ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Low-Overhead Compression of ECG Recordings for Implantable Medical Devices

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του
Σωμαράκη Αντώνιου

Επιβλέπων : Δημήτριος Ι. Σούντρης
Αναπληρωτής Καθηγητής ,Ε.Μ.Π

Αθήνα, Μάρτιος 2016

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΤΗΜΑΤΩΝ

Low-Overhead Compression of ECG Recordings for Implantable Medical Devices

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του
Σωμαράκη Αντώνιου

Επιβλέπων :Δημήτριος Ι. Σούντρης
Αναπληρωτής Καθηγητής ,Ε.Μ.Π

Εγκρίθηκε από την τριμελή επιτροπή την ημερομηνία εξέτασης.

....	....	....
Δημήτριος Ι. Σούντρης	Κιαμάλ Πεκμεστζή	Γεώργιος Μάτσόπουλος
Αναπληρωτής	Καθηγητής,Ε.Μ.Π.	Αναπληρωτής
Καθηγητής,Ε.Μ.Π.		Καθηγητής,Ε.Μ.Π.

Αθήνα ,Μάρτιος 2016

....

Σωμαράκης Αντώνιος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
Ε.Μ.Π.

Περίληψη

Στην εποχή μας οι εμφυτεύσιμες ιατρικές συσκευές, γνωστές παγκοσμίως και ως IMDs χρησιμοποιούνται ολοένα και περισσότερο στην Ιατρική. Τα IMDs συναντώνται σε μια μεγάλη ποικιλία ιατρικών εφαρμογών, βασίζονται στην απόκτηση ,επεξεργασία και μεταφορά δεδομένων ούτως ώστε να αναβαθμίσουν την καθημερινότητα των ασθενών αλλά και πολλές φορές να τους διατηρήσουν στη ζωή. Το πρόβλημα που παρουσιάζεται με τα IMDs είναι ότι έχουν περιορισμένη μνήμη, υπολογιστικές δυνατότητες και ενεργειακή επάρκεια ενώ πρέπει να συλλέγουν, επεξεργάζονται και να μεταδίδουν δεδομένα από διάφορους πιθανούς αισθητήρες. Όλοι αυτοί οι περιορισμοί καταδεικνύουν την ανάγκη οι πληροφορίες που συλλέγονται και μεταδίδονται από τα IMDs να συμπιέζονται όσο πιο αποδοτικά γίνεται. Στην προκειμένη περίπτωση η συμπίεση δεδομένων αποτελεί μια πρόκληση, καθώς το ανακτώμενο σήμα θα πρέπει να έχει υψηλή πιστότητα σε σχέση με το αρχικό. Επίσης, για να μπορεί να επιτευχθεί πιο αποδοτική συμπίεση η μέθοδος που θα χρησιμοποιηθεί θα πρέπει να είναι προσανατολισμένη στον τύπο των δεδομένων που θα συμπιέσει. Στην παρούσα διπλωματική θα προσπαθήσουμε να βελτιώσουμε τις υπάρχοντες μεθόδους συμπίεσης δεδομένων. Για να το καταφέρουμε αυτό εξετάζουμε διάφορους αλγορίθμους και συνδυασμούς αυτών για να βρούμε το πιο αποδοτικό σχήμα. Οι δύο κύριοι αλγόριθμοι που εξετάζουμε είναι ο LZO και ο SPIHT. Στην προσπάθεια μας ώστε να βελτιστοποιήσουμε την απόδοση τους ,τους συνδυάζουμε με διάφορες τεχνικές επεξεργασίας δεδομένων. Βασική μας επιδίωξη είναι να αξιολογήσουμε τους παραπάνω αλγορίθμους. Η αξιολόγηση αυτή γίνεται πάνω στην συμπίεση Ηλεκτροκαρδιογραφήματος, ένα από τα πιο διαδεδομένα βιολογικά σήματα το οποίο καταγράφεται από πολλά IMDs τα οποία είτε το μεταδίδουν απευθείας είτε το αποθηκεύουν για μελλοντική χρήση. Η κύριες παράμετροι στις οποίες βασίζεται η αξιολόγηση μας είναι η αναλογία συμπίεσης ενός αρχείου(CR) ,η ποσοστιαία ριζική μέση τετραγωνική διαφορά (PRD) και το ενεργειακό αποτύπωμα του κάθε αλγορίθμου. Εν τέλει, βασιζόμενοι στην διαδικασία αξιολόγησης συμπεραίνουμε ότι ο SPIHT αλγόριθμος με την μέθοδο ανακατάταξης με fuzzy C means Clustering προσφέρει τη μεγαλύτερη αναλογία συμπίεσης. Από την άλλη, την καλύτερη σχέση μεταξύ CR και PRD προσεγγίζει ο LZO αλγόριθμος με στρογγυλοποίηση.

Λέξεις κλειδιά:

εμφυτεύσιμες ιατρικές συσκευές, Ηλεκτροκαρδιογράφημα, συμπίεση βιοδεδομένων, συμπίεση δεδομένων με χαμηλή ενεργειακό αποτύπωμα, αναλογία συμπίεσης , ποσοστιαία ριζική μέση τετραγωνική διαφορά ,SPIHT , LZO

Abstract

It is an indisputable fact that Implantable Medical Devices (IMDs) are becoming an integral part of Medical science. IMDs are encountered in a great variety of medical applications. IMDs rely on data acquisition, processing and communication agents in order to sustain and ameliorate the life of the patients. IMDs have limited memory, computational and battery power resources, while collecting, processing and transmitting out information from potentially many sensors. These limitations require that information within the devices be efficiently compressed. Such data compression presents a challenging task, as it must provide high fidelity of the waveform reproduction and high compression ratios on limited size data frames. Also, it must be based on the type of data to be compressed, in order to provide bigger efficiency. In this thesis we try to better up the existing lossy and lossless compression methods. In order to manage that, we use various algorithms and combinations of those in order to find the most efficient scheme. The two main algorithms that we use are LZO encoding algorithm and SPIHT encoding algorithm. We combine these encoding algorithms with various data procession algorithms. Our main attempt is to evaluate the aforementioned algorithms and so we use Electrocardiography (ECG), an extremely widely used biodata which is recorded from IMDs and sent or saved from them. The main evaluation parameters of our thesis are the compression ratio, the Percent Root mean square Difference (PRD) and computational overhead of each algorithm. Finally, based on the evaluation process we conclude that SPIHT with Reordering with fuzzy C means Clustering offer the best compression ratio 25.95 with RPD 4.86 and the best tradeoff between compression ratio and PRD the LZO with Reordering technique with 10.67 compression ratio and 3.13 .As for the lossless algorithms LZO with Reordering with fuzzy C means clustering offers 2.42 compression ratio.

Tags: Implantable Medical Devices (IMDs), Electrocardiography (ECG), data compression, ECG compression ratio, low overhead, Percent Root Difference (PRD), compression ratio, SPIHT , LZO

Acknowledgments

First off all, I would like to thank Professor Dimitrios Soudris for giving me the opportunity to carry out my diploma thesis under his supervision. This diploma thesis has been a unique opportunity for me so as to be introduced in the process of scientific research.
Moreover, I would also like to thank Dr. Robert Seepers for the continuous guidance he has provided me with throughout the development of this thesis and for all the knowledge he has shared with me, the help he has offered and his constant engagement. I would also like to thank Dr.Christos Stydis for the valuable assistance he offered me.
Finally, I want to thank all the people that stood beside me throughout the years of my studies in NTUA. I want to thank my friends for all the experiences we have had during these years and especially my family who encouraged me to achieve my goals and ambitions.

Contents

List of figures

1. <u>Introduction</u>

The technological breakthroughs in the field of biomedical engineering have revolutionized the way that medicine is performed by scientists. Implementable medical devices (IMDs) are such a prominent example. If we would like to give a definition to implantable medical devices we could say that “Implantable medical devices are the medical devices that either partly or totally introduced, surgically or medically, into the human body and is intended to remain there after the procedure” [1]. Over the years, implantable medical devices have saved and also improved the quality of life to a big amount of patients. Nowadays, IMDs are used in many different parts of the body of a patient such as orthopaedics, pacemakers, cardiovascular stents, defibrillators, neural prosthetics or drug delivery system. Particular interest present IMDs that can record transmit or store data such as ECG monitoring systems, Holter , ElectroGraM (EGM) or Implantable hemodynamic monitors and Insertable loop recorders. The last would be and the main devices of interest in our thesis. Moreover, as the life expectancy has grown significantly the need for new and improvement of the existing technologies that are related to implantable medical devices such as treatments, implants, prostheses and long-term pharmaceutical usage has increased. One significant example is Implantable cardioverter defibrillators (ICDs). ICDs are devices that monitor and treat cardiac arrhythmia, when it is detected by sending a large jolt of electricity to the heart, and basically pressing the reset button. In 2009, according to the World Society of Arrhythmias, 133,262 ICDs were implanted in the U.S. with a total annual expenditure of $5.5 billion and average cost per procedure $40,000 and an increase of 12% from 2005.
Furthermore, we should mention that medical devices are categorized in classes upon the level of control required to assure safety and effectiveness for the device. Implantable medical devices are in Class III. [2].
One more worth-mention factor that characterizes the IMDs is the energy consumption. Energy consumption can be divided into three domains: sensing, communication and data processing. Wireless communication is usually the most power consuming among those three [3]. Another worth mention factor is that the size of the battery used to store the needed energy is in most cases the largest contributor to the size and weight of IMDs. As a result batteries should be kept small. Therefore energy consumption of the devices needs to be reduced .In order to minimize cost, patient trauma and risk associated with repeated surgeries for battery replacement, it is necessary to increase the lifetime of implanted batteries by conserving energy at the most power consuming part of an IMD, as these are designed to work for many years, powered by their original battery the energy consumption is a very crucial issue. Hence, compression of Biodata that are produced from IMDs is

necessary and should be done in an energy efficient way so as to tackle with both problems the big data rate and also the restrictions on the power consumption introduce.

There are 3 main types of data types that an IMD holds [4].

- Static Data
  - Device identification data(name, version of the device)
- Semi-static Data
  - Patient identification data(name, age of the patient)
  - Health condition data
  - Therapy configuration and Health Center Identification
- Dynamic Data
  - Patient readings(ECG, heart rate etc)
  - Audit log data(device's operational history)

From all the above data the Dynamic data consist the 70% percent of the entire data that an implantable medical device holds. Example of Dynamic Data could be an ECG .A typical cardiac IMD, can produce ECGs of data size up to 112 megabytes per day. Furthermore, the growth of IMDs in conventional medicine and their restricted capacity and energy consumption ability makes Data compression especially for IMDs necessary.

1.1. Problem Statement

After a thorough investigation on academic literature we found out that all the compression algorithms tailored to the ECGs are mostly orientated to manage big compression ratio and quite a few algorithms have point out the energy efficiency aspect of their algorithms used by Implantable Medical Devices. What is more, the aforementioned energy consumption problem for IMDs and their popularity makes the solution of the problem crucial.

1.2. Thesis goal

As we mentioned the problem which was the initiative of this thesis, we should now state and the goal and main endeavor of the thesis. So, our goal is to compare and create algorithms that can compress Biodata and more specifically ECGs in an energy efficient way with minimal loss of signal quality achieving decent compression ratio. Therefore we use a lightweight version of LZO encoding algorithm, which is designed for Implantable Medical Devices and from the existing academic literature it presents quite good compression results. Moreover, we make use of the SPIHT encoding algorithm which manages to offer pretty good results especially with the "Reordering" technique.

1.3. Outline

This dissertation is composed from 5 chapters. The first chapter is the Introduction chapter, which tries to give the reader a first but thorough look on the subject and shed some light on the problem that tries to solve and the goal of this dissertation. In chapter 2, the Background chapter is cited all the important information in order somebody to understand the main topic of the dissertation and the algorithms that we use. Also, it contains all the findings of the academic literature on the ECG compression field of research. Chapter 2 is followed by the Implementation chapter, chapter 3. In chapter 3 we describe analytically all the algorithms that can compress Biodata in an energy efficient way with minimal loss of signal quality. Also, we mention the tools that we have used in order to implement the aforementioned algorithms. Evaluation chapter comes after the Implementation chapter. The evaluation chapter contains all the results that come through the Implementation of the algorithms that we used. Moreover, the evaluation of the algorithms is done at this point. Finally, we have the Conclusion chapter. This chapter summarizes all the research that we have done. Furthermore, we state and the future work that can be done, regarding this research.

2. Background and related work

In this chapter we provide all the background information needed in order to understand the following research. We provide definitions for the main research terms and also we state all the background information for the main encoding algorithms that we would use excessively in the process. So it is a very important chapter for the further understanding of the Thesis. Moreover, in this chapter we try to provide our research findings related to our work.

