Data Compression in Multi-Hop Large-Scale Wireless Sensor Networks

Yimei Li
Purdue University

Follow this and additional works at: https://docs.lib.purdue.edu/open_access_dissertations

Recommended Citation
https://docs.lib.purdue.edu/open_access_dissertations/1756

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
DATA COMPRESSION IN MULTI-HOP LARGE-SCALE WIRELESS SENSOR NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Yimei Li

In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

May 2018

Purdue University

West Lafayette, Indiana
THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Yao Liang, Co-Chair
Department of Computer Science

Dr. Sonia Fahmy, Co-Chair
Department of Computer Science

Dr. Mohammad Al Hasan
Department of Computer Science

Dr. Zhiyuan Li
Department of Computer Science

Dr. He Wang
Department of Computer Science

Approved by:

Dr. Voicu Popescu by Dr. William J. Gorman
Head of Department Graduate Program
To my parents Shulan and Changsheng, and my daughter Joy, for their love and support.
ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my adviser, Professor Yao Liang, for his excellent guidance and unconditional support throughout my graduate study. Without his instruction and encouragement, this thesis would not have reached its current form.

I am also very grateful to many faculty members at IUPUI and PUWL for their help and contribution to my thesis work. My committee co-chair, Professor Sonia Fahmy, and committee members, Professor Mohammad Al Hasan, Professor Zhiyuan Li and Professor He Wang, were always a good resource for discussion and inspiration. I would like to thank them for their time and efforts in discussing the research problems with me, and for their challenging questions which have significantly enhanced many aspects of this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>xii</td>
</tr>
<tr>
<td><strong>1</strong> INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 WSN for Data Collection</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Major Contributions</td>
<td>3</td>
</tr>
<tr>
<td>1.3.1 Compressed Sensing Approach</td>
<td>3</td>
</tr>
<tr>
<td>1.3.2 A Generalized Predictive Coding Framework</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Organization</td>
<td>4</td>
</tr>
<tr>
<td><strong>2</strong> RELATED WORK</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Conventional Compression</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Compressed Sensing</td>
<td>9</td>
</tr>
<tr>
<td><strong>3</strong> MATHEMATICAL BACKGROUNDS</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Compressed Sensing</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Wavelets and Lifting Scheme</td>
<td>12</td>
</tr>
<tr>
<td><strong>4</strong> THE DESIGN OF MEASUREMENT MATRIX AND REPRESENTATION BASIS</td>
<td>15</td>
</tr>
<tr>
<td>4.1 Problem Formulation</td>
<td>16</td>
</tr>
<tr>
<td>4.2 Measurement Matrix</td>
<td>18</td>
</tr>
<tr>
<td>4.3 Representation Basis</td>
<td>20</td>
</tr>
<tr>
<td>4.3.1 GLE Algorithm</td>
<td>22</td>
</tr>
<tr>
<td>4.3.2 Construction of Underlying Graph</td>
<td>24</td>
</tr>
<tr>
<td><strong>5</strong> VALIDATION</td>
<td>26</td>
</tr>
<tr>
<td>5.1 Validation in ASWP Testbed</td>
<td>26</td>
</tr>
<tr>
<td>5.1.1 ASWP WSN Testbed and Deployment</td>
<td>26</td>
</tr>
<tr>
<td>5.1.2 Performance of Representation Basis</td>
<td>28</td>
</tr>
<tr>
<td>5.1.3 Signal Recovery Accuracy</td>
<td>33</td>
</tr>
<tr>
<td>5.2 Simulation in Large Scale Network</td>
<td>41</td>
</tr>
<tr>
<td>5.2.1 Compare with Different CS Approaches</td>
<td>42</td>
</tr>
<tr>
<td>5.2.2 Reconstruction Performance with Loops</td>
<td>44</td>
</tr>
<tr>
<td>5.3 Summary</td>
<td>47</td>
</tr>
</tbody>
</table>
### 6 LONG TERM EVALUATION IN THE OUTDOOR TESTBED
- 6.1 Different Training Size ........................................ 48
- 6.2 Different Laplacian Matrix ..................................... 49
- 6.3 Long Term Fitting .................................................. 52
- 6.4 Compare with Different CS Approaches ....................... 52
- 6.5 A Self-tuned approach in the Network ....................... 52

### 7 A GENERALIZED PREDICTIVE CODING FRAMEWORK
- 7.1 Algorithmic Framework .......................................... 60
  - 7.1.1 Two-Model Transmission ................................... 60
  - 7.1.2 Lossy Compression ........................................... 61
  - 7.1.3 GPC Framework .............................................. 64
- 7.2 Sequential Lossless Entropy Compression (S-LEC) .......... 67
  - 7.2.1 Overview of LEC ............................................ 67
  - 7.2.2 S-LEC Algorithm ........................................... 68
- 7.3 Performance Analyses ............................................ 70
  - 7.3.1 Residual Distribution Model ............................... 70
  - 7.3.2 Compression Radius ....................................... 71
  - 7.3.3 Lossy Compression Performance ......................... 74
- 7.4 Simulations and Evaluations ................................... 76
  - 7.4.1 Data Sets .................................................... 77
  - 7.4.2 Lossless Compression ..................................... 78
  - 7.4.3 Lossy Compression ......................................... 88
- 7.5 Energy Consumption Model and Reliable Transmission ...... 92
  - 7.5.1 Energy Consumption Model ................................. 92
  - 7.5.2 Reliable Transmission ..................................... 99

### 8 SUMMARY AND FUTURE WORK ............................... 102
- 8.1 Summary .......................................................... 102
- 8.2 Future Work ....................................................... 103

REFERENCES ............................................................ 104
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>The statistics of per-packet routing path recovery by RTR scheme in ASWP WSN</td>
</tr>
<tr>
<td>5.2</td>
<td>The differences between two networks</td>
</tr>
<tr>
<td>6.1</td>
<td>Training dataset information</td>
</tr>
<tr>
<td>7.1</td>
<td>Entropy coding table in LEC with $K = 14$</td>
</tr>
<tr>
<td>7.2</td>
<td>Sequential coding in S-LEC</td>
</tr>
<tr>
<td>7.3</td>
<td>Group code reduction</td>
</tr>
<tr>
<td>7.4</td>
<td>Entropy coding table in LEC with $K = 14$</td>
</tr>
<tr>
<td>7.5</td>
<td>Lossless compression performance on SensorScope data (block size: 264 samples)</td>
</tr>
<tr>
<td>7.6</td>
<td>Lossless compression performance on SensorScope data (block size: 32 samples)</td>
</tr>
<tr>
<td>7.7</td>
<td>Entropy coding table in LEC with $K = 24$</td>
</tr>
<tr>
<td>7.8</td>
<td>Entropy coding table in GPC/S-LEC with $M = 17$</td>
</tr>
<tr>
<td>7.9</td>
<td>Lossless compression performance on Volcano data (block size: 24 samples)</td>
</tr>
<tr>
<td>7.10</td>
<td>Lossless compression performance on Volcano data (block size: 24 samples)</td>
</tr>
<tr>
<td>7.11</td>
<td>Lossy Compression Performance on Volcano Data (block size: 24 samples)</td>
</tr>
<tr>
<td>7.12</td>
<td>Parameters of the energy models</td>
</tr>
<tr>
<td>7.13</td>
<td>Lossless energy comparisons (block size: 264 samples)</td>
</tr>
<tr>
<td>7.14</td>
<td>Lossless communication/computation energy tradeoff ratio (block size: 264 samples)</td>
</tr>
<tr>
<td>7.15</td>
<td>Lossy energy comparisons (block size: 32 samples; $e = 50%$SME)</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>2.1</td>
<td>Traditional predictive coding and decoding procedure.</td>
</tr>
<tr>
<td>3.1</td>
<td>Lifting scheme: one step of forward transform. Here, $a_l$ and $d_l$ denote the vectors of all approximation and detail coefficients of the lifted transform at level $l$. $U$ and $P$ are linear update and predict operators. HT and IHT are the Haar transform and its inverse.</td>
</tr>
<tr>
<td>4.1</td>
<td>Overview of our CSR design.</td>
</tr>
<tr>
<td>4.2</td>
<td>(a) An illustration of sensor network upward routing topology for data collection: solid circles (the sink and leaves) are the boundary nodes and dash circles are the intermediate nodes. (b) Bipartite graph corresponding to given routing matrix in (4.1).</td>
</tr>
<tr>
<td>4.3</td>
<td>An illustration of multiresolution decomposition of 1-dimensional space into a hierarchical structure.</td>
</tr>
<tr>
<td>5.1</td>
<td>Examples of node configurations in the ASWP testbed. A node hanging from a tree without external sensors (left) and a node mounted onto a PVC pipe with external sensors attached (right).</td>
</tr>
<tr>
<td>5.2</td>
<td>Location of the nodes in ASWP WSN testbed, sink locates at the red larger spot in the cabin house(Blue: Micaz; Yellow: TelosB; Red: IRIS).</td>
</tr>
<tr>
<td>5.3</td>
<td>Humidity raw signals (top) and the corresponding transform coefficients for our constructed representation basis using the GLE algorithm (bottom).</td>
</tr>
<tr>
<td>5.4</td>
<td>Soil moisture raw signals (top) and the corresponding transform coefficients for our constructed representation basis using the GLE algorithm (bottom).</td>
</tr>
<tr>
<td>5.5</td>
<td>The sparsification error of humidity datasets estimated by using only the largest $k$ components in magnitude in $s$.</td>
</tr>
<tr>
<td>5.6</td>
<td>The sparsification error of soil moisture datasets estimated by using only the largest $k$ components in magnitude in $s$.</td>
</tr>
<tr>
<td>5.7</td>
<td>An example of the routing path (measurement) when M=12.</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>5.8</td>
<td>34</td>
</tr>
<tr>
<td>5.9</td>
<td>35</td>
</tr>
<tr>
<td>5.10</td>
<td>35</td>
</tr>
<tr>
<td>5.11</td>
<td>36</td>
</tr>
<tr>
<td>5.12</td>
<td>37</td>
</tr>
<tr>
<td>5.13</td>
<td>37</td>
</tr>
<tr>
<td>5.14</td>
<td>38</td>
</tr>
<tr>
<td>5.15</td>
<td>38</td>
</tr>
<tr>
<td>5.16</td>
<td>39</td>
</tr>
<tr>
<td>5.17</td>
<td>39</td>
</tr>
<tr>
<td>5.18</td>
<td>40</td>
</tr>
<tr>
<td>5.19</td>
<td>40</td>
</tr>
<tr>
<td>5.20</td>
<td>42</td>
</tr>
<tr>
<td>5.21</td>
<td>43</td>
</tr>
<tr>
<td>5.22</td>
<td>43</td>
</tr>
<tr>
<td>5.23</td>
<td>44</td>
</tr>
<tr>
<td>5.24</td>
<td>44</td>
</tr>
<tr>
<td>5.25</td>
<td>45</td>
</tr>
<tr>
<td>5.26</td>
<td>46</td>
</tr>
</tbody>
</table>
### Figure

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Humidity dataset reconstruction errors with different training size: 10, 20, 50, 100, using SL0 as the solver, while M=20.</td>
</tr>
<tr>
<td>6.2</td>
<td>Humidity dataset reconstruction errors with different training size: 10, 20, 50, using SL0 as the solver, while M=20.</td>
</tr>
<tr>
<td>6.3</td>
<td>Humidity dataset reconstruction errors with different Laplacian matrix or Deep Learning scales, using SL0 as the solver, training size is 20, M=20.</td>
</tr>
<tr>
<td>6.4</td>
<td>Performance of CSR in cycle 1 (collected on 2017-08-12). Top: measurement matrix, the point means the entry value is 1; otherwise, 0. Middle: the reconstruction errors of each node. Bottom: the recovered humidity sensor reading.</td>
</tr>
<tr>
<td>6.5</td>
<td>Performance of CSR in cycle 350 (collected on 2017-10-28). Top: measurement matrix, the point means the entry value is 1; otherwise, 0. Middle: the reconstruction errors of each node. Bottom: the recovered humidity sensor reading.</td>
</tr>
<tr>
<td>6.6</td>
<td>Humidity dataset reconstruction errors with different $M$, using SL0 as the solver. The reconstruction error is the average value of 350 cycles.</td>
</tr>
<tr>
<td>6.7</td>
<td>Humidity dataset reconstruction errors with different $M$, using LP as the solver. The reconstruction error is the average value of 350 cycles.</td>
</tr>
<tr>
<td>6.8</td>
<td>Humidity dataset reconstruction errors with different $M$, using SL0 as the solver. The reconstruction error is from the 350th cycle.</td>
</tr>
<tr>
<td>6.9</td>
<td>Humidity dataset reconstruction errors with different $M$, using SL0 as the solver. The reconstruction error is from the 350th cycle.</td>
</tr>
<tr>
<td>6.10</td>
<td>Distribution of packets number.</td>
</tr>
<tr>
<td>7.1</td>
<td>TMT generalization of predicting coding and decoding procedure.</td>
</tr>
<tr>
<td>7.2</td>
<td>BLC generalization of predicting coding and decoding procedure.</td>
</tr>
<tr>
<td>7.3</td>
<td>Illustration of residue distribution models using L1 vs. L2 for SensorScope ambient temperature data sets.</td>
</tr>
<tr>
<td>7.4</td>
<td>The ideal lossless compression ratio for Laplacian distribution with zero mean under different $b$ values.</td>
</tr>
<tr>
<td>7.5</td>
<td>Comparison of compression ratios between ideal entropy coding and GPC/LEC entropy coding with different $b$ values ranging from 10 to 30.</td>
</tr>
<tr>
<td>7.6</td>
<td>Illustration of residue distribution models using L1 for SensorScope ambient temperature data sets.</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>7.7</td>
<td>The GPC/LEC lossy compression ratio for Laplacian distribution with zero mean under different $b$ values.</td>
</tr>
<tr>
<td>7.8</td>
<td>The distribution of the compression packet length (ambient temperature).</td>
</tr>
<tr>
<td>7.9</td>
<td>The distribution of the compression packet length (relative humidity).</td>
</tr>
<tr>
<td>7.10</td>
<td>The distribution of the compression packet length (seismic).</td>
</tr>
<tr>
<td>7.11</td>
<td>Lossy compression ratio under various error bound $e$ for SensorScope ambient temperature data set.</td>
</tr>
<tr>
<td>7.12</td>
<td>Lossy compression ratio under various error bound $e$ for SensorScope relative humidity data set.</td>
</tr>
<tr>
<td>7.13</td>
<td>Lossy compression ratio under various error bound $e$ for Node 203 data set.</td>
</tr>
<tr>
<td>7.14</td>
<td>Lossy compression ratio under various error bound $e$ for Node 251 data set.</td>
</tr>
<tr>
<td>7.15</td>
<td>Average performance of different predictors for all the 10 volcano data set.</td>
</tr>
<tr>
<td>7.16</td>
<td>Data transmission ratios under various BER with a frame size of 48 bytes.</td>
</tr>
<tr>
<td>7.17</td>
<td>Energy consumption ratio with a frame size of 48 bytes when BER is $10^{-2}$.</td>
</tr>
</tbody>
</table>
ABSTRACT

Li, Yimei Ph.D., Purdue University, May 2018. Data Compression in Multi-Hop Large-Scale Wireless Sensor Networks. Major Professor: Yao Liang.

Data collection from a multi-hop large-scale outdoor WSN deployment for environmental monitoring is full of challenges due to the severe resource constraints on small battery-operated motes (e.g., bandwidth, memory, power, and computing capacity) and the highly dynamic wireless link conditions in an outdoor communication environment. We present a compressed sensing approach which can recover the sensing data at the sink with good accuracy when very few packets are collected, thus leading to a significant reduction of the network traffic and an extension of the WSN lifetime. Interplaying with the dynamic WSN routing topology, the proposed approach is efficient and simple to implement on the resource-constrained motes without motes storing of a part of random measurement matrix, as opposed to other existing compressed sensing based schemes. We provide a systematic method via machine learning to find a suitable representation basis, for the given WSN deployment and data field, which is both sparse and incoherent with the measurement matrix in the compressed sensing. We validate our approach and evaluate its performance using our real-world multi-hop WSN testbed deployment in situ in collecting the humidity and soil moisture data. The results show that our approach significantly outperforms three other compressed sensing based algorithms regarding the data recovery accuracy for the entire WSN observation field under drastically reduced communication costs.

For some WSN scenarios, compressed sensing may not be applicable. Therefore we also design a generalized predictive coding framework for unified lossless and lossy data compression. In addition, we devise a novel algorithm for lossless compression to significantly improve data compression performance for various data collections
and applications in WSNs. Rigorous simulations show our proposed framework and compression algorithm outperform several recent popular compression algorithms for wireless sensor networks such as LEC, S-LZW and LTC using various real-world sensor data sets, demonstrating the merit of the proposed framework for unified temporal lossless and lossy data compression in WSNs.
1 INTRODUCTION

1.1 Motivation

Wireless Sensor Networks (WSNs), comprised of spatially distributed sensor nodes, are being increasingly deployed for continuous monitoring and sensing of physical variables of our world [1–4]. One of the critical challenges in large-scale outdoor WSN deployments is the energy consumption, since outdoor sensor nodes are mainly operated by battery power. Motivated by the breakthrough of compressed sensing (CS) [5, 6] in signal processing, CS based approaches for WSN data collection gain increasing attention from the research communities (e.g., [7–14]). However, existing CS solutions for WSNs are challenged by the following major difficulties in practice: (1) How to effectively and efficiently interplay with WSN routing so that per-packet routing path can be exploited as a random projection in CS measurement matrix to further reduce nodes transmissions? (2) How to design a suitable representation basis for real-world signals that satisfies good sparsification and incoherency with the measurement matrix in a WSN that interplay with dynamic routing? As Quer et al. put it finding a suitable transformation with good sparsification and incoherence properties remains an open problem [7]. Furthermore, although existing CS approaches for WSN data collection are evaluated with real sensor data, they are not tested with multi-hop WSN deployments in situ operated in real-world dynamic communication environments but rather by numerical simulations with the assumptions of some routing models. While useful, numerical simulations alone are not adequate. Thus, the need of practical validation and evaluation of CS approaches in real WSNs in situ is also urgent.

One goal of this work is to address the above challenges in CS for WSN data collection, and to provide a practical and efficient CS solution for real-world WSN
deployments. We focus on outdoor multi-hop WSNs in situ where communication environment is highly dynamic and harsh.

A limitation of CS is that it only works for the applications which are tolerable with lossy compression. It is important to note that both lossless and lossy compressions are necessary in energy-efficient WSNs. In many exploratory or reactive control tasks, the accuracy of observations is often critical in understanding the underlying physical processes. For any given precision of Analog to Digital Converters (ADCs) adopted in WSNs, lossless data gathering would be essential, particularly at the initial phase of WSN deployment and data collection. Here, it is critical that the compressed sensor data fully reserve the precision of the adopted ADCs in the WSNs without introducing any additional compression errors, while at the same time reduce the WSN energy consumption through lossless data compression. On the other hand, once scientists and engineers have a better understanding of the observed physical processes/phenomena over time, they gain more knowledge about what error margins of observation could be tolerable with regard to the missions and needs of the given tasks. By exploiting such tolerable data error margins, lossy compression schemes can then be considered in order to achieve a higher compression ratio than that of lossless compression schemes, which in turn further maximizes the lifetime of wireless motes.

