A Pattern-based Query Strategy in Wireless Sensor Network

Yanhong Ding[1], Tie Qiu[1], He Jiang[1*] and Weifeng Sun[1]
[1] School of Software, Dalian University of Technology
Dalian, 116620 - China
[e-mail: dyh2010@gmail.com; qiutie@dlut.edu.cn; jianghe@dlut.edu.cn]
*Corresponding author: He Jiang

Received January 6, 2012; revised March 14, 2012; revised May 1, 2012; accepted June 5, 2012; published June 25, 2012

Abstract

Pattern-based query processing has not attracted much attention in wireless sensor network though its counterpart has been studied extensively in data stream. The methods used for data stream usually consume large memory and much energy. This conflicts with the fact that wireless sensor networks are heavily constrained by their hardware resources. In this paper, we use piece wise representation to represent sensor nodes' collected data to save sensor nodes' memory and to reduce the energy consumption for query. After getting data stream's and patterns' approximated line segments, we record each line's slope. We do similar matching on slope sequences. We compute the dynamic time warping distance between slope sequences. If the distance is less than user defined threshold, we say that the subsequence is similar to the pattern. We do experiments on STM32W108 processor to evaluate our strategy's performance compared with naïve method. The results show that our strategy's matching precision is less than that of naïve method, but our method's energy consumption is much better than that of naïve approach. The strategy proposed in this paper can be used in wireless sensor network to process pattern-based queries.

Keywords: Pattern query, wireless sensor network, piecewise linear representation

This work is supported by National Natural Science Foundation of P.R. China (Grant No. 61175062 and 61103233) and the Fundamental Research Funds for the Central Universities.

http://dx.doi.org/10.3837/tiis.2012.06.004

1. Introduction

Wireless sensor network (WSN) is one of the most effective techniques for monitoring and controlling physical environments from remote locations with better accuracy [1][2]. WSNs are consisted of some energy-constrained sensor nodes and a few sink nodes. Sensor nodes collect data from the monitored area and transmit the data to sink nodes for future process. WSNs are widely used in many areas, such as wildlife monitoring [3] and clinical monitoring [4]. Despite of the fortunes that WSNs bring to industry, people show interests in getting useful knowledge from these networks. One of the most significant queries posted by users is based on data sequences collected by sensor nodes. For instance, Nodes that can sense temperature and humidity can show whether a forest is on fire; comparably, a clinical sensor can sense whether a patient is suffering from a heart attack. In this paper, we focus on how to search for sensor nodes with user specified patterns with low energy cost.

Unlike other kinds of queries in WSNs, such as MAX (MIN) value queries or IF HAS queries which only pay attention to single values and omit the correlations among sensor readings, sequence pattern queries can offer meaningful and abundant information, which is called query by example [5]: return all sensor nodes that have observed a particular pattern issued by users. Users may be interested in sensor nodes that have special patterns. These sensor nodes can be found through example queries. For example, one can ask to find all sensor nodes observing a temperature pattern "31.06, 34.81, 36.62, 38.31, 41.87", which may indicate that the sensor node may be on fire quickly. Another significant application resides in wide life monitoring. Wide animals can be monitored by wearable sensors. Query by example can find animals' abnormal actions. The pattern query can be defined as follow:

Definition 1: Given a data stream collected by a sensor node, $X = \{x_1, x_2, ..., x_n\}$, a query pattern $P = \{p_1, p_2, ..., p_m\}$ and a user-specified threshold ε, a pattern query operation retrieves all X's subsequences, $x_{i,j} = \{x_i, x_{i+1}, ..., x_j\}$, whose distance to P is less than ε, where $1 \le i \le j \le n$.

A normal approach to handle such queries is the centralized scheme. All sensor nodes send their data to sink node and the sink node stores all data in a central database where queries are processed. Transporting data consumes much more energy than processing data [6]. The centralized scheme quickly depletes batteries. Sensor networks are distributed environments producing multiple streams of data. We can consider the network as a distributed database where we are interested in query and mining. Above defined query has been widely studied in data streams and time series [7][8][9][10]. But data stream processing in wireless sensor network faces a number of research issues and challenges [11][12]: (1) Energy efficiency. Sensor nodes operate on low-power batteries. Data stream techniques in sensor networks should be energy-aware. (2) Processing power. Sensor nodes have limited processing power to perform advanced computational tasks. Limited processor speed has been an obstacle for large deployment of sensor network. (3) Memory size. A wide range of traditional processing techniques require data to be resident in memory for processing. Limited availability of memory of sensor nodes makes these techniques unsuitable for sensor network applications. It is necessary to provide an energy efficient and high performance strategy to process example queries in WSNs. Some approximation based methods [13][14] are proposed for data collection in WSNs, which are not expected to answer queries by example efficiently.

In this paper, we propose a novel approach to efficiently perform the query as defined in Definition 1 in WSNs. We treat each sensor node as a database, which stores collected data in

the memory for further process and does the matching process on each sensor node. As sensor nodes are heavily constrained by their memory size, it is impossible to store all raw data in the memory, we approximate the data sensed by sensor nodes based on Piecewise Linear Representation (PLR). We abstract the data stream to its basic shape descriptor. After getting two raw data, we do line fitting based on the two data and get a line function. We use the function to predict the coming data. If the difference between the approximated value and original data is larger than threshold, we start a new a line fitting process. We store lines' slopes and the number of original data that each line stands for in the memory. The patterns are all represented by linear segments also. Then we do similar matching on approximated line segments. We select a few line segments that stand for original data whose total number is equal to that of the pattern's original data. We compute dynamic timing warp distance between them. If the distance is less than the predefined threshold, we say the two sequences are similar. The users' queries are processed on approximated data. Approximation usually saves disk storage space, but it may not give a result with 100% precision. We compare our methods with naïve query method in match precision and energy consumption. The experiment results show that our strategy performs much better in energy consumption and the precision is also very high.

The rest of the paper is organized as follows. In Section 2, we talks about some related work on how the pattern query is processed in data stream. We review background material and state our problem in Section 3. Our PLR strategy and similar matching strategy is discussed in Section 4. In Section 5, we evaluate our strategies on microprocessor compared to naïve method. Finally, we conclude our work and describe the major future research directions that we intend to pursue in Section 6.