2.1. General description of Data compression

Firstly we would state the definition of Data compression. "Data compression is the process of modifying, encoding or converting the bits structure needed to store or transmit data". Data compression techniques are categorized according to the requirements of reconstruction as those in which the compressed data is reconstructed to form the original signal(lossless techniques) and techniques in which higher compression ratios can be achieved by producing reconstructed signal different to the original one(lossy techniques).In Lossless compression we do not have any loss of information and is often used for applications that cannot tolerate any difference between the original and the reconstructed signal, text compression is such an example. On the other hand, Lossy compression involves an escalated percentage of information loss. Depending on the quality required of the reconstructed signal, varying amounts of loss of information about the value of each sample can be tolerated and is used for applications where exact reconstruction is not so important, like image compression [5].
A compression algorithm could be evaluated in a number of different ways. We could measure the memory required to implement and run the algorithm, how fast the algorithm performs on a specific machine-both these metrics consist the computational overhead of an algorithm - ,the amount of compression, which is described in terms of compression ratio(CR) and how closely the reconstruction resembles the original, often measured as the percent mean-square difference (PRD). In the following chapters and especially in the Evaluation Chapter we would analyze in detail the aforementioned metrics.

2.2. LZO encoding algorithm

The LZO algorithm is a lossless widely used algorithm that manages the files to be compressed in blocks. This algorithm could theoretically manage big compression and decompression speed as it enables paralleling when the parallel procedures are running in different file-blocks at the same time.
The LZO algorithm is introduced in [6]. From the given source code of LZO library we can make out that the original file, which we would like to compress is divided into file-blocks which have the same size as the L2 cache memory of the processor. The compression process can be described by the following steps. Firstly, in every file-block it is given to each group of 4 bytes a hash value. The value of the hash function is affiliated with the value of the 4 bytes group. All these hash values form a compression a hash table is kept which is able to store one memory address for each hash value. In order the algorithm to secure its quick run the size of the hash table the hash table (memory address size * hash variations) is equal to the size of L1 cache.
We quote here an example as it is stated in [7].

"Given a text file with the following data is compressed:

SOMETHING IS A THING, THAT IS IMPORTANT

Let the pointer assigned to the beginning of the text be PTR0. Into the hash place ('S','O','M','E') of the hash table PTR0+3 is written, into the place of hash (O, M, E, T) PTR0+4 is written, etc. In order to detect the recurrences the algorithm checks the already existing values when writing into the hash table. If the new memory address and the initial address (a random number at the beginning) are close values, the algorithm will check whether there is a real recurrence, so the byte groups of four are compared (this comparison is necessary because of the initial random hash values and the hash collision). In the case of the example above the hash (T, H, I, N) is calculated in the 16th step, so the value PTR0+15 should be written into the table. However there already exists a memory address which is the value PTR0+7 (it has been written into the table at the 5th step, at the word "something"). In order to decide whether there is a hash collision or not, the algorithm compares the values: ^PTR0+7 == ^PTR0+15 and ^PTR0+6 == ^PTR0+14 and ^PTR0+5 == ^PTR0+13 and ^PTR0+4 == ^PTR0+12 If the condition above is fulfilled, it means that the same byte group of four is at both places (in this case the „thin" part of the word). At this place the file can be compressed in that way that instead of the recurrence the initial position and length of the original word-part is written into the given position.

SOMETHING IS A (RECURRENCE from the place back 11 bytes, length: 5 bytes), THAT IS IMPORTANT

For the determination of the length of the recurrence the check of the coincidence goes in the memory from bytes to bytes. In the case above ^PTR0+8 equals to ^PTR0+16 but in the next step this is not fulfilled anymore (space and comma characters are not coincide). It means that in the compressed file only the length and the object of the coincidence are stored.”

The LZO source code can be found at [6].The children of the LZO the miniLZO(mLZO) is an Lempel–Ziv family derivative designed with the processing, memory, and code size requirements of embedded systems in mind. According to the research from the scholarly article of Mr. Strydis [8] that has conducted a research on which he presents very promising results for the Lempel–Ziv–Oberhumer (LZO) and more specifically for the mLZO, which is a portable, lightweight subset of LZO library, suitable for implantable devices such as IMDs. More specifically after the comparison of general purpose compression algorithms, compressing ECG workloads it can seen from Figure 2.1 that mLZO has performed extremely well in the fields of total energy consumption, peak energy consumed and fair enough at compression ratio and rate.

compression ratio	compression rate	power	power2	size	total energy
lzari_oku	bclrle	mlzo	lzss	fin	mlzo
lzhuf_oku	slzw	arith	lzw15v	splay	bclrle
mlzo	mlzo	arith1e	fin	urban	bcllz
lzss	fin	arith1	mlzo	lzw12	slzw
fin	lzw12	urban	slzw	slzw	fin

Figure 2.1. Best-performing compression algorithms in descending order for 10-KB ECG Data

Especially in the field of total energy as for the ECG workload it is illustrated from Figure 2.2 [8] that mLZO outperforms all the other compression algorithms as regards the average energy consumption.

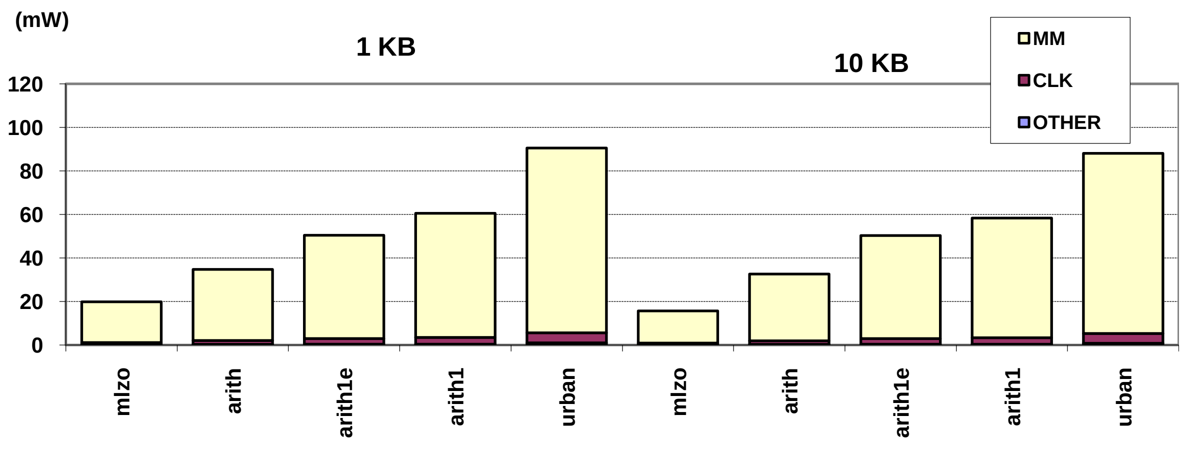


Figure 2.2 Average consumption for 1-KB and 10-KB ECG Dataset

From all the aforementioned it can be seen that LZO is the most appropriate solution for our low overhead and energy dependable devices, as IMDs.

2.3. DWT transformation algorithm –SPIHT encoding scheme

Discrete Wavelet Transform(DWT) is a linear fast(computationally) algorithm of any wavelet transform, in which the wavelets are discretely sampled. Wavelet, is a wave-like oscillation signal that has a short duration and has as a beginning and end the zero value.
DWT is a tool that separates data into different frequency components, with the use of low and high frequency filters. For each created component, a new high and low filtering is imposed followed by a factor 2 sub-sampling on the original signal and each component is studied with resolution matched to its scale.2-D DWT is computed the same way with the difference that firstly the algorithm is imposed for all rows and then for all columns. [9]

As for Set partitioning in hierarchical trees (SPIHT),SPIHT is one of the “state of the art” wavelet-based coding techniques, which exploits the inherent similarities across the subbands in a wavelet decomposition signal. SPIHT is basically transforms the real represented number of wavelet coefficients to a bit-rate. Some of the main characteristics of SPIHT is the following:

- It is progressively sending the wavelet coefficients starting from the low frequency ones, which contain and the most information.
- It does not need any training.
- The reconstruction is easily made, due to the fact that the most important (low frequency) coefficients are firstly sent.
- It can become from lossy to lossless, depending the threshold that we impose. As the more bits we are adding at the end of the bit-rate the better representation we get of the real number we want to compress.
- The “compressed” bit-rate can be terminated at any point, allowing any specific requirements of distortion or quantity parameters to be met.

It is worth-mentioning that, the energy of a wavelet-transformed signal is centered on the low frequency coefficients and coefficients are hierarchical ordered and has a parent-child relationship. This attribute of the wavelet-transformed signal uses SPIHT in order to save many bits from representing insignificant coefficients, as examining a parent coefficient we can determine whether or not the sub-band that this coefficient is related to worth any further representation in our “compressed” bit-rate.
We are presenting a brief SPIHT algorithm as it is described from S.Isa at [10]:

1) Initialization: Set the list of significant points (LSP) as empty. Set the roots of similarity trees in the list of insignificant points (LIP) and the list of the insignificant sets (LIS). Set the threshold$T_o = 2^n$, with $n = log_2(\text{max}|c(i,j)|)$ where c (i,j) denotes the coefficient at position (i,j).
2) Sorting pass in LIP: Each coefficient in the LIP is checked and the significant coefficients are moved to the LSP. The sign bits of the significant coefficients are encoded.
3) Sorting pass in LIS: If an entry in the LIS is significant one is sent and then its two offspring are checked like an entry in the LIP. If an entry in the LIS is insignificant, a zero is sent.
4) Refinement pass: Each old entry of LSP is checked. If it is significant under current threshold, a one is sent and its magnitude reduced by the current threshold. If it is insignificant, a zero is sent.

2.4. Related work

In this subchapter we present numerous previously proposed data compression algorithms all the methods and algorithms. In order this presentation to be fair and useful we would present these algorithms in 2 sections. The lossless algorithms and the lossy ones, according to the definitions that we have provided earlier in the previous subchapter.

2.4.1. Lossless algorithms

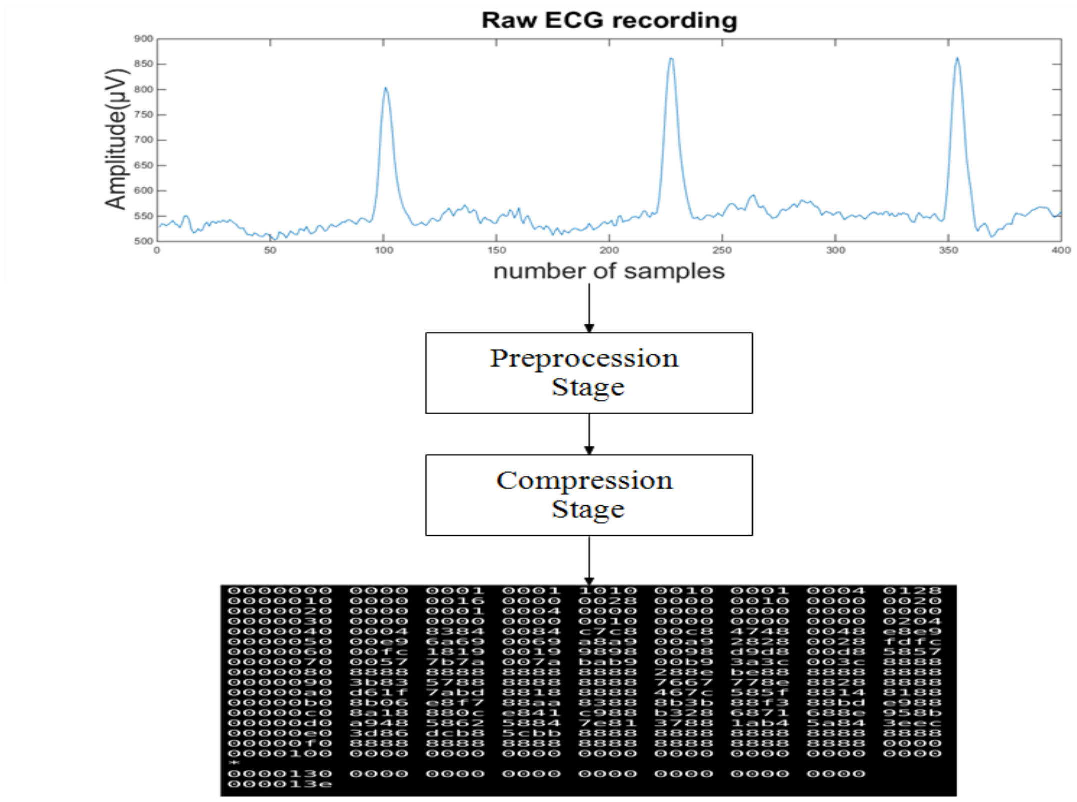
In this section we will mention the most renown and up-to-date Lossless algorithms that are used from the academic community for data compression.
At [11] Arnavut proposes a new technique that makes use of Based on Burrows-Wheeler Transformation, a block-sorting ,lossless data compression algorithm and also he uses Inversion Ranks of Linear Prediction. This technique manages to achieve better than the renowned bzip2 or gzip and also BWT with MTF instead of Inversion Ranks. It manages to achieve a percentage of 4.1 regarding the compression ratio percentage. We should mention that in Arnavut's algorithm there is no claim on the computational overhead of the algorithm. Duda at [12] proposes a lossless algorithm with Lifting Wavelet Transform. The new algorithm for lossless ECG compression is based on integers to integers lifting wavelet transform, thus quantization of wavelet coefficients (which normally cause the information lost) is avoided. Wavelet coefficients are entropy coded. In order to reduce the number of the symbols to be coded integer numbers is represented using the MS-VLI algorithm. This algorithms achieves about 2.77 of compression ratio. Also in this algorithm there is no mention on computational overhead. Koski at [13] proposes a new approach based on structural recognition and extraction of ECG complexes. She examines LZ77-Huffman encoding algorithm gamma encoding algorithm and complex-Huffman encoding algorithm. Among the aforementioned the LZ77-Huffman scores better with approximately 3.3 compression ratio.

