithm
(2) estimate the cost C_s for the single model query processing algorithm
(3) **if** $C_m < C_s$ **then**
(4) **return** Q = call the multi-model based algorithm
(5) **else**
(6) **return** Q = call the single model based algorithm
(7) **end if**

Algorithm 6: The range query processing algorithm augmented with cost model.

The energy cost of the multimodel based query processing algorithm C_m is given by

$$C_m = E_s + E_q. \tag{9}$$

3.3. Range Query Processing Algorithm Augmented with the Cost Model. The query processing algorithm augmented with the cost model, given in Algorithm 6, works as follows. The sink estimates the costs C_m and C_s for a particular range query according to formulas (9) and (3). If $C_m < C_s$, the multimodel based algorithm is adopted. Otherwise, the single model based algorithm is used.

4. Performance Evaluation

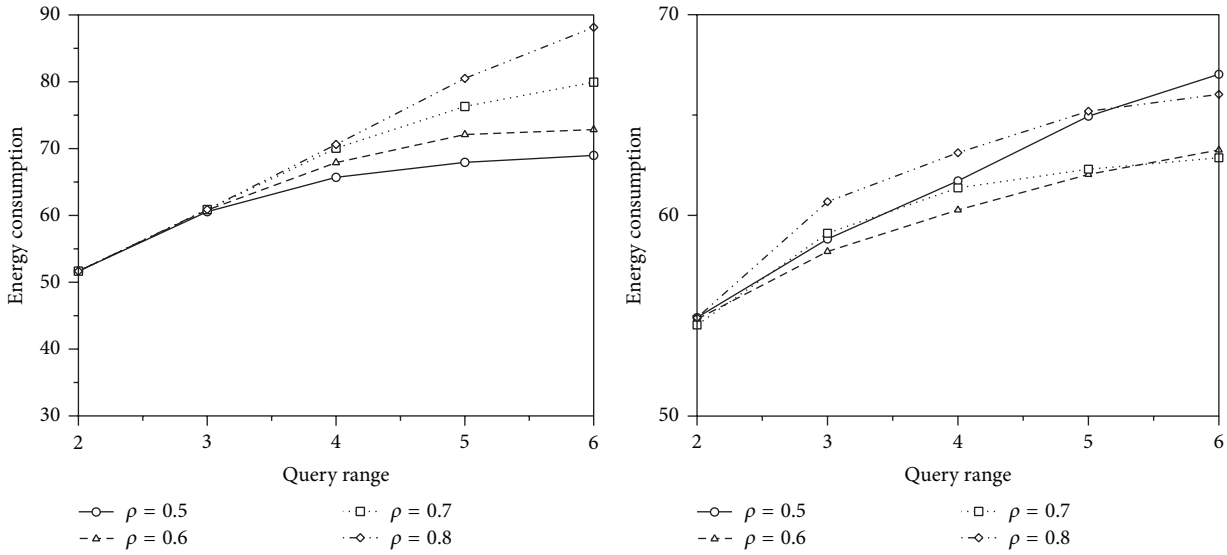
In this section, four experiments were done to verify the performance of the algorithms proposed in this paper. There are three factors that influence the performance of the multimodel based range query processing algorithm, which are the probability threshold ρ, the number of models m, and the coverage threshold σ. In the first three experiments, we test the influence of these factors on the multimodel based algorithm. In the last experiment, we compare the performance of the single model based algorithm, the multi-model based algorithm, and the query processing algorithm augmented with the cost model. The performance of the algorithms is measured by the energy consumption of these algorithms, which is the number of messages transmitted. We adopt the data set collected from 34 sensors deployed in the Intel Berkeley Research lab [20] in our experiments. There are 54 sensors in the data set. As a lot of data of some sensors is lost in the data set and our algorithm needs plenty of historical data to construct probability model, we only select 34 sensors from them. We randomly assign an integer number to each node as its number of hops to the sink, which is used to calculate the energy consumption of each of query processing algorithms.

