International Journal of Future Generation Communication and Networking
Vol. 9, No. 12 (2016), pp. 351-360
http://dx.doi.org/10.14257/ijfgcn.2016.9.12.32

Research on Data Transmission Security Technology of Wireless Sensor Networks

Jinsong Zhang and Hua Zhang

Qiqihar University, Qiqihar 161006, China
E-mail: 29373370@qq.com

Abstract

In this paper, regarding to the online data compression issue of wireless sensor networks, a disconnected segmented linear compression algorithm GDPLA is proposed. The algorithm uses the least number of disconnected segments to approximate describing the original time series under the condition of the error limit ε be guaranteed. The algorithm GDPLA is an optimal algorithm from the number of segments generated. In addition, the GDPLA algorithm requires linear run-time only, and the linear coefficient is 6, which makes it suitable for resource-constrained wireless sensor networks. Finally, the experiments on two real data sets show that the compression rate of our algorithm is obviously superior to other algorithms.

Keywords*: Wireless sensor network, data transmission, security technology, GDPLA*

1. Introduction

With the rapid development of sensing, embedded systems and low- consumption short-range wireless communications, sensor nodes with sensing, wireless communications and computing power are possible. A large number of sensor nodes with short-range wireless communication cooperate with each other to monitor a geographical area and transmit the perceptual data to the sink node by means of self-organized hop-by-hop forwarding. The wireless sensor networks (WSNs) are composed of these nodes with self- composition capacity.

In WSNs, data compression is a popular method to reduce communication overhead, but traditional techniques such as wavelet transform, discrete Fourier transform and discrete cosine transform have high space-time complexity and require high computation and storage, thus they are not suit for WSNs. Based on these observations, the researchers begin to explore new methods to compress the data in WSNs, namely the Piecewise Linear Approximation (PLA) method, which divides the time series into several segments, and then approximate describes each data points of the segment with a line segment. In recent years, the PLA methods used in WSNs are PMC-MR, Cache and Linear filter [1]. The PMC-MR scans the time series greedy and sequential from the first data point and places the scanned data points in a bucket until a data point is scanned, which makes the difference between the maximum value and the minimum value of the data points in the bucket larger than 2ε. In this case, the data points in the barrel are approximated by (Max + Min) / 2, and the absolute value of the difference between approximate value and the true value is equal to or less than (Max-Min) / 2 and less than or equal to ε. Then a new bucket is created from the last scanned data point, and operations like the first bucket continue to greedy scan the subsequent data points. Cache selects the value of the first data point as an approximation, if the subsequent data point within the ε range of the first data point it will be filtered out, until the data point which is not within the ε range of the first data point, then a new piecewise approximation is started from the new data point [2]. The linear filter selects the line connecting the first data point and the second data point as the approximation of the first segment. When the value of the subsequent data point is within the ε range of the vertical direction of the straight line, the data is filtered out, until the value of the new data point is not within the ε range of the vertical direction of the straight line, then a new Piecewise approximation is started from the new data point. As we known, there is no PLA method that can approximate describe a time series with a minimum number of line segments in linear time and guarantee the accuracy of the error (the absolute value of the difference between the true value and the approximation is less than or equal to one Error bounds) [3-4].

In order to solve this problem, a piecewise linear approximation algorithm, GDPLA (Greedy Disconnected Piecewise Linear Approximation), is proposed in this paper, which can describe a time series and guarantee the precision of error by using disconnected segment in linear time. The following briefly describes the main idea. Given a time series S, a maximum error bound ε, GDPLA starts at the first data point, greedily scans the data points in the sequence until a data point is scanned so that all data points before this data point can be approximate described by a line segment, and the accuracy is guaranteed [5-7]. However, if this data point is added, then there is no line segment can approximate describe all of the current data points which have not been approximate described yet. Thus, GDPLA approximates all data points prior to the last data point by a line segment, and greedy scanning begins at the last data point similarly until the entire time series ends.

