ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.5, May 2007, pp.1186−1197
DOI: 10.1360/jos181186

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种无线传感器网络中的多维 *K*-NN 查询优化算法*

赵志滨+, 于 戈, 李斌阳, 姚 兰, 杨晓春

(东北大学 信息学院,辽宁 沈阳 110004)

An Algorithm for Optimizing Multidimensional *K*-NN Queries in Wireless Sensor Networks

ZHAO Zhi-Bin+, YU Ge, LI Bin-Yang, YAO Lan, YANG Xiao-Chun

(College of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

+ Corresponding author: Phn: +86-24-83687776, Fax: +86-24-23895654, E-mail: zhaozhibin@ise.neu.edu.cn

Zhao ZB, Yu G, Li BY, Yao L, Yang XC. An algorithm for optimizing multidimensional *K*-NN queries in wireless sensor networks. *Journal of Software*, 2007,18(5):1186−1197. http://www.jos.org.cn/1000-9825/18/1186.htm

Abstract: This paper presents a filter-based algorithm called PREDICTOR for optimizing multidimensional *K*-NN queries in WSN. A filter installed at each sensor node is a node value distribution range. It is used to prevent the node from sending the data that belongs to the covering range of the filter, and so the node's energy is saved. The server keeps the historical sample data of all the nodes, and determines filters for them according to the query requirement and the samples. Three optimization strategies are proposed: (1) the method for dynamically adjusting the assignment strategies of the filter covering ranges so that a node with little chance to contribute to the final result is assigned with a larger covering range; (2) the method for sharing the filter among nodes so that the nodes with similar historical data are assigned with the same filter; (3) the method for transmitting filters in a compressed way to reduce the cost of filter updating for different *K*-NN queries. Evaluation experiment results prove the efficiency of the algorithm PREDICTOR in energy saving. Compared with the naive method, this approach can reduce the transmission volume dramatically.

Key words: WSN (wireless sensor network); *K*-NN; filter; compression

摘 要: 提出了一种基于过滤器的无线传感器网络多维 *K*-NN 查询优化算法 PREDICTOR.过滤器是设置在节点端的取值分布区间,用来屏蔽节点发送属于区间内的数据,从而节省节点能耗.在服务器端保存有各节点的历史样本数据,根据 *K*-NN 查询请求和样本数据的分布范围为节点定义过滤器.提出了 3 种优化策略:(1) 过滤器覆盖区间大小分配策略的动态调整方法,使得进入最终查询结果可能性小的节点拥有较大的覆盖区间;(2) 节点间过滤器共享方法,使得历史样本数据相近的节点使用相同的过滤器;(3) 过滤器压缩传输方法,减少为不同 *K*-NN 查询更新过滤器的代价.通过实验评价,验证了 PREDICTOR 算法的能量有效性,与朴素算法相比,极大地降低了数据传输量.

关键词: 无线传感器网络;*K*-NN;过滤器;压缩

* Supported by the National Natural Science Foundation of China under Grant Nos.60503036, 60473073 (国家自然科学基金); the Fok Ying Tong Education Foundation of China under Grant No.104027 (霍英东教育基金); the National Grand Fundamental Research 973 Program of China under Grant No.2006CB303000 (国家重点基础研究发展规划(973))

Received 2006-12-31; Accepted 2007-03-05

中图法分类号: TP393 文献标识码: A

1 Introduction

The recent advancement in wireless communication, sensor and electronics has provided us a new production of computing device: wireless sensor. It is battery-powered, small in size, multifunctional, and low in cost. Wireless Sensor Network (WSN) is composed of a large number of sensor nodes, and can be deployed into some special circumstances to carry out the tasks that are risky or high-cost for human beings. For example: scientists deploy a WSN that consists of 32 nodes on the Great Duck island off the coast Maine to monitor the habitat of seabird without any human disturbance[1]. Another instance is that an ecologist can collect data within one minute from 30 weather stations that distribute over the range of 1760 ha., which will cost 2 days at least by artificial acquisition[2].

A key challenge in wireless sensor network is how to make use of the limited power efficiently to prolong the node lifetime as long as possible[3,4]. In this direction, most research concentrates on single-object monitoring optimization, and less on multiple-object. However, in practice a sensor node may generate multidimensional data, which means that several readings from multiple observed objects are included. For example, in the system of safety monitoring of coal mine, numbers of wireless sensor nodes are scattered throughout saps, and they are collecting environmental data such as firedamp concentration, wind speed, temperature, and negative pressure. Obviously, in this application the data is multidimensional, and every item involved is important to the safety in production. At some time people plan to exploit a new working face in this coal mine, and a volume of environmental parameters that are suitable for human beings to work are given. Then, we should find out from thousands of nodes in a quick and energy-efficient way that which nodes have the physical readings nearest to the given parameters. It is a typical *K*-NN query, and the traditional *K*-NN query optimization algorithms seem invalid here because they are all based on the existing datasets. The PREDICTOR algorithm presented in this paper aims at this problem. Considering that in WSN data transmission is the predominant factor that leads to the loss of node energy[5], the main idea of the algorithm PREDICTOR is to prevent the nodes that are unlikely to enter the result set from sending data.