4.1. Evaluation of the Probability Threshold. In this subsection we evaluate the influence of the probability threshold ρ on the performance of the single model based algorithm (SMA) and the multimodel based range query processing algorithms (MMAs). In this experiment, we change the probability threshold ρ of the algorithms from 0.5 to 0.8. The number of models m and the coverage threshold σ of the MMA are fixed to 4 and 0.8. The sink sends queries with different ranges to the WSN. 10 queries are generated for each range. Figure 3 shows the energy consumption of the SMA and MMA corresponding to different ρ, respectively. The x-axis is the query range sent by a user and y-axis is the number of messages transmitted by the query processing algorithm, which is the average of the number of the messages to process the ten queries for each range.

The experimental results in Figure 3(a) show that, with the increasing of the ρ, the cost of the SMA increases. The SMA with $\rho = 0.5$ consumes the least energy. The energy consumption of the MMAs with different ρ is shown in Figure 3(b). The results show that the MMA with $\rho = 0.6$ consumes the least energy among all cases. Compared with the SMA, the MMA consumes more energy when the query range is small. For example, the query range is 2, while MMA saves more energy when the query range is large. When the query range is small, the number of nodes satisfying the query is small. The MMA needs to sample data from the typical nodes, so it consumes more energy than the SMA. When the query range is large, the number of nodes satisfying the range query is large and the MMA can select a suitable model to process the query. The saved energy of query processing for MMA is much more than the energy consumed by the data sampling, so the MMA is more efficient than the single based algorithm.

4.2. Evaluation of the Number of Models. In this subsection we evaluate the influence of the number of models m on the performance of the MMA. In this experiment, we set the number of models m of the MMA to 4, 6, and 8. The probability threshold ρ and the coverage threshold σ of the MMA are fixed to 0.6 and 0.8. The sink sends queries with different ranges to the WSN. 10 queries are generated for each range. Figure 4 shows the energy consumption of the MMAs corresponding to different m. The x-axis is the query range sent by a user and y-axis is the number of messages transmitted by the MMA, which is the average of the number of the messages to process the ten queries for each range.

The experimental results show that the algorithm with $m = 6$ is the most energy efficient one among all cases. When the number of models is small ($m = 4$), the granularity of the model is coarse. It means that the models constructed when $m = 4$ are not more accurate than those constructed when $m = 6$, which causes the energy waste. When the number of



(a) On the SMA (b) On the MMA

Figure 3: Influence of ρ on the different algorithms.

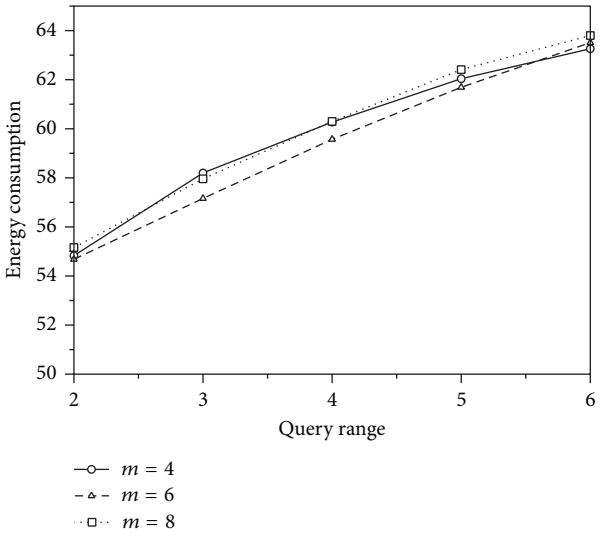


Figure 4: Influence of m on MMA.

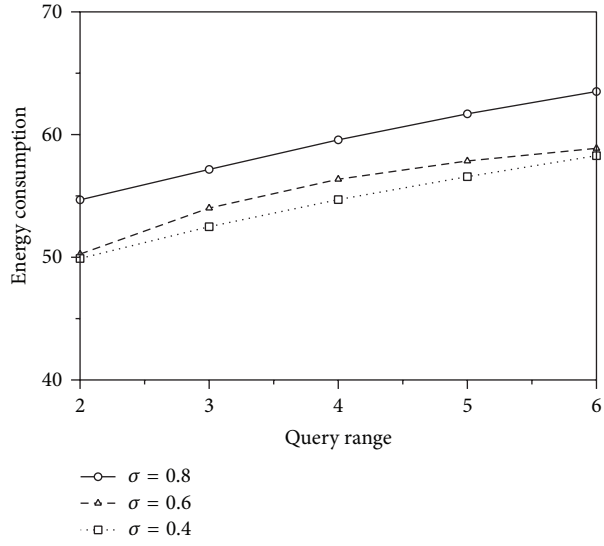


Figure 5: Influence of σ on MMA.

models is large ($m = 8$), the granularity of the model is too sensitive to the sampled noisy data to select a suitable model for the query, which cause the energy waste.