1.2 WSN for Data Collection

Wireless sensor networks (WSNs) are being used in a broad variety of domains including scientific, medical, commercial and military applications. There are a wide variety of applications for WSNs including environmental monitoring, overseeing of smart homes and offices, and managing of intelligent transportation systems. A key feature of all WSNs is that the individual wireless sensors are collecting data. These data can range from simple temperature readings in a smart home, to soil moisture readings relating in an extreme outdoor environment. With the advanced state of
wireless technology and computers in general, scientists are able to use WSNs to make data gathering from N sensors that are constantly taking measurements of their environment or surroundings. The sensors in WSNs are generally low-processing and low-power units which means special attention must be paid to the processing and energy consumption. WSNs allow for continually monitoring of an area at a rate much higher than would be possible by sending a person to physically gather data from the site.

Constantly monitoring and transmitting data to the base station can put a burden on the sensor nodes with limited power resources and therefore we need to minimize sensor data transmissions. Another aspect of most WSNs that can be exploited is the fact that it is highly probable that sensors within a certain radius of each other will be collecting highly correlated data readings that can be exploited later on in an effort to gain information on all sensor readings without having to explicitly get data from each sensor.

1.3 Major Contributions

This dissertation studies the problem of data compression in multi-hop large-scale wireless sensor networks for environmental monitoring. This dissertation presents a compressed sensing approach which can recover the sensing data at the sink with good accuracy when very few packets are collected. This dissertation also presents a generalized predictive coding framework for unified lossless and lossy data compression, which is suitable for single-hop WSN or the beginning data collection phase to study the feature of the sensor data before compressed sensing can be applied.

1.3.1 Compressed Sensing Approach

A novel compressed sensing approach is presented for multi-hop large-scale and dynamic WSNs in situ for data collection based on network topology tomography. A systematic method is proposed, based on deep learning, to find an optimized repre-
sentation basis which is both sparse and incoherent with the measurement matrix in
the CS. The approach is validated and evaluated in an environmental multi-hop WSN
deployment in a watershed, operating with TinyOS and an extended Collection Tree
Protocol (CTP), called CTP+EER. To the best of our knowledge, this represents the
first experiment of CS approach conducted on a real-world outdoor WSN in situ with
the deployed routing protocol and routing topology tomography for data collection.

1.3.2 A Generalized Predictive Coding Framework

A generalized predictive coding (GPC) framework is presented, which can per-
form unified lossless and lossy temporal compressions effectively and efficiently in a
systematic manner. This is desirable particularly for WSNs. Due to the extremely
limited resource on tiny motes, the implementation of two separate lossless and lossy
schemes on every mote is clearly not resource-efficient. The algorithmic framework
is quite simple as its additional complexity compared to the traditional predictive
coding is negligible, making it well suited for tiny motes in WSN data gathering.
In addition, a novel lossless data compression algorithm, called Sequential Lossless
Entropy Compression (S-LEC), is devised. To our knowledge, the proposed innova-
tion for the unification of both lossless and lossy compression is the first of its kind
for general, sustainable, and extensible WSN data collections. Moreover, we further
investigate the error distributions of sensor data prediction and propose a discrete
Laplacian distribution model which can better characterize those prediction errors.

1.4 Organization

This dissertation is organized as follows. Chapter 2 describes the related work.
Chapter 3 introduces the background knowledge of compressed sensing and wavelets.
Chapter 4 present our design of both the representation basis and the measurement
matrix in the compressed sensing approach. Chapter 5 validates our CS approach in
a real-world outdoor WSN testbed. Chapter 6 evaluates the performance of our CS
approach from both a long term deployment in the ASWP testbed and the simulation of a large scale network. Chapter 7 gives the design of our generalized predictive coding framework.
2 RELATED WORK

Wireless sensor networks (WSNs) are being increasingly deployed for enabling continuous monitoring and sensing of physical variables of our world. Energy efficiency and communication bandwidth are two critical challenges in the design and deployment of WSNs. First, sensor nodes in WSNs are typically battery-operated, in many real physical environments the replacement of batteries for sensor nodes is virtually impossible. Consequently, failure of a subset of sensor nodes due to their power depletion can result in the failure of the entire WSN. Secondly, even in those outdoor WSN deployments where the replacement of batteries is possible, the improved WSN energy efficiency can significantly reduce battery replacement frequencies and thus the WSN maintenance cost. Third, the low-power radios adopted in sensor nodes, such as IEEE 802.15.4, have very limited communication capacity, which significantly limits the WSN data collection rate, response time, as well as the network scalability.

Several approaches exist to address such power limitations, including energy-aware routing (e.g., [15, 16], energy-efficient MAC protocols (e.g., [17–19]), and adaptive sampling [20, 21]. A recent survey on these techniques can be found in [22]. Orthogonal to the above approaches, data aggregation and compression address the energy problem by exploiting the correlation structures in space and time in WSN sampled data [23–25]. As sensing signals usually exhibit strong temporal and spatial correlations, compression methods have been found very helpful to simultaneously address the issues of both power constraint and bandwidth limitation in WSNs. Data compression is a technique to reduce the number of bits required to represent any given information. In this work, we focus on compression in sensor networks, which has been shown to significantly improve WSN energy savings in real-world deployments (e.g. [26]). With data compression, sensor nodes only need to transmit compressed sensor readings, rather than the original sensor readings obtained from the analog-
to-digital (A/D) converter (ADC), saving both transmission and reception power by the reduction of data size. Compression schemes can be classified into two categories: lossless compression (e.g., [26–30]), and lossy compression (e.g., [31–35]). With lossless compression, compressed sensor data from sensor nodes can be perfectly restored at the data sink (or control center) in the compression/decompression process. With lossy compression, in contrast, some degree of information loss due to compression error will be introduced in the compression/decompression process in order to achieve higher compression ratios.

Thus far, lossless and lossy compression schemes are typically based on different principles. For example, one approach for WSN lossless compression, called Sensor LZW (S-LZW) [26], adopts and modifies the well-known dictionary-based lossless compression Lempel-Ziv-Welch (LZW) algorithm [36] for WSNs. Another lossless compression approach is typically based on traditional predictive coding [15,16], such as the recent Two-Modal Transmission (TMT) scheme [17,18], Lossless Entropy Compression (LEC) algorithm [29], and Adaptive Linear Filtering Compression (ALFC) algorithm [30]. For lossy compression, on the other hand, piecewise linear approximation approaches are typically adopted, such as Lightweight Temporal Compression (LTC) algorithm [33] and PLAMLiS algorithm [31] and its variant [32]. Another recently emerging technique for lossy compression is compressive sensing (e.g., [12,37]). Consequently, none of the existing compression algorithms for WSNs (e.g., [12,17–19,26,31–34]) are able to perform both lossless and lossy data compression scenarios, except for the preliminary work of [38].

Compression in sensor networks can also be categorized into two independent areas: conventional compression and compressed sensing, which are described in Section 2.1 and Section 2.2 respectively.
2.1 Conventional Compression

Conventional compression techniques utilize the correlation during the encoding process. In this section, we will focus on one of the many conventional compression techniques: predictive coding. Several compression approaches are typically based on traditional predictive coding [39,40], such as the recent Two-Modal Transmission (TMT) scheme [41,42], Lossless Entropy Compression (LEC) algorithm [29], and Adaptive Linear Filtering Compression (ALFC) algorithm [30].

Predictive coding was originally introduced by Elias in 1955 [39,40], based on two breakthrough contributions in the 1940s: Wiener’s work on prediction of random time series [43], and Shannon’s work on the mathematical theory of communication [44]. The communication procedure with predictive coding is illustrated in Figure 2.1. Each communication node has a predictor associated with its transmitter, which operates on past transmitted messages/signals stored locally to produce an estimate of its future message/signal. An error term (i.e., residue) is calculated by subtracting the predicted message/signal from the actual message/signal. This error term is then used as input to an entropy coder. It is this coded error term that is transmitted to the receiving node. The reversed process is followed at the receiving side, with an identical predictor as the sending node. When the error term is received, it is decoded and the original message is then obtained by adding the received error term to the predicted message produced at the receiving node. We can see that predictive coding can be viewed as a two-stage process: (1) message/signal prediction, and (2) error or residue signal coding. In the framework of predictive coding, linear or nonlinear prediction models can be used in the first stage, while a variety of coding schemes are applicable in the second stage. This provides flexibility for users to select the predictor and entropy encoder for given data sources to better capture the underlying temporal correlations.
2.2 Compressed Sensing

Several research efforts have been pursued to incorporate CS into data collection schemes in WSNs (e.g., [7–14]). Traditional CS based approaches such as [8–10] do not rely on the knowledge about WSN routing topology but rely on the use of dense measurement matrices, which results in a high transmission cost of sensor nodes, and also requires storing a part of measurement matrix in each resource-constrained sensor node [10]. On the other hand, Quer et al. [7] studied the interplay of routing with compressive sensing in multi-hop WSNs, where the measurement matrix is defined according to the routing paths. However, the authors of [7] found the results of their work were unsatisfactory due to the difficulty to find a suitable representation basis for real signals, and stated finding a suitable transformation with good sparsification and incoherence properties remains an open problem for WSN data gathering. The authors of [13] adopted an opportunistic routing model in WSNs, and conducted some theoretical analysis regarding the nonuniform random projection of CS due to the interplaying with WSN routing. However, in their approach [13], per-packet routing path is recorded in each data packet routed towards the sink, which makes the approach is not scalable and also reduce or eliminate the data compression.
performance. Zheng et al. [14] propose a random walk algorithm for data gathering in multi-hop WSNs, the measurements are collected along the random walks before they are sent to the sink using shortest path routing. This approach is also based on the idea of nonuniform random projection of CS, but does not interplay with WSN routing. Therefore it requires the length of each walk $t = O(n/k)$ for each packet in addition to its routing path towards the sink, which increases the energy consumption of WSNs due to the random walk transmissions. Some other researches [11,12] focused on temporal correlations in a sequence of samples taken by each sensor node in WSN. Existing approaches for multi-hop WSNs (e.g., [7–14]) are only evaluated by numerical simulations with some WSN routing models but not by experiments on real WSN deployments in situ with routing protocol in operation.
3 MATHEMATICAL BACKGROUNDS

3.1 Compressed Sensing

Compressed sensing is a breakthrough technique in signal processing [5,6]. CS theory asserts that for sparse or compressible signals, one can actually recover the original signals by using far fewer measurements or samples than required by the Nyquist rate. Consider an $N$-dimensional discrete sparse signal vector $x \in \mathbb{R}^N$, which is referred to as $k$-sparse if $x$ has no more than $k(k \ll N)$ nonzero items. Mathematically, the theory of CS has shown that if $x$ is sparse, under certain conditions, then it is possible to reconstruct the signal vector $x$ from $M$ measurements $y = [y^1, y^2, \ldots, y^M]^T$ with a quasi-random $M \times N$ measurement matrix $\Phi$, i.e., $y = \Phi x$, where $M(k < M)$ is much smaller than $N$. This can be achieved (with probability close to one) by solving the following optimization:

$$
\min_x = \|x\|_p \quad \text{s.t. } y = \Phi x,
$$

(3.1)

where $\|x\|_p (p = 0, 1)$ denotes $l_p$-norm of $x$. Often, a signal $x$ is not sparse but can be sparsely represented in an alternative domain. Specifically, if $x$ can be further written as $x = \Psi s$, for some $N \times N$ matrix $\Psi$, where $s$ is the $N \times 1$ coefficient vector in the $\Psi$-domain with $\|s\|_\infty = k$, where $\infty$ is referred to as the representation basis. We have $y = \Phi \Psi s = \tilde{\Phi} s$, where $\tilde{\Phi} = \Phi \Psi$ is also quasi-random. Then the associated signal recovery problem is to determine $s$ for given measurement $y$ and the defined matrices $\Phi$ and $\Psi$:

$$
\min_s = \|s\|_p \quad \text{s.t. } y = \tilde{\Phi} s.
$$

(3.2)

As $M$ is much smaller than $N$, this is an under-determined linear system. The reconstruction of the original signal $x$ is given by

$$
x = \Psi s.
$$

(3.3)
When \( p = 0 \) in (3.2), directly solving it is intractable, but fast approach exists by using smoothed \( l_0 \) norm, see e.g. the SL0 method proposed in [45]; When \( p = 1 \) in (3.2), it can be easily solved using linear programming (LP) methods [46]. Many algorithms can solve this problem in polynomial time, including interior-point methods; there are also faster algorithms aimed at large-scale systems. In addition to LP, the algorithms include Iterative Re-weighted Least Squares (IRWLS) [47], and Matching Pursuit (MP), see e.g. OMP [48]. They are considered faster than LP but with worse estimation quality, especially if the signal is not sufficiently sparse.

3.2 Wavelets and Lifting Scheme

Data collection in WSNs can be considered as the processing of signals defined on graphs. Wavelets can reveal many signal properties. Wavelets are well localized and few coefficients are needed to represent local transient structures. Therefore wavelets provide a flexible tool to find a sparse representation.

In 1910, Harr [49] constructed a piecewise constant function

\[
\psi(t) = \begin{cases} 
1 & \text{if } 0 \leq t < 1/2, \\
-1 & \text{if } 1/2 \leq t < 1, \\
0 & \text{otherwise.}
\end{cases}
\] (3.4)

the dilations and translations of which generate an orthonormal basis

\[
\left\{ \psi_{j,n}(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - 2^jn}{2^j}\right) \right\}_{n \in \mathbb{Z}}^j \] (3.5)

of the space \( L^2(\mathbb{R}) \) of signals having a finite energy

\[
\|f\|^2 = \int_{-\infty}^{+\infty} |f(t)|^2 dt < +\infty.
\] (3.6)

Let us write \( (f, g) = \int_{-\infty}^{+\infty} f(t)g^*(t)dt \) the inner product in \( L^2(\mathbb{R}) \). Any finite energy signal \( f \) can thus be represented by its wavelet inner-product coefficients

\[
\langle f, \psi_{j,n} \rangle = \int_{-\infty}^{+\infty} f(t)\psi_{j,n}(t) dt
\] (3.7)
and recovered by summing them in this wavelet orthonormal basis:

\[
f = \sum_{j=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} (f, \psi_{j,n}) \psi_{j,n}.
\]  

(3.8)

Wavelets on graphs [50] is designed to obtain a multiresolution [51] of \(L^2(G)\) the set of all functions/signals on graph \(G\). Namely, a nested sequence of approximation spaces from coarse to fine of the form \(V_1 \subset V_2 \subset \cdots \subset V_{l_{\text{max}}} = L^2(G)\) is constructed. Projecting a signal in the spaces \(V_l\) provides better and better approximations with increasing level \(l\). Associated wavelet/detail spaces \(W_l\) satisfying \(V_{l+1} = V_l \oplus W_l\) are also obtained.

Scaling functions \(\phi_{l,k}\) provide a basis for approximation space \(V_l\), and similarly wavelet functions \(\varphi_{l,k}\) for \(W_l\). As a result, for any signal \(f \in L^2(G)\) on graph and any level \(l_0 < l_{\text{max}}\), we have the wavelet decomposition

\[
f = \sum_k a_{l_0,k} \phi_{l_0,k} + \sum_{l=l_0}^{l_{\text{max}}-1} \sum_k l_{l,k} \varphi_{l,k}.
\]  

(3.9)

The coefficients \(a_{l,k}\) and \(d_{l,k}\) appearing in this decomposition are called approximation (also, scaling) and detail (also, wavelet) coefficients respectively. For simplicity, we use \(a_l\) and \(d_l\) to denote the vectors of all approximation and detail coefficients at level \(l\).

One powerful construction method of wavelets is based on the lifting scheme [52]. Starting with a given wavelet transform, which in our case is the Haar transform (HT), one can obtain lifted wavelets by applying the process illustrated in Figure 3.1 starting with \(l = l_{\text{max}} - 1\), \(a_{l_{\text{max}}} = f\) and iterating down until \(l = 1\). At every level the lifted coefficients \(a_l\) and \(d_l\) are computed by augmenting the Haar coefficients \(\bar{a}_l\) and \(\bar{d}_l\) (of the lifted approximation coefficients \(a_{l+1}\)) as follows

\[
a_l \leftarrow \bar{a}_l + U \bar{d}_l
\]  

(3.10)

\[
d_l \leftarrow \bar{d}_l - P \bar{a}_l
\]  

(3.11)
where update \((U)\) and predict \((P)\) are linear operators (matrices). Note that in adaptive wavelet designs the update and predict operators will vary from level to level, but for simplicity of notation we do not indicate this explicitly.

Figure 3.1. Lifting scheme: one step of forward transform. Here, \(a_l\) and \(d_l\) denote the vectors of all approximation and detail coefficients of the lifted transform at level \(l\). \(U\) and \(P\) are linear update and predict operators. HT and IHT are the Haar transform and its inverse.

Given \(n\) training functions \(f^n\), the linear operators \(U_l\) and \(P_l\) can be learned by solving the minimization problem [50]:

\[
\{U_l, P_l\} = \arg \min_{U_l, P_l} \sum_n \left( (\tilde{d}_l^n - P_l(\tilde{a}_l^n + U_l \tilde{d}_l^n)) \right),
\]

where \(z\) can be any sparse penalty function. Then the representation basis can be obtained by running the inverse process of Figure 3.1.
4 THE DESIGN OF MEASUREMENT MATRIX AND REPRESENTATION BASIS

To minimize the number of transmissions, our proposed compressed sensing approach for multi-hop WSN data collection, referred to as CSR (Compressed Sensing based on dynamic Routing topology tomography), closely interplays with the dynamic routing topology in a given WSN deployment.

Figure 4.1. Overview of our CSR design.

Figure 4.1 gives an overview of our CSR design. The measurement matrix is designed using the routing topology tomography, and we prove that the expected estimation error is bounded. While the design process of representation basis is more complex. compressed sensing theory has two requirement for the representation basis. First one is the basis can sparsify the signal; the other one is the basis should
be sufficiently incoherence with the measurement matrix. We adopt the algorithm from the paper [50]. The pre-collected signals and hierarchical decomposition are involved in the learning process. To find a proper hierarchical decomposition, we convert it to a embedding problem. The embedding process also uses the complement graph of undirected routing topology graph to satisfy the incoherence between the measurement matrix and the representation basis. Then we devise an algorithm, referred to as Graph Linear Embedding (GLE) to solve the embedding problem.