2. Related Work

There is a plethora of research on the problem of query processing in WSNs. Vaidehi et al. [15] treat the sensor network as a distributed database. According to the characteristics of sensor networks (mainly limited memory and battery energy), they propose Distributed Nested Loop Join Processing (DNLJP) algorithm. The algorithm portions the nodes into clusters based on geographic locations and performs the query processing in a distributed manner. The algorithm mainly concerns on how to process the query instead of how each sensor node process the query. The spatial and temporal correlations of sensor data are explored in [16][17]. Anurag et al. [16] propose a scheme to exploit the spatial correlation of data in a three dimensional wireless sensor network. The purpose is to reduce the number of transmissions in the network to save energy. The nodes are formatting a binary tree and the transmission of data is along the tree nodes. There are two types of sensor nodes in the system, the tree nodes and the non-tree nodes, also called sensing nodes. Then sensing nodes sense the attribute value and report it to the nearest tree node. Tree node runs the polynomial regression function to get the coefficients of regression. The polynomial may not represent the actual distribution of temperature in the space and tree nodes must consume much energy to run it. Teresa et al. [17] propose a relational framework to model and to analyze the data observed by sensor nodes of WSNs. Its purpose is to use relational learning techniques to discovery interesting patterns relating spatio-temporal correlations. The algorithm used in this paper is based on the same idea of the generic level-wise search method, known in data mining from the APRIORI algorithm. The genneration of the frequency patterns is based on a top-down approach. It starts with the most general patterns of length 1 generatd by adding to the empty pattern a non-dimensional atom. But this paper doesn't consider the pattern query problem.

Huamr et al [18] propose a new energy efficiency model for BS (Basic Station) energy consumption, emphasizing the embodied energy constituent. They give ways to estimate the embodied energy consumption and operating energy consumption. They apply this energy model to two scenarios and do simulations to optimize the total energy consumption by exploring the trade-off between operating energy and embodied energy. At last they get the conclusion that it is not good to use an increased number of BSs with lower transmission power and power-down strategy to perform energy savings. Kazem et al [19] propose a new query decomposition approach for optimizing query processing in multi-sink sensor networks. Message is sent to a single selected node and the selected node process the query and sends response back. The nodes between query region and sinks by sharing same query data, decide to decompose query and send whole or some part of query region nodes results back to the sinks. Experimental results show that the proposed approach highly reduces the query overhead in the network and save the energy consumption of sensing nodes. In our application, there is only one sink node. Prervin et al [20] propose a hybrid query processing framework that combines the advantages of both the approximation and aggregation based techniques and avoids their limitations. In this approach, authors construct a hierarchical probabilistic data model representing the overall sensor data characteristics across the network, which is query independent and is later used for selecting sensor nodes to process user queries. Experimental results illustrate the efficacy of the proposed framework compared to contemporary approximation and aggregation based query processing techniques.

There are mainly two problems in the pattern query process. One is that how to present the data collected by sensor nodes to save memory. Many data stream approximation strategies have been used in WSNs. Eugen et al. [21] propose a method that abstracts time series to its basic shape descriptor. They reduce the raw sensor signal by computing a piecewise linear approximation that preserves its shape. A sensor samples data at a specific fixed frequency. The new sampled data is forwarded to the emSWAP algorithm that decides wheter to store the value to a buffer or to run the bottom-up approximation step, thereby producing the next approximation linear segment. In emSWAP, the authors use sum of distances instead of Euclidean Distance. Also corresponding interpolation points on the segemnt is computed when computing the approximation cost between an approximating segment and raw data points. Wavelet transform is used in WSNs in [14] . Cai et al. [22] propose a novel data aggregation scheme based on wavelet-entropy for WSNs. Firstly, they use multi-scale wavelte transforms to spread signal in multi-sacale range, and then aggregate information using wavelet-entropy discriminance theorem in cluster members and cluster headers with LEACH [23] clustering algorithm to reduce transmitting packets. The authors decompose signal step by step. They regard wavelet transform as a set of enantiomorphism filters and use a set of high-pass and low-pass enantiomorphism filters to decompose signal. Walvelet entropy can be used to measure stabilization and reliabitlity of sensor ndoes, and distribute their weighted values in the fusion process. Zhang et al. [14] propose a data compression scheme with infinite norm error bound for WSNs by designing an error tree and the regression equations set. This algorithm can simultaneously explore the temporal correlations among the sensory data. They capture temporal correlation in one stream by the 1D Haar wavelet transform. As for multivariate sensor streams, some streams are selected as the base according to the correlation coefficient matrix, and other streams can be expressed with one of these bases using linear regression. All above mention techniques are appropriate to compress the data senses by sensor nodes and to process single sensor value queries or aggregation queries, which are based on sensor values at an instant time. They could not handle pattern based queries in WSNs.

Another key point in pattern based query is the approaches for measuring the similarity between the subsequence and patterns. Several distance functions have been proposed for different applications, such as Euclidean Distance (ED) [24], Dynamic Time Warping (DTW) [25], Longest Common SubSequence (LCSS) [26] and Edit distance with Real Penalty (ERP) [27]. ED is the most simple and classic distance function. It can be computed simply and be easy to understand. It satisfies the triangle inequality. But it requires the sequence to have the same length, which restricts its applications. Noises in the sequence affect the final result seriously. Keogh et al. [28] propose Weighted Euclidean Distance (WED). In this strategy, the weight changs accoring to users' feedback to support local time shifting. But the trainning always lags behind time series's changs, the WED is only can be used for static data streams. DTW can handle sequences with different lengths and local time shifting, but it doesn't follow triangle inequality, which is one of the important properties of a metric distance function. DTW is non-indexable with current techniques. Jeong et al. [29] propose a new distance measure, called weighted DTW (WDTE). In order to prevent minimum distance distortion caused by outliers, the appraoch penalizes points with higher phase difference between a reference point and a testing point. The authors also assign weights as a function of the phase difference between a reference point and a testing point. LCSS is proposed to handle possible noise that may appear in data. It allows some data points not to participate in the match process, so the final result may not accurate. Also LCSS can handle sequences with different lengthes and local time shifting, and it doesn't follow triangle inequality. LCSS use the longest common subsequence to measure the similarity between two streams. ERP represents a marriage of L1-norm and the edit distance. It uses real penalty between two non-gap elements, but a constant value for computing the distance for gaps. ERP can support local data shifting and it is a variant of DTW, but it follows triangle inequality.