2. Design of GDPLA Algorithm

We consider the data flow generated by a non-Sink node. Without loss of generality, we denote the number of the node by n0. Suppose that S=<x[t_1], x[t_2], x[t_3],……> is the time series of the monitoring value produced by the node n0. When the first data point (t_1, x[t_1]) arrives, it is stored. When the second data point (t_2, x[t_2]) arrives, it is easy to find a line segment where the vertical distance from the first two data points to this line segment is less than or equal to ε. When the third data point (t_3, x[t_3]) arrives, we examine whether the first three data points can be approximate described by a line segment and satisfy the ε error limit requirement [8-10]. If not, the first and second data point is approximated by a line segment, and then a new segment approximation is started from the third data point; contrarily, waiting for the arriving of the fourth data point. When the fourth data point arrives, similarly checking whether the first four data points can be approximate described by a line segment and satisfies the ε error margin requirement. In this way, whenever a new data point arrives, we perform a similar check operation until the entire time series ends.

2.2. UI lines and LI lines

Definition 1: Given a time series S[t_a:t_b]=< x[t_a], x[t_{a+1}],…, x[t_b]>, the straight line satisfying the following two conditions is U line which belongs to S[t_a:t_b].

(1) The straight line through point (t_p, x[t_p]−ε) and the point (t_q, x[t_q]+ε).

(2) a≤p≤q≤b.

We use upq to represent the U-line which passing the point (t_p, x[t_p]− ε) and the point (t_q, x[t_q]+ε).

Definition 2: Given a time series S[t_a:t_b]=< x[t_a], x[t_{a+1}],…, x[t_b]>, the line satisfying the following two conditions is L line which belongs to S[t_a:t_b].

(1) The straight line through point (t_p,x[t_p]+ε) and the point (t_q,x[t_q]−ε).

(2) a≤p≤q≤b.

We use lpq to represent the L-line which passing point(t_p,x[t_p]+ε) and the point (t_q, x[t_q]− ε).

Definition 3: Given a time series S[t_a:tb]=< x[t_a],x[t_{a+1}],…,x[t_b]>, l_{tk},2ε are segment connecting points (t_k, x[t_k]− ε) and (t_k,x[t_k]+ε), where a≤k≤b. We call l_{tk},2ε is a 2ε-bound line of S[t_a:t_b].

Definition 4: Given a time series $S[t_a:t_b]$=<x[ta],$x[t_{a+1}]$,…,$x[t_b]$>, U line upq of $S[t_a:t_b]$is UI line If and only if u_{pq} and each 2ε-bound line of $S[t_a:t_b]$ intersecting, that is

$$u_{pq} \cap l_{tk,2\varepsilon} \neq \Phi, a<k<b \quad (1)$$

Note: that the four lines and one line segment defined above are within the time series $S[t_a:t_b]$. For a better understanding of U-line, L-line, UI line, LI line, and the 2ε-bound line, we use a small example for a brief explanation. As shown in Fig. 1, there are three data points (t_k,$x[t_k]$) generated by a node, 1≤ k≤3. The three 2ε-bound lines (vertical thick line in the figure 1), three U-lines (solid line in the figure 1), and three L-lines (dashed lines in the figure 1). However, for $S[t_1 : t_3]$, there is only one UI line u12 of those three, there is only one LI line l23 too. We will prove that for any time series $S[t_a:t_b]$, it has at most one UI line and one LI line.

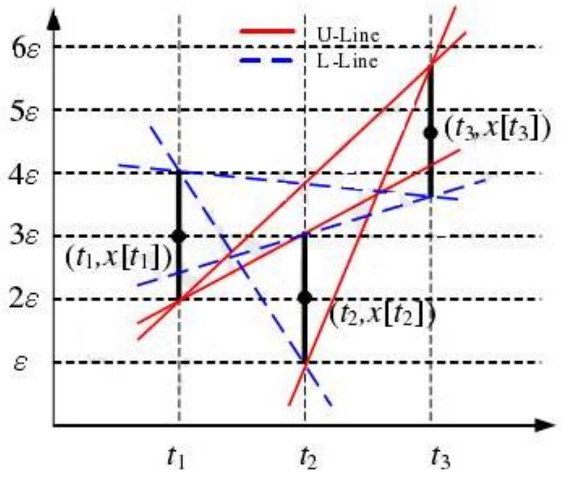


Figure 1. For S[t1:t3], there are Three 2-bound Line Segments, Three U-Lines and Three L-Lines

The main role of the UI and LI lines is to determine whether a new data point can be compressed with previously arrived data points that have not been compressed yet (namely, a line segment is used to approximate describe the points and guarantee the error limit ε).