2.4.2. Lossy algorithms

In this section we will mention the most renown and up-to-date Lossy algorithms that are used from the academic community. Among them will also introduce cutting edge technologies and also algorithms that will help us to understand better the current research. Firstly we state the Batista's algorithm [14].He proposes an ECG compressor based on optimized quantization of Discrete Cosine Transform (DCT) coefficients. The ECG to be compressed is partitioned in blocks of fixed size, and each DCT block is quantized using a quantization vector and a threshold vector that are specifically defined for each signal. The evaluation of this algorithm is based on compression ratio and PRD.As it is stated it manages to achieve an average CR of 9.3 for PRD equal to 2.5%. The energy efficiency neither of this algorithm is presented. Previously Hilton has presented his algorithm at [15].This algorithm is based on embedded zero-tree wavelet (EZW) coding. The proposed algorithm is used for compression of Holter ECG data He uses different wavelet packets in order to examine the efficiency of each one and he presents their results. He manages to achieve CR from 8 until 16, but with minimal clinical use, at the reconstruction of the ECG signal. Finally, we present a very similar research study to ours. Mr. Koyrakh at [16] proposes an algorithm which is orientated to Implantable Medical Devices and tries to compress in a lossy way ECG data. It is a very lightweight algorithm which can easily be implemented in any IMD. The proposed algorithm processes the data in the following way. Firstly, it transforms the coefficients with wavelet transformation, afterwards it changes the representation of the transformed coefficients and then the Quantization of the signal is imposed, which comes along with a threshold application. The quantization is followed by a Run-length encoding scheme in order to encode the quantized coefficients and finally there is an adaptive bit encoding algorithm. With PRD kept under 8%, the compression ratios, defined as ratios of total numbers of bits in the original and compressed waveforms, were 9.3 ± 2.5, consistently exceeding 85% of the theoretical limit determined by the bit entropy of the original data frames. Nevertheless, Koyrakh's algorithm does not present the computational overhead of the previous presented results. Therefore , it would be naïve to compare it with our research.