4.3. Evaluation of the Coverage Threshold. In this subsection we evaluate the influence of the coverage threshold σ on the performance of the MMA. In this experiment, we set the coverage threshold σ of the MMA to 0.8, 0.7, and 0.6, respectively. The probability threshold ρ and the number of models m of the MMA are fixed to 6 and 0.6. The sink sends queries with different ranges to the WSN. 10 queries are generated for each range. Figure 5 shows the energy costs of the MMA corresponding to different σ. The meaning of the x-axis and the y-axis is the same as the first two experiments.

The experimental results show that the energy consumption of the algorithm with $\sigma = 0.8$ consumes the most energy among all cases. If the coverage threshold is large, the data range of a model is large and the intersection of the data ranges for different models becomes large. The probability to select a suitable model is decreased. The energy consumption of the algorithm with $\sigma = 0.4$ is better than that of the algorithm with $\sigma = 0.6$.

4.4. Comparisons of Energy Consumption. In this subsection we evaluate the performance of the query processing algorithm augmented with the cost model, represented as CMA. In this experiment, we first evaluate which algorithm the CMA selects for different query ranges. Figure 6 shows the

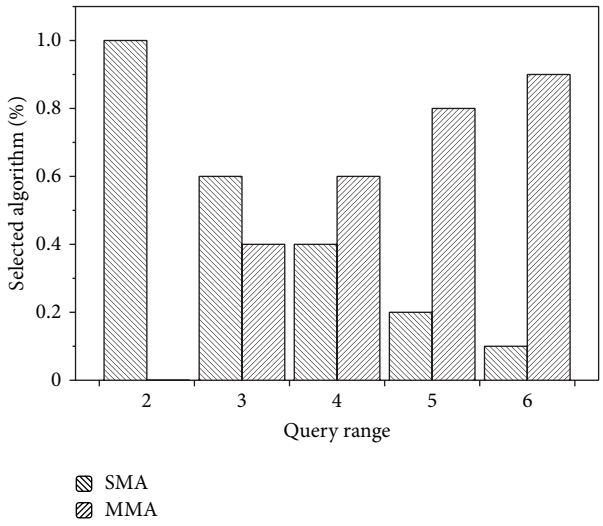


Figure 6: Per. of algorithm selected by CMA.

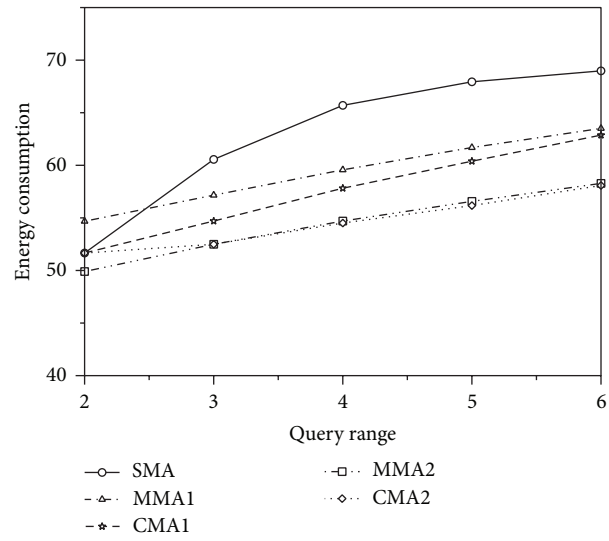


Figure 7: Comparison among SMA, MMA, and CMA.

results. When the query range is 2, CMA selects the SMA to process the query with 100%. When the query range is 3, CMA selects SMA with 60%. With the increasing of the query range, the percent of the SMA selected by CMA decreases. When the query range is 6, the percent of SMA selected by CMA is only 10%.

In the second experiment, we compare the energy consumption among the three kinds of algorithms. The parameter for the SMA is $\rho = 0.5$, which consumes the least energy among all ρs for the SMAs. The parameters for the MMA1 and the CMA1 are $\rho = 0.6$, $\sigma = 0.8$, and $m = 6$. The parameters for the MMA2 and CMA2 are $\rho = 0.6$, $\sigma = 0.4$, and $m = 6$. The results are shown in Figure 7. The meaning of the x-axis and the y-axis is the same as the first two experiments.