3. Analysis of GDPLA Algorithm

The space-time complexity of the algorithm depends on four subroutines in the algorithm, namely four subroutines in the 15th, 16th, 17th and 18th lines. We analyze the complexity of Update And Prune Up(U,t_j,$x[t_j]$) and Calculate LILine (U,t_j,$x[t_j]$), and similarly we can analyze the complexity of Update And Prune Low(L,t_j,$x[t_j]$) and Calculate UILine (L,t_j,$x[t_j]$).

3.1. Space Complexity

Suppose that g(h){1≤h≤H} H line segment which is the outputs of $S[t_1:t_n]$ is approximate described by GDPLA, and g(m) is the line segment with the most approximate data points. Assuming that Num(g(m)) is the total number of data points approximated by g(m), then the maximum number of elements in U (L) is Num(g(m)), the space complexity of the algorithm is O (max1≤ m≤ H(Num(g(m)))). In the worst case, the entire time series needs only one-line segment to be approximate described, in this case. The space complexity is O(n). However, this situation rarely occurs, in most cases max1≤m≤H(Num(g(m))) is less than or equal to a constant, so in most cases, the space complexity is O(1).

3.2. Time Complexity

Definition 5: Given the time series $S[t_1:t_n]$, the time complexity of the algorithm GDPLA is O(n).

Proof: For Update And Prune Up $(U,t_j,x[t_j])$, the main runtime of algorithm depends on the while loop from the 4th to 8th lines. Assume that while loop runs M times, M_1 loop condition is satisfied and M_2 loop condition is not satisfied. Since each time a new data point is reached, only one loop condition is not satisfied, so $M_2=n$, thus $M=M_1+M_2=M_{1+n}$.When the M_1 loop condition is satisfied, it means that there are M_1 pruning operations occurred in the Update and Prune Up (U, t_j, $x[t_j]$) algorithm. In essence, we compare the slope of two lines to determine whether a point is the pruning point, to simplify the description, we use 1 represent slope size (that is, the size of the two floating-point) to compare the time complexity. The time complexity of Update And Prune Up $(U,t_j,x[t_j])$ algorithm is M_{1+n}. Similarly, the time complexity of the Update and Prune Low (U, t_j, $x[t_j]$) algorithm is $M_{1'+n}$. where $M_{1'}$ is the number of times the while loop condition is satisfied. The main runtime of the Calculate LILine $(U,t_j,x[t_j])$ algorithm depends on the while loop from the 5th to 10th lines[11]. Assume that the while loop runs N times, where N_1 loop condition is satisfied and N_2 loop condition is not satisfied. Since each time a new data point is reached, only one loop condition is satisfied, so $N_2=n$. Besides, from the overall point of view, when the LI line is updated, no upper pruning point is scanned, At most $n-M_1$ data points are scanned, so that $N_1\leq n-M_1$.Therefore the time complexity of the Calculate LILine $(U,t_j,x[t_j])$ algorithm is at most $2n-M_1$. Similarly, the time complexity of the Calculate UILine (L, t_j, $x[t_j]$) algorithm is a most $2n-M_{1'}$. In conclusion, the time complexity of GDPLA algorithm is $O(M_1+n+M_{1'}+n+2n-M_1+2n-M_{1'})=O(6n)$, namely O(n).

3.3. Optimality

From definition 5 we can see that GDPLA requires only a linear run-time. Following we will prove that the number of disconnected segments produced by the GDPLA algorithm is the least. Linear time and optimality are the two greatest advantages of this algorithm, GDPLA, which make it suitable for WSNs with limited resources.