3. Implementation

This chapter will describe the Implementation process. Our implementation process starts from the ECG recordings. The preprocession stage follows in order to form a signal that its main attributes could easily be recognizable and detectable. Also preprocession aims to remove any noise that has distort the original and useful signal. After the preprocession stage, when the signal in as real as it could be it comes the compression stage where using different methods and algorithms like the LZO and SPIHT that we would use in our thesis. After the compression as it can be seen from Figure 3.1 the outcome would be a binary file which would contain the information of ECG signal in a smaller more “digitalized” size.



```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Figure 3.1 Implementation process

Our main goal is to manage low overhead compression on ECG recordings. In order to do so, we study existing and create new methods that can help us to that direction. Our main endeavor is to compare 2 compression algorithms. Beat reordering with SPIHT [10] and a low overhead encoding algorithm suitable for IMDs, LZO compression algorithm [8].In order to deal with that we have implemented different algorithms and we examine different settings of each algorithm, so as to find those settings and algorithms that are more suitable to our goal and purpose. In this Chapter we analyze the process of our work the purpose of each step and all the parts that it contains.

We start from our ECG recordings. For the purposes of our work we have made use of the MIT-BIH arrhythmia database. From the aforementioned database we used a default Lead (Lead I), due to the fact that we are interested in IMDs more than one lead would be useless for our research. As an IMD could not normally record the heart function of more than one place, so in order to produce a fair simulation to that environment we are using just one lead.

Secondly, from Figure 3.2 we can make out that in the raw ECG there is a great quantity of noise. This type of noise in the ECGs is called baseline wander. Prominent causes of that noise are patient breathing, body movement and also noise produced from the electrodes. The removal of the baseline wander is of great importance so as to make out better the characteristics of ECG [17]. Hence, we are using a method that uses a wavelet-base method in order to eliminate the baseline wander. According to [10] the spectrum of the baseline is below the spectrum of ECG signal, therefore through inverse wavelet transform of approximation coefficients we can estimate and remove the baseline wander.

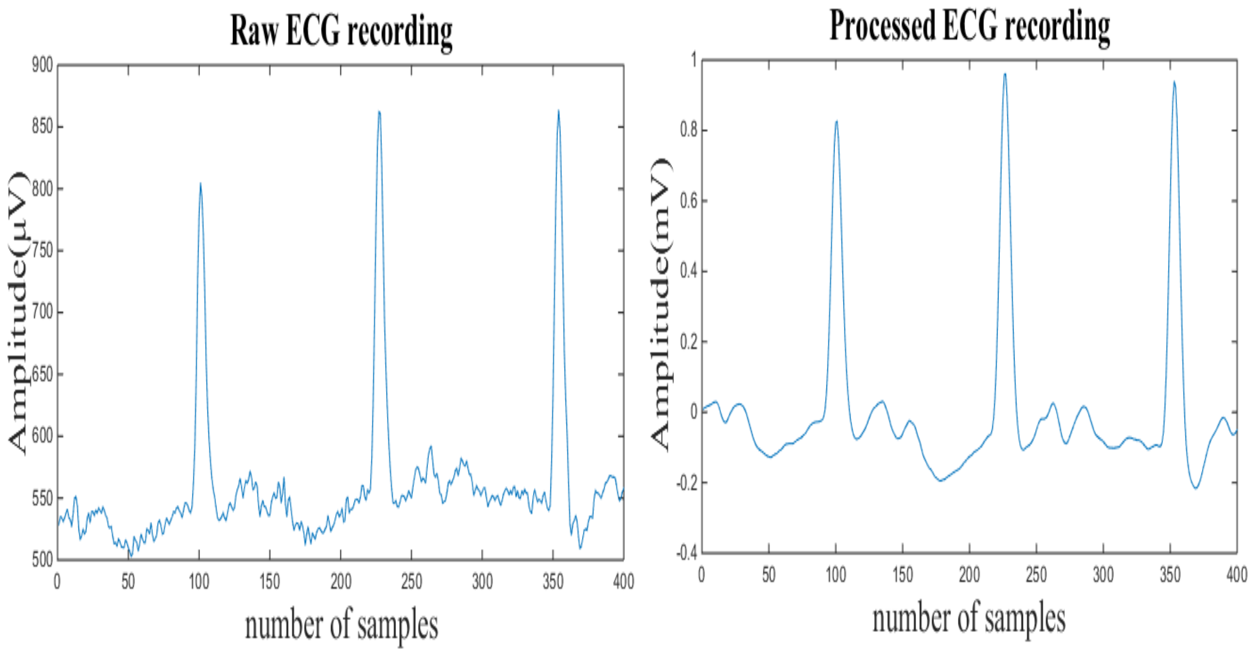


Figure 3.2 The ECG signal before and after the preprocession

After preprocession of the ECG signal, it follows the compression stage of the ECG. This section is basically the main subject of our research and also the topic of the whole Implementation chapter. In the process we examine separately different kinds of encoding and data procession algorithms. In this thesis we are mainly evaluate two encoding algorithms LZO encoding algorithm and also SPIHT encoding algorithm which follows DWT transformation. We have chosen these algorithms among the numerous compression algorithms that are released, because LZO algorithm and more

specifically the miniLZO algorithm as it stated in [8] it is ideal for embedded applications as the IMDs. As it is stated above IMDs are our main field of interest .On the other side the DWT-SPIHT compression algorithm can achieve very big compression ratios, which is and the main goal of every compression algorithm.
Furthermore, we present at this chapter and various data procession algorithms in order to examine the behavior of each algorithm.

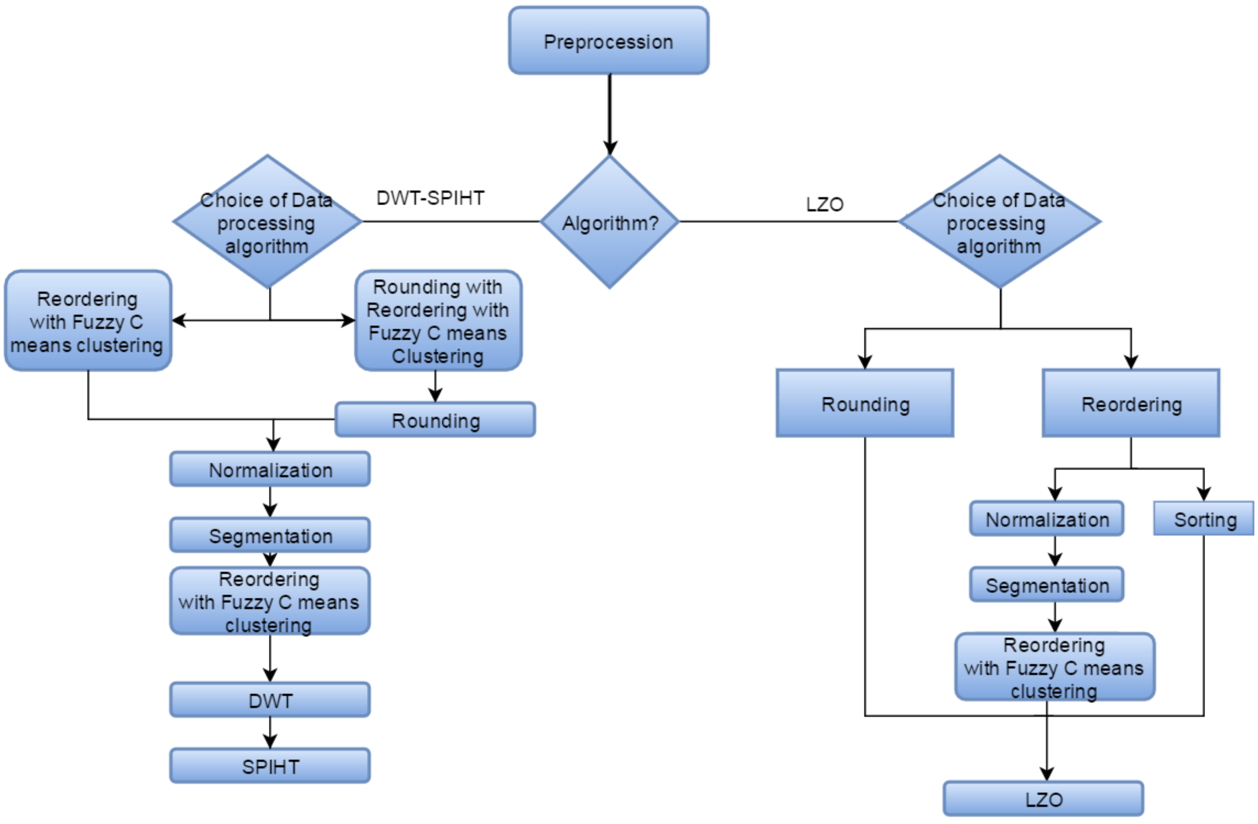

Preprocession
Algorithm?
DWT-SPIHT
LZO
Choice of Data processing algorithm
Reordering with Fuzzy C means clustering
Rounding with Reordering with Fuzzy C means Clustering
Rounding
Normalization
Segmentation
Reordering with Fuzzy C means clustering
DWT
SPIHT
Choice of Data processing algorithm
Rounding
Reordering
Normalization
Sorting
Segmentation
Reordering with Fuzzy C means clustering
LZO

Figure 3.3. Evaluation flowchart

3.1. *DWT-SPIHT data processing techniques*

To begin with, we should define the wavelet basis that we have chosen. As it is stated [10] the basis with the most efficient PRD is bior6.8 so it was chosen for the entire experiment.
Moreover, the experiments was made with 256 number of samples at is also stated in [10] that produces the best PRD results with the same compression

ratio. In order to verify this statement we have also run the experiments with 128 sample number.
Each tree node is calculated by the following equation

$$O(i, j) = \{C(2i, 2j), C(2i, 2j+1), C(2i+1, 2j), C(2i+1, 2j+1)\}$$

So it is more efficient when each parent 4 children and not less. Hence, the sample number in an SPIHT algorithm in order to work efficiently should be derivative of power of 2, as the SPIHT algorithm is developed in dyadic tree pyramid.
The dimensions of the matrix are defined from the sample number. That's why we have not chosen for example 255 or 257 number of samples.
Before the implementation of the DWT transformation and the SPIHT encoding algorithm we should process the input data in a specific way in order to achieve efficiency. For this reason in this section we present the reordering technique and the rounding before reordering technique. The first pre-process method reorders each beat so as to create a signal with less high frequency. We want a signal with less high frequency in order to be better encoded from SPIHT algorithm. The second algorithm rounds the coefficients before the reordering with the purpose of creating more correlation between the coefficients and with the aim to make the reordering easier, as after rounding there would be bigger similarities among coefficients.