4.1 Problem Formulation

As a data packet is routed from its source node towards the sink, the sensor reading of each traversed node adds up along the path. Let a dynamic WSN for data collection be modeled as a directed acyclic graph \( G(V, E) \), where \( V \) is a set of \( n \) nodes (the sink \( s \) and \( n-1 \) sensor nodes), and \( E \) is a set of edges. A directed edge \( e(u, v) \), an ordered pair \( (u, v) \in V \times V \), represents the wireless communication link from node \( u \) to node \( v \). Let \( p^i = [u_0, u_1, \ldots, u_j, \ldots, S] \) denote a routing path of packet \( i \) from a source node \( u_0 \) to the sink \( S \), which is a sequence of all nodes traversed along the route. For example, as shown in Figure 4.2, there are three data collection paths initiated from leaf nodes in a collection cycle:

\[
p^1 = [u_2, u_1, S],
\]

\[
p^2 = [u_4, u_3, u_1, S],
\]

\[
p^3 = [u_5, u_3, S].
\]

Let \( \Phi \) denote the routing matrix corresponding to the set of paths \( P_\Phi = \{p^1, p^2, p^3\} \). Then, for the sensor network shown in Figure 4.2(a), the routing matrix \( \Phi \) for the given data collection cycle is as follows:

\[
\Phi = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1
\end{bmatrix}
\]
A bipartite graph $B(V, P_\Phi, H)$ can be formed from a $G(V, E)$ with a bi-adjacency matrix $\Phi$, where $V$ is the set of nodes in $G(V, E)$, and $H \subset V \times P_\Phi$ is a set of coupled elements from $V$ to $P_\Phi$. Figure 4.2(b) represents the bipartite graph for the WSN in Figure 4.2(a) with the routing matrix $\Phi$.

Let $y^i_j$, carried by packet $i$, denote the aggregated compressed sensor reading measurement at node $u_j$ along the route $p^i$ towards the sink. We define the following in-network compressing for each data packet $i$ which carries sensor reading routed towards the sink:

\[ y^i_0 = \text{reading}(u_0), \quad j = 0, \quad (4.2) \]
\[ y^i_j = y^i_{j-1} + \text{reading}(u_j), \quad j > 0, \quad (4.3) \]

where $y^i_j$ is computed on the fly at each intermediate node $j$ along the dynamic route $p^i$ towards the sink. In our approach, $M(M \ll n - 1)$ data packets initiated from $M$ randomly selected source nodes of the WSN are collected in each data collection cycle, which carry $M$ compressed sensing measurements specified by (4.2) and (4.3), along their respective routing paths; the $M$ compressed sensing measurements received by the sink $S$ in each collection cycle are denoted by $y = [y^1, y^2, \ldots, y^M]^T$. As one can
see, each data collection routing path represents a random projection of the WSN data field in our compressed sensing. In general, a routing path in an outdoor WSN is inherently random due to the highly dynamic wireless link conditions of the WSN. In addition, some WSN routing protocols (e.g., CTP+EER [53], an extended CTP) could further induce more randomness in routing paths.

A critical issue of such an interplaying between WSN routing and compressed sensing is how to obtain such dynamic routing path information at the sink. Since we consider realistic WSN deployments in situ under time-varying communication environments, where wireless links available a moment ago for a previous packet transmission may not be available for the current packet in a random way, such on-the-fly routing information cannot be obtained in advance. Unlike the recent scheme in [13] which records the routing path of a data packet piggy back as the packet traverse along its path towards the sink, we propose to use WSN routing topology tomography e.g., [54–58] to obtain the dynamic routing information needed in the compressed sensing. The advantage is that the overhead of routing tomography techniques is usually very small per packet compared with the recording of the raw path trace, improving the energy efficiency. For example, the Routing Topology Recovery (RTR) introduced in [57] only has a fixed four-byte overhead of path measurement per each packet. More importantly, the fixed size of path measurement overhead means that it is scalable for multi-hop large-scale WSN deployments, as the widely used IEEE 802.15.4 communication protocol in WSNs has the maximum size of 127-byte MAC frame including the header.

4.2 Measurement Matrix

Two fundamental components of CS are the measurement matrix and the representation basis. We propose to introduce WSN routing tomography technique into the CS framework for constructing the $M \times N (N = n - 1)$ measurement matrix $\Phi = \{ \varphi_{ij} \} \ (1 \leq i \leq M, 1 \leq j \leq N)$ based on dynamic WSN routing. After $M$ data
packets are received in a WSN collection cycle, the routing paths for those $M$ packets are first reconstructed via an adopted routing topology reconstruction algorithm (e.g., RTR [57]). If node $j$ is on the path of packet $i$ received at the sink, then $\varphi_{i,j} = 1$; otherwise, $\varphi_{i,j} = 0$, the i-th row of the measurement matrix $\Phi$ represents the routing path of packet $i$ received at the sink in the given cycle, as illustrated in equation (4.1).

**Proposition:** Let $G(V, E)$ be a WSN with an upward routing matrix $\Phi$ for a given data collection cycle. Suppose that $B(V, P_\Phi, H)$ is a bipartite graph with bi-adjacency matrix $\Phi$. It is feasible to use routing matrix as measurement matrix in compressed sensing in recovering $k$-sparse sensor signals in the given data collection cycle, while the expected estimation error is bounded by equation (11) in [59].

**Proof:** The problem here is an isomorphism problem of the network link delay estimation via CS in [59]. First, the sink and the leaf nodes are boundary nodes, while the others are intermediate nodes. Thus, an upward routing path from a leaf node to a/the sink in the sensor network is equivalent to an end-to-end path in the network considered in [59]. Second, in our setting, the routing matrix $\Phi$ (i.e., measurement matrix) is defined in terms of the traversed nodes in each path in the network rather than the traversed links in each path defined in [59]; consequently, the formation of bipartite graph $B(V, P_\Phi, H)$ in our setting is based on the network node set $V$ as opposed to the bipartite graph $G(E, R, H)$ in [59] based on the network link set $E$.

All the theorems of [59] on the derivation of the error bounds would still be held when the $G(E, R, H)$ in [59] is replaced by our $B(V, P_\Phi, H)$. Thus the sensor signals on the nodes can be feasibly recovered using LP as the delays on the links in [59]. We note that, due to various random noises and interferences in outdoor WSN in situ, the constructed measurement matrix $\Phi$ for a different data collection cycle would be quite different due to wireless link dynamics, even if the deployed routing protocol does not induce any additional random effect on the routing topology.

As one can see, interplaying with WSN dynamic routing on the fly, each sensor node neither stores the matching column in the measurement matrix, nor performs vector multiplication and vector addition. As sensor nodes of outdoor WSNs are
usually battery-powered with very limited memory and low cost microcontroller, the compressed sensing able to effectively interplay with routing is particularly feasible and suitable to multi-hop and dynamic WSN deployments in situ.

4.3 Representation Basis

There are two main criteria in selecting a good representation basis $\Psi$: (1) its corresponding inverse has to sufficiently sparsify the signal $x$; and (2) it has to be sufficiently incoherent with the measurement matrix $\Phi$. A long-standing open question associated with the paradigm of compressed sensing for WSN data collection interplaying with routing is how to find an appropriate representation basis $\Psi$ with good sparsification and incoherence properties. To address this problem, we present a systematic method in building a suitable $\Psi$ which is based on wavelets on graphs via deep learning [50], since the sensor data collected from a WSN are signals defined on the graph of the WSN deployment topology. Rustamov and Guibas [50] recently introduced a machine learning framework, referred to as the GDL in this chapter, for constructing graph wavelets which is expected to sparsely represent a given class of signals on graphs. The basic idea is to use the lifting scheme as applied to the Haar wavelets. Their insight is that the recurrent nature of the lifting scheme gives rise to a structure resembling a deep auto-encoder network. One unique advantage of their framework is the constructed wavelets are adaptive to a class of signals on the underlying graph, which can obtain a multiresolution structure of the signals on a given graph.

However, the GDL framework proposed in [50], as expressed in (3.12), does not solve the problem of how to generally decompose an irregular underlying graph for constructing wavelets, but rather assumes that such a hierarchical decomposition of the underlying irregular and connected graph into connected regions is already provided in advance for the use of framework [50]. Rustamov and Guibas [50] find an approach [60] for their own dataset to obtain a hierarchical tree of partitions. They first
embed the graph into $\mathbb{R}^{n_{\text{max}}}$ as follows: $i \rightarrow (\xi_1(i)/\lambda_1, \xi_2(i)/\lambda_2, \ldots, \xi_{n_{\text{max}}}(i)/\lambda_{n_{\text{max}}})$, $\forall \in V$, where $\{\lambda_n, \xi_n\}_{n=0}^{n_{\text{max}}}$ are the eigen-pairs of the Laplacian Matrix of the graph, and $\xi_n(i)$ is the value of the eigenvector at the $i$-th vertex graph. To obtain a hierarchical tree of partitions, they start with the graph itself as the root. At every step, a given region (a subset of the vertex set) of graph $G$ is split into two children partitions by running the 2-means clustering algorithm ($k$-means with $k = 2$) on the above embedding restricted to the vertices of the given partition. This process is continued in recursion at every obtained region. This results in a dyadic partitioning except at the finest level.

Indeed, it is nontrivial to find an appropriate hierarchical decomposition of any highly irregular large-size graph in a general way. To overcome this difficulty, we devise a novel and effective algorithm that enables the partitioning of the multiscale structure imposed on the underlying irregular graph.

Our idea is to first embed the underlying irregular and connected graph into a linear graph (i.e., 1-dimensional space), in which any two consecutive vertices in this 1-dimensional space are, desirably, connected in the original graph. Then, signals on the original underlying irregular graph are now defined on the 1-dimensional regular space. Therefore, a standard multiresolution decomposition, such as the tree algorithm introduced by Mallat [51], can be readily applied (as shown in Figure 4.3) to generate a feasible hierarchical structure of signals on this transformed linear embedding graph, an approximate of the original underlying graph, upon which HT and (3.12) can be applied.

In this section, we present our schemes for embedding the connected underlying graph into a 1-dimensional space.

Our newly devised algorithm is referred to as Graph Linear Embedding (GLE), which is given in Section 4.3.1.
4.3.1 GLE Algorithm

Our devised algorithm, referred to as Graph Linear Embedding (GLE), is presented as follows. We consider the problem as finding a walk path visiting through all the vertices on the irregular and connected graph in an optimal way to reserve vertices neighborhood information. This problem can be formulated as getting a labeling of vertices which would closely reflect the structure of the graph. This question can be related to a graph labeling problem known as the cyclic bandwidth sum problem. It consists in finding a labeling of the vertices of an undirected and unweighted graph with distinct integers such that the sum of (cyclic) difference of labels of adjacent vertices is minimized [61]. Given an undirected and connected graph $G(V,E)$ with vertex set $V = \{u_i \in V | 1 \leq i \leq n\}$ and edge set $E$, our GLE algorithm is given as follows:

1. Sort the vertices $\{u_1, u_2, u_3, \ldots, u_n\}$ in the ascending order of their degrees, so that $d_{u_1} \leq d_{u_2} \leq d_{u_3} \leq \cdots \leq d_{u_n}$. Initialize list $A = \{u_1\}$ and list $B = \{u_2, u_3, \ldots, u_n\}$ and stack $C = \{u_1\}$. List $A$ keeps the vertices which have already been visited along the walk, while list $B$ keeps the remaining ones not traversed yet. The current vertex is defined as the top vertex in stack $C$. Stack $C$ maintains
the current walk path segment from the bottom vertex in the stack to the current vertex on top of the stack for further check.

(2) Search a vertex \( u_j \) in list \( B \) that matches the following conditions: (i) \( u_j \) is adjacent to the current vertex \( u_i \); and (ii) \( u_j \) has a neighborhood that is the most similar to the one of \( u_i \). Let \( \text{Adj}(u) \) return the all adjacent vertices of the vertex \( u \). The similarity index between vertices \( u \) and \( v \), denoted as \( J(u, v) \), is defined by [61]:

\[
J(u, v) = \frac{\#(\text{Adj}(u) \cap \text{Adj}(v) \cup \{u, v\})}{\#(\text{Adj}(u) \cup \text{Adj}(v))}.
\]

In other words, we are searching for vertex \( u_j \) in \( B \), which satisfies:

\[
u_j = \arg \max_v J(u_i, v), \quad \text{s.t. } v \in \text{Adj}(u_i).
\]

(3) If such a vertex \( u_j \) in \( B \) is found, add \( u_j \) into list \( A \) and then delete it from list \( B \). Push \( u_j \) to stack \( C \). If no vertex in \( B \) is found adjacent to the current vertex \( u_i \), pop \( u_i \) out from stack \( C \), and add the new current vertex in stack \( C \) into list \( A \).

(4) If \( B \) is not empty, repeat steps (2)-(3). If \( B \) becomes empty, the ordered sequence of vertices in \( A \) then forms the embedded 1-dimensional linear topology structure of the given irregular graph.

When a walk is generated by the GLE algorithm for the given connected graph, any two consecutive vertices in the resulting 1-dimensional topology structure are connected in the original graph.

**Lemma 4.3.1** Given a connected graph \( G(V, E) \) with vertex set \( V = \{u_i \in V|1 \leq i \leq n\} \), if our GLE algorithm can generate a walk path which has visited all the vertices only once, then the total time spend is \( O(n^2) \).

**Proof** This has been proved in [61].

**Theorem 4.3.2** Given a connected graph \( G(V, E) \) with vertex set \( V = \{u_i \in V|1 \leq i \leq n\} \), our GLE algorithm can generate a walk path with the total running time \( O(n^2) \).
Proof Lemma 4.3.1 has proved the case when the walk path doesn't have revisit vertices. Now we calculate the additional running time when revisit is needed to find a path. From step (3), a revisit means to pop the vertex or vertices out from the stack $C$. We only pop out a vertex when no vertex in list $B$ is adjacent to this vertex, assume at that moment there are $n_{B_i}$ vertices in $B$, then $O(n_{B_i})$ time is needed to verify that no vertex in list $B$ is adjacent to this vertex. Assume we need to pop out $n_{p_i}$ vertices from $C$ until the next vertex in $B$ is found, then the addition time to find it is $O(n_{p_i} \times n_{B_i})$. The total additional running time of revisiting is $O(\sum i n_{p_i} \times n_{B_i})$. 
\[
\forall i, n_{B_i} < n, \text{ s.t. } \sum_i n_{p_i} \times n_{B_i} < \sum_i n_{p_i} \times n = (\sum_i n_{p_i}) \times n
\]
From step (3), each vertex can only be pushed to stack $C$ only once, so a vertex can be popped out from no more than once. So we have the total number of pop out vertices: $\sum n_{p_i} < n$, then we have
\[
\sum_i n_{p_i} \times n_{B_i} < (\sum_i n_{p_i}) \times n < n \times n = n^2.
\]
Therefore the additional running time is $O(n^2)$ when revisit is needed. The total running time of our GLE algorithm is $O(n^2 + n^2)$, which is $O(n^2)$. 

4.3.2 Construction of Underlying Graph

Given a WSN node deployment topology, we consider how to construct the underlying graph of sensor signals from which an appropriate representation matrix $\Psi$ can be obtained via the GLE or Eigenmap algorithm and the GDL framework as described above. We start with the routing topology recovered at the sink for each data collection cycle in the WSN, which forms a routing topology graph (RTG). Change each directed edge in the RTG to an undirected edge, we have the corresponding undirected RTG, denoted by URTG. To maximize the incoherency between $\Psi$ and $\Phi$, we consider to construct the underlying graph as the complement graph (CG) of the WSN URTG.
For an undirected graph $G = (V, E)$ along with a function $w : E \rightarrow \mathbb{R}^+$, where $\mathbb{R}^+$ denotes the set of positive real numbers, the adjacency matrix $A_G$ of $G$ is [45]:

$$A_G(i, j) = \begin{cases} w(i, j) & \text{if } (i, j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The degree matrix $D_G$ of a weighted graph $G$ is a diagonal matrix such that

$$D_G(i, i) = \sum_j A_G(i, j).$$

The Laplacian matrix $L_G$ of a weighted graph $G$ is defined as

$$L_G = D_G - A_G.$$

Let $P$ training datasets be collected from the WSN deployment for constructing $\Psi$. A training dataset corresponds to a URTG graph $G_i = (V, E_i), i \in \{1, 2, \ldots, P\}$. The union of these $P$ graphs is

$$G_U = (V, E_U), \text{ while } E_U = E_1 \bigcup E_2 \bigcup \cdots \bigcup E_P.$$

The complement graph $C_G$ of $G_U$ is

$$G_{CG} = \overline{G_U} = (V, E_{CG}), \text{ while } (i, j) \in E_{CG}, \text{ if and only if } (i, j) \notin E_U.$$

Let the weight of all the edges be equal to 1, then the adjacency matrix $A_{CG}$ of this complement graph is

$$A_{CG} = \begin{cases} 1 & \text{if } (i, j) \in E_{CG}, \\ 0 & \text{otherwise.} \end{cases}$$

So that $D_{CG}(i, i) = \sum_j A_{CG}(i, j)$ and we get the a Laplacian matrix based on the complement graph of routing as

$$L_{CG} = D_{CG} - A_{CG}.$$

$G_{CG}$ and $L_{CG}$ will be used to find the sparse representation basis $\Psi$ by applying the GDL.
5 VALIDATION

We deploy our CSR approach in both an outdoor testbed and a simulation of large scale network, to validate the performance in different types of networks.

5.1 Validation in ASWP Testbed

To rigorously validate the proposed CSR approach based on WSN routing topology tomography, we deployed our approach in a real-world outdoor multi-hop WSN for environmental monitoring [3]. Section 5.1.1 describes the WSN deployment and our experiment setup. In Section 5.1.2, we study the sparsification performances of the constructed representation basis. In Section 5.1.3, we evaluate the data recovery performance of our CSR approach, in comparison with 3 other different CS based approaches.

5.1.1 ASWP WSN Testbed and Deployment

The multi-hop WSN in-situ used in our validation has been deployed at the Beechwood Farms Nature Reserve (BFNR) of the Audubon Society of Western Pennsylvania (ASWP), located in Fox Chapel in northern Allegheny County, Pennsylvania, USA. The BFNR is owned by the Western Pennsylvania Conservancy, and has 134 acres of protected land. In 2010, the ASWP testbed was deployed and since then the testbed has been operating with the objective of exploring the feasibility and challenges of using WSNs for collecting reliable long-term hydrological data and investigating the impacts of vegetation heterogeneity and soil properties on the status and trends of soil moisture and transpiration [62].
Our validation experiments were developed with TinyOS 2.1.2 [63]. As of August 2015, 84 heterogeneous WSN nodes (including MICAz [64], IRIS [65], and TelosB [66]) had been equipped and deployed. In terms of hardware of the WSN testbed, MicaZ [64] and IRIS motes [65], each one equipped with an MDA300 data acquisition board [67]. The MDA300 provides embedded temperature and humidity sensors. MicaZ and IRIS have 4K bytes and 8K bytes memory, respectively. TelosB motes with embedded temperature and humidity sensors. Both TelosB motes and MDA300 acquisition board have ADCs for external sensors, e.g. EC5 (soil moisture sensor) [68]. The base station (sink node), is an IRIS mote with a permanent power supply connected to a computer operated as the WSN gateway where the WSN gateway forwards the sensed data stream to the WSN data management system over the Internet.

The aesthetics of the nature reserve must be maintained, therefore the motes were camouflaged and discretely located either hanging from tree branches or mounted onto PVC pipes, as seen in Figure 5.1. Internet access is only available at cabin house of the Nature Center (as shown in Figure 5.2), while the locations of sensor motes are around 300 m away or farther.