3. Preliminaries and Problem Statement

A data stream collected by sensor node is an ordered sequence, $X = \{x_1, x_2, ..., x_i, ..., x_t\}$, where each x_i is a real value arriving at a specific time i, and t is the current timestamp. A subsequence $x_{i,j} = \{x_i, x_{i+1}, ..., x_j\}$ is similar to a pattern P if $dist(x_{i,j}, P) \le \varepsilon$, where $dist$ is a distance function and ε is a user-specified threshold. To solve the query defined in Definition 1, the most straightforward solution would be to consider all possible subsequence whose length is equal to the pattern's length. The sensor node stores recent coming data using sliding window. The sliding window model is shown in **Fig. 1**. A sliding window is denoted as $W_i = \{s_i, s_{i+1}, ..., s_{i+w-1}\}$, where w is pre-defined window size according to sensor nodes' memory size. In **Fig. 1**, the size of sliding window is w. When new data comes, the oldest data in the window is abandoned. We select subsequences from the sliding window and compute Euclidean distance, which can be computed by Formula (1), between the subsequence and the pattern. If the distance is smaller than pre-defined threshold, the subsequence is similar to the pattern. The algorithm is shown in Algorithm 1. The complexity of the algorithm is $O(w*l)$, where w is the length of sliding window and l is the length of pattern. A better solution, but still not good enough, is to speed up the ED calculation by doing early abandoning [30]. During the computation of Euclidean distance, if we note that the current sum of the squared differences between each pair of corresponding data points exceeds ε^2, where ε is the predefined distance threshold, we can stop the calculation.

$$D(Q,C)=\sqrt{\sum_{i=1}^{n}(q_i-c_i)^2} \tag{1}$$

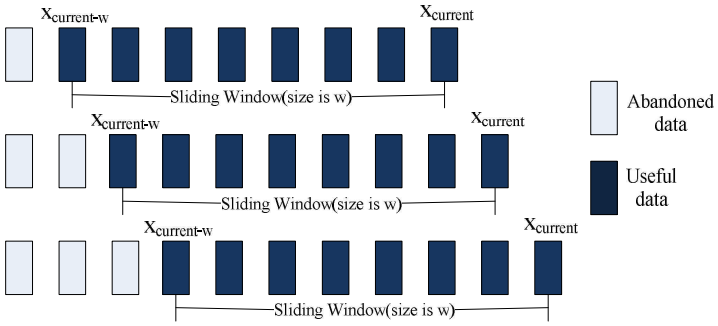

Fig. 1. Sliding window model

Algorithm 1: Naïve Method

Input: Sliding window W , Pattern P , Length of sliding window W_{length} , Length of pattern P_{length} , Predefined distance threshold ε .

Output: All subsequence of W that similar to P .

for $i = W_{start}$ to $W_{length} - P_{length}$, **do**

compute $d = D(W_i, P)$; // $D(W_1, P)$ is the Euclidean distance function

if $d < \varepsilon$

output position i

EndIf

EndFor

The naïve method can get most accurate results with appreciate threshold, but it is too complex for sensor nodes to run it. The complexity of Naïve method is $O(n^2)$. Its running time increases quickly with the increase of sliding window's size or pattern's size. We run above algorithm on STM32W108 Soc [31], whose micro processor is ARM Cortex-M3. The length of sliding window is 1432, and the lengths of patterns are range from 10 to 150. The running time of above algorithm is shown in **Fig. 2**. We can see that the running time increases quickly with pattern's length increasing. The microprocessor's speed is slow. When the pattern's length is 10, the running time is already upon to 0.43s, and when the pattern's length is 150, the running time goes up to 4.81s. The processor's speed is slow and energy of sensor nodes is limited. Sensor nodes need to spend much timeslot on finishing a query. This would limit sensor nodes doing other things and consume sensor nodes' energy quickly. We must reduce the complexity of Algorithm 1 to save processing time and thus to save energy. Another problem with upon algorithm is that sensor nodes store sensed data directly. This conflicts with the fact that wireless sensor network are heavily constrained by their memory size. We all know data sensed by sensor nodes has temporal correlation [17]. Storing all sensed data waste a lot memory. Also sensor nodes have limited memory size and only can store a little data. We must use compression strategies to store the raw data to improve the amount of stored data.

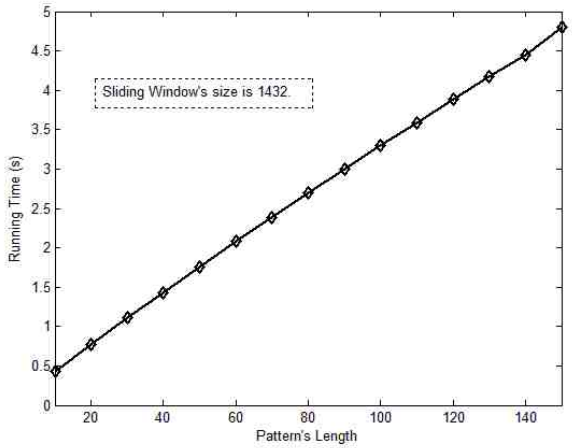

Fig. 2. Algorithm 1's running time on STM32W108 with different pattern's length

There are many compression strategies. Due to the advantage of easy understanding and implementation, PLR has been widely adopted in representing the data. Our approach is to approximate the sampled sensor data into a representation of linear segments which is efficient to manipulate and to process. After getting the linear segments representation for raw data, we use each line segment's slope and the number of original data that each line represents to present each segment. We choose a few line segments that stand for original data points whose number is equal to the pattern's length to do similar matching process, so the number of subsequence's line segments may not equal to that of pattern. We use dynamic time warp distance as similar standard.

4. Modeling and Strategy

The natural of physical phenomenon constitutes the temporal correlated between each consecutive observation of a sensor node. We can use collected data to predicate the coming data. As the memory of sensor nodes is limited, we use line approximated technology to represent the original data. The purpose of piece linear representation is to use several line segments to approximate the raw data collected by sensor nodes. Next we introduce our piecewise linear representation method which is called AD-PLR (Piecewise Linear Representation based on Adjacent Data). After this, we introduce our matching strategy based on piece lines.

4.1 Piecewise Linear Representation Based On Adjacent Data

Sensor nodes collect data at fixed rate. When we get x_i at t_i and x_{i+1} at t_{i+1}, we can do line fitting based on the two data points. We get linear equation $x_j^{'} = k * t_j + b$. Then at t_{i+2}, sensor node gets a new data x_{i+2}. Also we can get an approximated data by $x_{j+2}^{'} = k * j_{j+2} + b$. If $\left| x_{j+2} - x_{j+2}^{'} \right| \le \varepsilon$, we don't need to update the linear equation and use the equation to predict next coming data. If $\left| x_{j+2} - x_{j+2}^{'} \right| > \varepsilon$, we start to do new line fitting according to data x_{j+1} and x_{j+2}. We call x_{j+2} the key point. There will be two approximated values at the key time point. We drop the old value and use new equation to represent the value at key point to

reduce the linear fitting error rate. ε is predefined by users. It shows user's precision for the approximation process. The smaller ε is, more line segments there are. So the piecewise linear representation can show more details of the raw data and the similar matching results are more precise. But the compression is not obvious and only can save a little memory space. Hence, it is a tradeoff between the compression and matching precision. The user must define the threshold according to real applications. PLR-AD supports online compression. An example of above process is shown in **Fig. 3**. When the sensor node's memory is full, if the coming sample data can be approximated by the latest linear function, we drop the data directly and the data in the sliding window don't need to be updated. But if the coming sample data can't be approximated by the latest linear function, we start a new linear fitting process and store the latest function in the sliding window and drop the oldest linear function.