Initially, as soon as the wireless sensor network has been deployed, each node is asked to sense its surroundings immediately to collect some data as samples. Then, all the samples are converged onto the server. At a certain time, user initiates a multidimensional *K*-NN query, which includes a query vector with *m* attributes corresponding to the *m* monitored objects. Next, server will calculate the distance between the samples and the query vector using Euclid distance formula and determine filters for nodes accordingly. In order to guarantee a node could receive its filter correctly, we define the format of the query requirement message. On receiving the query requirement message, the node will sense its surroundings for instantaneous data. Similarly, it calculates the distance between the physical data and the query vector. In any case, the nodes that rank the front *K* in samples must feedback. Others will determine whether to report their data by comparing the distance value with filter. If the distance value goes beyond the filter, then the node will update its data. Otherwise, the node will keep silent. After the server has received all the feedbacks, it can determine the final answer to the *K*-NN query. With the steps above PREDICTOR can prevent most nodes from sending useless data. Undoubtedly, PREDICTOR may add some computing burden to server and nodes. However, we consider that the cost is acceptable because of the following two reasons: (1) the energy cost in calculation is much less than that in data transmission; (2) If a proper method is adopted, the node filters can be pushed down into the sensor network along with the query requirement message, which means that no more messages are needed for filter updating.

2 Preliminaries

The *K*-Nearest Neighbor query is a central problem in the area of similarity search[6,7]. *K*-NN query in sensor network is to find *k* sensor nodes with values nearest to the given conditions or nodes. Assume that there is a wireless sensor network *S* with *n* sensor nodes, i.e. $S=\{s_1,s_2,\ldots,s_n\}$. $|S|$ represents the number of nodes, i.e. $|S|=n$. There are *m* objects being monitored, and each object can be regarded as an attribute and recorded as *A*. According to the definition above, at any time data got by node s_i is an *m*-dimensional vector: $v_{s_i}=\langle A_{s_i}^1, A_{s_i}^2,...,A_{s_i}^m\rangle$. Suppose *q* is a user query requirement, then $v_q=\langle A_q^1, A_q^2,...,A_q^m\rangle$.

Without loss of generality, we consider that a K-NN query is to find a node set $R=\{s_1,s_2,\ldots,s_k\}$, $R\subseteq S$, $|R|=K$, where $\forall s_i\in R$ and $\forall s_j\in(S-R)$, $d(v_{s_i},v_q)\le d(v_{s_j},v_q)$. $d(v_{s_i},v_q)$ is the distance between v_{s_i} and v_q. If $v_{s_i}=\langle A_{s_i}^1, A_{s_i}^2,...,A_{s_i}^m\rangle$, $v_{s_j}=\langle A_{s_j}^1, A_{s_j}^2,...,A_{s_j}^m\rangle$, then $d(v_{s_i},v_{s_j})$ can be calculated as

$$d(v_{si},v_{sj})=\left(\sum_{p=1}^{m}\omega_p(A_{si}^p-A_{sj}^p)^2\right)^{1/2} \tag{1}$$

ω_p is the weight factor that the attribute A^p owns. It is saved symmetrically on both server and nodes. The value of ω_p is determined by users according to the physical meanings. In addition, nodes should be informed of any changes of ω_p in time for its influence on filter setting.

Furthermore, we can calculate the distance between two matrices. Given two matrices: $X_{i\times j}=[x_{ij}]$ and $Y_{i\times j}=[y_{ij}]$, then $d(X,Y)$ makes of a new matrix $D_{i\times j}$. Elements in $D_{i\times j}$ are the distance values generated by the two elements with the same position in $X_{i\times j}$ and $Y_{i\times j}$, i.e.

$$D_{i\times j}=[d(x_{ij},y_{ij})] \tag{2}$$

3 The PREDICTOR Algorithm

There are five phrases in PREDICTOR: initialization phrase, filter generation phrase, filter update phrase, data update phrase and result generation phrase. For the simplicity of discussion, we assume that the server messages could reach node in a single-hop way, while those from node must be delivered to server in a multi-hop way.

3.1 Initialization phrase

As mentioned above, as soon as the wireless sensor network is built, PREDICTOR will initiate sampling from across the network. Let *l* denote the number of sampling times, and t_i denote the interval for sampling. After the time of $(l-1)*t_i$, we can get the samples space $T_{l\times n}$,

$$T_{l\times n}=\begin{bmatrix} v_{1,s_1} & v_{1,s_2} & \cdots & v_{1,s_n} \\ v_{2,s_1} & v_{2,s_2} & \cdots & v_{2,s_n} \\ \cdots & \cdots & \cdots & \cdots \\ v_{l,s_1} & v_{l,s_2} & \cdots & v_{l,s_n} \end{bmatrix}.$$

In order to guarantee that $T_{l\times n}$ can reflect the current situation precisely and timely, we can keep incremental maintenance on $T_{l\times n}$. An optional maintenance scheme is that we can collect samples from across the network with a fixed frequency and use time window to select data for $T_{l\times n}$. The parameters, *l*, t_i and the sampling frequency, are strongly related to the practical applications and the energy left on the node. Their values should be adjustable at the run time. In this paper we suppose that the samples in $T_{l\times n}$ are always the newest and efficient. The server keeps the global samples, and each sensor node keeps its own locally, as depicted in Fig.1.

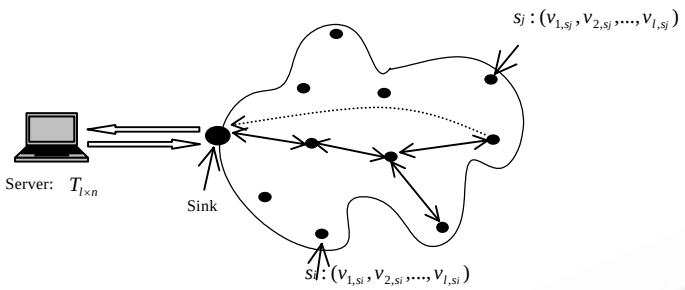


Fig.1 Samples are collected and kept on both server and nodes

3.2 Filter generation phrase

After initialization, PREDICTOR is ready for accomplishing the task of *K*-NN queries. At a certain time, there arrives a new query with vector $v_q=\langle A_q^1,A_q^2,...,A_q^m\rangle$. First, server should translate v_q into a query matrix $Q_{l\times n}$ as

$$Q_{l\times n}=\begin{bmatrix} v_q & v_q & \cdots & v_q \\ v_q & v_q & \cdots & v_q \\ \cdots & \cdots & \cdots & \cdots \\ v_q & v_q & \cdots & v_q \end{bmatrix}.$$