The reordering technique consists of some steps. First step is the normalization of the ECG recording, the second step is the segmentation of the 2D ECG that has been created from the normalization to frames and after that the reordering of each frame separately. We would analyze each step explicitly in the process.
In an ECG signal we can find out 2 types of correlation.

1. Correlation in a single ECG cycle (intrabeat correlation).
2. Correlation among ECG cycles (interbeat correlation).

Creating a 2D ECG array where each row would depict a heartbeat and each column a specific part of the ECG (for example column 1 for every row depicts the highest point of R wave) should help to the decorrelation of ECG signal.

<u>Normalization</u>

As all the coefficients would be aligned and would share similar attributes. Therefore, since each heartbeat can have a different duration, it should be normalized into constant number in order to construct 2D ECG array. For our normalization process we used the PAN method [18]to normalize each heartbeat duration without amplitude normalization step. As it is shown from the Figure 3.4 there is no significant difference neither in morphology nor in amplitude of the unnormalized and normalized ECG signal. During the upscaling of the signal.

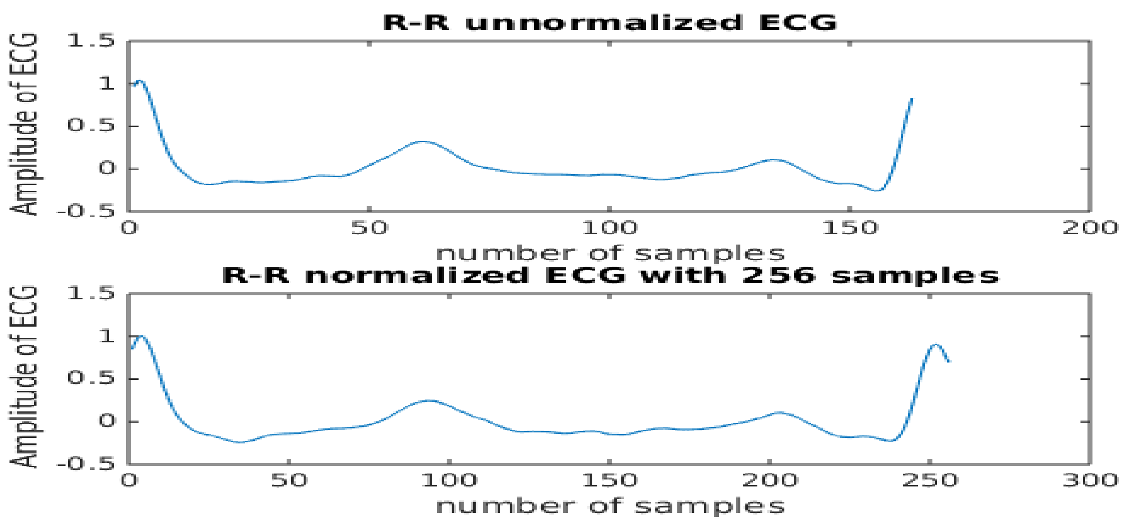


Figure 3.4.Example of Normalized R-R heart beat with 256 number of samples

Segmentation

We are segmenting the 2D ECG array in frames.Each frame is a square matrix with columns and rows equals to the sample number that we have chosen for our normalization.Hence,the 2D square matrix that is produced has in its rows the intrabeat coefficients and each column the coefficients of each normalized ECG heart beat.

Reordering

In our research we are using a beat reordering technique to optimize SPIHT coding for ECG signal compression. Beat reordering rearranges beat order in 2D ECG array based on similarity among adjacent beats. The rearrangement reduces variances among adjacent beats so that the 2D ECG array contains less high frequency. SPIHT coding work more efficiently on the signal with less high frequency component.Working in that way, we are using fuzzy c-means clustering as beat reordering technique to optimize SPIHT coding by rearranging beat or ECG cycle order in 2D ECG array, according to their similarities.We have chosen fuzzy c-means clustering algorithm for the proposed reordering ,because fuzzy c-means clustering not only cluster the beats but also provides the each beat with a degree of their belonging to each cluster [19]. According to that degree, we rearrange the beats inside each cluster,since the frequency distribution is only affected by the order of beats inside each cluster.
We can see the difference between an ECG signal before and after beat reordering in Figure 3.5 [10] .

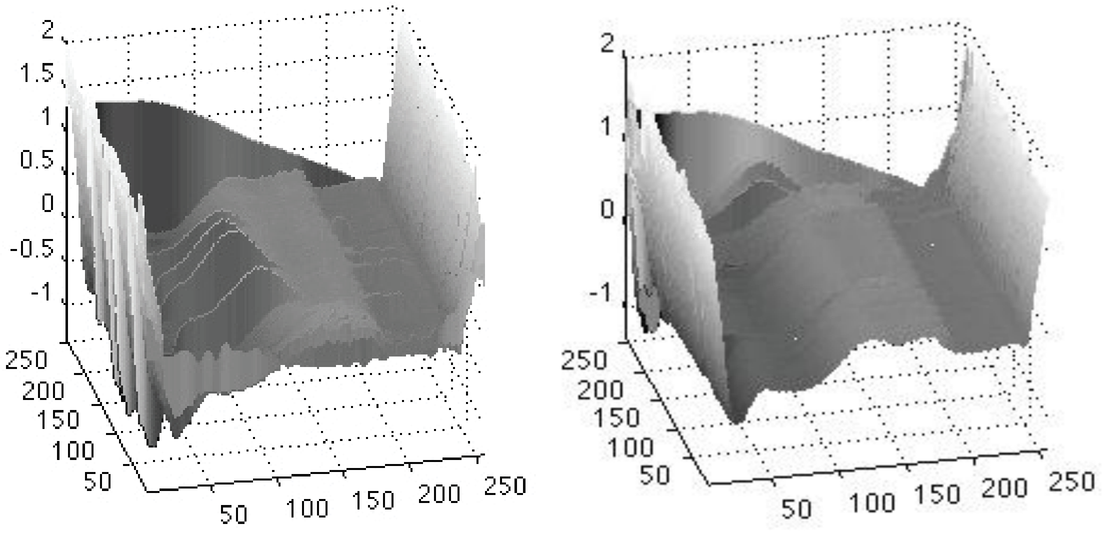


Figure 3.5. 2D ECG array of ECG before(right) and after(left) beat reordering(x-axis depicts number of samples per heart beat, z-axis depicts the number of the heart beats and y-axis the amplitude of the each ECG sample) [10]

3.1.1. Rounding with Reordering with fuzzy C-means clustering technique

This technique is exactly the same as Reordering technique with the difference that before the normalization step the coefficients of the ECG are rounded. Figure 3.3 explains explicitly the steps that we followed for that technique

3.2. LZO Data processing techniques

As we have already mentioned some general attributes of LZO in the Background chapter, LZO is a lossless data compression algorithm originally written in ANSI C. It is a block compression algorithm. It compresses and decompresses blocks of data. Block size must be the same for compression and decompression. LZO compresses a block of data into *matches* (a sliding dictionary) and *runs* of non-matching literals to produce good results on highly redundant data. In this section we will state the methods that we use before the encoding process. Some of the methods mentioned here have already been explained in the previous chapter.
Moreover, LZO is a lossless compressor opposed to SPIHT which in our experiments is lossy. So some of the methods proposed below use some

preprocessing algorithms like the rounding in order to create lossy algorithms with the use of LZO compressor.

3.2.1. Reordering

In our research we are using 2 types of Reordering. We reorder the coefficients sorting them ascending and reordering them with fuzzy C-means clustering.

Sorting

We are sorting in ascending order the coefficients of the ECG recording in order to create correlations as the LZO is a block compressor that uses sliding dictionary. So, sorting them the adjacent coefficients would appear similarities. The LZO compressor could take advantage of these similarities and produce better compression ratio.

Reordering with fuzzy C-means clustering

This technique is exactly the same as the technique that we have described in the previous section ,when we described the Fuzzy C means cluster algorithm with the difference that now at the end we are not encoding the coefficients with DWT-SPIHT encoding scheme but with the LZO.

3.2.2. Rounding

As in reordering we are proposing 2 implementation schemes. We can combine rounding with reordering or just use reordering as it is depicted in Figure 3.3.

Rounding

In order to round the data we are using fixed-point values.
The aforementioned technique has the following steps:

- We take the initial values
- We are creating diffirent fixed point values according to the number of the digits that we want to give to the fraction number of the values.

So,we round the values using 2-5 bits for the fraction part.

e.g:For 2 bits we are rounding the values to
1)XXX.750
2)XXX.500
3)XXX.250
4)XXX.000
For the other bits is done by similar way.

- We continue with the encoding algorithm.