Definition 6: Given the time series $S(n)=S[t_1:t_n]$, error limit ε and error function E(S(n), $S\tilde{S}$(n)), $S\tilde{S}$(n)is the approximate description of S(n). In all of the disconnected piecewise linear ε approximation algorithm which satisfy $E(S(n),S\tilde{S}(n))\leq\varepsilon$, if the number of disconnected segments generated by the algorithm GDPLA is H, there is no algorithm for piecewise linear approximation S (n) with less than H segments.

Proof: Counter-evidence. Suppose that there is a piecewise linear approximation algorithm approximates S (n) with a shorter length of disconnected line segments $(d_1,d_2,\ldots,d_{H'})(H'<H)$ and satisfies $E(S(n),S\tilde{S}(n))\leq\varepsilon$, where $S\tilde{S}$(n) is the time series obtained by approximating S (n) with$(d_1,d_2,\ldots,d_{H'})$. The line d1 the approximate subsequence $<x[t_1],x[t_2],\ldots,x[t_{l1}]>$. Line segment dh′($2\leq h'\leq H'$) approximate subsequence $x[t_{lh-1+}1],x[t_{lh-1+2}],\ldots, x[t_{lh}]>$. Assume that the algorithm GDPLA produces disconnected segment sequence c1, c2,…, cH), where segment c1 approximates the subsequences$<x[t_1],x[t_2],\ldots,x[t_{j1}]$ >, segment ch($2\leq h\leq H$) approximates subsequences $<x[t_{jh-1}+1],x[t_{jh-1}+2],\ldots, x[t_{jh}]>$; where $j_H=n$.

By mathematical induction we prove that for $\forall 1\leq m\leq H'-1$, $l_{H'-m}>j_{H-(m+1)}$ holds.

(1) When m=1, it is easy to know that $l_{H'-1}>j_{H-2}$ holds. Otherwise, if $l_{H'-}1\leq j_{H-2}$, the subsequence $S[t_{jH-2}:t_{jH}]\subseteq S[t_{jH`-1}+1:t_{jH1}]$, the subsequences $S[t_{jH-2}:t_{jH}]$ can be approximated by $d_{H'}$. and the error is not greater than ε. However, $S[t_{jH-2}:t_{jH}]\subseteq(S[t_{jH`-1}+1:t_{jH1}])$ can’t be approximated by one line segment and ensured error is less than or equal to ε. $S[t_{jH-2}:t_{jH}]$ can’t be approximated by a line segment and ensured error is less than or equal to ε. Draw the contradiction, when m=1, $l_{H'-1}>j_{H-2}$ holds.

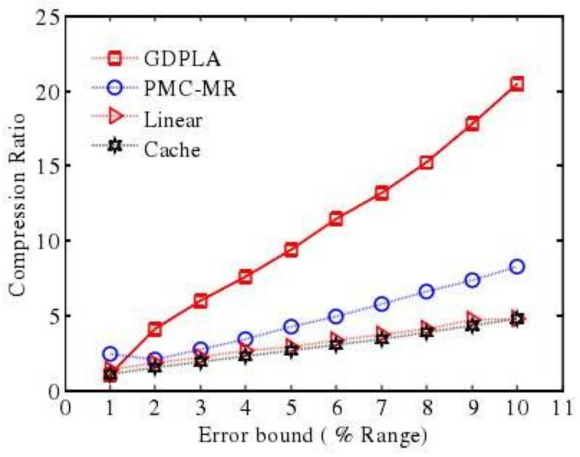
(2) Suppose $m=k<H'$, $l_{H'-k}> j_{H-(k+1)}$ holds. Then $l_{H'-(k+1)}>j_{H-(k+2)}$ holds too. Otherwise, $l_{H'-(k+1)}\leq j_{H-(k+2)}$, By the induction hypothesis that $l_{H'-(k+1)}\leq j_{H-(k+2)}<j_{H-(k+1)}<l_{H'-k}$, and $S[t_{jH-(k+2)}:t_{jH-(k+1)}]\subseteq S[t_{jH`(k+1)} +1:t_{jH`-k}]$, $S[t_{jH-(k+2)}:t_{jH-(k+1)}]$ can be approximated by $d_{H'-k}$ and the error is less than or equal to ε. But from the algorithm, $S[t_{jH`(k+1)} +1:t_{jH`-k}]$ can't be approximated by a line segment and ensured error is less than or equal to ε. Draw the contradiction, thus $l_{H'-(k+1)}> j_{H-(k+2)}$ holds.