Data collected in one cycle of ASWP WSN, where each WSN node sampled and sent its sensor readings once, forms a dataset. In our CS validation, we collected datasets from ASWP WSN in situ for 87 cycles. Table 5.1 shows the statistics of per-packet routing path recovery by RTR in our validation experiments conducted on ASWP WSN in August 2015. The longest routing path of packet had 8 hops.

Table 5.1.
The statistics of per-packet routing path recovery by RTR scheme in ASWP WSN

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total cycles</strong></td>
<td>87</td>
</tr>
<tr>
<td><strong>Average path recovery ratio</strong></td>
<td>98.38%</td>
</tr>
<tr>
<td><strong>Best cycle recovery</strong></td>
<td>100%</td>
</tr>
<tr>
<td><strong>Worst cycle recovery</strong></td>
<td>93.20%</td>
</tr>
</tbody>
</table>
Figure 5.1. Examples of node configurations in the ASWP testbed. A node hanging from a tree without external sensors (left) and a node mounted onto a PVC pipe with external sensors attached (right).

The first 10 datasets were used as the training datasets for constructing the representation basis in our approach while the other 77 datasets were used as the test datasets. Humidity data were collected from 75 nodes, while soil moisture data were collected from 48 nodes equipped with soil moisture sensor EC5. The original sensor readings of each node were also collected in the same data packet in each cycle in addition to the compressed data to provide the base for the accuracy analysis of our CS approach.

5.1.2 Performance of Representation Basis

We first evaluate the sparsity of the representation basis. As described in Section 3, s is the $N \times 1$ coefficient vector in the $\Psi$-domain with $\|s\|_0 = K$, where $K \ll N$. By keeping only the largest $K$ components in $s$, we can get the approximation $s'$ of
s, and thus the approximation $x' = \Psi s$. Comparing $x'$ with $x$ gives the performance of the representation basis $\Psi$.

The representation basis in our CS approach is constructed as follows: First, construct the underlying graph of WSN with recovered WSN URTGs from path measurements in the given training datasets; second, use our devised GLE algorithm to obtain an appropriate hierarchy decomposition of the underlying graph; third, apply GDL [50] to the obtained hierarchy decomposition of the underlying graph to construct graph wavelets with the given WSN training datasets, and then construct
the sparse representation basis based on the constructed graph wavelets. Figure 5.3 shows an example of the humidity data collected by 75 nodes and the corresponding transform coefficients for the representation basis obtained by using GLE algorithm with the 10 humidity training datasets. Similarly, Figure 5.4 shows an example of the soil moisture data from 48 nodes and the corresponding transform coefficients for the representation basis constructed by using GLE algorithm with the 10 training datasets of soil moisture. As we can see, only very few coefficients have large absolute values.

Next, we select the largest k transform coefficients in magnitude of both humidity and soil moisture data, respectively, to evaluate the sparsification performance of our representation basis. We further compare the sparsification performance of our CS representation basis with those adopting Haar wavelet transformation and DCT (Discrete Cosine Transformation), the two popular transformations used in existing
Soil moisture raw signals (top) and the corresponding transform coefficients for our constructed representation basis using the GLE algorithm (bottom).

CS schemes such as CDC [13] and CDG [8]. The approximation error (%), defined as in (5.1), is employed to evaluate the performance for different CS approaches.

\[
\text{Error} = \frac{\sqrt{\sum_n (x'_n - x_n)^2 / N}}{\sqrt{\sum_n x_n^2 / N}} \times 100\%.
\]  

(5.1)

Figures 5.5 and 5.6 show the average sparsification errors of the 77 test datasets for humidity and soil moisture signals, respectively. As we can see, the representation basis constructed by our method (GLE plus graph wavelets via deep learning by GDL) can always lead to very small approximation error even when only keeping a few largest transform coefficients in magnitude. While the performances of all three different bases are improved when \(k\) becomes larger, our representation basis always significantly outperforms the Haar and DCT transformations. Our method also has very stable performance on both humidity and soil moisture datasets. For humidity data, when keeping only the largest three transform coefficients in magnitude (out of total 75), the approximation error is less than 3.3%, while for soil moisture data,
keeping only the largest two transform coefficients in magnitude (out of total 48), the approximation error is always less 1.7%. This indicates that the humidity and soil moisture signals can be well sparsely represented using their respective basis obtained by the proposed method.

![Humidity](image1)

Figure 5.5. The sparsification error of humidity datasets estimated by using only the largest $k$ components in magnitude in $s$.

![Soil Moisture](image2)

Figure 5.6. The sparsification error of soil moisture datasets estimated by using only the largest $k$ components in magnitude in $s$. 
5.1.3 Signal Recovery Accuracy

After collecting $M \times 1 (M < N)$ measurements $y = [y_1, y_2, \ldots, y^M]^T$ from the ASWP WSN in each cycle, we first recover the routing path of each received packet using the RTR scheme [57], from which the measurement matrix $\Phi$ is reconstructed for the sensor dataset in this cycle. Then, two CS solvers SL0 [45] and LP [46, 69] are adopted to obtain an approximation $s'$ of $s$ in the transform domain. Finally, the original signal is recovered by computing $x' = \Psi s'$. The recovery accuracy is evaluated here using the approximation error defined in Equation (5.1).

![Figure 5.7. An example of the routing path (measurement) when M=12.](image)

For the evaluation of our compressed sensing approach CSR, three existing CS schemes CDG [8], RS-CS (with Horz-diff transformation) [7] and CDC [13] are used
for the comparison. While CSR, CDC and RS-CS approaches interplay with routing, CDG does not, and relies on dense random projections which need to collect data from all WSN nodes.

We set $M = 12$ for both humidity and soil moisture data collection in our CS approach. Figure 5.7 shows an example of the WSN routing topology for the 12 measurements collected from the ASWP WSN in our experiments, which was reconstructed by the RTR scheme running at the sink. As we can see, many nodes were not visited in this cycle, which means their data were not collected in CSR, CDC and RS-CS approaches jointly with routing. Fewer visited nodes generally can make it more difficult to recover the entire data field.

![Figure 5.7: WSN routing topology for the 12 measurements collected.](image)

**Figure 5.7.** WSN routing topology for the 12 measurements collected from the ASWP WSN in our experiments, which was reconstructed by the RTR scheme running at the sink.

Figures 5.8 ~ 5.11 show the reconstruction error for humidity and soil moisture signals using four different CS approaches, with two solvers: SL0 and LP. As we can see, our CSR with LP solver can always achieve the data recovery with the error less than 7.7% on humidity datasets and less than 5.0% on soil moisture datasets for any collection cycle, significantly outperforms CDG, CDC and RS-CS. Note that
Figure 5.9. Humidity dataset reconstruction error when M=12, using LP as the solver.

Figure 5.10. Soil moisture dataset reconstruction error when M=12, using SL0 as the solver.

both CDC and RS-CS perform worse than CDG and they are sensitive to different datasets collected in different cycles, because only a subset of the nodes was visited
in each individual collection cycle. Our CSR approach overcomes this problem by constructing a much better representation basis as shown in Section 5.2, therefore CSR always outperforms the other three CS approaches for both humidity and soil moisture data. We also note that CDC are very sensitive to the solver, it performs much better with LP than SL0. Generally, the solver LP outperforms SL0, but LP takes longer computation time.

Figure 5.12 shows the reconstruction errors for humidity using different partitioning approaches. As we can see, GLE algorithm performs much better than the k-mean approach using in [50], which indicates that GLE can achieve high quality partitions on the graph.

Figures 5.13 ∼ 5.16 show the reconstruction errors for humidity and soil moisture signals using four different CS approaches, with different numbers (M) of collected measurements. As we can see, our CSR has excellent performance even when M is very small, with reconstruction errors being an order of magnitude less than those of the other three CS approaches even with much larger M. Generally, the performance
Figure 5.12. Humidity dataset reconstruction error when M=12, using different partitioning approaches.

Figure 5.13. Humidity dataset reconstruction errors with different $M$, using SL0 as the solver.

will improve when $M$ becomes larger, the only exception is CDG with solver LP on soil moisture data.
Figure 5.14. Humidity dataset reconstruction errors with different $M$, using LP as the solver.

Figure 5.15. Soil moisture dataset reconstruction errors with different $M$, using SL0 as the solver.

Figure 5.17 gives an example of the reconstructed humidity data field by our CSR when only 12 data packets are collected at the sink, in comparison with the original
Figure 5.16. Soil moisture dataset reconstruction errors with different $M$, using LP as the solver.

Figure 5.17. Original signals vs. the reconstructed signals by CSR.

humidity data field. As we can see, while the original humidity data change drastically from sensor node to node, our CSR is still able to recover the entire data field with high fidelity using only 12 measurements.
Figures 5.18 and 5.19 show the transmission numbers of CSR and CDG for collecting humidity and soil moisture measurements, respectively, for different numbers of received measurements. As we can see, the CSR leads to the drastic reduction of
data packet transmissions, by an order of magnitude less than those of CDG. This means the drastic radio communication energy conservation by CSR, the great advantage of any CS approach if successfully interplaying with routing. While both CDC and RS-CS have the same transmissions as CSR in our experiments, due to the employment of the same routing protocol CTP+EER in the experiments for CDC and RS-CS as well, CDC and RS-CS have significantly larger data packet size than that of CSR. This is because CDC and RS-CS record the original packet path in the data packet along the route, whereas our CSR uses routing topology tomography for path information. Consequently, CSR is not only scalable for large-size WSNs and big data acquisition, but also more resource efficient than CDC and RS-CS.

5.2 Simulation in Large Scale Network

To further evaluate our CS approach in a larger network, we run a simulation using TOSSIM. This network has 225 nodes, and node 0 is the sink, the other 224 are motes. The topology is very different from ASWP testbed as shown in Table 5.2 and Figure 5.20. The sensor data is NASA GISS surface temperature reading [70]. The test runs for 120 cycles, and the first 20 cycles are used as training set. In this simulation, we also consider loops in the network, and the result is in Section 5.2.2.

<table>
<thead>
<tr>
<th></th>
<th>ASWP Testbed</th>
<th>Simulation network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of motes</td>
<td>75</td>
<td>224</td>
</tr>
<tr>
<td>Number of 1-hop and 2-hop neighbors</td>
<td>3 and 3</td>
<td>8 and 34</td>
</tr>
<tr>
<td>Longest path (hops)</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>Consider loops?</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 5.2. The differences between two networks
5.2.1 Compare with Different CS Approaches

Figures 5.21 and 5.22 show the reconstruction errors for temperature signals using four different CS approaches, with different numbers (M) of collected measurements. Figure 5.23 shows reconstruction error when $M = 20$. For temperature dataset, both CDC and CDG are very sensitive to the solvers. Using the LP solver, CDC performs as well as our CSR approach when $M$ is larger than 17; CDG performs slightly better than our CSR approach, this can be explained by the dense measurement matrix of CDG. To compare with all other three CS approaches, the loops are not considered in the results of Figure 5.21-5.23. In Section 5.2.2, we will see the reconstruction performance when loop happens. Figure 5.24 shows the reconstruction errors for temperature using different partitioning approaches. As we can see, GLE algorithm also performs much better than the k-mean approach using in [50], which indicates that GLE can also achieve high quality partitions on this large scale network.
Figure 5.21. Temperature dataset reconstruction errors with different M, using SL0 as the solver. The reconstruction error is the average value of 100 cycles.

Figure 5.22. Temperature dataset reconstruction errors with different M, using LP as the solver. The reconstruction error is the average value of 100 cycles.
5.2.2 Reconstruction Performance with Loops

In the 38th cycle of the test dataset, we found 14 measurements with loops. The reconstruction performance using only these 14 measurements is shown in Figure
5.25 and 5.26. In our CSR approach, all the path information is compressed in the packet, so the loops only change the 0-1 measurement matrix to a matrix with natural number. As shown in top of Figure 5.25, white means the entry value is 0; the darker the point is, the higher value the entry is, which means the matching node is revisited more often in that collection path. From Figure 5.25, our CSR approach handle the loops very well. Figure 5.26 shows the performance of CDG, in their approach, the measurement matrix is pre-assigned, so the sink cant notice that loop happens. When loop happens, the measurement value changes with the loops, but CDG will recover the signals using the original measurement matrix, this leads to the large error as shown in Figure 5.26.

Figure 5.25. Performance of CSR in cycle 38 when all the measurements have loops.
Figure 5.26. Performance of CDG in cycle 38 when all the measurements have loops.
5.3 Summary

In this section, rigorous validation of our CSR approach versus three existing CS approaches CDG, CDC and RS-CS was conducted in both a real-world outdoor WSN deployment in situ and a large scale network simulation. The results clearly demonstrate that CSR approach significantly outperforms CDG, CDC and RS-CS by reducing data reconstruction errors by an order of magnitude for the entire WSN data field, while drastically reducing wireless communication costs, by an order of magnitude, at the same time. This indicates that our CSR approach is a reliable and practical solution to energy efficient data acquisition in multi-hop large-scale WSNs. In our experiments, CSR can successfully recover the entire data field of the real-world multi-hop WSN in situ with very small errors, when only 16% of data packets (i.e., 12 randomly selected nodes out of total 75 sensor nodes in the WSN testbed) needed to be collected at the sink.
6 LONG TERM EVALUATION IN THE OUTDOOR TESTBED

To further evaluate the performance of our CS approach, we will present the performance of our CS approach from a long term deployment in the outdoor ASWP testbed.

Our CS approach is deployed in the same ASWP testbed as in Chapter 5. The sensor data is collected from 2017-08-01 to 2017-10-28. There are 97 nodes in this testbed during this period. However, an outdoor testbed will always suffer data loss for many reasons. We only use data collected from 68 nodes, which we have the original sensor reading for comparison.

6.1 Different Training Size

In our CSR approach, the sparse representation basis is learned from training dataset. To learn the impact of the training size, we choose 4 different training size: 10, 20, 50, 100, as shown in Table 6.1. The test set are collected from 2017-08-12 to 2017-10-28, include 350 cycles.

Table 6.1.
Training dataset information

<table>
<thead>
<tr>
<th>Training Size (cycles)</th>
<th>Start Time</th>
<th>End Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2017-08-11 12:44:02</td>
<td>2017-08-12 20:43:29</td>
</tr>
<tr>
<td>20</td>
<td>2017-08-10 21:41:32</td>
<td>2017-08-12 20:43:29</td>
</tr>
<tr>
<td>50</td>
<td>2017-08-06 19:33:18</td>
<td>2017-08-12 20:43:29</td>
</tr>
<tr>
<td>100</td>
<td>2017-08-01 00:03:02</td>
<td>2017-08-12 20:43:29</td>
</tr>
</tbody>
</table>

From Figure 6.1, we can see that generally the reconstruction error will increase as the cycle (day) increases. The one with training size 100 performs worse than
other three. Figure 6.2 zooms in the details of the other three. As we can see in Figure 6.2, the one with training size 20 performs better than the other two. This indicates that the representation basis derived from the 20 training sets (2-day-long datasets) fits the future test dataset than the other three. We also noticed that all the three has some similar peak points, it mainly because in that cycle, the collected measurements covered fewer nodes, in another word, fewer nodes were visited by the collection routing path.

![Humidity dataset reconstruction errors with different training size](image)

Figure 6.1. Humidity dataset reconstruction errors with different training size: 10, 20, 50, 100, using SL0 as the solver, while M=20.

6.2 Different Laplacian Matrix

Based on the results of Section 6.1.1, we choose 20 as the training size for the following test. In Figure 6.3, we first compare the performance of two different Laplacian
Figure 6.2. Humidity dataset reconstruction errors with different training size: 10, 20, 50, using SL0 as the solver, while M=20.
matrices. Routing refers the one built from the complement graph of routing, while Distance refers the one built from distance. In our testbed, if the distance between two nodes is shorter than 60 meters, we consider these two nodes are connected. As we can see, the Laplacian matrix derived from routing is more stable than the one derived from distance. Furthermore, we also tried different scales of Deep Learning in Figure 6.3. Using Figure 4.2 in Section 4.1.3 as an example, the cluster $A^2_3$ can be consider as $A^2_3$'s 1st order of neighbors, while $A^3_3$ and $A^3_4$ are $A^2_3$'s 2nd order neighbors. As we can see, the scale of 1st order of neighbors is more stable, this can be explained by that the correlation in small cluster is more stable even in a long term test, while the larger the cluster is, the correlation may change more with the time.

![Graphs showing error rates for different scales](image)

Figure 6.3. Humidity dataset reconstruction errors with different Laplacian matrix or Deep Learning scales, using SL0 as the solver, training size is 20, M=20.
6.3 Long Term Fitting

Our test dataset is collected from 2017-08-12 to 2017-10-28, including 350 cycles and lasts for about two and a half month. To investigate the fitting performance of our representation basis during this long term test, we show the results of two cycles, the first (2017-08-12) (Figure 6.4) and the last (2017-10-28) (Figure 6.5). From Figure 6.4, we can see that in cycle 1, the reconstruction error is very small, especially on the visited nodes. From Figure 6.5, we can see that the reconstruction error grows larger, especially on the visited nodes. It indicates that the re-training is needed after a long term running to capture the new correlations between the sensor reading of nodes. We noticed that in both cases, although the measurement matrix is very sparse, our CSR approach is able to recover the sensor reading with tolerable error.

6.4 Compare with Different CS Approaches

Figures 6.6 – 6.9 show the reconstruction errors for humidity signals using four different CS approaches, with different numbers (M) of collected measurements. In Figure 6.6 and 6.7, the reconstruction error is the average value of 350 cycles; while in Figure 6.8 and 6.9, the reconstruction error is only from the 350th cycle. As we can see, our CSR has excellent performance even when M is very small, with the two solvers: SL0 and LP. CDC is very sensitive to the solver, it performs much better with the LP solver. Generally, the performance will improve when M becomes larger, but the LP solver doesn't follow this rule strictly.

6.5 A Self-tuned approach in the Network

In this long term evaluation, we also deploy self-tuned approach in the network to adjust the collected packets (or measurements) from cycle to cycle.

As shown in Procedure 1, each mote is assigned with an initial probability $p_0$ (or an integer threshold in practical). Each cycle, a mote will first check if it has
Figure 6.4. Performance of CSR in cycle 1 (collected on 2017-08-12). Top: measurement matrix, the point means the entry value is 1; otherwise, 0. Middle: the reconstruction errors of each node. Bottom: the recovered humidity sensor reading.
Figure 6.5. Performance of CSR in cycle 350 (collected on 2017-10-28). Top: measurement matrix, the point means the entry value is 1; otherwise, 0. Middle: the reconstruction errors of each node. Bottom: the recovered humidity sensor reading.
Figure 6.6. Humidity dataset reconstruction errors with different $M$, using SL0 as the solver. The reconstruction error is the average value of 350 cycles.

Figure 6.7. Humidity dataset reconstruction errors with different $M$, using LP as the solver. The reconstruction error is the average value of 350 cycles.
Figure 6.8. Humidity dataset reconstruction errors with different $M$, using SL0 as the solver. The reconstruction error is from the 350th cycle.

Figure 6.9. Humidity dataset reconstruction errors with different $M$, using SL0 as the solver. The reconstruction error is from the 350th cycle.
forwarded packet in previous cycle: if yes, then it will reduce the probability to the half of the current value. We also set a lower bound of the probability value as $p_{min}$. Then the mote will generate a random number, and compare it with the probability (or the threshold) to decide if it will send out its own packet in this cycle, or only wait to forward the other motes’ packet.