Then, $d(T_{l\times n},Q_{l\times n})$, the distance between the matrices $Q_{l\times n}$ and $T_{l\times n}$, is calculated as Eq.(3)

$$d(T_{l\times n},Q_{l\times n})=D_{l\times n}=\begin{bmatrix} d(v_{1,s_1},v_q) & d(v_{1,s_2},v_q) & \cdots & d(v_{1,s_n},v_q) \\ d(v_{2,s_1},v_q) & d(v_{2,s_2},v_q) & \cdots & d(v_{2,s_n},v_q) \\ \cdots & \cdots & \cdots & \cdots \\ d(v_{l,s_1},v_q) & d(v_{l,s_2},v_q) & \cdots & d(v_{l,s_n},v_q) \end{bmatrix} \tag{3}$$

The element in $d(T_{l\times n},Q_{l\times n})$ denotes the distance from each sample to the query vector. The row in $d(T_{l\times n},Q_{l\times n})$ is the set of distance values of all nodes for sampling once. The column in $d(T_{l\times n},Q_{l\times n})$ is the set of distance values of any one node in l samples.

Next, PREDICTOR will determine filters according to $D_{l\times n}$. In order to eliminate the chanciness, we sum up the values of distance in l samples. Let Γ be the summation vector, and then it can be described as $\Gamma=\left(\sum_{j=1}^{l}d(v_{j,s_1},v_q),\sum_{j=1}^{l}d(v_{j,s_2},v_q),...,\sum_{j=1}^{l}d(v_{j,s_n},v_q)\right)$. A filter is in fact a range which most instantaneous values of a node fall into. If the physical environment varies smoothly and the sample space is the newest, then Γ can be used to calculate the filters for its statistical regularity.

We sort the elements in Γ, and for simplicity, the sorted result can be recorded as

$$\Gamma'=\left(\sum_{j=1}^{l}d(v_{j,s_1},v_q)<...<\sum_{j=1}^{l}d(v_{j,s_i},v_q)<...<\sum_{j=1}^{l}d(v_{j,s_n},v_q)\right).$$

Figure 2 shows the relative element positions in value domain.

There are several ways to determine a filter for a node, and theoretically, each node can be assigned with a unique filter. For example, we can use the median of the elements in neighbor in Γ' as the boundaries of filter.

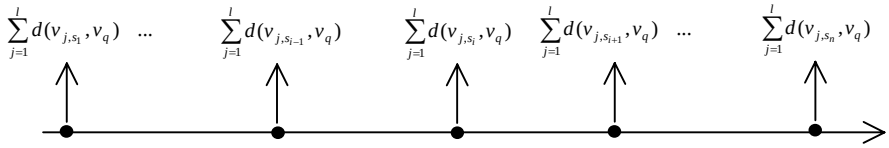


Fig.2 The relative positions of the elements in Γ'

For any node s_i, $s_i \in S$, $s_i \neq s_1$ and $s_i \neq s_n$, let ub_i denote the upper boundary of filter for node s_i, and lb_i denote the lower boundary of filter for node s_i. Then, ub_i and lb_i are

$$ub_i = \frac{\left(\sum_{j=1}^{l} d(v_{j,s_i}, v_q) + \sum_{j=1}^{l} d(v_{j,s_{i+1}}, v_q)\right)}{2} \tag{4}$$

$$lb_i = \frac{\left(\sum_{j=1}^{l} d(v_{j,s_{i-1}}, v_q) + \sum_{j=1}^{l} d(v_{j,s_i}, v_q)\right)}{2} \tag{5}$$

For node s_1, lb_1 is

$$lb_1 = \frac{\sum_{j=1}^{l} d(v_{j,s_1}, v_q)}{2} \tag{6}$$

For node s_n, ub_n is

$$ub_n = 2\sum_{j=1}^{l} d(v_{j,s_n}, v_q) - lb_n \tag{7}$$

Unfortunately, the method above has two serious disadvantages. Firstly, to receive filter will consume node energy, which makes it impossible for us to inform each node of its filter. Secondly, it is obvious that the more backward a node lies in Γ', the little chance it has to enter the final result set. However, the method above is unaware of the node importance level to the query, and so it will lead to some useless transmission.

In fact, the whole domain in Fig.2 can be divided into two ranges: $\left(\sum_{j=1}^{l} d(v_{j,s_1}, v_q), \sum_{j=1}^{l} d(v_{j,s_k}, v_q)\right]$ and $\left(\sum_{j=1}^{l} d(v_{j,s_k}, v_q), \sum_{j=1}^{l} d(v_{j,s_n}, v_q)\right]$. Then, all the nodes can be divided into two groups according to these two ranges. The first group consists of the front k nodes in Γ', and other nodes compose the second group. Obviously, the first group seems more likely to be the final result than the second group. Furthermore, nodes in the second group have different importance levels. Those with summations close to $\sum_{j=1}^{l} d(v_{j,s_k}, v_q)$ are more likely to be the final result.