In **Fig. 3**, the size of sliding window is 40 and there have already 40 data in the window and the user defined threshold is 1. After getting sample data x_1, x_2, we can get fitting linear equation $X_1 : x = 0.25 * t + 19.68$. At $t = 3$, the predicted data $x_3^{'} = 20.43$, but the sample data $x_3 = 21.56$, $\left| x_3 - x_3^{'} \right| > 1$, so we can't use equation X_1 to predict next coming sample data. We do linear fitting again based on sample data x_2 and x_3, then we use the new equation to predict coming sample data. The process is repeated until all the stored sample data are represented. At last we only use 7 line segments to represent it. The raw data can be approximated by $\{(x_1, x_2, t_1), (x_2, x_4, t_2), (x_4, x_8, t_4), (x_8, x_{27}, t_8), (x_{27}, x_{33}, t_{27}), (x_{33}, x_{40}, t_{33})\}$. $x_2, x_4, x_8, x_{27}, x_{33}$ are the key data points. Computing the slope for each segment and the number of original data that each segment stand for, the original data are also can be represented by $\{(0.25,1), (1.38,2), (0.57,4), (0.25,19), (0.00,6), (0.44,4), (0.12,4)\}$ .If the coming data can be predicted by seventh segment function, we don't need to update the data in the window. But if the data can't be predicted, we start a new linear fitting process. After getting new segment function, we store it in the memory and drop the oldest segment, segment 1 in the figure.

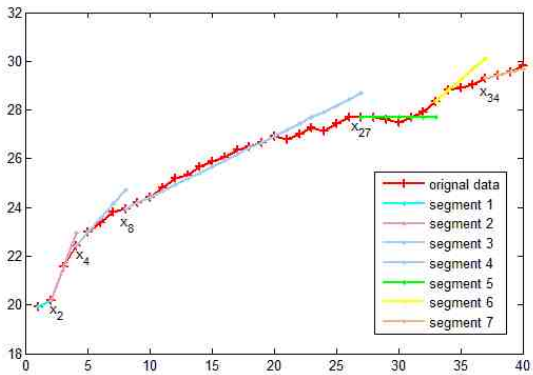

Fig. 3. An example for piecewise line approximation based on adjacent data, threshold $\varepsilon = 1$

4.2 Matching Strategy For AD-PLR

After doing piecewise linear representation, a data sequence can be represented by $\{(x_{1L}, x_{1R}, t_1), (x_{2L}, x_{2R}, t_2), ..., (x_{mL}, x_{mR}, t_m)\}$, where x_{mL}, x_{mR} are the two feature points for m^{th} line segment, t_m is the end point of the m^{th} line segment and m is the total number of line segments. According to two feature points, we get each segment's slope. So the data sequence also can be

presented by $\{(s_1,l_1),(s_2,l_2),...,(s_m,l_m)\}$, where s_i is the slope for i^{th} line segment and l_i is the number of original data point that i^{th} segment represents. Likewise, the pattern also can be presented by slope sequence, $\{(p_1,l_1),(p_2,l_2),...,(p_t,l_t)\}$. We can do similar matching process based on segments' slopes.

One of match strategies is that we can choose a few line segments whose total number is equal to that of the pattern to do similar matching. But the raw data points in these two data stream may differ a lot. This may affect the similar matching precision seriously. So when doing similar matching, we always choose nearly a few original data points whose total number is equal to that of the pattern. We start from the i^{th} line segment and the i^{th} segment stands for l_i original data points. When $l_i + l_{i+1} + ... + l_{i+j} > P.length$, we start the similar matching process. These two original data points may be represented by different number of line segments. One example is shown in **Fig. 4**. The number of original data points in the pattern is 40 and the number of approximated line segments is 7. One similar subsequence's length is 38, but its line segments' number is 4. ED is a traditional and simple standard for similar matching, but it only can compute distance between sequences with the same length. So we use dynamic timing warp to compute the distance between subsequences and patterns. Given two sequence $X = \{x_1, x_2, ...x_m\}$ of length m and $Y = \{y_1, y_2, ..., y_n\}$ of length n, their DTW distance $D(X,Y)$ is defined as:

$$\begin{aligned} &D(x,y) = f(m,n) \\ &f(t,i) = \|x_t - y_i\| + \min\begin{cases} f(t,i-1) \\ f(t-1,i) \\ f(t-1,i-1) \end{cases} \\ &f(0,0) = 0, f(t,0) = f(0,i) = \infty \\ &(t = 1,...,m; i = 1,...,n) \end{aligned} \tag{1}$$

where $\|x_t - y_i\| = (x_t - y_i)^2$ is the distance between two numerical values. When getting a pattern represented by $\{(p_j,t_j),(p_{j+1},t_{j+1}),...,(p_{j+n},t_{j+n})\}$ from user, sensor node searches its stored data and chooses line segments $\{(s_i,t_i),(s_{i+1},t_{i+1}),...,(s_{i+m},t_{i+m})\}$, where $(t_{i+m} - t_i) \approx (t_{j+n} - t_j)$, to compute DTW distance, $D(m,n)$, with the pattern. If $D(m,n) < \varepsilon$, the subsequence is similar to the pattern.

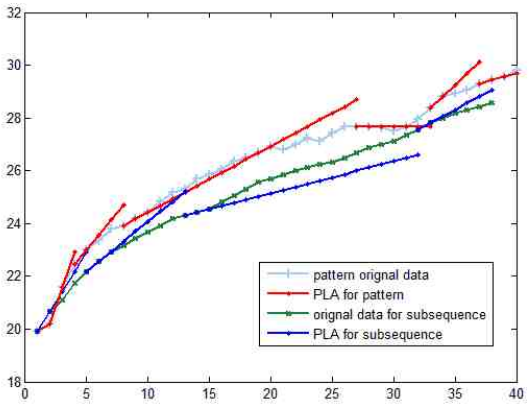

Fig. 4. Similar subsequences have the same number of data points, but have different number of approximated line segments.

4.3 Algorithms For AD-PLR And Similar Matching Strategy

In order to save sensor nodes' memory, we use piecewise linear representation to compress the data collected by sensor nodes. Our strategy supports online compression. The algorithm for AD-PLR is shown as Algorithm 2. AD-PLR only need to go through the original data for only once to get approximated line segments. Its complexity is *O(n)*.