3.3. *Tool Flow*

This section provides a step-by-step guide to what tools, scripts and files are required and used for the implementation and evlauation of our work. The tool flow used throughout the project is illustrated in Figure 3.6 .
Starting from the top, we are loading in Matlab environment the MIT-BIH arrhythmia database, which was saved locally to our PC in Matlab data files (.mat).Then, we are executing in Matlab all the needed preprocession functions, such as selection of the lead that we want to use or the removal of baseline wander with. After the preprocession is over we saved the ECG recordings in binary.
Following that we are going under the compression stage. All the procession that is following that stage(see Figure 3.3) is done in Matlab until we decide which encoding algorithm we use. Whether we are using DWT-SPIHT encoding or LZO encoding would bring us in different paths.
If we continue with LZO after any data procession algorithm is imposed we call through Matlab a python script. This python script executes the LZO library, which is written in C and produce the compressed files.
Otherwise, if we choose to continue our compression algorithm using DWT-SPIHT, we continue in Matlab as the DWT and SPIHT are Matlab functions and after the encoding is over we are saving the compressed ECG signal, like we have done with LZO so as later to evaluate the methods from quantitive scope.

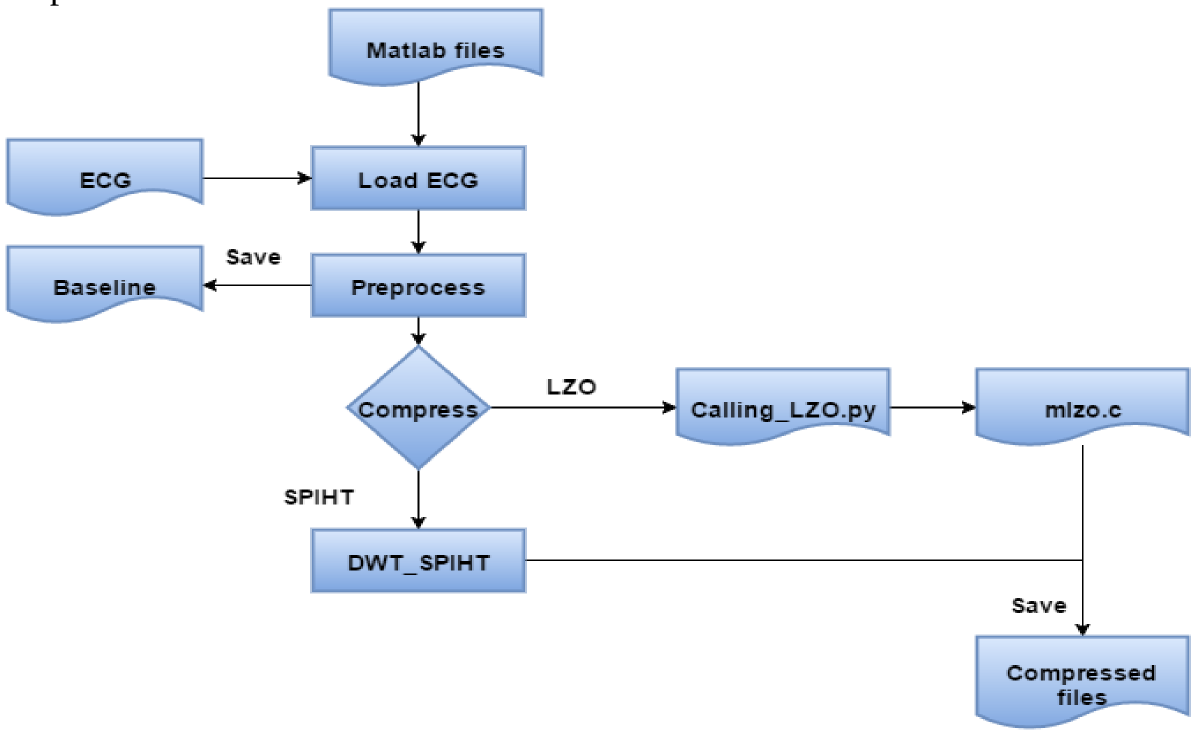


Figure 3.6 Tool flow figure

4. Evaluation

At this chapter of this thesis we would evaluate the proposed methods that we mentioned in the aforementioned chapter, so as to compare our proposed compression techniques, which is and the main goal of our research. Hence, we have to declare firstly explicitly the evaluation criteria and the tools that we used in order to compare the compression techniques and afterwards to present the evaluation results for each of the proposed algorithms.

4.1. Experimental Setup

It is very important from the beginning to point out clearly the criteria and the tools that we used in our evaluation process. First of all, we would explain the metrics that we would use in order to evaluate the algorithms and then we would explain the tools that we used to acquire these metrics and all the needed assumption that we did.

4.1.1. Metrics

At this section we describe the metrics which we use to decide which algorithm or combination of algorithms is the most suitable for IMDs.

4.1.1.1. Compression ratio

When we are referred to compression algorithms the compression ratio is the most renowned metric that we use to evaluate them.
CR depicts how much space we save when we impose a data compression algorithm. The compression ratio (CR) is defined by the following equation [20]:

$$CR = \frac{SIze\ of\ the\ original\ file}{Size\ of\ compressed\ file}$$

In case that we use a reordering technique has as a consequence the production of indexes files. So CR equation is transformed to:

$$\mathrm{CR} = \frac{\text{Size of the original file}}{(\text{Size of compressed file} + \text{Size of the indexes})}$$

4.1.1.2. File size

Compression ratio is not enough in order to evaluate the results from a quantitive prospect, so we are saving also the size of each file. The process followed to acquire these information is cited above.
For compression ratio and the file size we just calculated the size of each file, we didn’t use any complex tools for acquiring these information

4.1.1.3. PRD

The Percent **R**oot mean square **D**ifference [10] is used frequently in academic literature in order to examine the quality of a compression technique, especially for compression of biological signals.
The PRD is defined by the following equation:

$$PRD = \sqrt{\frac{\sum_{i=1}^{n}[x_1(i) - x_2(i)]}{\sum_{i=1}^{n} x_1^2(i)} \times 100}$$

,where x_1 is the coefficient of the original signal and x_2 is the coefficient of the reconstructed signal.
In order to calculate the PRD we use a Matlab function which calculate the PRD based on the equation we provide above. In order to calculate the PRD of each algorithm we called the PRD function of Matlab after the decompression of each compressed file. The input values are the original file and the reconstructed file.

4.1.1.4. Computational Metrics

It is very important for our research the energy footprint of each algorithm. These metrics would help us come to a conclusion whether algorithms copy with the requirements of the Implantable Medical Devices in order to compress ECG recordings in an energy efficient way. More specifically we present the total time that each algorithm needs, the peak memory and also the total amount of memory it uses. As for the calculation of computational metrics as it was more complex to acquire them we used different techniques.
For the algorithms that are affiliated to LZO we used a combination of Profile tool of Matalab and Massif Heap profiler from Valgrind framework.
More specifically, the preprocession of LZO algorithms as we have mentioned in the implementation chapter is done in Matlab but the compressor itself is in C, so we should find a way to aggregate the energy footprint of compressor

itself to the energy footprint of preprocession algorithm. So, we followed the steps below:

1. Firstly, we measured with the help of Massif Heap profiler the execution time and the memory footprint of SPIHT compressor in C and LZO compressor in C.
2. Afterwards, we measure the total execution time and memory footprint of SPIHT in Matlab.
3. From the execution time and memory footprint of SPIHT in C and in Matlab we extracted a scaling factor of the algorithm between Matlab and C.
4. Finally we multiplied with this scaling factor the computational metrics of LZO in C in order to find in a theoretical way the computational metrics of LZO in Matlab and aggregated them with the computational metrics of preprocession in order to find the final values of computational metrics.