Therefore, $\left(\sum_{j=1}^{l} d(v_{j,s_1}, v_q), \sum_{j=1}^{l} d(v_{j,s_k}, v_q)\right]$ can be assigned as one shared filter for the nodes in the first group, and $\left(\sum_{j=1}^{l} d(v_{j,s_k}, v_q), \sum_{j=1}^{l} d(v_{j,s_n}, v_q)\right]$ should be divided into several filters for the nodes in the second group. For that sake, we introduce λ and α into PREDICTOR. λ is a parameter of distance, $\lambda \in \left[0, \sum_{j=1}^{l} d(v_{j,s_n}, v_q) - \sum_{j=1}^{l} d(v_{j,s_k}, v_q)\right]$. α represents an adjusting coefficients vector $(\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots)$. Conclusively, the filters can be calculated as

$$\begin{cases} filter_0 = \left(\sum_{j=1}^{l} d(v_{j,s1}, v_q), \sum_{j=1}^{l} d(v_{j,sk}, v_q) \right] \\ filter_i = \left(\left(\sum_{j=1}^{l} d(v_{j,sk}, v_q) + \lambda * \sum_{j=0}^{i-1} \alpha_j \right), \left(\sum_{j=1}^{l} d(v_{j,sk}, v_q) + \lambda * \sum_{j=0}^{i} \alpha_j \right) \right], \alpha_0 = 0 \end{cases} \tag{8}$$

In Eq.(8), the maximum of i is the minimum of the natural numbers that make the inequality $\left(\sum_{j=1}^{l} d(v_{j,s_k}, v_q) + \lambda * \sum_{j=0}^{i} \alpha_j \right) \geq \sum_{j=1}^{l} d(v_{j,s_n}, v_q)$ true.

We can see from the Eq.(8) that, given the same λ, if $\alpha_1=\alpha_2=\ldots=\alpha_i=\ldots$, then the range $\left(\sum_{j=1}^{l} d(v_{j,s_k}, v_q), \sum_{j=1}^{l} d(v_{j,s_n}, v_q) \right]$ is divided equally. If $\alpha_1\neq\alpha_2\neq\ldots\neq\alpha_i\neq\ldots$, then the range is divided unequally. Especially, if $\alpha_1<\alpha_2<\ldots<\alpha_i<\ldots$, then the filters will change larger and larger. The larger filter a node owns, the less data it is inclined to report.

Up to here, we have determined the filters for all the nodes. The server needs to save the corresponding relationship between the node and its filter. For that sake, we assign node boxes for each of the filters. Figure 3 shows the scenario of equant filters and their corresponding node boxes.

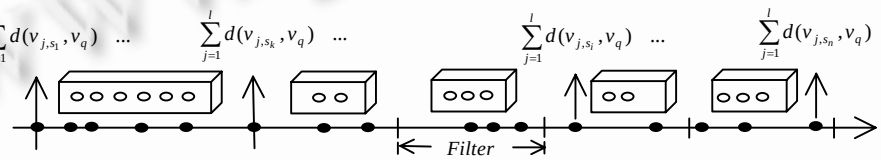


Fig.3 The equant filters and their node boxes

3.3 Filter update phrase

Filters, along with the query vector, are enveloped in the query requirement message. In order to guarantee that a node could parse its filter correctly, we should define the query requirement message format first. There are four parts in the query message: v_q, $\sum_{j=1}^{l} d(v_{j,s_k}, v_q)$, λ and α. α represents the types of the adjusting coefficient vectors, see Table 1. The mapping relationships are saved not only on the server but also on all nodes.

Table 1 The values of α

α	The vector of adjusting coefficients
1	(1,1,1,…,1,…)
2	(1,2,3,…,*i*,…)
3	(1,2,4,…,2*i*,…),*i*=1,2,3,…
4	(F_1,F_2,…,F_n,…),F_1=F_2=1,F_n=F_{n-1}+F_{n-2}

3.4 Data update phrase

When the sensor node s_i receives the query requirement message, it will extract its filter first, and then determine whether to report its readings by the following steps:

Step 1. Calculate the summation of distance value between its samples and v_q, i.e. $\sum_{j=1}^{l} d(v_{j,s_i}, v_q)$;

Step 2. Reduce all filters according to Eq.(8);

Step 3. Determine whether to update its readings or not. If $\sum_{j=1}^{l} d(v_{j,s_i}, v_q) < \sum_{j=1}^{l} d(v_{j,s_k}, v_q)$, then s_i must send its

data back. Otherwise, the node s_i should judge which filter the value $\sum_{j=1}^{l} d(v_{j,s_i}, v_q)$ belongs to. If $\sum_{j=1}^{l} d(v_{j,s_i}, v_q) \in \left(\left(\sum_{j=1}^{l} d(v_{j,s_k}, v_q) + \lambda * \sum_{j=0}^{i-1} \alpha_j \right), \left(\sum_{j=1}^{l} d(v_{j,s_k}, v_q) + \lambda * \sum_{j=0}^{j} \alpha_j \right) \right)$, then $filter_i$ is the right one. Next, we should compare $d(v_{s_i}, v_q) \times l$ with $filter_i$. If $d(v_{s_i}, v_q) \times l \in filter_i$, then s_i keeps silent. If not, s_i sends $\langle s_i, v_{s_i} \rangle$ back.

3.5 Result generation phrase

After sending out the query requirement message, the server will come into the state of waiting. Within the maximum time delay, the server could receive some feedbacks from some of the nodes. For those nodes that have data reported, the corresponding relationship between node and node box should be adjusted. In other words, some nodes should be relocated.

For example, if the server receives a message $\langle s_i, v_{s_i} \rangle$ from the node s_i, then it should be relocated. In other words, s_i should be moved out of the original node box and put into the new one that the value of $d(v_{s_i}, v_q) \times l$ corresponds to. This procedure is depicted in Fig.4.