Algorithm 2: AD-PLR

Input: sliding window W , user defined threshold ε
Output: approximated line segments
for $i = 1$ to W_{length} , **do**
  if $i == 2$
    $k = w_i - w_{i-1}$;
    $b = w_i - k * i$;
  else
    $w_i^{'} = k * i + b$;
    if $abs(w_i - w_i^{'}) > \varepsilon$ // abs is the equation to get absolute value
      output k, i ;
      $k = w_i - w_{i-1}$;
      $b = w_i - k * i$;
    EndIf
  EndIf
EndFor

After getting approximated line segments for original data streams and patterns, we can do similar matching process on line segments. We always choose a few line segments that represent original data points whose number is nearly equal to pattern's length. We compute DTW distance between the subsequence's slope sequence and the pattern's slope sequence. If the value is smaller than user predefined threshold, we say that the subsequence is similar to the pattern. The similar matching algorithm is shown as Algorithm 3. The complexity for Algorithm 3 is *O(m*l)*, where *m* is the total number of line segments for original data and *l* is the number of original data points in the pattern. Although the complexity is similar to naïve method, but amount of approximated data is much less than that of original data.

Algorithm 3: MatchStrategy

Input: slope sequence for data stream collected by sensor node, S ; slope sequence for pattern, $S_{pattern}$, user defined threshold ε .
Output: similar subsequence
$SubSeq\_Len = 0$;
for $i = 1$ **to** S_{length} ,**do**
  for $j = i$ **to** S_{length} ,**do**
    $SubSeq\_Len = SubSeq\_Len + S_j.length$;
    if $SubSeq\_Len > S_{pattern}.length$
      compute $DTW(j, S_{pattern})$ between $< s_j, s_{j+k} >$ and $S_{pattern}$;
      if $DTW(j, S_{pattern}) < \varepsilon$
        output $i, SubSeq\_Len$;

```
        SubSeq _ Len = 0 ;
      EndIf
    EndIf
  EndFor
EndFor
```

5. Performance Analysis

We do experiments on STM32W108 microprocessor to evaluate the efficiency of our strategy. First we run our strategy on STM32W108 to get our strategy's running time. According to the running time, we can see whether our strategy can run on sensor node. Then we use STM32W108 to collect large number of data. We exact special patterns from the collected data and use our strategy to do similar matching and compare our strategy's precision and energy consumption with that of naïve method.

5.1 Collect Dataset And Patterns

We use STM32W108 to collect indoor temperature to get the dataset. The collecting frequency is 0.2Hz. During collecting process, we use hot handbag and cup to heat up the temperature sensor. We collect about 4500 data, and we extract four patterns from the dataset. The dataset and four patterns are shown in **Fig. 5** and **Fig. 6**. Pattern 1 stands for that we use hot cup to heat up the temperature sensor. The temperature goes up slowly. Pattern 2 shows the process of using hot handbag to heat up the temperature sensor. The temperature goes up much faster than using cup to heat up the sensor. The cooling process after using hot handbag to heat up the sensor is shown in Pattern 3. Pattern 4 shows the true indoor temperature, which is smooth.

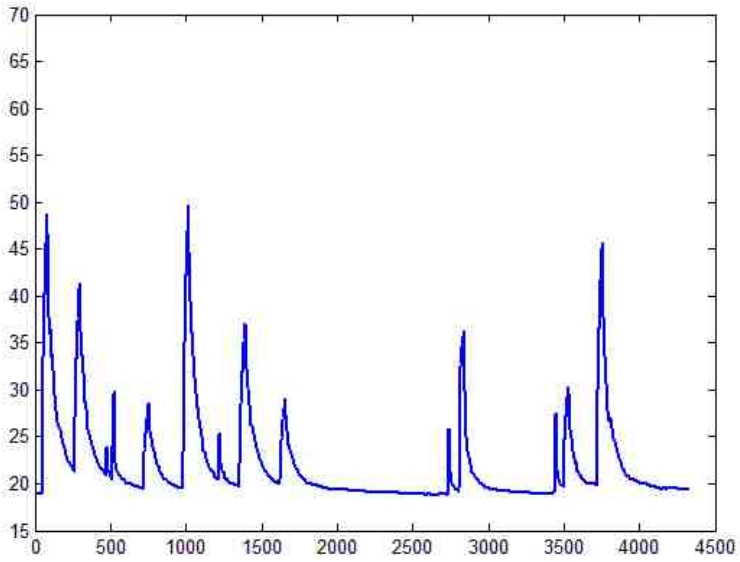

Fig. 5. Collected dataset

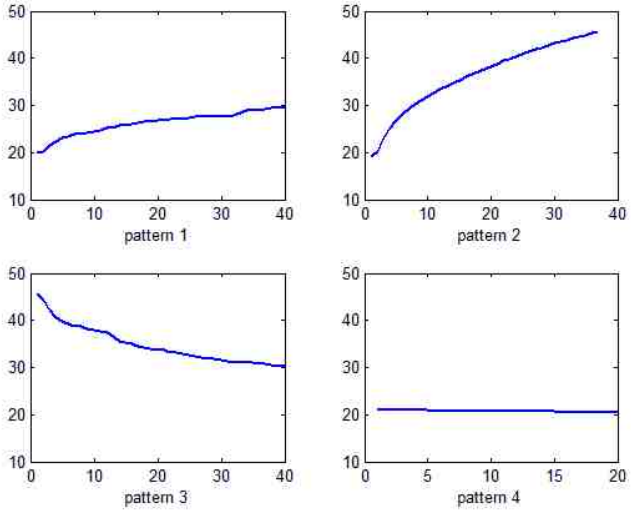

Fig. 6. Four patterns extracted from the collected dataset

5.2 Performance For AD-PLR Strategy

We do experiments to evaluate AD-PLR's performance. We mainly use following three metrics to evaluate performance of AD-PLR strategy: (1) AD-PLR's running time on STM32W108; (2) the compression rate of AD-PLR strategy: the total amount of original data is *m*, after running AD-PLR strategy we can *n* bytes to represent it, so the compression rate can be computed by $compression\_rate = 1 - \frac{n}{m}$; (3) the matching precision: We analyze the returned subsequences, total amount is *m*, and get sequences that are truly similar to the pattern, total amount is *n*. So the matching precision can be computed by $pecision = \frac{n}{m}$.

We run our piecewise linear representation strategy with different length of data stream on STM32W108. The result is shown in **Fig. 7**.

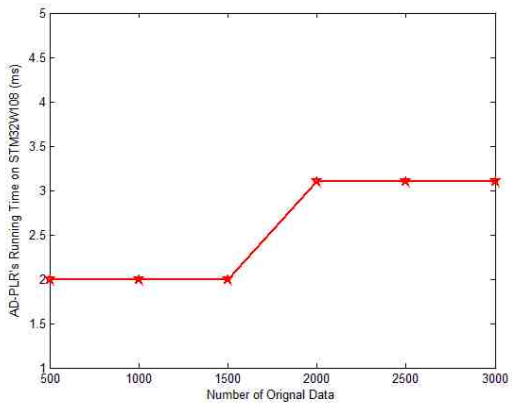

Fig. 7. AD-PLR's running time on STW32W108 with different number of original data points

AD-PLR's running time is very little on STW32108, less than 10ms. When the data stream's

length is 500, its running time is only 2ms, and when the length goes up to 3000, the running time increases only about 1ms. Compared to naïve method's running time in **Fig. 2**, AD-PLR's running time is much less, so it consumes much less energy.