Figure 6.10 shows the distribution of packets number after deploying our self-tuned approach. In our test, we set $p_0 = 0.2$ and $p_{min}=0.1$. As we can see, generally the distributions follows a normal distribution, which indicates that random generator controls the packets number as expected. We notice that in $[0,5]$, the number of cycles seems to be more than expected. It is because in these cycles, the network condition is very poor, we also only receive very few other types of packets in these cycles.

Figure 6.10. Distribution of packets number.
Procedure 1: Self-tuned procedure at the mote

Notations:
\( p_0 \): initial probability.
\( p \): current probability.
\( p_{\text{min}} \): lower bound of the probability.
\( \text{random()} \): generate a random number in \((0,1)\).
\( \text{send()} \): send out a packet using current mote as the source node.
\( \text{sampling_{timer\_start}()} \): a sampling timer is set to sample the sensor reading periodically.
\( \text{sampling_{timer\_fired}()} \): return \( TRUE \) when the sampling timer is fired, otherwise return \( FALSE \).
\( \text{pack\_forward()} \): pack the sensor reading and the path information into the forwarding packet, then forward it.

Initial:

\[ p \leftarrow p_0 \]
\[ \text{flag} \leftarrow 0 \]

\begin{verbatim}
SelfTuned():
begin
while mote is running do
    sampling_{timer\_start()}
    flag \leftarrow 0
    while sampling_{timer\_fired()} == FALSE do
        if forwarding a packet then
            pack\_forward()
            flag \leftarrow 1
        end
    end
    if flag == 1 then
        if \( p/2 \geq p_{\text{min}} \) then
            \[ p \leftarrow p/2 \]
        else
            \[ p \leftarrow p_{\text{min}} \]
        end
    else
        \[ r \leftarrow \text{random()} \]
        if \( r \leq p \) then
            \[ \text{send()} \]
        end
    end
end
\end{verbatim}
7 A GENERALIZED PREDICTIVE CODING FRAMEWORK

In this section, we extend our previous work of [38] and present a generalized predictive coding (GPC) framework, which can perform unified lossless and lossy temporal compressions effectively and efficiently in a systematic manner. This is desirable particularly for WSNs. Due to the extremely limited resource on tiny motes, the implementation of two separate lossless and lossy schemes on every mote is clearly not resource-efficient. Our algorithmic framework is quite simple as its additional complexity compared to the traditional predictive coding is negligible, making it well suited for tiny motes in WSN data gathering. In addition, a novel lossless data compression algorithm, called Sequential Lossless Entropy Compression (S-LEC), is devised. The proposed GPC framework, with the devised S-LEC algorithm, can form a powerful unified lossless and lossy compression solution to practical WSNs, although each can be used individually as well. We thoroughly evaluate our GPC framework and S-LEC algorithm through rigorous simulation using diverse real WSN data, and show that the GPC framework with S-LEC significantly outperforms the recent popular lossless compression algorithms S-LZW [26] and LEC [29], and the lossy compression algorithm LTC [33]. To our knowledge, the proposed innovation for the unification of both lossless and lossy compression is the first of its kind for general, sustainable, and extensible WSN data collections. Moreover, we further investigate the error distributions of sensor data prediction and propose a discrete Laplacian distribution model which can better characterize those prediction errors. We also provide energy consumption analyses for our framework and algorithm to justify their efficacy.
7.1 Algorithmic Framework

7.1.1 Two-Model Transmission

In our generalized predictive coding (GPC) framework, the first generalization is the introduction of the concept of two-modal transmission (TMT). The basic idea is to, for a given residue distribution model, encode only those residues falling inside a relatively small range \([-R, R]\) \(R > 0\) and is called compression radius hereafter) by entropy coding (referred to as predictive compression mode in the residue domain); otherwise, the raw samples are transmitted in the original data domain (referred to as normal mode), as illustrated in Figure 7.1. Hence, the GPC is concerned with the adoption of the optimum compression radius \(\text{R}_{\text{op}}\) to maximize the compression ratio, subject to the resource constraints of tiny sensor nodes. Usually, for WSNs, an atomic unit of sensing information in the normal mode of transmission is one raw sensing observation, represented in fixed \(K\) bits specified by the precision/resolution of the A/D converter used for sampling. The maximum compression radius \(\text{R}_{\text{max}}\) is then \(2^K - 1\) by default.

![Diagram of TMT generalization of predicting coding and decoding procedure.](image)
Entropy coding is defined on a set of symbols called an *alphabet*, or symbol table. The basic alphabet adopted for entropy coding shall contain *one* symbol $S$ for each *residue* $r$ to be encoded. Given a compression radius $R$, one distinct symbol $S_i$ is used to represent residue $r_i \in -R, -R + 1, \ldots, 0, \ldots, R - 1, R$. The size of the alphabet is $2R + 1$. It can also be easily seen that given a precision $K$ of A/D converters used in a WSN, the basic alphabet for the traditional predictive coding would then be $-2^K + 1, -2^K + 2, 0, \ldots, 2^K - 2, 2^K - 1$, and the size of the alphabet is $2^{K+1} - 1$.

It is desirable to work with a very small alphabet for entropy coding, to reduce the usage of the already limited resources in sensor nodes. A large alphabet would use more memory and would also complicate the computation of a compression algorithm. To cope with this issue, Liang and Peng [41] introduces the use of an appropriate or optimized number system (e.g., decimal system), called the $M$-based alphabet system, in representing individual residues to achieve a reduced alphabet, where $M$ is the base of the number system. In another approach, a basic alphabet of residues is divided into $M$ groups with binary code in each group, such as an Exponential-Golomb type of coding adopted in [29]. Hence, we can use $M$ to represent the resource constraints of sensor nodes (e.g., required memory).

Thus, the two-modal transmission can be formulated as the following conditional optimization problem: given the constraint $M \leq M_{\text{max}}$, find a solution of $(M, R)$ where $R \leq R_{\text{max}}$ maximizes the compression ratio under the GPC framework.

7.1.2 Lossy Compression

The second generalization of our GPC is how to effectively deal with lossy compression in a predictive coding framework, as we attempt to build a unified framework for both lossless and lossy compression based on predictive coding. In general, to control compression errors in a lossy compression scheme, a control knob in lossy compression is needed for data quality. Intuitively, in a predictive coding framework, when the absolute value of error $r_i$ (i.e., residue) generated for a sample $x_i$ at a sensor node
does not exceed a given margin $e(e > 0)$ specified by the control knob, this residue $r_i$ can be approximated as zero to save bits for transmission. That is, we may have

$$\hat{r}_i = \begin{cases} 
0, & |r_i| \\
 r_i - e, & |r_i| > e \land r_i > 0 \\
 r_i + e, & |r_i| > e \land r_i < 0
\end{cases}$$

where $[\hat{r}_i]$ can be transmitted instead of $[r_i]$ in a block/packet for lossy compression, so that the sink can effectively and approximately recover original samples $[x_i]$ from the received $[\hat{r}_i]$. In the earlier work of [38], the proposed approach was based on synchronized iterative multi-step prediction at both sensor nodes and the data sink, in which the predicted output for a given time step will be used as an input for computing the sensor reading series at the next time step. That is, approximated residues $[\hat{r}_i]$ are produced by using iterative multi-step prediction following the first sample in a packet at a sensor node. However, this causes compression performance to suffer, because errors propagate and accumulate sample by sample and quickly exceed the specified error margin $e$. To address this issue, approximated residues $[\hat{r}_i]$ transmitted in the same block/packet should be computed independently from each other. Based on this insight, we introduce a new lossy compression called the block lossy compression (BLC) scheme into the GPC. A sensor data block is a sequence of sensor readings to be compressed as one unit in the proposed BLC scheme. For each data block to be compressed, a predicted anchor value will be attached, which is the average value of the all sensor readings in the block. This anchor is used for computing all residues for individual samples of the block in lossy compression. In WSNs, one packet carries a complete compressed data block, so that the data of each packet can be independently decompressed. The proposed GPC then employs the predictor $\bar{x} = \sum_{i=1}^{J} x_i / J$ for each packet for lossy compression, in which $J$ is the number of samples to be transmitted in a packet. We have

$$r_i = x_i - \bar{x} = x_i - \frac{\sum_{i=1}^{J} x_i}{J}, i = 1, 2, \ldots, J$$  (7.1)
This way, the residue errors of individual samples in a packet will be independent from each other, and thus not propagate and accumulate sample by sample. Another method to overcome the accumulation of errors is to modify the encoder to calculate residue $r_i$ by subtracting the most recent reconstructed $\hat{x}_{i-1}$ rather than the original $x_{i-1}$ [71]. When this solution is applied to lossy compression in WSNs, it requires every sensor node to reconstruct every compressed sample value, which inevitably increases operations on each resource-constrained sensor node. In Section 6, we will evaluate our BLC scheme in comparison with Salomon’s solution in terms of compression performance.

We apply the idea of quantization to residues $[r_i]$ computed by (7.1) at sensor nodes. In our quantization mechanism, a quantization index $q_i$ for $r_i$ is computed as follows:

$$q_i = Q(r_i) = \text{round} \left( \frac{r_i}{2\lceil c \rceil + 1} \right), i = 1, 2, \ldots, J$$

That is, $q_i$ is the rounding of the quotient $r_i/(2\lceil c \rceil + 1)$ to the nearest integer. The same entropy encoder for lossless compression in the GPC is used for the transmission of $\hat{x}$ and quantization indices $[q_i]$ for lossy compression. At the WSN sink(s), approximated $[\hat{r}_i]$ and $[\tilde{x}_i]$ are computed from the received $[q_i]$ as follows, respectively

$$\hat{r}_i = q_i(2\lceil c \rceil + 1), i = 1, 2, \ldots, J$$

$$\tilde{x}_i = \bar{x}_i + \hat{r}_i, i = 1, 2, \ldots, J$$

Figure 7.2 illustrates the BLC generalization of predictive coding for lossy compression. The compression errors in the lossy compression are quantization errors. Quantization indices $[q_i]$ are much smaller integers compared to $[r_i]$, which can significantly improve the compression performance. Note that in the proposed GPC framework, there should be total of $J + 1$ items (i.e., one average value $\bar{x}$ and $J$ quantization indices) in a packet for lossy compression, whereas there are only $J$ residues in a packet for lossless compression.
7.1.3 GPC Framework

Putting the above two generalizations together, in our unified GPC algorithmic framework, a compression error bound (denoted as $e$) can be used as the control knob, where lossless compression can be processed as $e = 0$. The algorithmic procedure at source nodes of the $GPC(R, e)$ is presented as Procedure 2, while the corresponding decoding algorithmic procedure at the sink(s) is presented in Procedure 3. Note that, like traditional predictive coding, the proposed GPC is a general framework which provides users with flexibility to choose any appropriate predictor for lossless compression and any entropy encoder, based on given tasks. However, given a resource constraint $M \leq M_{\text{max}}$, the optimal $R$ is usually determined based on the chosen entropy encoder approach. In Section 7.5, we will illustrate, given $M \leq M_{\text{max}}$, how to obtain $R$ for the S-LEC encoder which will be presented in Section 7.4. Readers can also check [41] for how to obtain $R$ with an $M$-based alphabet using an arithmetic coding encoder.
**Procedure 2:** Sender procedure in GPC(R,e) framework at sensor nodes

**Notations:**
- R: optimal compression radius, satisfying resource constraint \( M \leq M_{\text{max}} \).
- e: given compression error bound \( (e \leq R) \).
- \( x_i \): sensor reading.
- \( \hat{x} \): prediction of \( x_i \).
- predictor(): any predictor for lossless compression; for lossy compression, the predictor will be \( \hat{x} = \sum_{j=1}^{J} x_i / J \), where \( J \) is the total number of items in a packet.
- encode(): encode residues with any encoder being used.
- pack(y): packing item \( y \) at source node into a packet.

```plaintext
GPC_Sender(R, e):
begin
  if \( e = 0 \) then
    pack(encode(\( x_1 \))) /*lossless compression*/
    \( \hat{x} \leftarrow \) predictor()
  end
  if \( e > 0 \) then
    pack(\( \bar{x} \)) /*lossy compression*/
    if \( |\text{round}(\frac{x_1 - \bar{x}}{2^e + 1})| \leq R \) then
      pack(encode(\( \frac{x_1 - \bar{x}}{2^e + 1} \)))
    else
      pack(encode(\( x_1 \)))
      \( \hat{x} \leftarrow \bar{x} \)
    end
  end
  \( i \leftarrow 2 \)
  while not done do
    residue \( \leftarrow x_i - \hat{x} \)
    if \( e = 0 \) then
      if \( |\text{residue}| \leq R \) then
        pack(encode(\( \text{residue} \)))/*lossless compression*/
      else
        pack(encode(\( x_i \)))
      end
      \( \hat{x} \leftarrow \) predictor()
    end
    if \( e > 0 \) then
      if \( |\text{round}(\text{residue}/(2^e + 1))| \leq R \) then
        pack(encode(\( \text{round}(\text{residue}/(2^e + 1)) \)) /*lossy compression*/
      else
        pack(encode(\( x_i \)))
      end
    end
    \( i \leftarrow i + 1 \)
  end
end
```
**Procedure 3:** Receiver procedure in GPC(R,e) framework at the sink

**Notations:**
R: optimal compression radius.
e: given compression error bound (e ≤ R);
xᵢ: restored sensor reading.
\( \hat{x} \): prediction of xᵢ.
predictor(): any predictor for sensor observation prediction for lossless compression.
flag: the flag to indicate the first item of the packet.
decode(): decode encoded residues by the selected encoder.
read(y): reading item y from the packet at the sink node.

```
GPC_Receiver(R, e):
begin
  z ← read(y)
  if e == 0 then
    x₁ ← x /*lossless decompression*/
  end
  if e > 0 then
    \( \hat{x} \leftarrow x /\text{lossy decompression*} \)
    z ← decode(read(y))
    if |z| ≤ R then
      x₁ ← \( \hat{x} + z(2[e] + 1) \)
    else
      x₁ ← x
    end
  end
  i ← 2
  while not_done do
    z ← decode(read(y))
    if e == 0 then
      if |z| ≤ R then
        xᵢ ← z + predictor() /*lossless decompression*/
      else
        xᵢ ← z
      end
    end
    if e > 0 then
      if |z| ≤ R then
        x₁ ← \( \hat{x} + z(2[e] + 1) /\text{lossy decompression*} \)
      else
        xᵢ ← z
      end
    end
    i ← i + 1
  end
end
```
7.2 Sequential Lossless Entropy Compression (S-LEC)

We present our S-LEC algorithm, a novel extension and modification of LEC [29], for lossless entropy compression based on predictive coding. Both LEC and S-LEC adopt the simple and popular differential predictor to generate residue series. That is \( \hat{x}_i = x_{i-1} \), and the residue is the difference \( r_i = x_i - x_{i-1} \). Our insight is that the simple and generic differential predictor would not be able to completely capture and remove the temporal correlations among an arbitrary sensor data sequence. As a result, the generated residues are usually not independent. However, in LEC, residues \([r_i]\) (\(i = 1, 2, 3, \ldots, J\), where \(J\) is the size of data block) in the series, being coded by the entropy encoder, are considered to have no correlation, and thus are encoded independently. Hence, while LEC would perform reasonably if the correlation characteristic of a sensor data stream can be largely captured by the differential predictor, it would perform poorly otherwise. To address the robustness issue of the original LEC, we devise the S-LEC algorithm to exploit valuable sequential context information among adjacent residues for sensor data compression. In the following, before we present our S-LEC algorithm, we first briefly overview LEC.

7.2.1 Overview of LEC

The entropy encoder in LEC is a modified version of the Exponential-Golomb code of order 0 [72]. LEC organizes the alphabet of integer residues obtained from the differential predictor into groups which have exponentially increased sizes [29]. Generally, an LEC codeword consists of two parts: the entropy code specifying the group and the binary code representing the index in the group. Assuming that any sensor reading \(x_i\) is represented in \(K\) bits, \(K + 1\) groups are to be formed. Residue \(r_i\) other than zero is represented as \(h_i|a_i\), where \(h_i = h(n_i)\) encodes the group number \(n_i = \lfloor \log_2 |r_i| \rfloor + 1\) and \(a_i\) represents the following computed index within the group as binary code:
\[ \text{index} = \begin{cases} r_i, & r_i > 0 \\ 2^{n_i} - |r_i| - 1, & r_i < 0 \end{cases} \]

When residue \( r_i \) equals zero, the corresponding group number \( n_i = 0 \) and no index (i.e., no \( a_i \) code) is needed. For example, assuming that any sensor reading \( x_i \) is represented in \( K \) bits, an LEC entropy coding table for \( K = 14 \) is shown in Table 7.1.

<table>
<thead>
<tr>
<th>( n_i )</th>
<th>( s_i )</th>
<th>( r_i (n_i \in 0,1,\ldots,14) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>-1, +1</td>
</tr>
<tr>
<td>2</td>
<td>011</td>
<td>-3, -2, +2, +3</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>-7, \ldots, -4, +4, \ldots, +7</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>-15, \ldots, -8, +8, \ldots, +15</td>
</tr>
<tr>
<td>5</td>
<td>110</td>
<td>-31, \ldots, -16, +16, \ldots, +31</td>
</tr>
<tr>
<td>6</td>
<td>1110</td>
<td>-63, \ldots, -32, +32, \ldots, +63</td>
</tr>
<tr>
<td>7</td>
<td>11110</td>
<td>-127, \ldots, -64, +64, \ldots, +127</td>
</tr>
<tr>
<td>8</td>
<td>111110</td>
<td>-255, \ldots, -128, +128, \ldots, +255</td>
</tr>
<tr>
<td>9</td>
<td>1111110</td>
<td>-511, \ldots, -256, +256, \ldots, +511</td>
</tr>
<tr>
<td>10</td>
<td>11111110</td>
<td>-1023, \ldots, -512, +512, \ldots, +1023</td>
</tr>
<tr>
<td>11</td>
<td>111111110</td>
<td>-2047, \ldots, -1024, +1024, \ldots, +2047</td>
</tr>
<tr>
<td>12</td>
<td>1111111110</td>
<td>-4095, \ldots, -2048, +2048, \ldots, +4095</td>
</tr>
<tr>
<td>13</td>
<td>11111111110</td>
<td>-8191, \ldots, -4096, +4096, \ldots, +8191</td>
</tr>
<tr>
<td>14</td>
<td>111111111110</td>
<td>-16383, \ldots, -8192, +8192, \ldots, +16383</td>
</tr>
</tbody>
</table>

7.2.2 S-LEC Algorithm

The proposed S-LEC introduces a novel sequential code \( s_i \) to generally extend the LEC codeword of \( r_i \) from \( h_i|a_i \) into a new general form \( s_i|h_i|a_i \), in order to exploit the correlation among adjacent residues to achieve better compression performance. The essential idea is that if a subsequent residue \( r_{i+1} \) belongs to the same or neighboring group as residue \( r_i \), which is highly likely due to sequential correlation between adjacent residues, the group code \( h_{i+1} \) for \( r_{i+1} \) can be inferred from the previous \( h_i \).
and thus omitted in its codeword. That is $r_{i+1}$'s codeword is $s_{i+1}|a_{i+1}$ instead of $s_{i+1}|h_{i+1}|a_{i+1}$, thus reducing the size of the codeword. To code such sequential context information, two bits of sequential code $s_i$ are devised in our S-LEC encoder, as specified in Table 7.2. It is worthy to note that $s_1$, for the first residue $r_1$ of a sensed data block, will be always omitted, and hence $h_1$ has to be present.