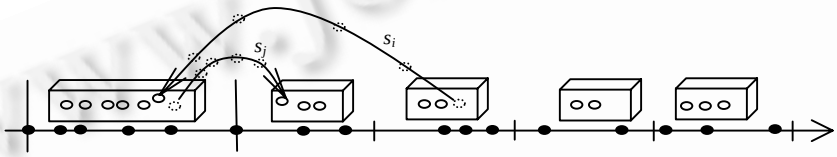


Fig.4 The procedure of node relocation

After the node relocation, a node box may get or lose some nodes. In terms of the distance to v_q, any two nodes that lie in the different box are in order because node boxes are in order, while those that lie in the same node box are out of order. Since the answer of *K*-NN query only has *k* nodes, which is the minority of the whole, it is unnecessary to sort all the nodes now. It will be postponed until we have determined the candidate result set.

Finally, It starts to search the node box in turn from the lowest one and pick out the nodes inside into *R* until $|R| \geq k$. When the searching procedure stops, there are two situations. If $|R|=k$, then *R* is the final result set. If $|R|>k$, then we needs to visit the nodes that are in *R* and did not give any feedbacks in data update phrase so as to tell the frontal *k* nodes. Though the visiting is directed and refers to little nodes, we hope that it can be avoided. Thus, we make a small improvement on PREDICTOR.

We define *P* as the once-query success rate of the algorithm PREDICTOR. *P* is the ratio of the number of the nodes that meet the situation of $d(v_{s_i}, v_q) \times l < \sum_{j=1}^{l} d(v_{j,s_k}, v_q)$ to *k*. If *p*=100%, then the procedure of directed visiting can be avoided. Deep analysis shows that *P* is affected by *k*. Then we can increase *P* by increasing *k* properly. The relationship between *P* and *k* is described in Experiment 4 in detail.

3.6 Benefit and cost analysis

In this subsection, we will build the models to evaluate the benefit and the cost that PREDICTOR brings to a node. Let *SC* denote the energy cost (mJ) that is used for sending one byte, and *RC* denote the energy cost for receiving one byte. *byte*(*x*) stands for the total bytes that *x* occupies.

(1) The total cost if no filter is used.

$$totalcost_1 = byte(v_q) \times RC + byte(v_{s_i}) \times SC \tag{9}$$

(2) The total cost if a filter is used.

$$totalcost_2=\left(byte(v_q)+byte\left(\sum_{j=1}^{l}d(v_{j,s_k},v_q)\right)+byte(\lambda)+byte(\alpha)\right)\times RC+byte(v_{s_i})\times SC \tag{10}$$

In Eq.(10), If filter prevents s_i from sending v_{s_i}, then $byte(v_{s_i})\times SC=0$. The cost for filter update is $\left(byte\left(\sum_{j=1}^{l}d(v_{j,s_k},v_q)\right)+byte(\lambda)+byte(\alpha)\right)\times RC$.

(3) The benefit that a node gains in energy saving.

$$benefit=totalcost_1-totalcost_2 \tag{11}$$

4 Performance Evaluation

For evaluating the performance of PREDICTOR, we made four simulative experiments which are based on two real traces: TAO project dataset and Intel Lab dataset. TAO (tropical atmosphere ocean) project is set up for improved detection, understanding and prediction of El Niño and La Niña[8]. It consists of about 70 moored ocean buoys in the Tropical Pacific Ocean. Instruments embedded in each buoy can measure over 20 physical phenomena at the same time. We selected five of them in our experiment as attributes. They are air temperature, dynamic height, isotherm depth, relative humidity and surface met. The second is Intel Lab data[9]. It comes from 54 nodes deployed in the Intel Berkeley Research lab. We selected three attributes in our experiment: humidity, temperature and light. Comparatively, TAO data is more dispersive in distribution than Intel Lab data.

Experiment 1. Comparison on the number of nodes that send data for two basic methods of filter determination.

We adopt two basic methods to determine the value of λ. The first is that the nodes except for the front k ones share one big filter, i.e. $filter_1=...=filter_i=\lambda_1=\left(\sum_{j=1}^{l}d(v_{j,s_n},v_q)-\sum_{j=1}^{l}d(v_{j,s_k},v_q)\right)$. The second method is to divide the range $\left(\sum_{j=1}^{l}d(v_{j,s_k},v_q),\sum_{j=1}^{l}d(v_{j,s_n},v_q)\right)$ equally, and each part serves as an individual filter, i.e. $filter_1=...=filter_i=\lambda_2=\left(\sum_{j=1}^{l}d(v_{j,s_n},v_q)-\sum_{j=1}^{l}d(v_{j,s_k},v_q)\right)\Big/(n-k)$. The experiment result is depicted in Fig.5.

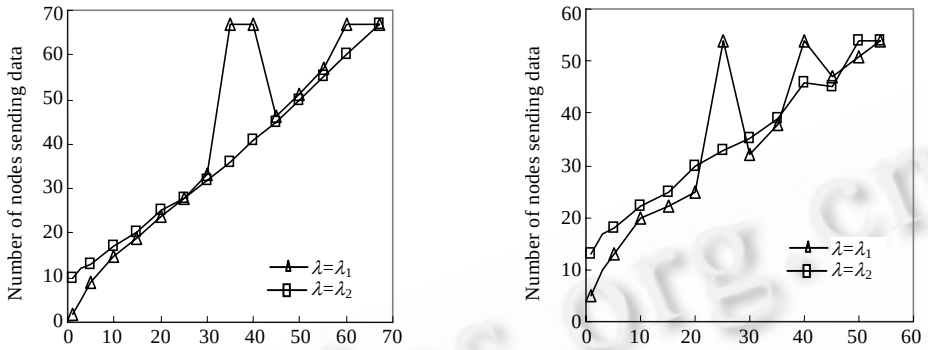
Figure 5(a) is the result based on TAO project data, and Fig.5(b) is on Intel Lab data. For both Fig.5(a) and Fig.5(b), we can see that when $k\leq 20$, the first method sends less messages than the second one. This is because there are more nodes sharing one big filter, and fewer nodes could violate it. However, with the increment of the value of k, more nodes are asked to report their data, and at the same time, the shortening of the shared filter makes more nodes violate their filters. Especially in the first method, if the number of the nodes with $d(v_{s_i},v_q)\times l<\sum_{j=1}^{l}d(v_{j,s_k},v_q)$ is less than k, then the directed visiting is needed, which means that all the nodes have been disturbed. In other words, the second method is more stable in the column of communication. This is more obvious in the application of niche monitoring because the data is relatively concentrative.