The number of approximated line segments is based on user's precision for the approximation process. If the threshold is smaller, there are more line segments and the precision is higher. We use average Euclidean distance between the original data and approximated data as the precision stand. Euclidean distance between two sequences can be computed by Formula (1). We run AD-PLR for collected dataset and the extracted four patterns with different thresholds. The results are shown in **Table 1**.

Table 1. AD-PLR's effect on the collected dataset and the extracted four patterns with different threshold (segNo stands for total number of line segments; avgED stands for the average ED)

	$\varepsilon = 0.5$		$\varepsilon = 1$		$\varepsilon = 1.5$		$\varepsilon = 2$	
	segNo	**avgED**	**segNo**	**avgED**	**segNo**	**avgED**	**segNo**	**avgED**
Collected Dataset	312	0.076	180	0.298	149	0.747	128	1.046
Pattern 1	11	0.049	7	0.15	6	0.559	3	0.573
Pattern 2	9	0.046	7	0.163	6	0.430	5	0.582
Pattern 3	9	0.051	8	0.171	6	0.472	3	0.535
Pattern 4	1	0.078	1	0.078	1	0.078	1	0.078

We can see that we only need one segment to represent the original sequence when the data is smooth where the compression effect is the best. But when the data stream is changing, it needs several segments to represent the original data. It is noted that though smaller value of threshold can increase the matching precision, but produce smaller compression. Hence, larger value of threshold improve the compression rate, it makes the matching error rate higher. When the threshold is 0.5, the error rate is less than 0.1 for five sequences; but when the threshold is 2, the error rate goes up to 0.5. The number of approximated line segments decreases with user defined threshold increasing. For Pattern 1, when the threshold is 0.5, the number of line segments is 11; but when the threshold is 2, the number of line segments is only about 3. The compression rate increases about 72.7%.

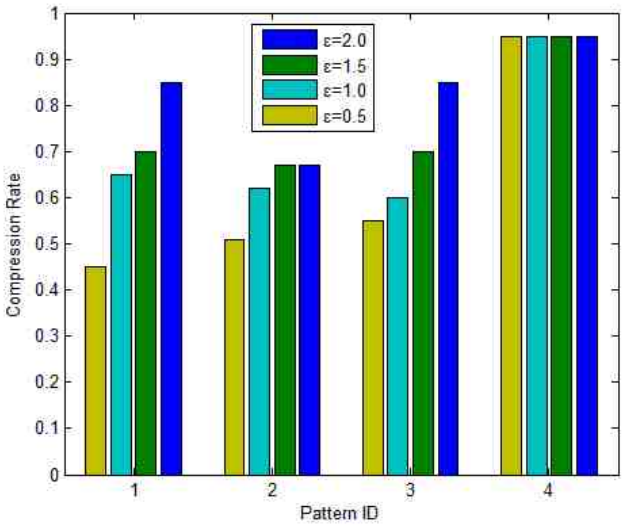

Fig. 8. Each pattern's compression rate for different ε

The compression rate and matching precision of AD-PLR strategy for different ε are shown in **Fig. 8** and **Fig. 9**. From **Fig. 8** we can see that with the increase of ε, the compression rate is increased. For pattern one, when ε=0.5, its compression is 45%, but when ε=2, its compression is 85%. But for pattern four, its compression is always 95%, because its basic shape is smooth, so we only need one segment to represent it. According to **Fig. 8**, our conclusion from **Table 1** is true. **Fig. 9** shows each pattern's matching precision for different ε. We can see that each pattern's matching precision increases with ε's decreasing. As for pattern 2, when ε=0.5, its matching precision is 0.8, but when ε increases to 2, its matching precision is only 0.38. The users should define the threshold according to their demand for the approximation precision and compression rate. In the matching process, we set the threshold to 1 to do line approximation.

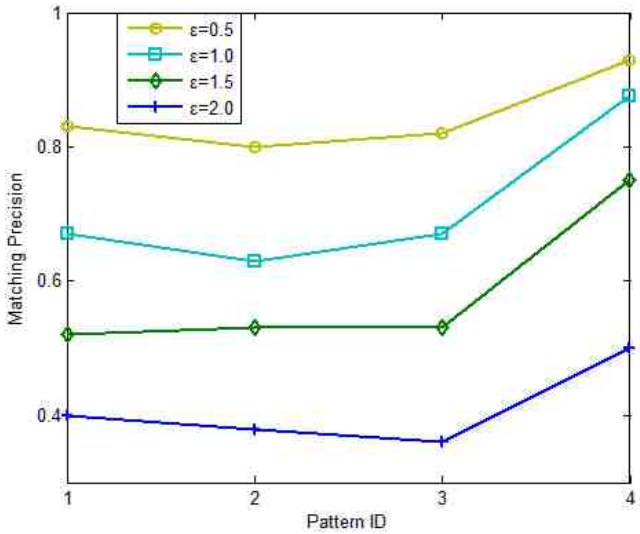

Fig. 9. Each pattern's matching precision for different ε

5.3 Performance For The Similar Strategy

After doing piecewise linear representation, the data streams of the sensor nodes and the patterns are all represented by line segments. We compute all line segments' slopes to stand for lines. When doing similar matching process, we select the same amount of data points as the pattern. We compute the dynamic timing warp distance between the two sequences. If the distance is smaller than the threshold, we say that the subsequence is similar to the pattern. We compare our strategy's matching precision with that of naïve method. The similar matching strategy returns subsequences.

The precision performance is shown in **Fig.10**. We can see that the precision for naïve method is higher than our method. When using naive method, the precision for each pattern is above 0.9. Our matching strategy based on line segments' slope is about 0.65 for each pattern. When the pattern is simple, for example Pattern 4, the precision is much higher.

We also analyze the energy consumption of naïve method and our matching method. We use the running time as the metric for energy consumption. If the algorithm's running time is longer, it consumes much more energy. Our method's running time equals to AD-PLR's running time plus pattern's matching time. The result is shown in **Fig. 11**. We can see that our matching strategy based on slope runs much faster than native method. As for Pattern 1,

running time of our strategy is about 50ms, but that of naïve method is about 1400ms. Our time is about 3.5% of naïve method's running time. This can save sensors' energy a lot. It consumes much less energy than naïve method. It suits for energy-limited sensor network.