The experimental results show that, when the query range is 2, the MMA1 consumes more energy than SMA. The CMA1 selects SMA as the algorithm to process the query, while the MMA2 consumes less energy than SMA when query range is 2. The CMA2 also selects SMA as the algorithm to process the query, which wastes the energy. For other kinds of query range, the energy cost of the CMA is less than that of the corresponding MMA. Compared with the SMA, the MMA1 can save about 5.8% energy and the CMA1 can save about 8.7% energy. The MMA2 can save about 13.6% energy and CMA2 can save about 13.3% energy. The results show that the CMA is the energy efficient algorithm on average among all algorithms. Even though the energy consumption of the CMA is not better than the MMA when the MMA is optimal, the energy consumption of the two algorithms is very close. In other cases, the energy consumption of CMA is much better than that of the MMA, so CMA is the best algorithm to be used in reality.

5. Related Work

The DIM [12] divides the whole network into a lot of zones and embeds a k-d tree-like index in the network. There is only one node in each zone, which acts as the index node of the zone. With the k-d tree-like index, events with comparable attribute values are stored nearby and the DIM can fulfill the range query easily. In the comb-needle algorithm [13], when a sensor detects an event, it distributes the data of the events to the nodes in the vertical direction within h hops around it. If a query is sent to the sensor network along multiple lines in the horizontal direction and the distance between any two neighboring lines is smaller than h hops, the query will be transmitted to the event storage node and get the results. The double ruling algorithm [14] distributes the data of the events around the network in a circle. The query also traverses the network in a circle. The double ruling algorithm guarantees that the circle of the query can intersect with all circles of the queried events. The work in [21] constructs a single dimensional address space of sensor nodes through a zigzag traversing such that geographically near nodes are located near in the linear address space. The multidimensional query is transformed into a single dimensional data space using Hilbert space-filling curves. The work in [22] builds a distributed k-d tree based index structure over sensor network and proposes a dynamic programming based methodology to control the granularity of the index tree in an optimized approach. The work in [23] proposes a bloom filter based algorithm to reduce the number of messages transmitted during the procedure of query processing. The work in [24] proposes the bloom filter based approximate algorithms, which can save the energy even further. The work in [25, 26] considers the security problem of the range query. The work in [25] presents a novel spatiotemporal cross-check approach to ensure secure range queries in event-driven two-tier sensor networks. The work in [26] employs the order-preserving symmetric encryption and a novel data structure called authenticity and integrity tree to preserve authenticity and integrity of query results.

6. Conclusions

In this paper, a multimodel based query processing algorithm is proposed to solve the range query problem. The cost model of the multimodel based query processing algorithm is analyzed and a range query processing algorithm augmented with cost model is proposed to save energy even further. The experimental results show that the cost model based algorithm can save 13.3% energy consumption more than that of the single model based algorithm.

Notations

N: The set of all nodes in the WSN
V_t: The vector of data collected at timestamp t
H: The set of V_t for all historical data
M_i^j: The ith model of H_i of node n^j
m: The number of subclasses
μ_i^j: The mean of the data set D_i^j
R_i^j: Data range of node n^j for model M_i^j
I: The set of (l, k) for all 2 combinations from 1 to m
CN_{lk}: The candidate node set distinguishing model l from k
CM: The candidate model set
n^j: The jth node in the WSN
v_t^j: The data collected from n^j at timestamp t
H_i: The ith subclass of H
M_g^j: The general model of node n^j
D_i^j: $\{v_t^j \mid \forall V_t \in H_i, v_t^j \in V_t\}$
σ: Coverage ratio
R_{lk}^j: $R_l^j \cap R_k^j$
T: The set of all typical nodes
MCN: The merged candidate node set
Q: The set of node IDs for a query.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by a grant from the National Natural Science Foundation of China (nos. 61100032, 61202142), Joint Funds of the Ministry of Education of China and China Mobile (no. MCM20122081), the National Key Technology R&D Program Foundation of China (no. 2013BAH44F00), and the Fundamental Research Funds for the Central Universities (nos. 2010121070, 2010121072, and 2013121030).