In the case of $s_i = 11$, while group code $h_i$ cannot be omitted completely, it can be reduced in some context situations without any ambiguity. To investigate this possibility, let all groups of the LEC alphabet be further grouped into three clusters as follows: $C_1 = \{n_i| i = 0, 1, 2, 3\}$, $C_2 = \{n_i| i = 4, 5\}$, and $C_3 = \{n_i| i = 6, \ldots, K\}$, where $K$ is the number of bits of a sensor reading, that is, the precision of A/D converter (ADC). Then we give $h_i$ reduction rule based on $r_{i-1} \in C_j (j = 1, 2, 3)$ in Table 7.3 when $s_i = 11$ and $n_i > n_{i-1}$.

**Table 7.2.**
Sequential coding in S-LEC

<table>
<thead>
<tr>
<th>$s_i$</th>
<th>Context Information</th>
<th>$h_i = h_{i-1}$</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>$h_i = h_{i-1}$</td>
<td>same group</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>$h_i = \begin{cases} h(n_{i-1} - 1), &amp; n_{i-1} \geq 1 \ h(n_{i-1} + 2), &amp; n_{i-1} = 0 \end{cases}$</td>
<td>neighboring group</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>$h_i = \begin{cases} h(n_{i-1} + 1), &amp; n_{i-1} &lt; K \ h(n_{i-1} - 2), &amp; n_{i-1} = K \end{cases}$</td>
<td>neighboring group</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>$h_i$ cannot be omitted in codeword, $s_i</td>
<td>h_i</td>
<td>a_i$ is required</td>
</tr>
</tbody>
</table>

**Table 7.3.**
Group code reduction

<table>
<thead>
<tr>
<th>$r_{i-1}$</th>
<th>Reduced $h_i$ when $s_i = 11$ and $n_i &gt; n_{i-1}$</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1-reduced $h_i$</td>
<td>Remove the first 1(^{st}) from LEC $h_i$, e.g., 1110 reduced to 110</td>
</tr>
<tr>
<td>$C_2$</td>
<td>11-reduced $h_i$</td>
<td>Remove the first two 1’s from LEC $h_i$ if possible, e.g., 11110 reduced to 110</td>
</tr>
<tr>
<td>$C_3$</td>
<td>111-reduced $h_i$</td>
<td>Remove the first three 1’s from LEC $h_i$ if possible, e.g., 1111110 reduced to 110</td>
</tr>
</tbody>
</table>
Thus, in addition to the entropy coding table used in LEC, our S-LEC encoder creates two new coding logics described in Tables 7.2 and 7.3, respectively. To illustrate, for example, let us consider a series of residues $r_1 = 35$, $r_2 = 106$, $r_3 = -72$, $r_4 = 12$, and $r_5 = -54$, which will be encoded by the S-LEC encoder as 1110|100111, 10|1101010, 00|0001000, 11|1011100, 11|10|010110. In contrast, the same residue series would be encoded by the LEC encoder as 1110|100111, 1110|1101010, 1110|0001000, 101|1100, 1110|010110.

7.3 Performance Analyses

To analytically study the compression performance of the proposed GPC, we investigate the relationship of the compression ratio with respect to the selection of optimal compression radius $R_{op}$ in our GPC for lossless compression. To this end, a residual distribution model is needed. In this section, we first present and study a residual distribution model based on discrete Laplace distribution. Then, we show the relationship between the compression performance and the compression radius $R$ in our GPC for lossless compression, based on the Laplace residual distribution. Finally, we provide an analysis of the lossy compression performance of our GPC.

7.3.1 Residual Distribution Model

Laplace distribution was used in [41] to model residues. Since residue is a discrete variable, the discrete Laplace distribution [73] is adopted in this chapter.

The probability mass function (pmf) of the discrete Laplace distribution (in the special case $\mu = 0$) takes on an explicit form in terms of the parameter $p = e^{-1/b}$,

$$f(k) = P(X = k) = \frac{p^{|k|}}{1 + p^{|k|}}, k \in Z = \{0, \pm 1, \pm 2, \ldots\}$$

Given $N$ independent and identically distributed samples $x_1, x_2, \ldots, x_N$, the maximum likelihood estimator of $b$ is derived by Least Absolute Deviation (LAD) [74]:

$$\hat{b} = \frac{1}{N} \sum_{i=1}^{N} |(x_i - \mu)|$$
Alternatively, we can derive the parameter $p$ using $p = \frac{1-P(X=0)}{1+P(X=0)}$ then derive $b$ using $b = \frac{-1}{\log p}$. In essence, $b$ characterizes the shape of the discrete Laplace distribution (i.e., the smaller the value of $b$, the narrower the Laplace distribution), and thus characterizes the nature of the original sensor data series. In the following analytical studies, we will use different values of $b$ to represent different sensor data series to be obtained in the real applications.

Figure 7.3 shows the discrete Laplace distributions with the parameter $b$ derived by Least Absolute Deviation (LAD, i.e., L1) and Least Squares (L2). The real residue (Empirical) is calculated using various real-world data sets from SensorScope [75]. It can be observed that LAD characterized the real residue distribution better. Least absolute deviation is robust in that it is resistant to large residue in the data, which is why we use LAD to estimate the parameter $b$.

### 7.3.2 Compression Radius

We characterize the relationship of the compression ratio $CR$ with respect to the selection of compression radius $R$ for lossless compression using the Laplace residual model. The compression performance is usually evaluated by the compression ratio, defined as follows:

$$CR = \left(1 - \frac{s'}{s}\right) \times 100\% = \left(1 - \frac{s'}{K \times N}\right) \times 100\%,$$

where $s$ and $s'$ denote the original raw data size and the compressed data size in bits, respectively; $N$ is the number of total data samples, and $K$ is the precision – the fixed bits per raw sample usually specified by the resolution of the A/D converter used for sensor sampling. The sensor sample precision $K$ determines the ultimate maximum compression radius $R_{\text{max}}$ ($R_{\text{max}} = 2^K - 1$) for the sensor samples. Figure 7.4 shows the ideal source entropy coding compression ratio $CR$ for Laplace residual distribution with zero mean under different parameter $b$. As we can see, the compression ratio is non-decreasing as the compression radius $R$ increases, which means that a smaller compression radius would degrade the compression ratio. In this sense, the general
principle of selecting optimal compression radius $R_{op}$ is to choose the largest compression radius $R$ ($R < R_{max}$) that can accommodate the given resource constraints on sensor nodes before WSN applications. However, we observe that there exists an integer $R_t$ for any given parameter $b$ of the residual distribution, when $R_2 > R_1 \geq R_t$, we have

$$|CR(R_2) - CR(R_1)| < 1\%.$$ 

This indicates that there exists a threshold $R_t$, and the improvement of $CR$ for any compression radius $R$ larger than $R_t$ is basically negligible, justifying the merit
Figure 7.4. The ideal lossless compression ratio for Laplacian distribution with zero mean under different $b$ values.

of TMT generalization of the GPC. In particular, $R_t$ could be rather small, implying that a very small coding table would be sufficient to achieve a satisfactory compression result. For example, when $b \leq 15$, $R_2 > R_1 \geq R_t = 64$, we have $|CR(R_2) - CR(R_1)| \leq 0.74\%$. Therefore, $R_t = 64$ is sufficient for achieving the near optimal $CR$ within 1% of the ideal entropy coding. The resulting compact coding table is easier to implement in the sensor node, and hence energy efficient. In practice, when given a resource constraint $M \leq M_{max}$, the optimal $R_{op}$ can be determined accordingly, which will be illustrated in Section 5.

Figure 7.5 compares the lossless compression ratios between the ideal entropy coding and the GPC/LEC entropy coding with respect to $R$, in which LEC is used in the proposed GPC framework. We note that GPC/S-LEC cannot be easily shown analytically when only a residual distribution is known, since S-LEC exploits additional
Figure 7.5. Comparison of compression ratios between ideal entropy coding and GPC/LEC entropy coding with different \( b \) values ranging from 10 to 30.

sequential correlation not expressed in the residual distribution. It can be seen that as \( R_t \) becomes larger, GPC/LEC entropy coding results are close to the ideal entropy coding results.

7.3.3 Lossy Compression Performance

In Section 7.4.1, we show that discrete Laplace distribution can be used as a residue distribution model for lossless compression. In fact, this model can also be used for lossy compression as well. In lossy compression BLC, when we use the average value of the entire block readings as the predictor, the residue distribution (without quantization) is shown in Figure 7.6. It can be observed that the discrete Laplace distribution with L1 characterized the real residue distribution well.
Figure 7.6. Illustration of residue distribution models using L1 for SensorScope ambient temperature data sets.

In our lossy compression, we also use the idea of quantization. We analytically show that the entropy of residues will be reduced due to quantization, where the discrete Laplace distribution is used as the residue model. Let $H_P$ and $H_Q$ denote the entropy of residues of source data, before and after applying quantization, respectively. We have

\[
H_P = - \sum_i \left( \frac{1-p}{1+p}p_i \right) \log_2 \left( \frac{1-p}{1+p}p_i \right),
\]

\[
H_Q = - \sum_Q \left( \sum_{Q(i)} \left( \frac{1-p}{1+p}p_i \right) \right) \log_2 \sum_{Q(i)} \left( \frac{1-p}{1+p}p_i \right),
\]

where $SQ(i)$ is the subset of residues which have the same quantization index $q$. $H_P$ and $H_Q$ can be written as
\[
H_P = - \sum_{SQ} \left( \sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right) \log_2 \left( \frac{1-p}{1+p} p^i \right) \right) = \sum_Q H_P(SQ(i)), \\
H_Q = - \sum_{SQ} \sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right) \log_2 \sum_{SQ(i)} \left( \frac{1-p}{1+p} p^i \right) = \sum_Q H_Q(SQ(i)).
\]

For any given \( SQ(i) \), we have
\[
H_P(SQ(i)) = - \sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right) \log_2 \left( \frac{1-p}{1+p} p^i \right) \\
= - \sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right) \log_2 \frac{1-p}{1+p} + \log_2 p^i \\
= - \sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right) \log_2 \frac{1-p}{1+p} + \sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right) \log_2 p^i,
\]
\[
H_Q(SQ(i)) = -(\sum_{SQ(i)} \left( \frac{1-p}{1+p} p^i \right)) \log_2 \sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right) \\
= -(\sum_{SQ(i)} \left( \frac{1-p}{1+p} p^i \right)) \log_2 \frac{1-p}{1+p} + \log_2 \sum_{Q(i)} p^i \\
= -(\sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right)) \log_2 \frac{1-p}{1+p} - (\sum_{SQ(i)} \left( \frac{1-p}{1+p} p^i \right)) \log_2 \sum_{Q(i)} p^i.
\]

As we can see, the first part of \( H_P(SQ(i)) \) and \( H_Q(SQ(i)) \) are the same, so we focus on the second part.

For any given i, we have \( p^i < \sum_{Q(i)} p^i \leq 1 \), so that
\[
- \log_2 p^i > - \log_2 \sum_{Q(i)} p^i \\
- \frac{1-p}{1+p} p^i \log_2 p^i > - \frac{1-p}{1+p} \log_2 \sum_{Q(i)} p^i \\
- \sum_{Q(i)} \left( \frac{1-p}{1+p} p^i \right) \log_2 \sum_{Q(i)} p^i > - (\sum_{SQ(i)} \frac{1-p}{1+p} p^i) \log_2 \sum_{Q(i)} p^i
\]
\[
\log_2 \sum_{Q(i)} p^i \text{ is the same for any } i \text{ in set } SQ(i). \text{ So for any given } SQ(i), \text{ we always have } H_Q(SQ(i)) > H_Q(SQ(i)). \text{ Therefore, } H_P > H_Q.
\]

Figure 7.7 shows the GPC/LEC lossy compression ratio CR for Laplace residual distribution with zero mean under different parameter b. Indeed this gives a good prediction of GPC/LEC performance for the real world sensor data corresponding to different b values, which can be seen in Section 7.4.

7.4 Simulations and Evaluations

We first evaluate the proposed GPC framework and S-LEC algorithm for lossless compression. We consider S-LEC based on GPC (i.e., GPC/S-LEC), versus the LEC algorithm (based on traditional predictive coding) and S-LZW algorithm, using diverse real-world WSN datasets from SensorScope [75] and volcanic monitoring [76].
The lossy compression ratio using GPC/LEC

Figure 7.7. The GPC/LEC lossy compression ratio for Laplacian distribution with zero mean under different $b$ values.

Since LEC and S-LZW represent two different approaches in lossless compression for WSN data, as briefly described in Section 1, they form a good set of existing algorithms for lossless compression evaluation. We then evaluate the performance of GPC/S-LEC for lossy compression compared with the popular LTC algorithm using the same real-world WSN data sets.

7.4.1 Data Sets

First, real-world environmental monitoring WSN data sets from SensorScope [75] are used in our simulations. To compare with the recently proposed algorithms LEC and S-LZW, the same ambient temperature and relative humidity measurements from the identical sensor nodes used in [29] are tested in the following four SensorScope deployments: HES-SO FishNet, LUCE, Grand-St-Bernard, and Le Genepi. A Sensirion
SHT75 sensor module [77] was adopted with TinyNode sensor node platform [78] in the SensorScope deployments. Both the ambient temperature and relative humidity sensors are connected to a 14-bit ADC (i.e., $K = 14$). However, the outputs of ADC for the raw temperature ($\text{raw}_t$) and raw relative humidity ($\text{raw}_h$) are represented with resolutions of 14 and 12 bits, respectively. These raw outputs $\text{raw}_t$ and $\text{raw}_h$ are then converted into physical measures $t$ and $h$ expressed, respectively, in Celsius degrees and percentage (%) as described in [77]. The surface temperature measurements from the same sensor nodes are also used in this chapter. A Zytemp TN901 infrared module was used for sensing surface temperature with the resolution of 16 bits. The data sets published on SensorScope corresponding to the four deployments are in the format of physical measures $t$ and $h$. Therefore, one needs to convert such physical measures back to their corresponding raw measures to evaluate compression algorithms, which can be realized by using the inverted versions of the conversion functions in [77]. For more details about characteristics of these data sets, please see [78].

Next, real-world volcanic monitoring WSN seismic data sets were used [76], which were collected via a 19-day WSN deployment at Reventador, an active volcano in Ecuador [79, 80]. The resolution of sampled seismic signals is 24 bits (i.e., $K = 24$), and sample rate is 100Hz. The size of the seismic data sets used ranges from 7885 samples to 8268 samples. Unlike the relatively smooth SensorScope ambient temperature and relative humidity data, the seismic data of volcanic eruptions are highly dynamic, presenting challenges to data compression approaches. The volcanic data 2005-08-11_03.36.40 from [76] is used in our experiment, whose range is projected from the normalized $(-1, 1)$ to their original $(0, 2^{24} - 1)$ to obtain raw ADC readings.

7.4.2 Lossless Compression

First, to make a direct comparison with the results of LEC and S-LZW reported with SensorScope data sets in [29], we follow the same assumptions made in [29]: (1) a
raw temperature (raw\_t) sample and a raw relative humidity sample will take 16 bits when transmitted uncompressed; and (2) the size of each data block for compression is 264 samples. In the predictive compression mode of the GPC, if the same size of LEC coding table (i.e., \( K + 1 \)) in [29] is used in our simulations, representing the same resource constraint condition, the optimal compression radius \( R_{op} \) is accordingly \( 2^{K-1} - 1 \), based on the assumption of Laplace or Normal distribution of residues. In the normal mode of the GPC, uncompressed raw samples are transmitted, following the same assumption of byte-aligned representation of uncompressed raw samples in [29]. Entropy coding tables for both LEC (with traditional predictive coding) and S-LEC with GPC (i.e., GPC/S-LEC) are given in Tables 7.1 and 7.4, respectively, in which their last table entry differs. While Table 7.4 has the same size (i.e., 15 entries) as that of Table 7.1, S-LZW uses significantly more memory space for its dictionary entries and mini-cache entries (e.g., MAX\_DICT\_ENTRIES being 512, and MINI\_CACHE\_ENTRIES being 32, [26, 29]).

Table 7.4. Entropy coding table in LEC with \( K = 14 \)

<table>
<thead>
<tr>
<th>( n_i )</th>
<th>( s_i )</th>
<th>( r_i(n_i0, 1, \ldots, 13) ) /Original raw sample (( n_i = 14 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>-1, +1</td>
</tr>
<tr>
<td>2</td>
<td>011</td>
<td>-3, -2, +2, +3</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>-7, \ldots, -4, +4, \ldots, +7</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>-15, \ldots, -8, +8, \ldots, +15</td>
</tr>
<tr>
<td>5</td>
<td>110</td>
<td>-31, \ldots, -16, +16, \ldots, +31</td>
</tr>
<tr>
<td>6</td>
<td>1110</td>
<td>-63, \ldots, -32, +32, \ldots, +63</td>
</tr>
<tr>
<td>7</td>
<td>11110</td>
<td>-127, \ldots, -64, +64, \ldots, +127</td>
</tr>
<tr>
<td>8</td>
<td>111110</td>
<td>-255, \ldots, -128, +128, \ldots, +255</td>
</tr>
<tr>
<td>9</td>
<td>1111110</td>
<td>-511, \ldots, -256, +256, \ldots, +511</td>
</tr>
<tr>
<td>10</td>
<td>11111110</td>
<td>-1023, \ldots, -512, +512, \ldots, +1023</td>
</tr>
<tr>
<td>11</td>
<td>111111110</td>
<td>-2047, \ldots, -1024, +1024, \ldots, +2047</td>
</tr>
<tr>
<td>12</td>
<td>1111111110</td>
<td>-4095, \ldots, -2048, +2048, \ldots, +4095</td>
</tr>
<tr>
<td>13</td>
<td>11111111110</td>
<td>-8191, \ldots, -4096, +4096, \ldots, +8191</td>
</tr>
<tr>
<td>14</td>
<td>11111111111</td>
<td>Original raw sample</td>
</tr>
</tbody>
</table>
The lossless compression performance is listed in Table 7.5 for the ambient temperature and relative humidity data sets from the four WSN deployment sites (the compression ratios by S-LZW adopted from [29]). As we can see, our GPC/S-LEC outperforms both LEC and S-LZW for every WSN data set.

The above results are not very realistic, however, because of the two assumptions of the uncompressed raw sample size of 16 bits and the block size of 264 samples (i.e., 528 bytes) made in [29]. The first assumption is not necessary since each raw sample of temperature and humidity reading is actually 14 bits and 12 bits, respectively. The second assumption of a large size of data block means that a compressed data block has to be transmitted as a sequence of multiple packets in a WSN, since the packet size in WSNs is usually quite small. For example, the widely used IEEE 802.15.4 communication protocol for WSNs has the maximum size of 127-byte MAC frame including the header. If any packet of a compressed data block is lost in a WSN, all subsequent packets in a sequence cannot be decoded at the sink even if they are correctly received. To overcome this problem, each compressed packet should be decodable independently; thus, the large data block size is not practical. To this end, we use 32 samples per block, so that the entire compressed data block can be carried by a single packet in WSNs. The benefit is that this frame can tolerate occasional packet loss in practical WSNs under unreliable communication channel conditions, even when retransmission method is employed.