Seen from (1) and (2), $\forall 1\leq m\leq H'-1$, $l_{H'-m+1} > j_{H-(m+1)}$ holds. Specially, when $m = H'-1$, $l1>j_{H-H'}$ holds, then $S[t_1:t_{jH-H`}]\subseteq S[t_1+1:t_{l1}]$, $S[t_1:t_{j1}]\subseteq S[t_1:t_{H-H`}]$ can be approximated by d_1 and the error is less than or equal to ε. But from the algorithm, $S[t_1:tj_1]$ can't be approximated by a line segment and ensured error is less than or equal to ε[12]. Draw the contradiction, so there is no other piecewise linear algorithm approximating S (n) with a sequence of lengths less than H and satisfying $E(S(n), S\tilde{S}(n))\leq\varepsilon$. End of proof.

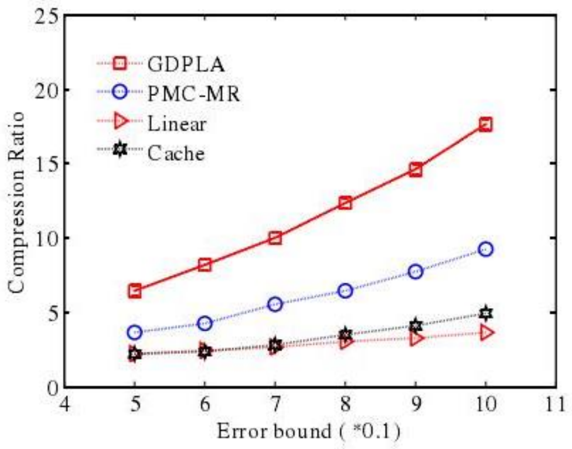
4. Experimental Results

4.1. Contrast Results of Compression Ratio

As shown in Figure 2, the first set of experiments measures the variation of the compression rate with error ε. Figure 2(a) describes the variation of compression ratio of GDPLA, PMA-MR, Cache, and Linear filter with error ε when the original data size of the Intel TEM dataset is 8000. As expected, for all algorithms, the compression ratio increases with error ε. When ε=0.1, the compression ratio of GDPLA is 2.5, 4.2 and 5.0 times of PMA-MR, Cache and Linear filter respectively. The reason is that GDPLA is an optimal algorithm, which uses a minimum number of segments to describe the time series in a piecewise linear approximation. When ε=0.05, GDPLA only needs to transfer about 10% of the original data, that means the compression rate is about 10; when ε = 0.1, GDPLA only need to transfer about 5% of the original data, the compression rate is about 20. When the data size of Wash HUM is 10000, Figure 2(b) describes the variation of compression ratio of GDPLA, PMA-MR, Cache, and Linear filter with error ε. The observed experimental results are similar to those shown in Figure 2(a). When ε = 0.5, the compression ratio of GDPLA is 1.75, 2.87 and 2.88 times of PMA-MR, Cache and Linear filter respectively. We observe that GDPLA's compression ratio increase is larger than PMA-MR, Cache, and Linear filter. When ε = 1, the compression ratio of GDPLA is 1.9, 4.81 and 3.54 times of PMA-MR, Cache and Linear filter respectively. Figure 3 (a) depicts how the compression ratio of GDPLA, PMA-MR, Cache and Linear filter changes with the size of the Intel TEM dataset when ε=0.05. For all algorithms, the compression rate decreases as the size of the data set increases. This may be related to the distribution of the data. Figure 3 (b) shows the variation of compression ratio of GDPLA, PMA-MR, Cache, and Linear filter with the size of WashTEM data set, when ε = 0.5. It can be seen that the compression rate of all the algorithms changes slowly. From Figure.2 and Figure.3, we can see that the compression ratio of GDPLA is always larger than that of PMA-MR, Cache and Linear filter. This further validates our theoretical analysis.