Fig. 12 shows part of matching results for four patterns by using our method. From the figure, we can see that the shape of much matching results is similar to mother pattern. As for pattern 2, much of results are increases quickly which is true for Pattern 2; and results of Pattern 4 are stay smooth.

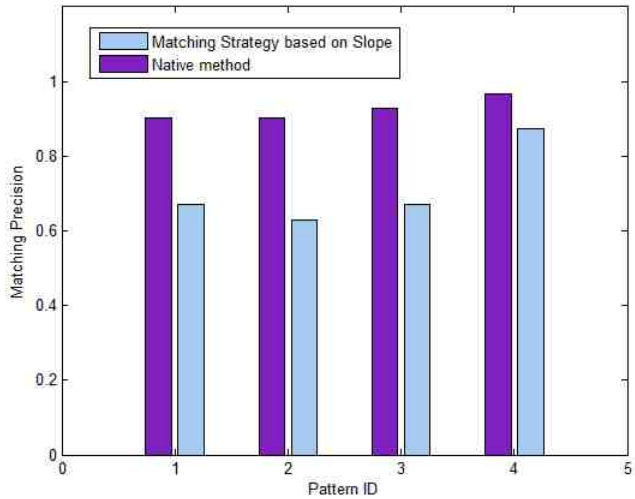

Fig. 10. Each pattern's matching precision in the dataset

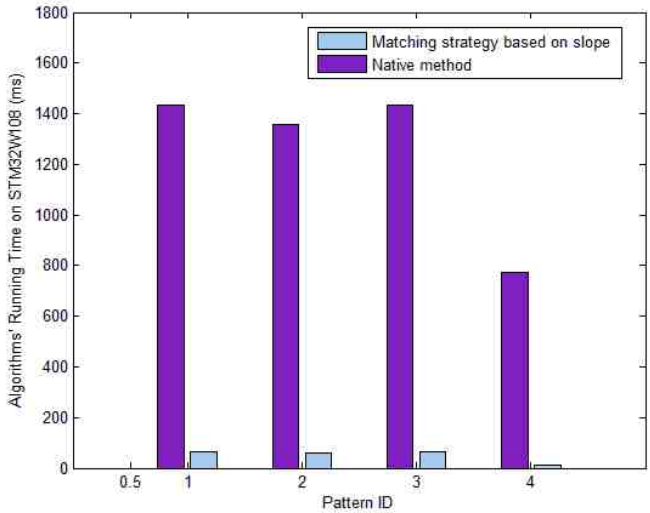

Fig. 11. Similar matching strategy's energy consumption on STW32W108 for each pattern

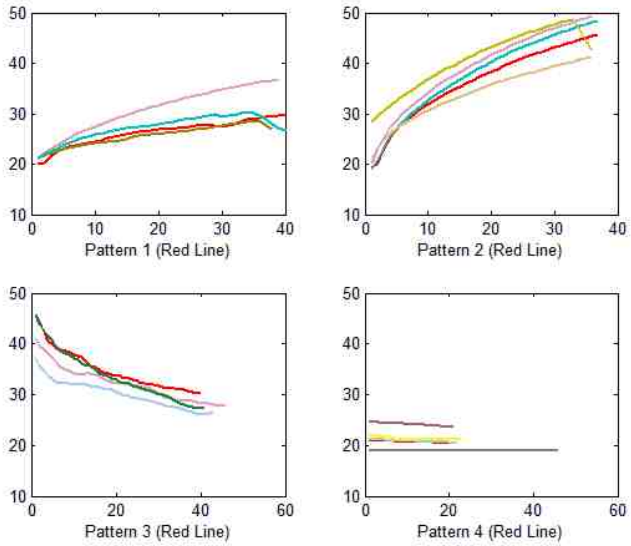

Fig. 12. Matching results for four patterns by using our strategy

5. Conclusion and Future Work

In this paper we delve into the problem of sequence pattern query processing in WSNs. As traditional strategies for data stream or time sequence need too much memory space and consumes too much energy for sensor nodes, we propose a novel approach to process the pattern queries. We represent original data by line segments, which can compress the data and reduce the energy consumption for similar matching process. Also the patterns are represented by line segments. We use each line's slope to stand for each segment and do similar matching based on the slope sequence. We choose several line segments which stand for nearly the same number of original data points as the pattern has. We compute DTW distance between the two slope sequences. If the distance is less than the threshold, we say the subsequence is similar to the pattern. We do experiments on STW32W108 to compare our method with traditional approach. Although our strategy's matching precision is less than that of the naïve method, the energy consumption is much less than that of naïve method. The experiments show that our strategy can be used in wireless sensor networks.

The precision of our strategy is still not very high. In future, we will do research on improving the precision of our strategy. The complexity of our similar matching method is $O(n^2)$, when the pattern's length increase a lot, the energy consumption increases quickly. In future we force on reducing the complexity of our strategy.

References

[1] J.D. Freeman, and S. Simi, "Remote monitoring of indoor environment using mobile robot based wireless sensor network," in *Proc. of 6th International Conference on Computer Science and Education, Final Program and Proceedings*, pp.1080-1084, Aug.2011. Article (CrossRef Link)

[2] L. Shan, Z. Li, and H. Hu, "Converged mobile cellular networks and wireless sensor networks for machine-to-machine communications," *KSII Transactions on Internet and Information Systems*, vol.6, no.1, pp.147-161, 2012. Article (CrossRef Link)