References

[1] E. Lee, "Cyber physical systems: design challenges," in *Proceedings of the International Symposium on Object Oriented Real-Time Distributed Computing*, 2008.

[2] J. Li, S. Cheng, H. Gao, and Z. Cai, "Approximate physical world reconstruction algorithms in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3099–3110, 2014.

[3] M. Li and Y. Liu, "Underground coal mine monitoring with wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 2, article 10, 2009.

[4] Y. Keung, B. Li, and Q. Zhang, "The intrusion detection in mobile sensor network," in *Proceedings of the 11th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '10)*, pp. 11–20, ACM, New York, NY, USA, September 2010.

[5] L. Tang, X. Yu, S. Kim, J. Han, C. Hung, and W. Peng, *Tru-Alarm: Trustworthiness Analysis of Sensor Networks in Cyber-Physical Systems*, ICDM, 2010.

[6] C. Ai, L. Guo, Z. Cai, and Y. Li, "Processing area queries in wireless sensor networks," in *Proceedings of the 5th International Conference on Mobile Ad-hoc and Sensor Networks (MSN '09)*, pp. 1–8, Fujian, China, December 2009.

[7] Y. Liu, J. Li, and H. Gao, "Enabling ε-approximate querying in sensor networks," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 169–180, 2009.

[8] B. Yu, J. Li, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," in *Proceedings of the 28th Conference on Computer Communications (INFOCOM '09)*, pp. 2159–2167, Rio de Janeiro, Brazil, April 2009.

[9] S. Cheng, J. Li, and Z. Cai, "O(ε)-Approximation to physical world by sensor networks," in *Proceedings of the 32nd IEEE Conference on Computer Communications (INFOCOM '13)*, pp. 3084–3092, Turin, Italy, April 2013.

[10] L. Guo, Y. Li, and Z. Cai, "Minimum-latency aggregation scheduling in wireless sensor network," *Journal of Combinatorial Optimization*, 2014.

[11] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," in *Proceedings of the ACM MobiCom*, 2006.

[12] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 63–75, usa, November 2003.

[13] X. Liu, Q. Huang, and Y. Zhang, "Combs, needles, haystacks: balancing push and pull for discovery in large-scale sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 122–133, November 2004.

[14] R. Sarkar, X. Zhu, and J. Gao, "Double rulings for information brokerage in sensor networks," in *Proceedings of the 12th ACM Annual International Conference on Mobile Computing and Networking (MobiCom '06)*, Los Angeles, Calif, USA, September 2006.

[15] A. Boukerche, X. Cheng, and J. Linus, "Energy-aware data-centric routing in microsensor networks," in *Proceedings of the 6th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '03)*, pp. 42–49, September 2003.

[16] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 56–67, August 2000.

[17] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 491–502, June 2003.

[18] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," in *Proceedings of the INFOCOM*, 2005.

[19] A. Deshpande, C. Guestrin, and S. R. Madden, *Modeldriven Data Acquisition in Sensor Networks*, VLDB, 2004.

[20] S. Madden, Intel lab data, http://db.csail.mit.edu/labdata/labdata.html.

[21] J. Lee, Y. Lim, Y. Chung, and M. Kim, "Data storage in sensor networks for multi-dimensional range queries," in *Proceedings of the 2nd International Conference (ICESS '05)*, 2005.

[22] L. Xie, L. Chen, D. Chen, and L. Xie, "A decentralized storage scheme for multi-dimensional range queries over sensor networks," in *Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS '09)*, pp. 166–173, 2009.

[23] H. Chen, M. Li, H. Jin, Y. Liu, and L. Ni, "MDS: efficient multi-dimensional query processing in data-centirc wsns," in *Proceedings of the 29th IEEE Real-Time Systems Symposium (RTSS '08)*, December 2008.

[24] G. Li, L. Guo, X. Gao, and M. Liao, "Bloom filter based processing algorithms for the multi-dimensional event query in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 37, pp. 323–333, 2014.

[25] J. Shi, R. Zhang, and Y. Zhang, "Secure range queries in tiered sensor networks," in *Proceedings of the (INFOCOM '09)*, 2009.

[26] J. Bu, M. Yin, D. He, F. Xia, and C. Chen, "SEF: a secure, efficient, and flexible range query scheme in two-tiered sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2011, Article ID 126407, 12 pages, 2011.