Experiment 2. How does the continuous changes of λ affect on the column of communication?

In the second experiment, we reviewed the relationship between λ and the number of the disturbed nodes for different k. Theoretically, the value of λ will affect the size of filters, and thus affect the performance of the algorithm PREDICTOR. The experiment result is in Fig.6(a) and Fig.6(b). In order to make it clear, we select arithmetical progression as the vector of adjusting coefficients. Moreover, we set k to 1, 5, 10 and 20, which are more common in *K*-NN queries.

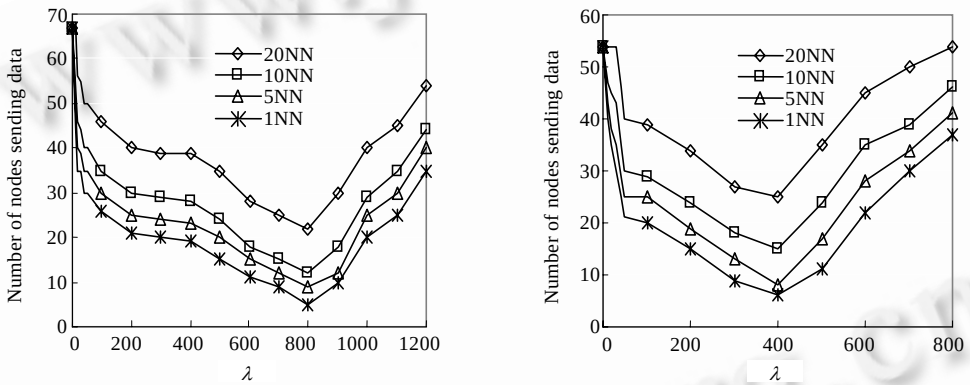
From Fig.6, we can see that whether the λ is too great or too small will both lead to the increment of the

number of the nodes that are asked to report data. When λ=0 there are no filters and all the nodes are asked to send their data. The reasonable value of λ depends on the specific applications, just as that in TAO project (see Fig.6(a)), when λ=800, the transmission column is the minimum, while in Intel Lab data, it goes to 400.



(a) TAO, α=1 (b) Intel Lab, α=1

Fig.5 Comparison of two basic methods of filter determination



(a) TAO, α=2 (b) Intel Lab, α=2

Fig.6 Effect of λ on the number of the nodes sending data

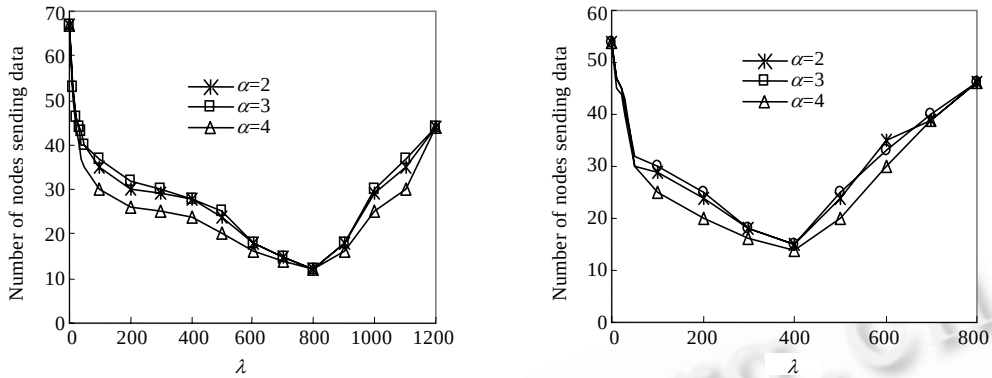
Experiment 3. How does the different vectors of α affect on the column of communication?

In the third experiment, the value of k is fixed to 10, and we choose three different values of α to initialize filters, which means that three different vectors of adjusting coefficient are used. They are arithmetical progression (1,2,3,…,i,…), geometric progression (1,2,4,…,2i,…), and Fibonacci. Among the three arrays, filters generated with the geometric progression increase most quickly, while those generated with the arithmetical progression increase most slowly. With the help of the tilted filters, the backer a node lies in Γ', the less it violates its filer, and the less possibly it sends data. The result is shown in Fig.7.

The experiment in Fig.7 shows that among the three arrays, Fibonacci is more efficient, because it makes the size of filters increase smoothly. If it increases too fast as with geometric progression, directed visiting may be needed, which will lead to some extra transmission. If it increases too slowly, more nodes will violate their filters, which will lead to more transmission, too. Fibonacci seems more suitable to the data distribution than the other two.

Experiment 4. How does it influence on the once-query success rate if k is increased properly to k'?

In this experiment, we increase k to k', i.e. $k'=\lceil \beta * k \rceil$, to avoid the procedure of directed visiting. As mentioned in Section 3, P denotes the once-query success rate. For eliminating the chanciness, we repeat the experiment with ten groups of testing data, and select the average of P. Table 2 shows the result on TAO data.