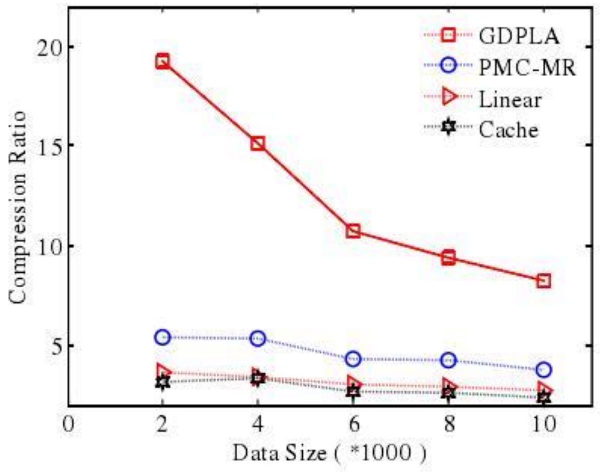


(a) Compression ratio vs. ε (IntelTEM)

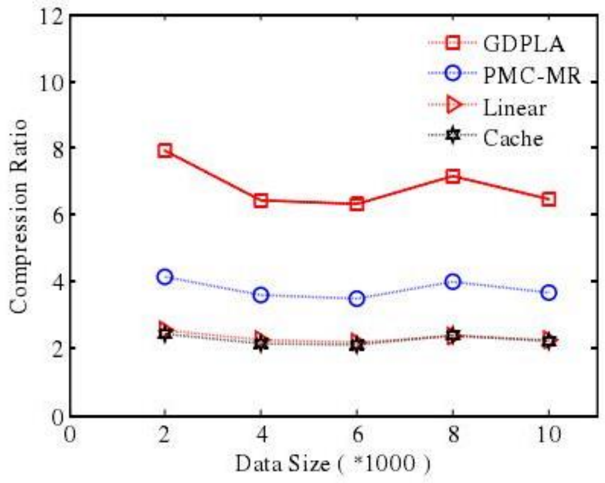


(b) Compression ratio vs. ε(WashHUM)

Figure 2. Compression Ratio as a Function of the Error Bound ε



(a)compression ratio vs. data size (IntelTEM)



(b) compression ratio vs. data size(WashHUM)

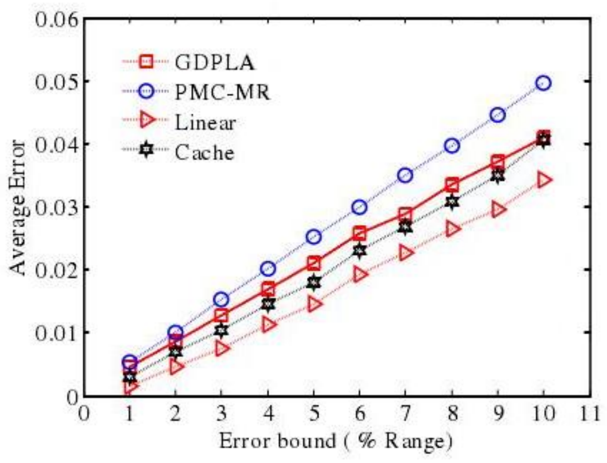
Figure 3. Compression Ratio as a Function of Data Size