After the more realistic assumptions, we re-conducted the comparison simulations between our GPC/S-LEC and the LEC on the same real WSN data sources, and the new compression performance is shown in Table 7.6. The results are closer to the reality. Furthermore, two adaptive LEC versions GAS-LEC and FAS-LEC recently proposed in [28] are also evaluated together with SensorScope surface temperature data sets used in the evaluation, as reported in Table 7.6. For ambient temperature and relative humidity data sets, we note that the experimental compression ratios are lower than the previous compression ratios shown in Table 7.5. This can be explained by: (1) adoption of smaller block size and smaller uncompressed raw data
size; and (2) the requirement of each packet to be independently decompressible in consideration of unreliable communication channels, where the first sample of each packet is transmitted as its original raw value. Since we use 14 and 12 bits instead of 16 bits for original raw data as in [29], the raw data sizes are 12.5% and 25% smaller, while the corresponding compressed sizes do not change that much. As shown from Table 7.6, our GPC/S-LEC framework still yields significantly better performance than that of LEC. Our approach outperforms FAS-LEC for seven out of eight data sets, and achieves similar compression performance as GAS-LEC. For surface temperature data, our approach outperforms LEC, GAS-LEC and FAS-LEC for every data set.

Table 7.5.
Lossless compression performance on SensorScope data (block_size: 264 samples)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data set</th>
<th>Compression ratio CR (%) 16 bit/sample</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GPC/S-LEC</td>
</tr>
<tr>
<td>Ambient Temperature</td>
<td>FN_ID101</td>
<td>68.03</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>72.77</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>61.18</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>56.64</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td>FN_ID101</td>
<td>65.90</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>64.64</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>55.49</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>51.29</td>
</tr>
</tbody>
</table>

Figures 7.8 and 7.9 show the distributions of compressed data block sizes for ambient temperature and humidity respectively from four sensor sites. As we can see, GPC/S-LEC packet size curves shift to the left of the corresponding LEC packet size curves.

For volcanic data sets, each seismic reading is 24 bits. Assuming the given resource constraint $M_{max} = 17$, we have $M = 17$ and thus the corresponding optimal compression radius $R = 2^{15} − 1$ in GPC framework. Therefore, entropy coding tables for the LEC and GPC/S-LEC are shown in Tables 7.7 and 7.8, where the LEC with the traditional predictive coding framework actually cannot satisfy the given
Table 7.6.
Lossless compression performance on SensorScope data (block_size: 32 samples)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data set</th>
<th>Compression ratio CR (%) AT: 14 bit/sample, RH: 12 bit/sample, ST: 16 bit/sample</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GPC/S-LEC</td>
<td>LEC</td>
</tr>
<tr>
<td>Ambient Temperature</td>
<td>FN_ID101</td>
<td>58.37</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>65.53</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>51.53</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>46.15</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td>FN_ID101</td>
<td>51.33</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>49.36</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>38.46</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>33.25</td>
</tr>
<tr>
<td>Surface Temperature</td>
<td>FN_ID101</td>
<td>73.41</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>72.36</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>69.19</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>70.62</td>
</tr>
</tbody>
</table>

![Graphs](image1.png)

![Graphs](image2.png)

Figure 7.8. The distribution of the compression packet length (ambient temperature).

resource constraint, since its coding table must have \( K + 1 = 25 \) entries (i.e., code groups) instead of 17 code groups (i.e., entries). Here, we want to demonstrate that a smaller compression radius \( R \) in our GPC/S-LEC is chosen to accommodate the
Figure 7.9. The distribution of the compression packet length (relative humidity).

given resource limitation of sensor nodes (e.g., the GPS/S-LEC coding table reduced to 17 code groups for volcano data, compared to the LEC coding table requiring 25 code groups). In fact, S-LEC alone, using the same maximum compression radius as the LEC, would achieve a better lossless compression ratio than GPC/S-LEC. In the following evaluation, data sampled from 10 sensor nodes are employed with 24 samples per block in our simulations, and the lossless compression performances of GPC/S-LEC, S-LEC (without the compression radius constraint), LEC, GAS-LEC and FAS-LEC are listed in Table 7.9. We observe that GPC/S-LEC and S-LEC have drastically different performance results from LEC on volcano WSN data sets. As we can see from Table 7.9, even though the LEC uses a larger coding table, GPC/S-LEC improves the compression ratios by a factor of 2 to 6 when compared to the LEC for all sensor nodes data sets except for node 251. For node 251, GPC/S-LEC can achieve more than 25% compression ratio, whereas the LEC completely fails. Figure 7.10 shows the distributions of compressed data block sizes for seismic data.

While both GAS-LEC and FAS-LEC perform significantly better than LEC, overall they perform worse than GPC/S-LEC or S-LEC alone. Although GAS-LEC/FAS-LEC can also exploit temporal correlation among residues, their approach leads to
the consideration among a broader range of distant neighboring groups. In contrast, S-LEC exploits the temporal correlations more significantly between two direct neighboring groups (i.e., the temporal correlations between any consecutive residues \( r_i \) and \( r_{i+1} \) which belong to two direct neighboring groups). Out of the 10 data sets, GPC/S-LEC (with a smaller coding table) performs better than both GAS-LEC and

<table>
<thead>
<tr>
<th>( n_i )</th>
<th>( s_i )</th>
<th>( r_i(n_i \in {0, 1, \ldots, 24}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>-1, +1</td>
</tr>
<tr>
<td>2</td>
<td>011</td>
<td>-3, -2, +2, +3</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>-7, \ldots, -4, +4, \ldots, +7</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>-15, \ldots, -8, +8, \ldots, +15</td>
</tr>
<tr>
<td>5</td>
<td>110</td>
<td>-31, \ldots, -16, +16, \ldots, +31</td>
</tr>
<tr>
<td>6</td>
<td>1110</td>
<td>-63, \ldots, -32, +32, \ldots, +63</td>
</tr>
<tr>
<td>7</td>
<td>11110</td>
<td>-127, \ldots, -64, +64, \ldots, +127</td>
</tr>
<tr>
<td>8</td>
<td>111110</td>
<td>-255, \ldots, -128, +128, \ldots, +255</td>
</tr>
<tr>
<td>9</td>
<td>1111110</td>
<td>-511, \ldots, -256, +256, \ldots, +511</td>
</tr>
<tr>
<td>10</td>
<td>11111110</td>
<td>-1023, \ldots, -512, +512, \ldots, +1023</td>
</tr>
<tr>
<td>11</td>
<td>111111110</td>
<td>-2047, \ldots, -1024, +1024, \ldots, +2047</td>
</tr>
<tr>
<td>12</td>
<td>1111111110</td>
<td>-4095, \ldots, -2048, +2048, \ldots, +4095</td>
</tr>
<tr>
<td>13</td>
<td>11111111110</td>
<td>-8191, \ldots, -4096, +4096, \ldots, +8191</td>
</tr>
<tr>
<td>14</td>
<td>111111111110</td>
<td>-16383, \ldots, -8192, +8192, \ldots, +16383</td>
</tr>
<tr>
<td>15</td>
<td>1111111111110</td>
<td>-32767, \ldots, -16384, +16384, \ldots, +32767</td>
</tr>
<tr>
<td>16</td>
<td>11111111111110</td>
<td>-65535, \ldots, -32768, +32768, \ldots, +65535</td>
</tr>
<tr>
<td>17</td>
<td>111111111111110</td>
<td>-131071, \ldots, -65536, +65536, \ldots, +131071</td>
</tr>
<tr>
<td>18</td>
<td>1111111111111110</td>
<td>-262143, \ldots, -131072, +131072, \ldots, +262143</td>
</tr>
<tr>
<td>19</td>
<td>11111111111111110</td>
<td>-524287, \ldots, -262144, +262144, \ldots, +524287</td>
</tr>
<tr>
<td>20</td>
<td>111111111111111110</td>
<td>-1048575, \ldots, -524288, +524288, \ldots, +1048575</td>
</tr>
<tr>
<td>21</td>
<td>1111111111111111110</td>
<td>-2097151, \ldots, -1048576, +1048576, \ldots, +2097151</td>
</tr>
<tr>
<td>22</td>
<td>11111111111111111110</td>
<td>-4194303, \ldots, -2097152, +2097152, \ldots, +4194303</td>
</tr>
<tr>
<td>23</td>
<td>111111111111111111110</td>
<td>-8388607, \ldots, -4194304, +4194304, \ldots, +8388607</td>
</tr>
<tr>
<td>24</td>
<td>1111111111111111111110</td>
<td>-1677215, \ldots, -8388608, +8388608, \ldots, +1677215</td>
</tr>
</tbody>
</table>

Table 7.7. Entropy coding table in LEC with \( K = 24 \)
FAS-LEC for six data sets, respectively; S-LEC performs better than both GAS-LEC and FAS-LEC for seven data sets, respectively. In Table 7.9, we have also provided the average and the standard deviation of compression ratios over 10 data sets for each evaluated algorithm. As we can see, GPC/S-LEC or S-LEC has not only the better average of compression ratio but significantly smaller deviation than GAS-LEC, FAS-LEC and LEC, indicating that our approach not only has better compression performance but also significantly better robustness. It can also been seen clearly that S-LEC always performs better than GPC/LEC, due to its usage of more resource (i.e., a larger coding table than GPC/S-LEC).

We also note that GAS-LEC and FAS-LEC would suffer from adding more complexity on already resource-limited sensor nodes as well as the network sink. For example, if each sensor node connects \( t \) different types of sensors (e.g., temperature, humidity, soil moisture), then the node would have to concurrently manipulate \( t \) ro-

---

**Table 7.8.**
Entropy coding table in GPC/S-LEC with \( M = 17 \)

<table>
<thead>
<tr>
<th>( n_i )</th>
<th>( s_i )</th>
<th>( r_i(n_i \in {0,1,\ldots,15}) ) Original raw sample (( n_i = 16 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>-1, 1</td>
</tr>
<tr>
<td>2</td>
<td>011</td>
<td>-3, -2, 2, 3</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>-7, -4, 4, 7</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>-15, -8, 8, 15</td>
</tr>
<tr>
<td>5</td>
<td>110</td>
<td>-31, -16, 16, 31</td>
</tr>
<tr>
<td>6</td>
<td>1110</td>
<td>-63, -32, 32, 63</td>
</tr>
<tr>
<td>7</td>
<td>11110</td>
<td>-127, -64, 64, 127</td>
</tr>
<tr>
<td>8</td>
<td>111110</td>
<td>-255, -128, 128, 255</td>
</tr>
<tr>
<td>9</td>
<td>1111110</td>
<td>-511, -256, 256, 511</td>
</tr>
<tr>
<td>10</td>
<td>11111110</td>
<td>-1023, -512, 512, 1023</td>
</tr>
<tr>
<td>11</td>
<td>111111110</td>
<td>-2047, -1024, 1024, 2047</td>
</tr>
<tr>
<td>12</td>
<td>1111111110</td>
<td>-4095, -2048, 2048, 4095</td>
</tr>
<tr>
<td>13</td>
<td>11111111110</td>
<td>-8191, -4096, 4096, 8191</td>
</tr>
<tr>
<td>14</td>
<td>111111111110</td>
<td>-16383, -8192, 8192, 16383</td>
</tr>
<tr>
<td>15</td>
<td>1111111111110</td>
<td>-32767, -16384, 16384, 32767</td>
</tr>
<tr>
<td>16</td>
<td>1111111111111</td>
<td>Original raw sample</td>
</tr>
</tbody>
</table>
Table 7.9.
Lossless compression performance on Volcano data (block size:24 samples)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Compression ratio CR (%) 24 bit/sample</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GPC/S-LEC</td>
</tr>
<tr>
<td>Node 200</td>
<td>28.45</td>
</tr>
<tr>
<td>Node 201</td>
<td>25.91</td>
</tr>
<tr>
<td>Node 202</td>
<td>28.75</td>
</tr>
<tr>
<td>Node 203</td>
<td>31.87</td>
</tr>
<tr>
<td>Node 205</td>
<td>24.16</td>
</tr>
<tr>
<td>Node 207</td>
<td>26.05</td>
</tr>
<tr>
<td>Node 210</td>
<td>28.54</td>
</tr>
<tr>
<td>Node 212</td>
<td>29.18</td>
</tr>
<tr>
<td>Node 214</td>
<td>28.94</td>
</tr>
<tr>
<td>Node 251</td>
<td>22.52</td>
</tr>
<tr>
<td>Average CR</td>
<td>27.44</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>2.61</td>
</tr>
</tbody>
</table>

tatings of the coding table as each individual rotating state adapts to specific sensor readings. The sink of WSN with n nodes would then have to concurrently operate \( n \times t \) rotatings of the coding table for its operation. In contrast, our approach only needs to operate one single static coding table in a stateless manner at each sensor node and the sink, which dramatically reduces the operating complexity at both sensor nodes and the sink. This then has great practical significance in real-world WSN applications.

Next, we provide an analysis to explain the improvement achieved by the proposed S-LEC. As indicated in Section 7.4.2, the improvement comes from exploiting the sequential correlation between adjacent residues. In the devised four sequential contexts (via two-bit sequential coding) of any two adjacent residues as shown in Table 7.2, the first three contexts will save the group code of the second residue. Since a group code of LEC is at least two bits, S-LEC will either use the same number of bits in the case that the group code of the second residue is two bits, or use fewer bits otherwise (called gain situation), compared to LEC. In the fourth sequential context listed in
Figure 7.10. The distribution of the compression packet length (seismic).

Table 7.2, S-LEC would use more bits compared to LEC (called loss situation) in most cases. However, a gain situation is likely to save many bits, as many group codes are long in LEC; on the other hand, for a loss situation, only one or two bits
are wasted. Thus, due to the sequential correlations existing in a residue sequence, the gain probability would be high, resulting in significant gain in compression performance. We provide the statistical analysis of gain probability and loss probability of S-LEC, as well as the corresponding total number of bits saved and wasted, compared to LEC, on the volcano data from all 10 nodes, as listed in Table 7.10. As we can see, the gain probability of S-LEC sequential coding is significantly higher than the loss probability, resulting in the significant improvement of compression performance, as indicated by the column of net gain with respect to LEC.

Table 7.10.
Lossless compression performance on Volcano data (block size: 24 samples)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Gain Probability (%)</th>
<th>Loss Probability (%)</th>
<th>Number of Bits Saved</th>
<th>Number of Bits Wasted</th>
<th>Net Gain (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 200</td>
<td>72.58</td>
<td>22.73</td>
<td>37212</td>
<td>3346</td>
<td>33866</td>
</tr>
<tr>
<td>Node 201</td>
<td>72.12</td>
<td>23.32</td>
<td>38402</td>
<td>3502</td>
<td>34900</td>
</tr>
<tr>
<td>Node 202</td>
<td>72.02</td>
<td>23.23</td>
<td>36271</td>
<td>3490</td>
<td>32781</td>
</tr>
<tr>
<td>Node 203</td>
<td>71.48</td>
<td>23.75</td>
<td>32371</td>
<td>3575</td>
<td>28796</td>
</tr>
<tr>
<td>Node 205</td>
<td>75.03</td>
<td>20.55</td>
<td>44788</td>
<td>3048</td>
<td>41740</td>
</tr>
<tr>
<td>Node 207</td>
<td>72.60</td>
<td>22.77</td>
<td>40346</td>
<td>3398</td>
<td>36948</td>
</tr>
<tr>
<td>Node 210</td>
<td>72.33</td>
<td>22.91</td>
<td>36930</td>
<td>3437</td>
<td>33493</td>
</tr>
<tr>
<td>Node 212</td>
<td>73.01</td>
<td>22.27</td>
<td>38382</td>
<td>3332</td>
<td>35050</td>
</tr>
<tr>
<td>Node 214</td>
<td>71.87</td>
<td>23.38</td>
<td>32520</td>
<td>3177</td>
<td>29343</td>
</tr>
<tr>
<td>Node 251</td>
<td>76.79</td>
<td>18.85</td>
<td>48005</td>
<td>2641</td>
<td>45364</td>
</tr>
</tbody>
</table>

7.4.3 Lossy Compression

All types of sensors used in sensor networks introduce sensing errors. Typically available only from a sensor’s data sheet provided by its vendor, the only information that can be found describing a sensor’s accuracy is the so-called sensor manufactured error (SME). SME is the maximum margin error that could be produced by a sensor operated normally under given conditions. In lossy compression, as larger tolerable compression errors lead to higher compression ratios, the control knob of compression
error bound $e$ can be conveniently specified as a percentage of SME. According to the application’s requirements on data quality, the extent of compression errors is bounded and adjustable by the control knob in order to maximize the lossy compression ratio. For example, if $e = 50\%$ of SME, the possible maximum compression error for each individual raw sample is bounded by the degree of its SME/2.

First, Figures 7.11 and 7.12 show the compression ratios for ambient temperature and relative humidity data sets, respectively, by our GPC/S-LEC and the LTC with different values of $e$. To demonstrate that GPC is general and can work with any other lossless entropy encoders, we also include the evaluation of GPC/LEC in Figures 12 and 13. In addition, we include the evaluation of Salomon’s solution by replacing our average predictor in BLC scheme with Salomon’s method to calculate the difference, denoted as GPC/S-LEC/Salomon. Note that in Figures 12 and 13, $e = 0$ indicates lossless compression in our GPC, whereas LTC only works for lossy compression. The raw sample size of 14 and 12 bits for ambient temperature and relative humidity are used in the lossy compression respectively. We again assume that there are 32 samples per packet.

As shown in Figures 7.11 and 7.12, the lossy compression ratio $CR$ of the GPC/S-LEC can quickly reach about 70% when the error bound $e$ is only 20% of SME for most data sets, while the LTC algorithm requires a higher error bound to achieve $CR = 70\%$. This indicates that GPC/S-LEC is generally better than LTC for lossy compression when only small compression errors are allowed. Even when the error bound $e$ is increased up to 100% of SME, GPC/S-LEC still outperforms LTC for four out of the eight data sets from SensorScope.

Comparing the GPC/LEC performance in Figures 7.11 and 7.12 with our performance predictions in Figure 8, it can be seen that we predict the performance quite accurately. This shows again that our discrete Laplace model captures the residue distribution very well.

In general, GPC/S-LEC and GPC/LEC performs better than LTC when a given error bound is moderate. As the error bound increases, LTC then has some advantage.
On the other hand, GPC/S-LEC always outperforms GPC/LEC, demonstrating the merit of S-LEC. We also observe from the figures that GPC/S-LEC/Salomon performs better than GPC/S-LEC, indicating the advantage of Salomon’s calculation method of difference in lossy compression for smooth data sets. One interesting observation is that when a data set has a lower CR for lossless compression, evidence of the high entropy of the data set, its lossy CR by GPC/S-LEC can be significantly better than LTC even for a large error bound (e.g., left bottom of both Figures 7.11 and 7.12).