(a) TAO, k=10 (b) Intel Lab, k=10

Fig.7 Effect of different values of α on the number of the nodes sending data

Table 2 Comparison on P between k and k'

k \ k' / P	k'=⌈1k⌉	k'=⌈1.2k⌉	k'=⌈1.5k⌉	k'=⌈2k⌉	k'=⌈3k⌉
1	85%	98.5%	98.5%	98.5%	100%
3	83.5%	90.5%	95%	97.5%	100%
5	82%	87%	93.5%	96.7%	100%
10	81.5%	85%	92%	95%	99.8%

5 Related Work

There is much work in the research on optimizing *K*-NN query. In 1984, Guttman, *et al*. proposed R-tree structure[10]. The structure of tree can help to speed up *K*-NN query. Other similar algorithms include R*-tree[11], K-D-B-tree[12], SS-tree[13], X-tree[14] and BBD-tree[15]. The main idea of these algorithms is to reduce iteration times at the cost of space. Aiming at the multidimensional *K*-NN query, Cui, *et al*. presented Δ-tree structure[16]. In Refs.[17,18], the query vector is regarded as a point in multidimensional space, and further as a centre of circle. Then, a fixed—Or unfixed-length radius is chosen to probe around until the final answer is gotten. All the methods above are based on the existing dataset and do not take data acquisition into consideration. However, data acquisition is a significant problem in WSN, because we usually cannot obtain the whole of the data. Therefore, it needs to combine query optimization with data acquisition optimization in WSN applications.

WSN is a typically distributed environment. Brian Babcock formalized the definition of distributed monitoring in multiple data flow environment in Ref.[19]. As for data acquisition in WSN, there are three types of methods in the main: model-driven data acquisition, sampling-based data acquisition and filter-based data acquisition. Deshpande, *et al*. used the model-driven data acquisition method in Ref.[20]. The basic idea is to predict the readings of nodes according to a certain probability distribution, such as independent Gaussians, and visit the target nodes. Chu, *et al*.[21] proposed the Ken architecture for continuous monitoring. It uses a joint probability model to predict data for temporal suppression and a disjoint-clique model for spatial suppression. The main disadvantages of the model-driven method are that they are expensive in model selection and update. The sampling-based data acquisition method is to construct visiting models dynamically instead of using some fixed probability model. It is more flexible and adaptive to the changes of surroundings. Silberstein, *et al*.[22] succeeded in using it to answer Top-k queries. The sampling-based approach is reliable in the situation where 1) samples can be renewed in time, and 2) the environment changes smoothly. The last is the filter-based data acquisition method. Wu, *et al*.[23] proposed the algorithm FILA to answer Top-*k* query in WSN. Though FILA is similar to PREDICTOR in that they both use filters to prevent nodes from sending useless data, they are quite different in many other ways: Firstly,

FILA is designed for the Top-*k* query which is stable, whereas PREDICTOR is for the multidimensional *K*-NN queries in which case the query points may change every time. Secondly, FILA only gives one method for filter setting, while PREDICTOR presents several strategies for that. FILA fixes the filter number to *k*+1, i.e. non-top-k nodes share one big filter. This may lead to a whole network disturbance as in the scenario 1 in Ref.[23]. In PREDICTOR, the filter number is adjustable. Filters can be merged or divided according to the parameters λ and α. Therefore, filters are more precise. Finally, PREDICTOR gives a novel method for filter updating. Silberstein A. suggested the algorithm HAT in Ref.[24]. HAT uses threshold to suppress node from reporting their values with little difference from its previously reported. Though the Ken architecture and the HAT algorithm can both be used to answer *K*-NN query, they are much different from the approach presented in this paper. The significant one is that PREDICTOR is designed for real-time response for snapshot query, while the other two are for continuous monitoring.

6 Conclusion and Future Work

In this paper, we have introduced an energy-efficient multidimensional *K*-NN query optimization algorithm in WSN, which is called PREDICTOR. Its most significant character is to combine query optimization with data acquisition. It uses the distance between the samples and the query vector to determine filters for nodes. With the help of the filters the algorithm could prevent most nodes from sending useless data. Simulative experiment results show that it is more energy-efficient than the naive approach.

As for future, we are interested in several extensions in PREDICTOR. 1) Support of the hierarchical topology. We acknowledge that nodes communicate to the root by a relatively-fixed route in a multiple-hop way. The advanced PREDICTOR should empower middle nodes to filter the data passing by, rather than act as transceivers. The transmission volume will further decrease if the middle nodes abandon some useless data; 2) Detection of node failure or link failure. When the server does not hear from a node for a long time, PREDICTOR needs to distinguish between node or link failure and the case that the data is always within its filter. We plan to add node registration mechanism into PREDICTOR to help making this distinction.

References:

[1] Mainwaring A, Polastre J, Szewczyk R, Culler D, Anderson J. Wireless sensor networks for habitat monitoring. In: Proc. of the 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications. Atlanta: ACM Press, 2002. 88−97.

[2] SMER sensor network advances bioscience discoveries. http://fs.sdsu.edu/kf/news/view/index.php?id=0021

[3] Li JZ, Li JB, Shi SF. Concepts, issues and advance of sensor networks and data management of sensor networks. Journal of Software, 2003,14(10):1717−1727 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1717.htm

[4] Cui L, Ju HL, Miao Y. Overview of wireless sensor networks. Journal of Computer Research and Development, 2005,42(1): 163−174 (In Chinese with English abstract).

[5] Crossbow. MPR (mote processor radio board) and MIB (mote interface/programming board) user's manual. Crossbow Technology Inc., 2003. http:// www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf

[6] Roussopoulos N, Kelley S, Vincent F. Nearest neighbor queries. In: Carey MJ, Schneider DA, eds. Proc. of the '95 ACM SIGMOD Int'l Conf. on Management of Data. San Jose: ACM Press, 1995. 71−79.