4.2. Experimental Results

We would progressively quote the results of all the proposed algorithms. It is important at this point to clarify our baseline, namely the input we are using for our experiments. The input file size is 370080 bytes.

4.2.1. Results of SPIHT algorithms

We should mention that on SPIHT algorithms the compression ratio and the file size do not change over the methods as we define the size of the compressed file in the beginning of the algorithm. Before the beginning of the algorithm we define the maximum amount of bits we want to grant to our compressed file. As we have already in Background chapter SPIHT algorithm does progressive compression, so we have the ability to define the size of our compressed files.
We have chosen 10 different compression ratios(2.67, 5.33, 8, 10.67, 13.33, 16, 18.67, 21.33, 24, 26.67) in order to examine the differences in PRD and computational overheads for these specific compression ratios for various values of samples and clusters. Those compression ratios have been chosen and calculated randomly according to the range of compression ratios that we have encountered in our related work.

4.2.1.1. Results on Reordering with fuzzy C means clustering

Firstly we can see the differences in PRD when we impose the reordering with fuzzy C means clustering.

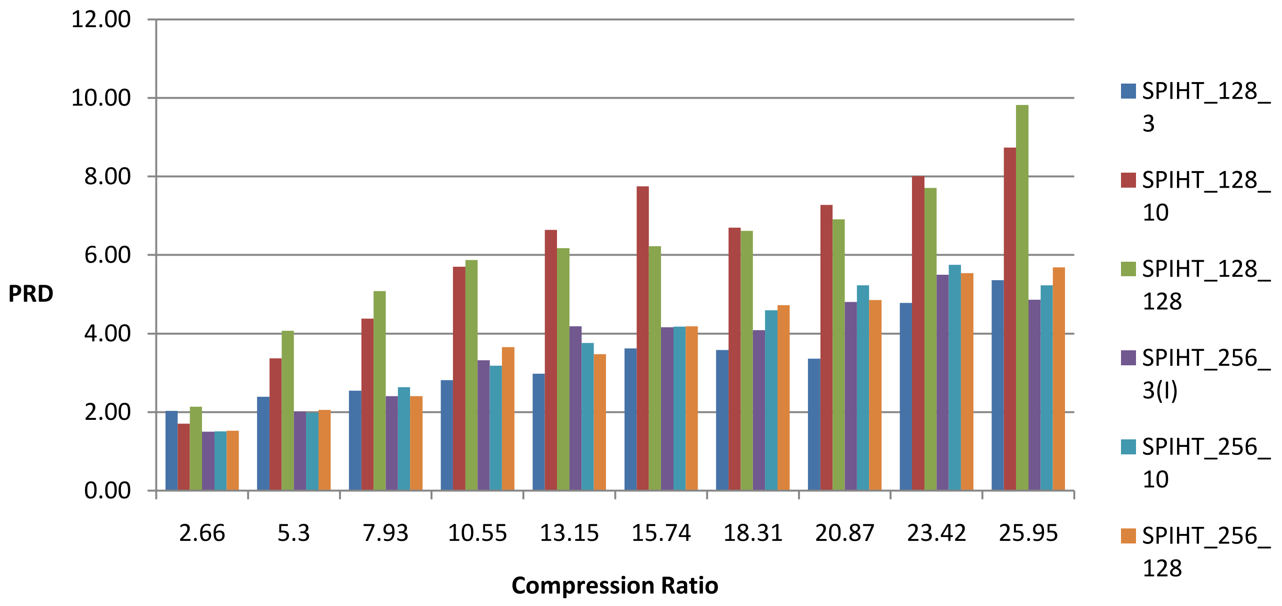


Figure 4.1 PRD over compression ratio for SPIHT Reordering with fuzzy C-means clustering

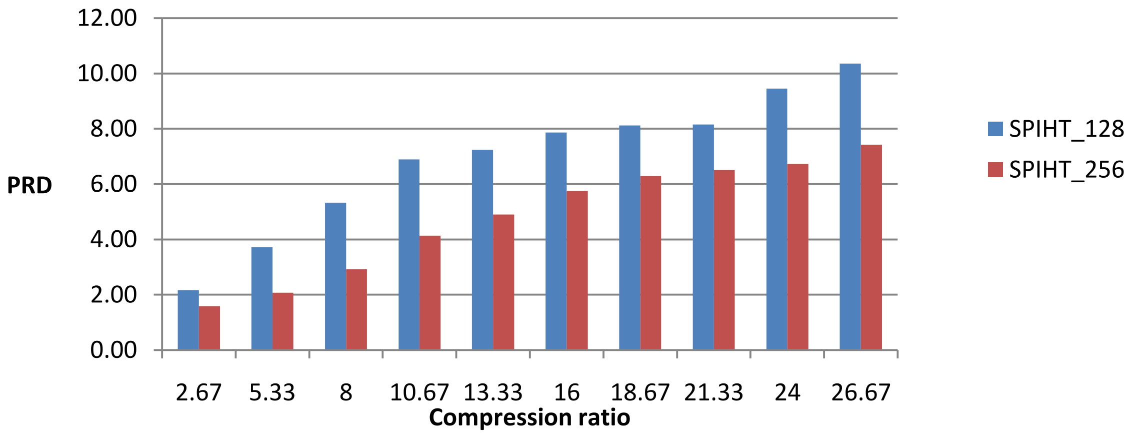


Figure 4.2 PRD over compression ratio for SPIHT without Reordering with fuzzy C-means clustering

Before start explaining each figure separately we should clarify the fact that compression ratio is different for the Figures with Reordering with fuzzy C-means clustering and without. The difference is outcome of the indexes that

are produced with Reordering with fuzzy C-means clustering, which worsen the compression ratio a little when the Reordering is used.
As it can be seen from the Figure 4.1 and Figure 4.2 we illustrate all the different values of PRD for different values of Compression Ratio. As it can be seen from the legend SPIHT is the method we used, first number the sample number, second number the number of clusters we used (if there is not second number we haven't imposed reordering).
The pattern of the PRD values is clear, the more clusters we use the bigger PRD values we get no matter the sample number. Moreover, when we use reordering the PRD values are smaller comparing to those that we have without reordering with fuzzy C-means ordering. Also, it can be seen that we agree experimentally with [10] as the best PRD values are encountered when we use 256 samples. Finally, it is more than obvious that the bigger compression ratio we get the bigger the PRD becomes. It is recognizable that in 15.76 compression ratios is bigger than 13.15 and nevertheless we get smaller PRD values. That is happening due to the fact that in the reordering with fuzzy C means clustering the initialization on the cluster centers in done randomly and this parameter influences the reordering and therefore the PRD values.
As much as computational overheads it concerns, from Figure 4.3 and Figure 4.4 we can see that the bigger the sample number is the most time it gets to compress the data and also the number of cluster influences the Execution time of an algorithm. As it needs more time for the clustering as the calculations are more in order to see each coefficient in which cluster it belongs. Surprisingly enough we can see that, the better compression ratio we acquire less time we need. This happens due to the fact that as we have mentioned SPIHT sends progressively the coefficients and the sooner we stop the compression process the better compression ratio we get ,but with the worst PRD. Moreover, we can see that the trend of Execution time is followed by Total Memory used and Peak Memory needed calculations too.

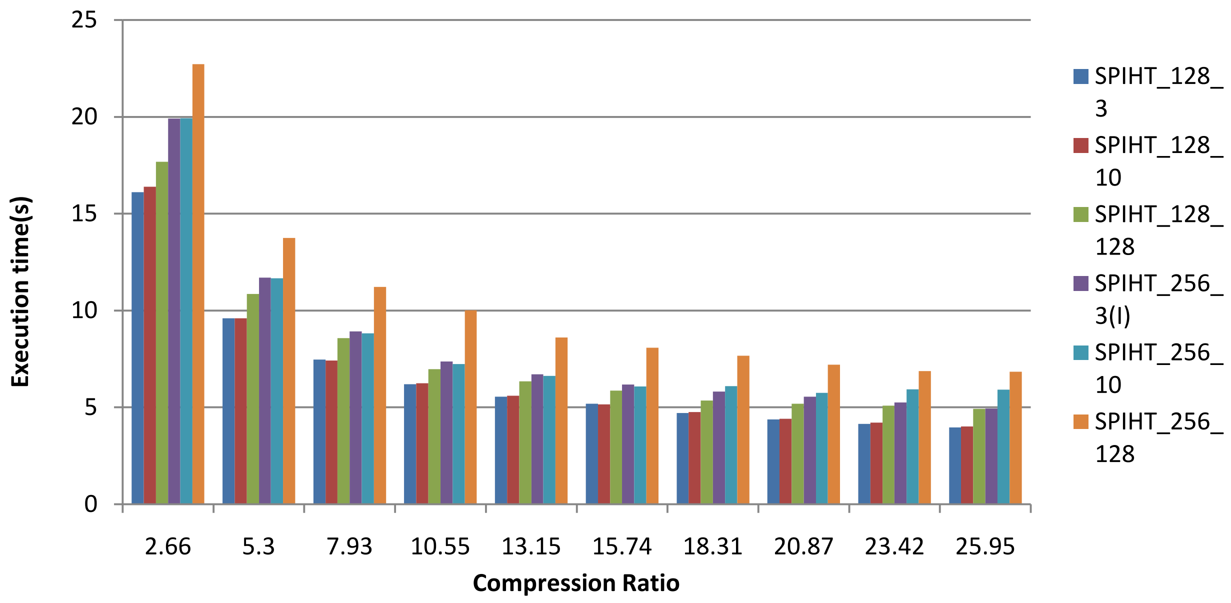


Figure 4.3 Execution time over Compression ratio for SPIHT Reordering with fuzzy C-means clustering.

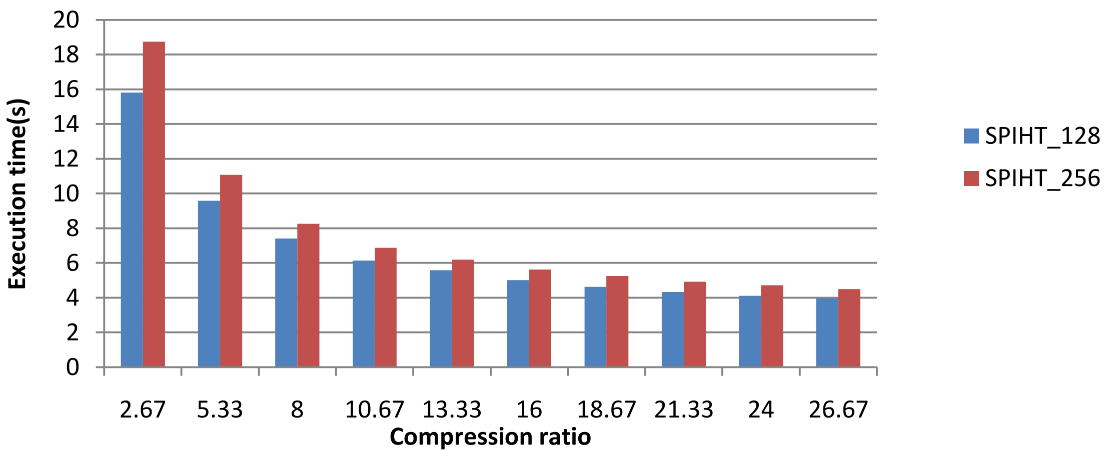


Figure 4.4 Execution time over Compression ratio for SPIHT without Reordering with fuzzy C-means clustering.

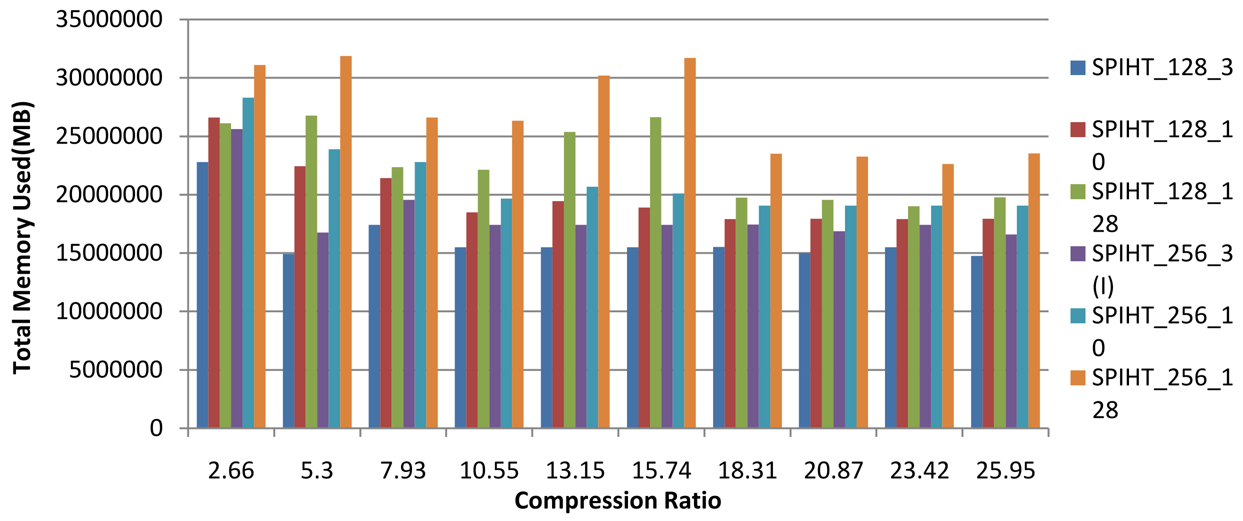


Figure 4.5 Total Memory Used over compression ratio for SPIHT Reordering with fuzzy C-means clustering

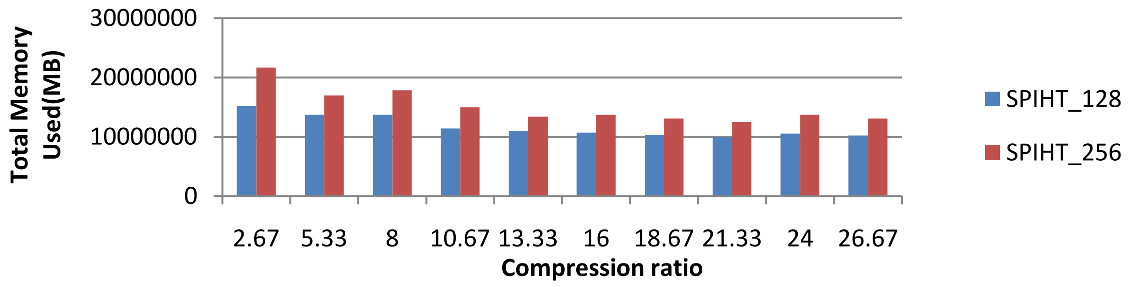


Figure 4.6 Total Memory Used over compression ratio for SPIHT Reordering without fuzzy C-means clustering

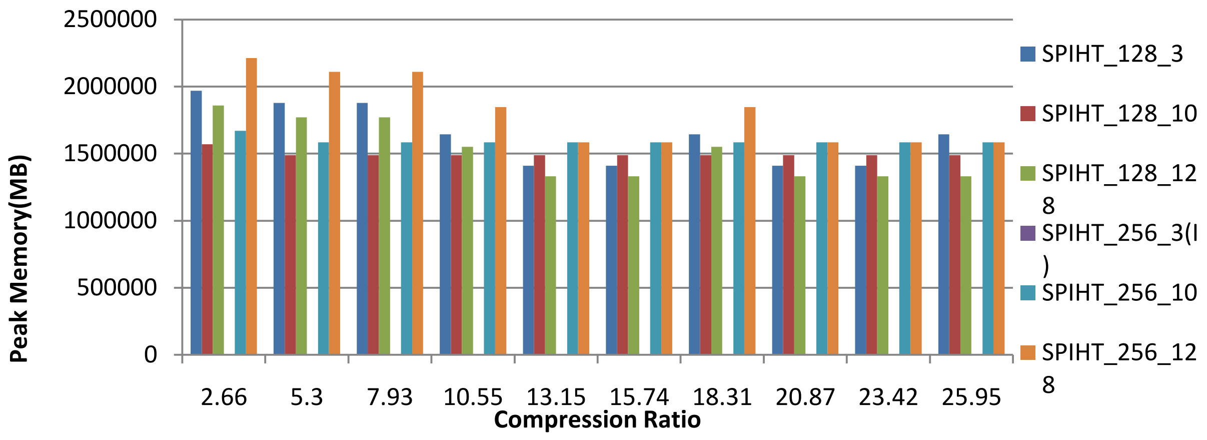


Figure 4.7 Peak Memory used over compression ratio for SPIHT Reordering with fuzzy C-means clustering

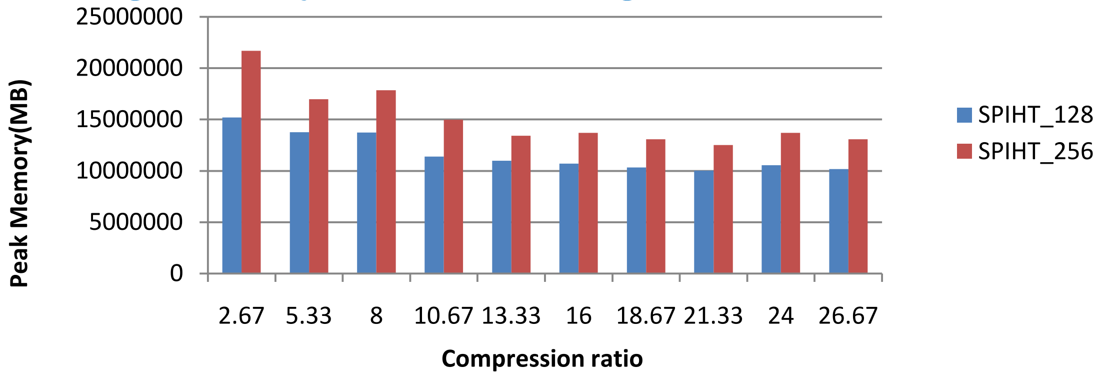


Figure 4.8 Peak Memory used over compression ratio for SPIHT Reordering without fuzzy C-means clustering

4.2.1.2. Results on Reordering with fuzzy C means clustering after Rounding.

Firstly in Figure 4.9 we can see that the rounding in the coefficients worsen the PRD is getting worst 7-10% worst, compared to the PRD results with reordering(Figure 4.1) with reordering with fuzzy C-means clustering, without prior rounding. Moreover, we observe an approximately 5% increase in the execution time(approximately 1 second more) in Figure 4.10(compared to Figure 4.3) , a 8% increase in total memory used from the algorithm in Figure 4.11(compared to Figure 4.5), peak memory used from the algorithm seems to be the same in Figure 4.12(compared to Figure 4.7) compared to the similar figures of SPIHT reordering algorithm with fuzzy C-means clustering. Finally, it is shown in the graphs that the rounding in the coefficients does not help our goal. With the method specified in this section the computational overheads stay more or less the same, compared with the results on results with reordering with fuzzy C-means clustering. Although, there is an increase on the computational overheads as the rounding demands more calculations which means more time and memory, the rounding makes the reordering easier. So it saves some time in that stage of the algorithm, but it does not over-exceed the total execution time. To sum it up, we gain no improvement but we consume more energy.