[3] G. Tolle, J. Polastre, R. Szewczyk et al., "A macroscope in the redwoods," *In Sensys*, pp.51-63, 2005. Article (CrossRef Link)
[4] O. Chipara, C. Brooks, S. Bhattacharya, and T. Bailey, "reliable real-time clinical monitoring using sensor network technology," *AMIA Annu. Symp. Proc. 2009*, pp.103-107, Nov.2009. Article (CrossRef Link)
[5] Y.Y. Yu, S.F. Shi, J.Z. Li, and C.K. Wang, "HIBOR: An efficient approach to sequence pattern query processing in wireless sensor networks," *IET International Conference on Wireless Sensor Network 2010*, pp.161-166, Nov.2010. Article (CrossRef Link)
[6] R.K. Kushwaha, and S. Kar, "Development of the grooming techniques for reducing network traffic in sensor networks," in *Proc. of 6th International Conference on Wireless Communication and Sensor Networks*, Dec.2010. Article (CrossRef Link)
[7] C. Faloutsos, M. Ranqanathan, and Y. Manolopoulos, "fast subsequence matching in time-series databases," in *Proc. of ACM SIGMOD International Conference on Management of Data*, vol.23, no.2, pp.419-429, Jun.1994. Article (CrossRef Link)
[8] H. Geoff, S. Laurie, and D. Pedro, "Mining time changing data streams," in *Proc. of 2001 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.97-106, 2001. Article (CrossRef Link)
[9] C. G. Anna, K. Yannis, S. Muthukrishnan, and S. Martin, "Surfing wavelets on streams one-pass summaries for approximate aggregate queries," in *Proc. of the 27th International Conference on Very Large Data Bases*, Sep.2001.Article (CrossRef Link)
[10] X. Lian, L. Chen, J.X. Xu, G.R. Wang, and G. Yu, "Similarity match over high speed time-series streams," in *Proc. of 23rd International Conference on Data Engineering*, pp.1086-1095, Apr.2007. Article (CrossRef Link)
[11] S. Datta, K. Bhaduri, C. Giannella, H. Karqupta, and R. Wolff, "Distributed data mining in peer-to-peer networks," *IEEE Internet Computing*, vol.10, no.4, pp.16-26, Jul.2006. Article (CrossRef Link)
[12] G. Jonannes, and M. Samuel, "Query processing in sensor networks," *IEEE Pervasive Computing*, vol.3, no.1, pp.46-55, Jan.2004. Article (CrossRef Link)
[13] A. Silberstein, R. Braynard, and J. Yang, "Constraint chaining: On energy-efficient continuous monitoring in sensor networks," in *Proc. of SIGMOD 2006-Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp.157-168, Jun.2006. Article (CrossRef Link)
[14] J.M. Zhang, Y.P. Lin, S.W. Zhou, and J.C. Ouyang, "Haar wavelet data compression algorithm with error bound for wireless sensor networks," *Journal of Software*, vol.21, no.6, pp.1364-1377, Jun.2010. Article (CrossRef Link)
[15] V. Vaidehi, and D.S. Devi, "Distributed database management and join of multiple data streams in wireless sensor network using querying techniques," in *Proc. of International Conference on Recent Trends in Information Technology*, pp.594-599, Jun.2011. Article (CrossRef Link)
[16] A. Sharma, T. Banerjee, and D.P. Agrawal, "exploiting spatial correlation in a three dimensional wireless sensor network," in *Proc. of 2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, Oct.2007. Article (CrossRef Link)
[17] T.M.A. Basile, N. Di Mauro, and S. Ferilli, F. Esposito, "Relational temporal data mining for wireless sensor networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol.5883, pp.416-425, Dec.2009. Article (CrossRef Link)
[18] I. Humar, X. Ge, L. Xiang, M. Jo, and M. Chen, "rethinking energy-efficiency models of cellular networks with embodied energy," *IEEE Network Magazine*, vol.25, no.3, pp.40-49, Mar.2011. Article (CrossRef Link)
[19] A.A.P. Kazem, R.F. Beyrami, and A. Ghaffari, "A new approach for query decomposition and optimization in multi-sink wireless sensor networks," in *Proc. of the 10th IASTED International Conference on Parallel and Distributed Computing and Networks*, pp.148-154, Feb.2011. Article (CrossRef Link)
[20] S. Pervin, J. Kamruzzaman, G. Karmakar, and A.K.M. Azad, "Hybrid in-network query processing framework for wireless sensor networks," in *Proc. of IEEE International Conference*

on Communications, Jun. 2011. Article (CrossRef Link)
[21] E. Berlin, and K. Van Laerhoven, "An on-line piecewise linear approximation technique for wireless sensor networks," in *Proc. of 2010 IEEE 35th Conference on Local Computer Networks*, pp.905-912, Oct.2010. Article (CrossRef Link)
[22] W.Y. Cai, and M.Y. Zhang, "Data aggregation mechanism based on wavelet-entropy for wireless sensor networks," in *Proc. of 2008 International Conference on Wireless Communications, Networking and Mobile Computing*, Oct.2008. Article (CrossRef Link)
[23] T.Y. Wu, K.H. Kuo, H.P. Cheng, J.W. Ding, and W.T. Lee, "Increasing the lifetime of ad hoc networks using hierarchical Cluster-based Power Management," *KSII Transactions on Internet and Information Systems*, vol.5, no.1, pp.5-23, 2011. Article (CrossRef Link)
[24] A. Rakesh, F. Christos, and S. Arun, "Efficient similarity search methods in sequence databases," *Foundations of Data Organization*, vol.930, pp.69-84, 1993. Article (CrossRef Link)
[25] J. B. Donald, and C. James, "Finding patterns in time series: a dynamic programming approach," *Advances in knowledge discovery and data mining*, pp.229-248, 1996. Article (CrossRef Link)
[26] V. Michail, G. Dimitrios, and K. George, "Discovering similar multidimensional trajectories," in *Proc. of the 18th International Conference on Data Engineering*, Feb.2002. Article (CrossRef Link)
[27] C. Lei, N. Raymond, "On the marriage of Lp-norms and edit distance," in *Proc. of the Thirtieth international conference on Very large data bases*, vol.30, pp.792-803, Sep.2004. Article (CrossRef Link)
[28] E. Keogh, and M. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," in *Proc. of the 4th International Conference of Knowledge Discovery and Data Mining*, pp.239-241. Article (CrossRef Link)
[29] Y.S. Jeong, M.K. Jeong. And O.A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognition*, vol.44, no.9, pp.2231-2240, Sep.2011. Article (CrossRef Link)
[30] W. Liu, E. Kergh, H. Van Herle, and A. Mafra-Neto, "Atomic wedgie efficient query filtering for streaming time series," *Fifth IEEE International Conference on Data Mining*, pp.490-497, Nov.2005. Article (CrossRef Link)
[31] C.L. Zhou, and J.H. Shen, "The design and realization of ZigBee Wi-Fi wireless gateway," *2011 International Conference on Electric Information and Control Engineering*, pp.1786-1790, Apr.2011. Article (CrossRef Link)

Yanhong Ding received B.E. degree from Dalian University of Technology, China. Currently he is a Master student in School of Software, Dalian University of Technology. His research interest covers wireless sensor networks and Internet of Things.

Tie Qiu is a lecturer and Ph. D in computer science and technology at Dalian University of Technology, Dalian, China. His research interests cover Embedded system architecture, Internet of Things(IoT) and High-performance computing based on multi-core SoC. He is a member of China Computer Federation (CCF) and ACM.

He Jiang is an associate professor at School of Software, Dalian University of Technology. His research interests include computational intelligence and its applications in software engineering and data mining. Jiang received his PhD in computer science from University of Science and Technology of China. He's a program cochair of the 2012 International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE 2012). He is also a member of the IEEE, the ACM and the CCF.

Weifeng Sun is a lecturer in software school at the Dalian University of Technology. Most of his research centers around high quality wireless communictions on wireless multihop network(such as WMN,WSN,MIPv6 and so on) and scheduling and applications on distributed network. He got the PH.D degree and bachelor degree at University of Science and Technology of China.