4.2. Contrast Results of Average Error

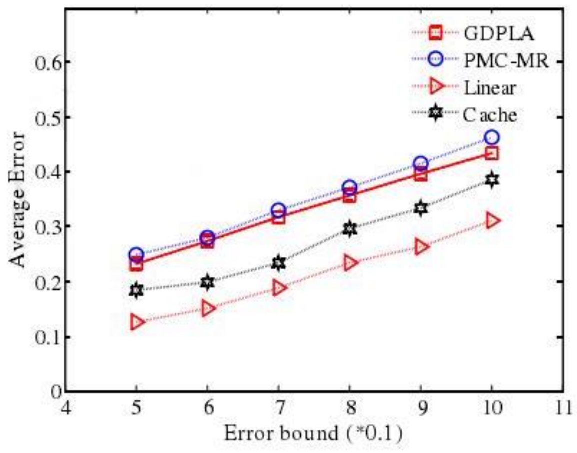
Figure.4 (a) describes the variation of average error of GDPLA, PMA-MR, Cache and Linear filter with error ε when the original data size of the Intel TEM dataset is 8000. From this we can observe that for all algorithms, the average error is approximately proportional to the error ε, and that the average error of all the algorithms is less than half of the maximum absolute value error. Figure 4(b) depicts similar experimental results. Figure 5 (a) depicts the variation of the average error of GDPLA, PMA-MR, Cache, and Linear filter when ε=0.05 and the size of the Intel TEM dataset increases from 2000 to 10000. From the figure we can observe that for all algorithms, the average error is less than 0.03, and when the size of the Intel TEM dataset increases from 2000 to 10000, the average error does not change much. The average error of GDPLA is slightly higher than the average error of Cache, which is lower than the average error of PMC-MR. Figure 5 (b) shows the variation of the average error of GDPLA, PMA-MR, Cache and Linear filter with the size of WashTEM dataset when ε=0.5, we can see that the average error of the four algorithms is less than 0.3, and when the size of the Wash HUM dataset increases from 2000 to 10000, the average error value increases gradually, but does not exceed 0.55 times the maximum error.

4.3. Contrast Results of Processing Time

Figure 5(a) depicts the average processing time for each data point in the Intel TEM dataset when the size of the Intel TEM dataset is 8000 and ε is incremented from 0.01 to 0.1. The average processing time of GDPLA for each data point is slightly higher than the other three algorithms. Figure 5 (b) depicts the average processing time for each data point in the Wash HUM data set when ε is gradually change from 0.5 to 1 and the size of the Wash HUM data set is 10000. Similarly, the average processing time of GDPLA for each data points is slightly higher than the other three algorithms. However, the computational energy cost of the algorithm GDPLA is worth because the energy cost of WSNs depends mainly on the communication overhead, moreover the compression ratio of GDPLA is far more than that of other algorithms, so GDPLA will transmit much less data than the other algorithms, which makes GDPLA communication overhead is far less than other algorithms communication overhead.

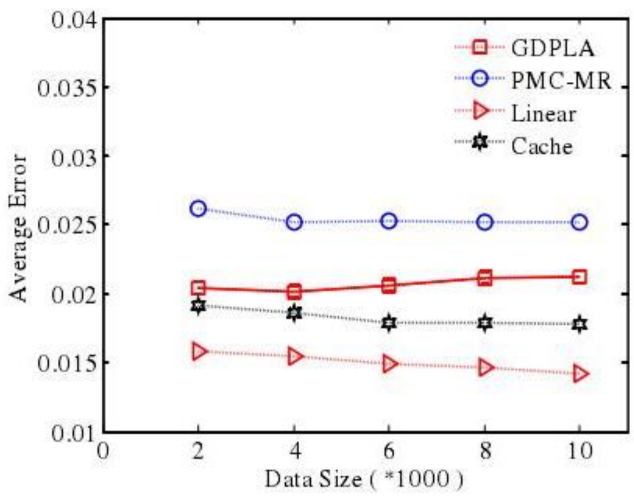


(a) Average error vs. ε (IntelTEM

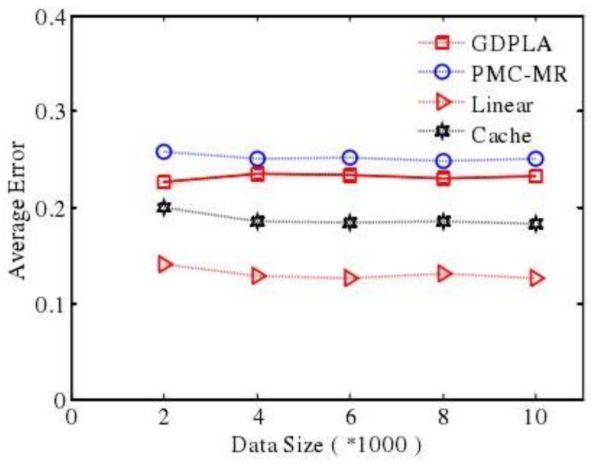


(b) Average error vs. ε(WashHUM)