[7] Song ZX, Roussopoulos N. *K*-Nearest neighbor search for moving query point. In: Jensen CS, Schneider M, Seeger B, Tsotras VJ, eds. Proc. of the 7th Int'l Symp. on Spatial and Temporal Databases. Berlin: Springer-Verlag, 2001. 79−96.

[8] TAO (tropical atmosphere ocean) project. http://www.pmel.noaa.gov/tao/index.shtml

[9] Intel Berkeley Research Lab. http://berkeley.intel-research.net/labdata

[10] Guttman A. R-Trees: A dynamic index structure for spatial searching. In: Yormark B, ed. Proc. of the '84 ACM SIGMOD Int'l Conf. on Management of Data. Boston: ACM Press, 1984. 47−57.

[11] Beckmann N, Kriegel HP, Schneider R, Seeger B. The *R**-tree: An efficient and robust access method for points and rectangles. In: Garcia-Molina H, Jagadish HV, eds. Proc. of the '90 ACM SIGMOD Int'l Conf. on Management of Data. Atlantic City: ACM Press, 1990. 322−331.

[12] Robinson JT. The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In: Edmund Lien Y, ed. Proc. of the ’81 ACM SIGMOD Int’l Conf. on Management of Data. Ann Arbor: ACM Press, 1981. 10−18.

[13] White DA, Jain R. Similarity indexing with the SS-tree. In: Su SYW, ed. Proc. of the 12th Int’l Conf. on Data Engineering. New Orleans: IEEE Computer Society, 1996. 516−523.

[14] Berchtold S, Keim DA, Kriegel H. The X-tree: An index structure for high-dimensional data. In: Vijayaraman TM, Buchmann AP, Mohan C, Sarda NL, eds. Proc. of the 22nd VLDB Conf. Bombay: Morgan Kaufmann Publishers, 1996. 28−30.

[15] Arya S, Mount D, Netanyahu NS, Silverman R, Wu AY. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. Journal of the ACM, 1998,45(6):891−923.

[16] Cui B, Ooi BC, Su J, Tan KL. Contorting high dimensional data for efficient main memory K-NN processing. In: Alon Y, Halevy, *et al*., eds. Proc. of the 2003 ACM SIGMOD Int’l Conf. on Management of Data. San Diego: ACM Press, 2003. 479−490.

[17] Zhang R, Kalnis P, Ooi BC, Tan KL. Generalized multidimensional data mapping and query processing. ACM Trans. on Database Systems, 2005,30(3):661−697.

[18] Shi QX, Nickerson BG. Decreasing radius k-nearest neighbor search using mapping-based indexing schemes. 2006. http://www.cs.unb.ca/tech-reports/reportpage2006.shtml

[19] Babcock B, Olston C. Distributed top-k monitoring. In: Halevy AY, *et al*., eds. Proc. of the 2003 ACM SIGMOD Int’l Conf. on Management of Data. San Diego: ACM Press, 2003. 28−39.

[20] Deshpande A, Guestrin C, Madden S, Hellerstein J, Hong W. Model-Driven data acquisition in sensor networks. In: Mario A, *et al*., eds. Proc. of the 2004 Int’l Conf. on Very Large Databases. Toronto: Morgan Kaufmann Publishers, 2004. 588−599.

[21] Chu D, Deshpande A, Hellerstein JM, Hong W. Approximate data collection in sensor networks using probabilistic models. In: Liu L, Reuter A, *et al*., eds. Proc. of the 2006 Int’l Conf. on Data Engineering. Atlanta: IEEE Computer Society, 2006.

[22] Silberstein A, Braynard R, Ellis C, Munagala K. A sampling-based approach to optimizing top-*k* queries in sensor network. In: Liu L, Reuter A, *et al*., eds. Proc. of the 2006 Int’l Conf. on Data Engineering. Atlanta: IEEE Computer Society, 2006.

[23] Wu MJ, Tang XY. Monitoring top-*k* query in wireless sensor networks. In: Liu L, Reuter A, *et al*., eds. Proc. of the 2006 Int’l Conf. on Data Engineering. Atlanta: IEEE Computer Society, 2006.

[24] Silberstein A, Munagala K, Yang J. Energy-Efficient monitoring of extreme values in sensor networks. In: Chaudhuri S, Hristidis V, *et al*., eds. Proc. of the 2006 ACM SIGMOD Int’l Conf. on Management of Data. Chicago: ACM Press, 2006. 169−180.

附中文参考文献:

[3] 李建中,李金宝,石胜飞.传感器网络及其数据管理的概念、问题与进展.软件学报,2003,14(10):1717−1727. http://www.jos.org.cn/1000-9825/14/1717.htm

[4] 崔莉,鞠海玲,苗勇.无线传感器网络研究进展.计算机研究与发展,2005,42(1):163−174.

ZHAO Zhi-Bin was born in 1975. He is a Ph.D. candidate at Institute of Computer Software and Theory, Northeastern University and a CCF member. His current research areas are distributed database and wireless sensor network.

YU Ge was born in 1962. He is a professor and doctoral supervisor at the Institute of Computer Software and Theory, Northeastern University and a CCF senior member. His researches areas are database theory and application, data stream management.

LI Bin-Yang was born in 1982. He is a master candidate at the Institute of Computer Software and Theory, Northeastern University. His current research area is wireless sensor network.

YAO Lan was born in 1977. She is a Ph.D. candidate at the Institute of Computer Software and Theory, Northeastern University and a CCF student member. Her current research areas are computer networks and wireless sensor network.

YANG Xiao-Chun was born in 1973. She is an associate professor at the Institute of Computer Software and Theory, Northeastern University and a CCF senior member. Her current research areas are distributed data management and wireless sensor network.