Figure 7.11. Lossy compression ratio under various error bound $e$ for SensorScope ambient temperature data set.

Next, we evaluate our GPC/S-LEC and the LTC for lossy compression using real-world WSN seismic data sets [75]. While SensorScope temperature and relative
humidity data are smooth, the seismic data of volcanic eruptions are highly dynamic. We assume that there are 24 samples per packet as the raw sample size is 24 bits.

The comparison results for sensor nodes 203 and 251 are shown in Figures 7.13 and 7.14, respectively. Table 7.11 lists the results for all 10 nodes, as the detailed comparison results for the other nodes are similar with those of Figures 7.13 and 7.14 and hence are omitted. While LTC can perform reasonably well for relatively smooth temperature and relative humidity data sets from SensorScope, it completely failed for all dynamic seismic data sets. In contrast, the GPC/S-LEC can still perform quite well, achieving about $CR = 60\%$ or higher when the error bound $e$ is just $40\%$ of SME. The evaluation results clearly indicate that the proposed GPC/S-LEC is much more
robust than LTC for lossy compression. Also, in contrast with the performance on the smooth data sets, our GPC/S-LEC overall performs slightly better than GPS/S-LEC/Salomon for these highly dynamic data sets, as shown in Figure 7.15 In practice, the average predictor devised in our BLC scheme is simpler to implement than the Salomon solution on sensor nodes.

![Figure 7.13. Lossy compression ratio under various error bound e for Node 203 data set.](image)

7.5 Energy Consumption Model and Reliable Transmission

7.5.1 Energy Consumption Model

A simplified energy consumption model on source sensor nodes for the proposed GPC/S-LEC can be obtained by counting the number of basic operations (e.g., shifts, additions) conducted in the following outlined algorithmic logic.
Table 7.11.
Lossy Compression Performance on Volcano Data (block_size:24 samples)

<table>
<thead>
<tr>
<th>Node ID</th>
<th>GPC/S-LEC</th>
<th>GPC/LEC</th>
<th>GPC/S-LEC/Salomon</th>
<th>LTC</th>
<th>GPC/S-LEC</th>
<th>GPC/LEC</th>
<th>GPC/S-LEC/Salomon</th>
<th>LTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>62.89</td>
<td>60.92</td>
<td>59.15</td>
<td>-17.21</td>
<td>68.50</td>
<td>67.66</td>
<td>64.79</td>
<td>-12.55</td>
</tr>
<tr>
<td>201</td>
<td>61.17</td>
<td>58.87</td>
<td>62.90</td>
<td>-17.94</td>
<td>66.99</td>
<td>66.11</td>
<td>68.71</td>
<td>-14.45</td>
</tr>
<tr>
<td>202</td>
<td>63.34</td>
<td>61.97</td>
<td>63.90</td>
<td>-17.52</td>
<td>69.04</td>
<td>68.46</td>
<td>69.61</td>
<td>-12.82</td>
</tr>
<tr>
<td>203</td>
<td>66.08</td>
<td>65.86</td>
<td>65.70</td>
<td>-16.57</td>
<td>71.54</td>
<td>71.39</td>
<td>71.29</td>
<td>-11.01</td>
</tr>
<tr>
<td>205</td>
<td>58.99</td>
<td>54.7</td>
<td>56.74</td>
<td>-18.63</td>
<td>64.87</td>
<td>63.09</td>
<td>62.49</td>
<td>-15.32</td>
</tr>
<tr>
<td>207</td>
<td>60.99</td>
<td>57.92</td>
<td>61.41</td>
<td>-17.96</td>
<td>66.56</td>
<td>65.33</td>
<td>67.15</td>
<td>-14.06</td>
</tr>
<tr>
<td>210</td>
<td>63.27</td>
<td>61.53</td>
<td>64.64</td>
<td>-17.57</td>
<td>68.81</td>
<td>68.18</td>
<td>70.30</td>
<td>-13.00</td>
</tr>
<tr>
<td>212</td>
<td>63.32</td>
<td>61.16</td>
<td>63.88</td>
<td>-16.62</td>
<td>68.91</td>
<td>67.71</td>
<td>69.51</td>
<td>-10.88</td>
</tr>
<tr>
<td>214</td>
<td>63.68</td>
<td>62.48</td>
<td>65.09</td>
<td>-16.98</td>
<td>69.25</td>
<td>68.96</td>
<td>70.76</td>
<td>-12.00</td>
</tr>
<tr>
<td>251</td>
<td>56.87</td>
<td>50.52</td>
<td>52.19</td>
<td>-18.92</td>
<td>62.76</td>
<td>59.49</td>
<td>57.95</td>
<td>-16.65</td>
</tr>
</tbody>
</table>
For S-LEC entropy encoder, it needs $n$ branches to obtain the code if the residue is in the $n^{th}$ group of the coding table. Based on the simulation, we obtain the actual operation numbers which are used in the following compression energy consumption. Consider the widely used CC2420 radio transceiver [81], and both ARM7TDMI microprocessor [82] and MSP430 micro-controller [66,83] of motes. Based on their data sheets and reported experience [84–86] we summarize the parameters in Table 7.12. $I_{TX}$ and $I_{RX}$ are the current draw of sending and receiving by the radio respectively; $T_{TX}$ and $T_{RX}$ are the corresponding operating time over 1 byte; $V$ is the voltage supply, which we assume to be constant throughout the transmission. For simplicity, the model does not include possible variable levels of transmission power but only the highest power level. It does not include the effect of header and CRC error code overhead. The simplicity of our model, however, would be feasible for the comparison of different data compression algorithms, where the same assumptions are made for
Figure 7.15. 15 Average performance of different predictors for all the 10 volcano data set.

individual compression algorithms under comparison. For example, we assume the same number of samples per packet, and thus the same number of total packets in comparison. As a result, the effect of header and CRC error code can be eliminated in such relative performance comparisons without affecting the comparison result. While the model discussed in this subsection assumes a desirable channel condition where the bit-error-rate (BER) can be negligible, the impact of BER is specifically discussed in the next subsection. We only consider one hop transmission in the following calculation. Therefore, transmitting $k$ bytes per hop requires:

$$E_{\text{ratio}}(k) = kI_{TX}V_{TX} + kI_{RX}T_{RX}.$$  (7.5)

Let us denote $N_{\text{add}}, N_{\text{mul}}, N_{\text{shl}}, N_{\text{BR}}$ the number of needed addition (also including bit or, load, etc., which have the same number of clock cycle as addition), multiplica-
tion, shift, and branch operations, respectively, in microprocessor computation. The total computation energy consumption is as follows:

\[ E_{\text{comp}} = N_{\text{add}} \epsilon_{\text{add}} + N_{\text{mul}} \epsilon_{\text{mul}} + N_{\text{sht}} \epsilon_{\text{sht}} + N_{BR} \epsilon_{BR}, \]  

(7.6)

where \( \epsilon_{\text{add}}, \epsilon_{\text{mul}}, \epsilon_{\text{sht}}, \) and \( \epsilon_{BR} \) are the energy consumption of addition, multiplication, shift, and branch instruction, respectively (see Table 7.12). The total energy consumption is:

\[ E = E_{\text{radio}} + E_{\text{comp}} \]  

(7.7)

Table 7.12.
Parameters of the energy models

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_{TX} )</td>
<td>17.4 mA</td>
<td>( I_{CPU} )</td>
<td>30.5 mA</td>
</tr>
<tr>
<td>( I_{RX} )</td>
<td>19.7 mA</td>
<td>( f_{CPU} )</td>
<td>48 MHz</td>
</tr>
<tr>
<td>( T_{TX} )</td>
<td>( 3.2 \times 10^{-5} ) s</td>
<td>( \epsilon_{\text{add}} )</td>
<td>1.91 nJ</td>
</tr>
<tr>
<td>( T_{RX} )</td>
<td>( 3.2 \times 10^{9} ) s</td>
<td>( \epsilon_{\text{mul}} )</td>
<td>5.73 nJ</td>
</tr>
<tr>
<td>( V )</td>
<td>3.0 V</td>
<td>( \epsilon_{BR} )</td>
<td>5.73 nJ</td>
</tr>
<tr>
<td>( \epsilon_{sht} )</td>
<td>3.82 nJ</td>
<td>( \epsilon_{sht} )</td>
<td>0.68 nJ (1 bit)</td>
</tr>
</tbody>
</table>

Table 7.13 shows the total energy consumption of lossless compression, including the energy consumption portions for radio and computing respectively. Clearly, GPC/S-LEC obtains a higher energy gain than LEC. Even in a one hop WSN, the computation energy cost of GPC/S-LEC is significantly less than its communication energy gain.

When comparing any two data compression schemes A and B, if \( E_{\text{comp}}^A > E_{\text{comp}}^B \) and \( E_{\text{radio}}^A < E_{\text{radio}}^B \), we can introduce a new measure, defined as \( \eta_{A,B} = \frac{E_{\text{comp}}^A - E_{\text{comp}}^B}{E_{\text{radio}}^B - E_{\text{radio}}^A} \), to represent the tradeoff between the computation and communication for the energy consumption evaluation of schemes A and B. If \( \eta_{A,B} < 1 \), it means that scheme A overall has less energy consumption than scheme B because A reduces more communication energy than its increased computation energy; on the other hand, if \( \eta_{A,B} > 1 \), it means that scheme A overall has more energy consumption than scheme B because the reduced communication energy by A is smaller than its increased computation.
### Table 7.13.
Lossless energy comparisons (block size: 264 samples)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data set</th>
<th>ARM7TDMI: Energy(mJ) ($E_{radio} + E_{comp}$)</th>
<th>MSP430: Energy(mJ) ($E_{radio} + E_{comp}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GPC/S-LEC</td>
<td>LEC</td>
</tr>
<tr>
<td>Ambient Temperature</td>
<td>FN_ID101</td>
<td>31.69 (30.98+0.71)</td>
<td>31.99 (31.47+0.52)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31.22 (30.98+0.24)</td>
<td>31.65 (31.47+0.18)</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>132.67 (129.40+3.27)</td>
<td>138.94 (136.62+2.32)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>130.50 (129.40+1.10)</td>
<td>137.40 (136.62+0.78)</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>70.10 (68.64+1.46)</td>
<td>71.14 (70.04+1.10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>69.16 (68.64+0.52)</td>
<td>70.44 (70.04+0.40)</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>70.77 (69.36+1.41)</td>
<td>72.29 (71.22+1.07)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>69.87 (69.36+0.51)</td>
<td>71.62 (71.22+0.40)</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td>FN_ID101</td>
<td>31.98 (31.31+0.67)</td>
<td>34.19 (33.67+0.52)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31.55 (31.31+0.24)</td>
<td>33.86 (33.67+0.19)</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>171.78 (168.05+3.73)</td>
<td>176.06 (173.29+2.77)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>169.36 (168.05+1.31)</td>
<td>174.28 (173.29+0.99)</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>77.23 (75.71+1.52)</td>
<td>81.02 (79.85+1.17)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76.27 (75.71+0.56)</td>
<td>80.29 (79.85+0.44)</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>76.02 (74.57+1.45)</td>
<td>80.21 (79.09+1.12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>75.11 (74.57+0.54)</td>
<td>79.52 (79.09+0.43)</td>
</tr>
</tbody>
</table>
Table 7.14.
Lossless communication/computation energy tradeoff ratio (block_size:264 samples)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data set</th>
<th>$\eta_{A,B} = \frac{E_{comp}^A - E_{comp}^B}{E_{radio}^A - E_{radio}^B}$</th>
<th>A: GPC/S-LEC</th>
<th>B: LEC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>ARM7TDMI</td>
<td>MSP430</td>
</tr>
<tr>
<td>Ambient Temperature</td>
<td>FN_ID101</td>
<td>0.394</td>
<td>0.131</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>0.131</td>
<td>0.044</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>0.258</td>
<td>0.087</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>0.179</td>
<td>0.060</td>
<td></td>
</tr>
<tr>
<td>Relative Humidity</td>
<td>FN_ID101</td>
<td>0.065</td>
<td>0.021</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>0.184</td>
<td>0.062</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>0.083</td>
<td>0.028</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>0.072</td>
<td>0.024</td>
<td></td>
</tr>
</tbody>
</table>
energy. Table 7.14 shows $\eta_{GPC/S-LEC, LEC}$ for various data sets. As we can see, $\eta_{GPC/S-LEC, LEC}$ is much smaller than one for all data sets, indicating that GPC/S-LEC overall has less energy consumption than LEC, due to the fact that GPC/S-LEC has a much better energy consumption tradeoff between the computation and communication in comparison with LEC.

Table 7.15 shows the total energy consumption of lossy compression where the error bound $e$ is 50% of SME. As we can see, for most WSN data sets, GPC/S-LEC obtains higher energy gain than LTC, except for LU_ID84 ambient temperature data set, which can also be justified from Figure 7.12, where LTC performs better when $e$ is 50% of SME.

### 7.5.2 Reliable Transmission

WSNs are often operated in dynamic and harsh outdoor environments, where the bit-error-rate (BER) is typically high. To ensure reliability in noisy WSNs, automatic repeat request (ARQ) is often applied at the data link layer. In the following discussions, only the payload of a frame is considered. Let $s$ be the frame payload size, the frame-error-rate (FER) is characterized as

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data set</th>
<th>ARM7TDMI: Energy(mJ) ($E_{\text{radio}} + E_{\text{comp}}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambient Temperature</td>
<td>FN_ID101</td>
<td>15.55 (15.06+0.49) 16.42 (15.51+0.91)</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>75.14 (73.19+1.95) 54.98 (50.29+4.70)</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>34.16 (33.21+0.95) 59.19 (57.49+1.71)</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>34.44 (33.49+0.95) 74.82 (73.28+1.54)</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td>FN_ID101</td>
<td>17.33 (16.85+0.47) 24.40 (23.49+0.92)</td>
</tr>
<tr>
<td></td>
<td>LU_ID84</td>
<td>80.22 (78.08+2.13) 81.08 (76.37+4.71)</td>
</tr>
<tr>
<td></td>
<td>GSB_ID10</td>
<td>36.12 (35.11+1.01) 64.41 (62.70+1.71)</td>
</tr>
<tr>
<td></td>
<td>LG_ID20</td>
<td>34.99 (34.02+0.97) 68.84 (67.30+1.54)</td>
</tr>
</tbody>
</table>
\[ FER = 1 - (1 - BER)^s. \]

Then, the expected number of transmission \( E(k) \) to get one frame successfully transferred over a wireless link is

\[
E(k) = \sum_{k=1}^{\infty} k \times FER_{k-1}(1 - FER) = \frac{1}{1 - FER} = \frac{1}{(1 - BER)^s}. \tag{7.8}
\]

(7.8) represents a theoretical characterization of the ultimately expected number of transmissions to achieve a reliable data transfer. However, in practice, the number of retransmissions is often limited by a threshold, which means the completely reliable data transfer would not be possible. Nevertheless, (7.8) can be effectively adopted as the underlying basis for the discussion of reliable transmission under unreliable channels regarding energy consumption. Let \( s' \) and \( E(k') \) be the corresponding frame payload size and expected number of transmission for the compressed frame, respectively. Then we can define data transmission ratio (DTR) \( \phi \) as follows to evaluate the overall energy gains from the data compression in noisy WSNs:

\[
\phi = \frac{s' \times E(k')}{s \times E(k)} = (1 - CR)((1 - BER)^s)^{CR}, \tag{7.9}
\]

in which \( CR \) is the compression ratio. A smaller \( \phi \) indicates a higher energy saving by the data compression in noisy WSNs.

The relationship characterized by (7.9) between compression ratio and data transmission ratio under various link conditions is shown in Figure 7.16. When the link condition is bad (i.e., BER is high), DTR decreases very rapidly with the increasing of compression ratio, indicating that even a moderate compression ratio \( CR \) can achieve substantial energy savings in noisy WSNs due to retransmissions.

Figure 7.17 shows the relationship between energy consumption ratio and compression ratio given \( BER = 10^{-2} \), when GPC/S-LEC is applied. In fact, the energy consumption ratio curve shown in Figure 7.18 almost overlaps with the data transmission ratio curve with \( BER = 10^{-2} \) in Figure 7.16. This indicates that indeed the energy cost for computation can be negligible.
Figure 7.16. Data transmission ratios under various BER with a frame size of 48 bytes.

Figure 7.17. Energy consumption ratio with a frame size of 48 bytes when BER is $10^{-2}$. 
8 SUMMARY AND FUTURE WORK

8.1 Summary

In this thesis, we address these critical open questions and present a novel CS approach called CSR for multi-hop WSN data acquisition based on dynamic routing topology tomography. Our CSR approach has two distinguishing characteristics. First, CSR introduces the use of WSN routing topology tomography into CS approach and thus provides a practical and elegant solution for large-scale WSN data acquisition based on effective interplaying with dynamic routing. We show that the adoption of routing matrix as measurement matrix in compressed sensing in recovering $k$-sparse sensor signals in WSN can achieve feasible estimation with bounded errors. As shown in our real-world WSN experiments, our CSR approach not only considerably reduces transmissions, resulting in an order of magnitude less in energy consumption compared to CDG, but also significant reduction of transmission costs compared to CDC and RS-CS as well due to the WSN routing topology tomography, thus extending the lifetime of real-world outdoor WSN deployments. Second, CSR provides a systematic method to construct an optimized representation basis with both good sparsification and incoherence properties for various given classes of signals, and therefore drastically reduces WSN data recovery errors by an order of magnitude compared to existing CS schemes CDG, CDC and RS-CS. Therefore, the proposed approach is expected to significantly improve the state of art of CS based approaches for WSN data acquisition, and to facilitate the CS application in large-scale multi-hop outdoor WSN systems for various data gathering.

Our approach is deployed for a real-world outdoor WSN testbed and is rigorously validated and evaluated via the long term WSN deployment in situ operated under highly dynamic communication environment for environmental monitoring. To the
best of our knowledge, our work represents the first demonstration and performance analysis of CS approach applied to real-world WSN deployment in situ jointly with routing for data acquisition with actual routing protocol in operation. It is expected that the presented systematic method in our CSR approach for constructing an optimized representation basis can be in general adopted to any other CS schemes to significantly improve their data recovery fidelity in big data acquisition.

We also develop a generalized algorithmic framework referred to as generalized predictive coding for unified lossless and lossy compression. This proposed GPC framework is simple and efficient, and is particularly suited for highly resource-constrained WSN nodes where both lossless and lossy compressions are needed in applications. To our knowledge, this is the first work on the unification of lossless and lossy compression for WSNs. In addition, GPC is able to accommodate the resource limitation of the tiny sensor nodes, at a small expense of slightly degraded compression ratio. This is accomplished through the selection of a smaller compression radius than the default maximum compression radius as specified by the precision of sensor samples.

8.2 Future Work

In Chapter 7, the predictive coding framework allows us to choose an error margin, then decided how to compress the data. But with compressed sensing approaches, we cant precisely control the error margin as we want. Can we do this in the future? Many error bound theorems in compressed sensing area are proved with assumption of strictly k-sparse signals, while in practice we may only find the basis which can transform the original signals to nearly k-sparse. How can we find the error bound in practice?
REFERENCES
REFERENCES