Figure 4. Average Error as a Function of the Error Bound ε



(a) Average error vs.data size(IntelTEM)



(b) Average error vs.data size(WashHUM)

Figure 5. Average Error as a Function of Data Size

5. Conclusion

This paper proposes a piecewise linear compression algorithm, GDPLA. The main task of the wireless sensor network is to collect the data, timely and accurate feedback to the user. In most wireless sensor network applications, the data is continuously transferred from the data source node to the sink node. As the energy of the wireless sensor network node is very limited and wireless data transmission is a major part of node energy consumption, so under the premise of without damaging the network application tasks, how to reduce the amount of data transmission is an important research topic. To solve this problem, we propose an online piecewise linear approximation algorithm, GDPLA, to reduce the amount of data transmission from the source. Under the condition of guaranteeing the error limit ε, even if the absolute value of error between the true value and the approximate value of each data point is less than or equal to the user-specified threshold ε, the original time series is approximate described by using the least disconnected line segments. We prove the optimality of the algorithm, that is, the number of line segments generated by the algorithm GDPLA is the least in the algorithm which approximate describing the time series by disconnected segments. In addition, we also prove that the time complexity of the algorithm GDPLA is O (n), where n is the length of the time series. The low time complexity of the algorithm makes it suitable for wireless sensor networks with limited resources. Finally, the experiments on two real data sets show that the compression rate of our algorithm is better than other algorithms. Thus, it has greatly reducing the amount of data transmission in the wireless sensor network from the source.

References

[1] J. N. Alkaraki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey", IEEE Wireless Communications, vol. 11, no. 6, **(2004)**, pp. 6-28.
[2] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva, "Directed diffusion for wireless sensor networking", IEEE/ACM Transactions on Networking, vol. 11, no. 1, **(2013)**, pp. 2-16.
[3] W. Ye, J. Heidemann and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks", IEEE/ACM Transactions on Networking, vol. 12, no. 3, **(2004)**, pp. 493-506.
[4] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures", Ad Hoc Networks, vol. 1, no. 2–3, **(2003)**, pp. 293-315.
[5] K. Sohrabi, J. Gao, V. Ailawadhi and G. J. Pottie, "Protocols for self-organization of a wireless sensor network", Personal Communications IEEE, vol. 7, no. 5, **(2000)**, pp. 16-27.
[6] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks", IEEE Signal Processing Magazine, vol. 22, no. 4, **(2005)**, pp. 54-69.

[7] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks", Acm Transactions on Information & System Security, vol. 8, no. 2, **(2005)**, pp. 228-258.

[8] H. Zheng, S. Xiao and X. Wang, "Energy conservation in wireless sensor networks: a survey", Ad Hoc Networks, vol. 7, no. 3, **(2009)**, pp. 537-568.

[9] J. Kulik, W. Heinzelman and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks", Wireless Networks, vol. 8, no. 2, **(2002)**, pp. 169-185.

[10] D. Ganesan, R. Govindan, S. Shenker and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks", Acm Sigmobile Mobile Computing & Communications Review, vol. 5, no. 4, **(2001)**, pp. 72-85.

[11] O. Boyinbode, H. Le, A. Mbogho, M. Takizawa and R. Poliah, "A survey on clustering algorithms for wireless sensor networks", Computer Communications, vol. 30, no. 14–15, **(2007)**, pp. 2826-2841.

[12] N. Patwari, A. O. I. Hero, M. Perkins, N. S. Correal and R. J. O'Dea, "Relative location estimation in wireless sensor networks", IEEE Transactions on Signal Processing, vol. 51, no. 8, **(2003)**, pp. 2137-2148.