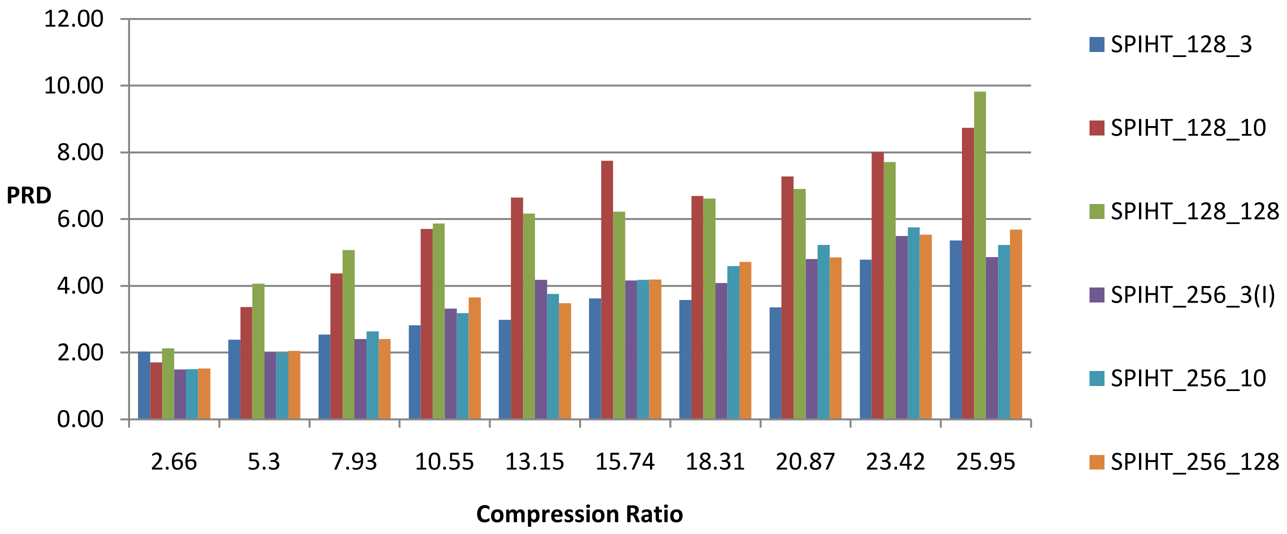


Figure 4.9 PRD over compression ratio for SPIHT Reordering with fuzzy C-means clustering after Rounding.

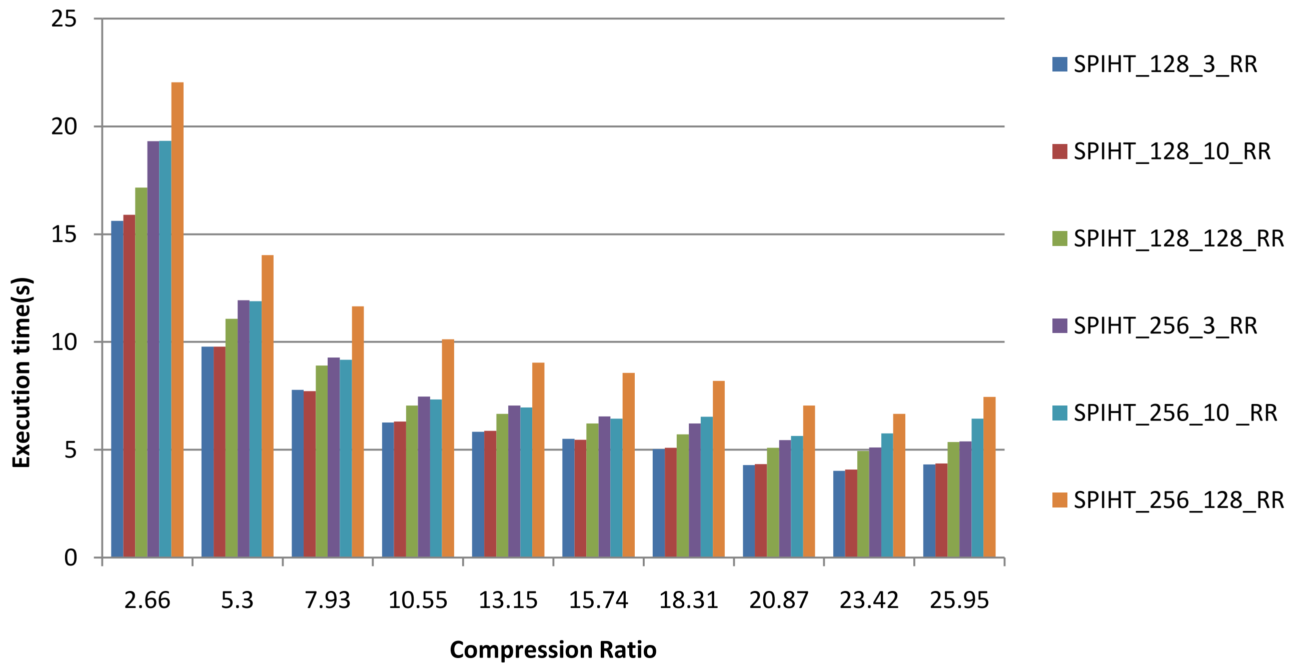


Figure 4.10 Execution time over compression ratio for SPIHT Reordering with fuzzy C-means clustering after Rounding

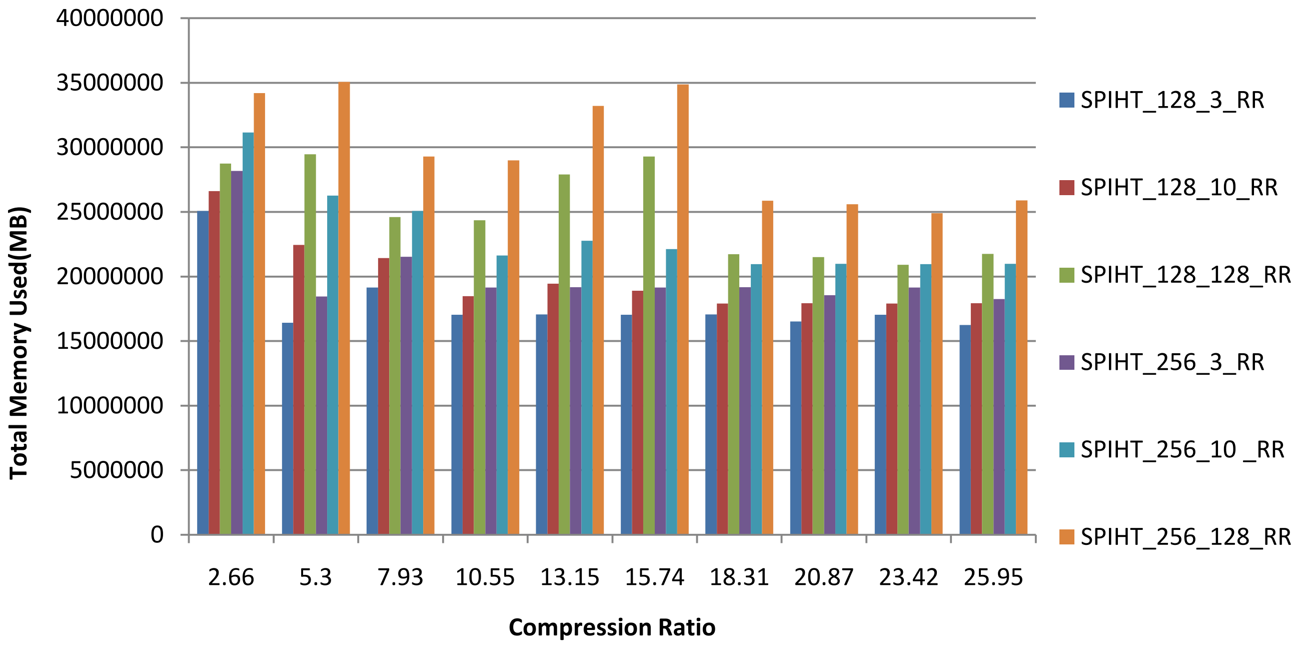


Figure 4.11 Total Memory Used over compression ratio for SPIHT Reordering with fuzzy C-means clustering after Rounding

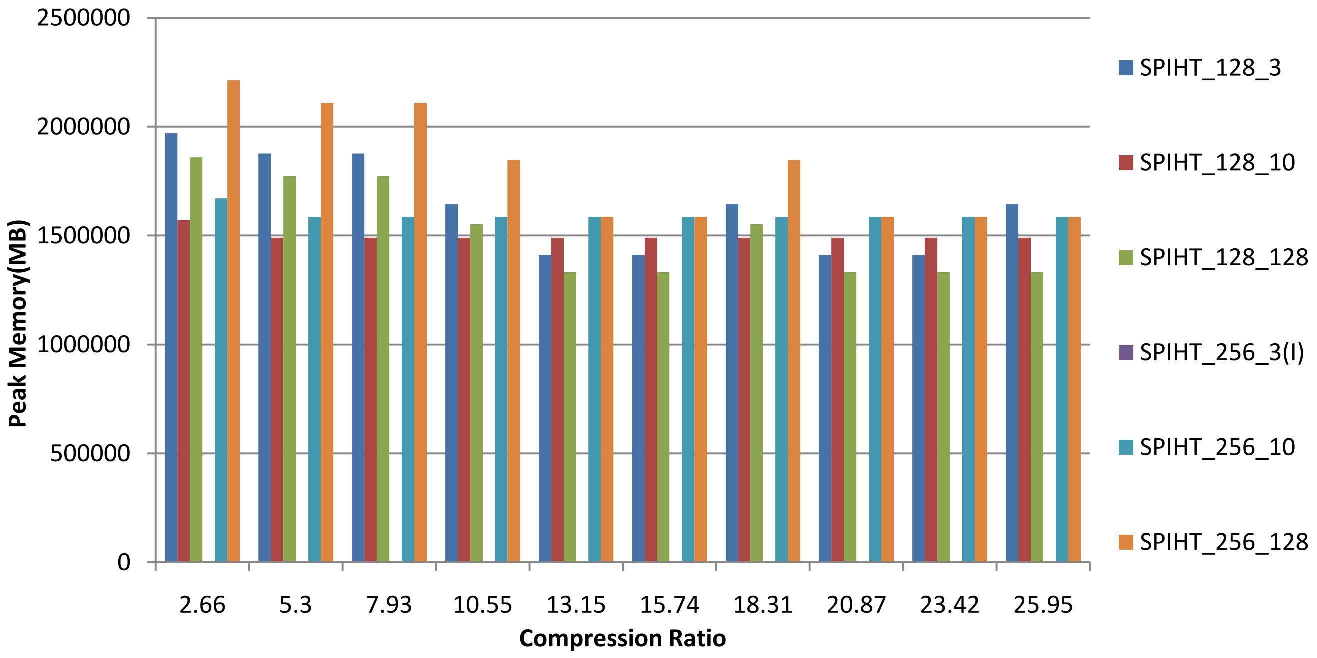


Figure 4.12 Peak Memory used over compression ratio for SPIHT Reordering with fuzzy C-means clustering after Rounding

4.2.2. Results of LZO algorithms

In this section we present the results from all the proposed algorithms that they use the LZO compressor. Moreover as our main concern is to examine which compressor and which preprocessing algorithm is more suitable for our goal, we have tried to compare them as equally as we could. Hence, we have tried to create a “lossy” LZO compressor.
LZO is a lossless compressor opposed to SPIHT which in our experiments is lossy. So we have used some preprocessing algorithms like the rounding in order to create lossy algorithms with the use of LZO compressor.
Moreover, as we have already mentioned in Implementation chapter, in order to use the LZO compressor we should firstly save the data and then call it in order to compress them.

4.2.2.1. Rounding

As we have mention on Reordering on Implementation chapter with the reordering we basically create a lossy algorithm, as we try to find an effective way where we could have adequate compression ratio and a decent PRD value. From the

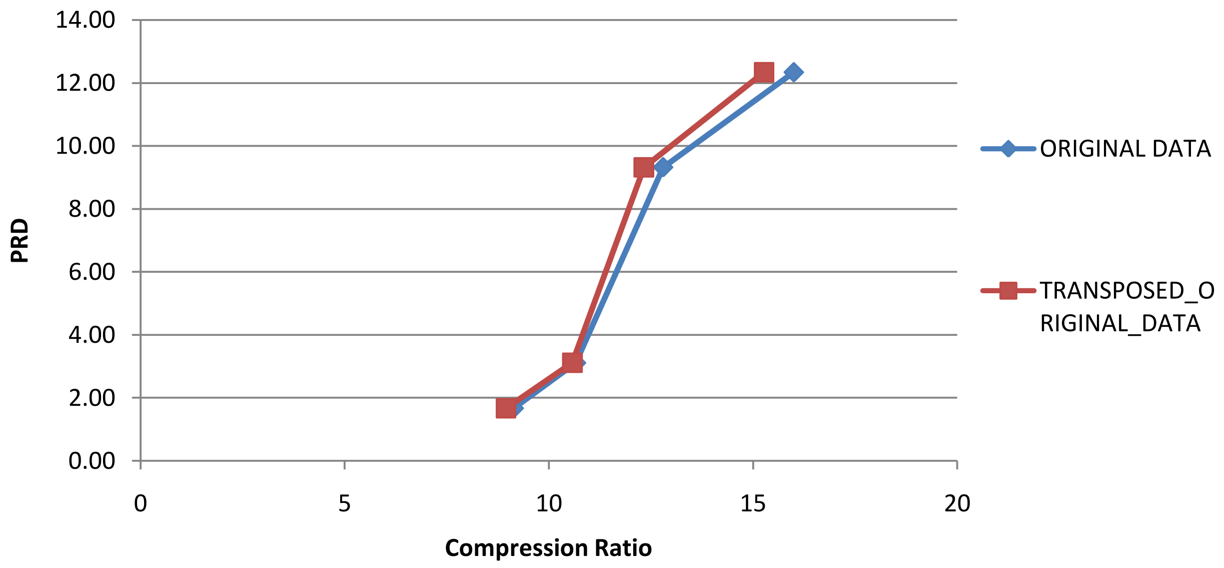


Figure 4.13 we can see that Rounding provides very good compression ratio and with an efficient PRD. More specifically, given 5 bits after the decimal point we manage to acquire compression ratio 8.95 and PRD 1.67 , a very promising trade off and also with 4 bits after the decimal point we get 10.57 compression ratio and 3.11 PRD The transposed data does not offer better

ratios from the original one. In the next Chapter we will compare more thoroughly the results of Rounding with the other methods.
From the computational overheads we can figure out that the more we round the coefficients the more time we consume, approximately 4 seconds per bit we grant for the decimal part. Moreover, we can see that the peak memory does not change over the bits significantly, whereas the total memory used from the 5 bits is two and a half times bigger from the one used from the 2 bits. Also, it can be seen from Figure 4.14 that the computational overheads have not any difference either for transposed or original data.

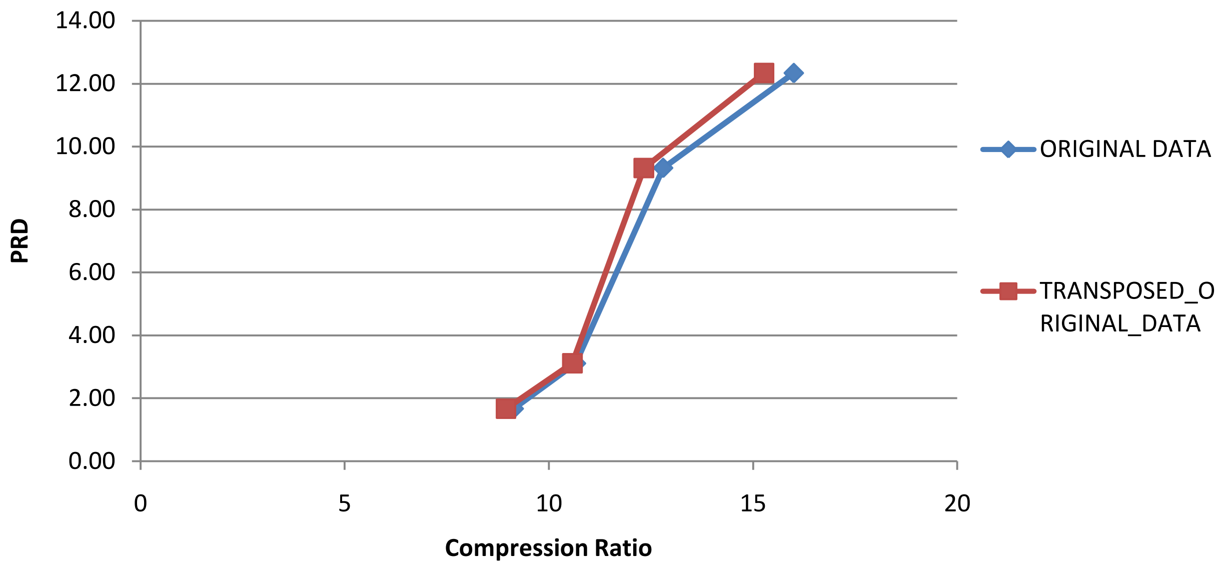


Figure 4.13 PRD over compression ratio for LZO Rounding

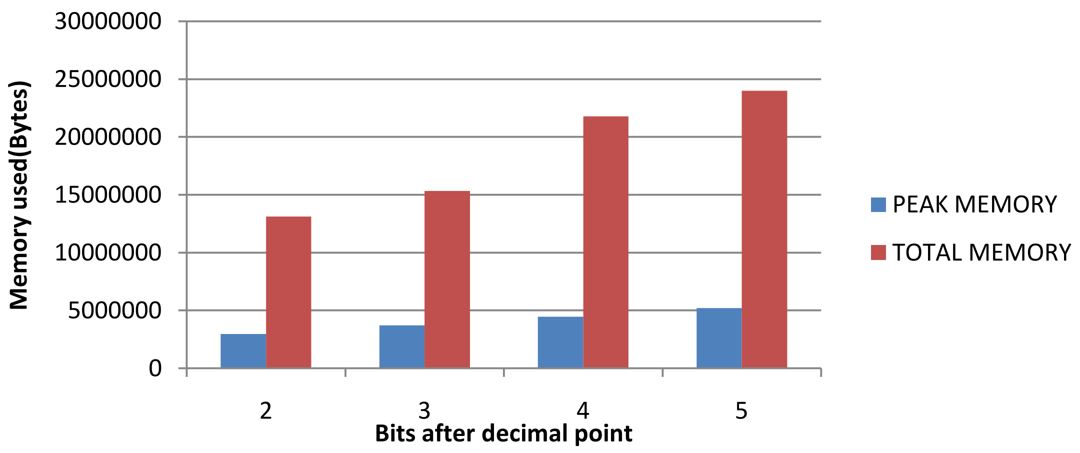


Figure 4.14 Memory consumption over bits given after decimal point for LZO Rounding.

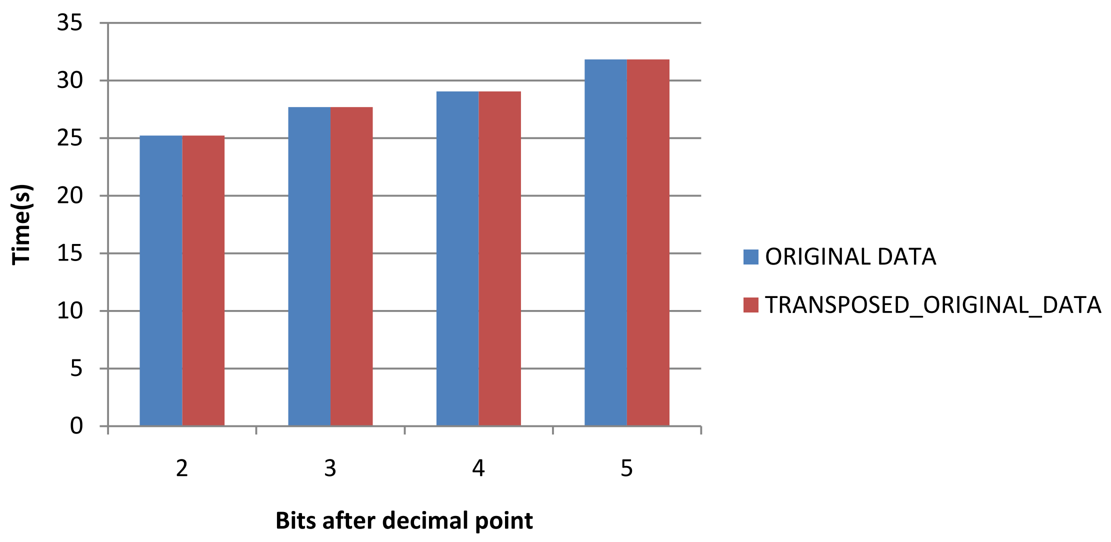


Figure 4.15 Execution time over bits given after decimal point for LZO Rounding.

4.2.2.2. Reordering

Sorting

We have tried to find out the best way of Reordering in order to make use of the attributes of LZO compressor in order to achieve high compression ratios and of course lossless, PRD value is 0. As we have pointed out in the beginning of the Implementation chapter LZO compressor produces big compression ratios when the coefficients of a block of data are the same in order the sliding dictionary to recognize similarities. The Figure 4.16 shows us that Sorting method produces a compression ratio which is including the indexes that are produced from the Sorting method less than 1 ,which means that the original file was smaller than the compressed one .This makes the Reordering with Sorting useless for data compression.

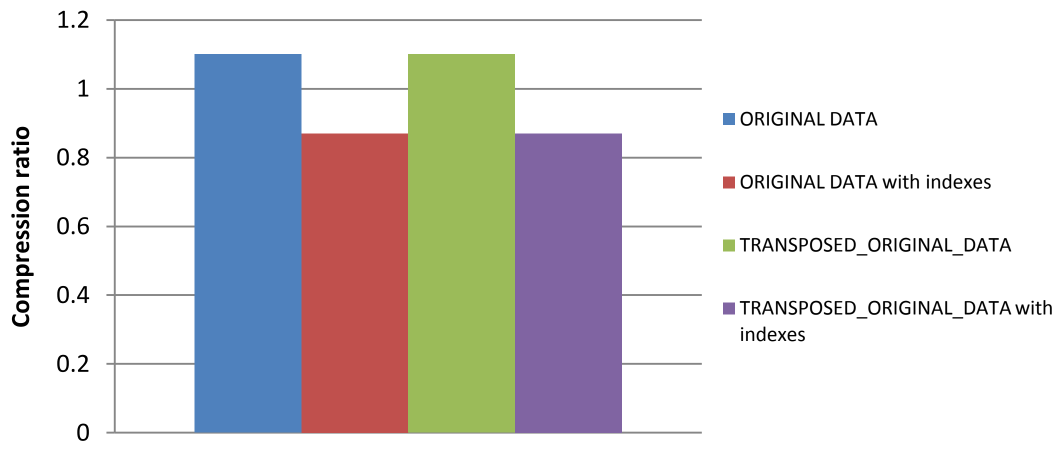


Figure 4.16 Compression ratio for different data of Sorting method

Although we have proved that the Sorting method is useless in terms of compression ratio , we state also the results from the computational overhead in order to observe differences between the Original and Transposed data. Finally, we can make out no differences between the 2 different types of saved types.

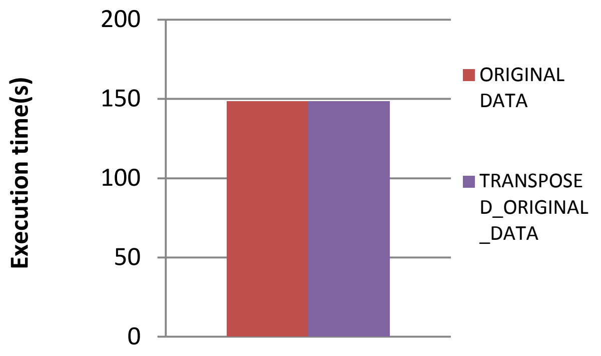


Figure 4.17 Execution time for different data of Sorting method

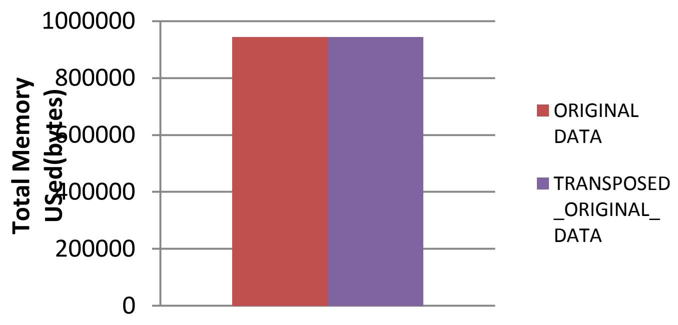


Figure 4.18 Total Memory used for different data of Sorting method

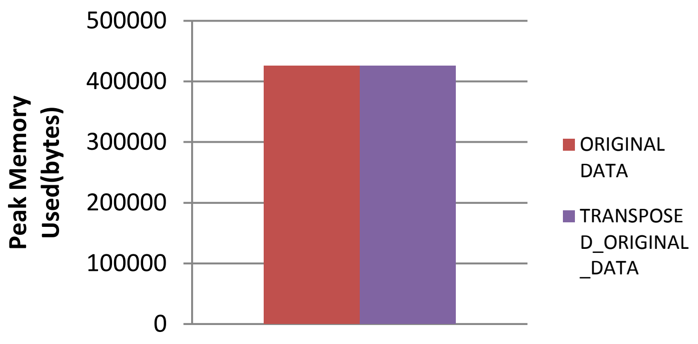


Figure 4.19 Peak Memory used for different data of Sorting method

Fuzzy C means clustering

We continue with the presentation of the results of the Reordering technique. A technique which has as main goal to reorder the data in a way to create more correlation to the coefficients so as the compressors to make use of this

correlation to produce better compression ratio. In this section we will present the results for Reordering with Fuzzy C means clustering, as in the results presented in the SPIHT section for this method we will present the results for different values of sample and cluster number in order to understand the reaction of LZO compressor to the change of these variables.
Reordering with fuzzy C means clustering using the LZO compressor is a lossless method that obviously produces zero value PRD.

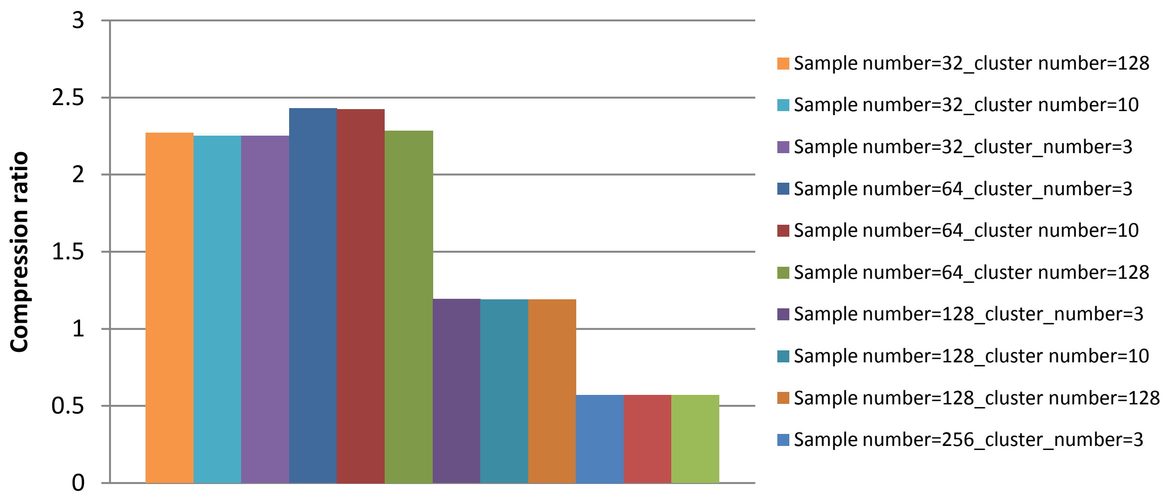


Figure 4.20 Compression ratio for different values of sample and cluster number

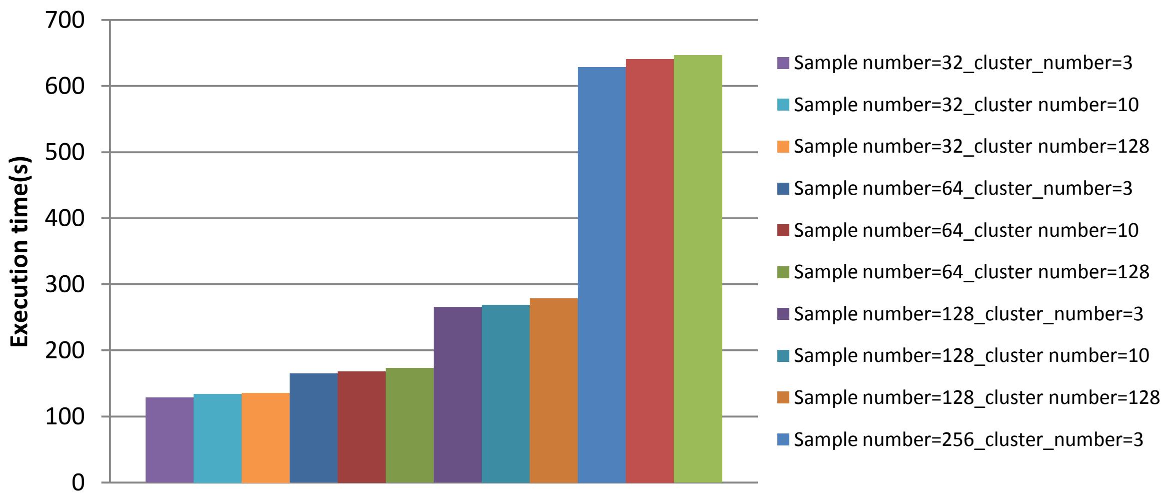


Figure 4.21 Execution time for different values of sample and cluster number

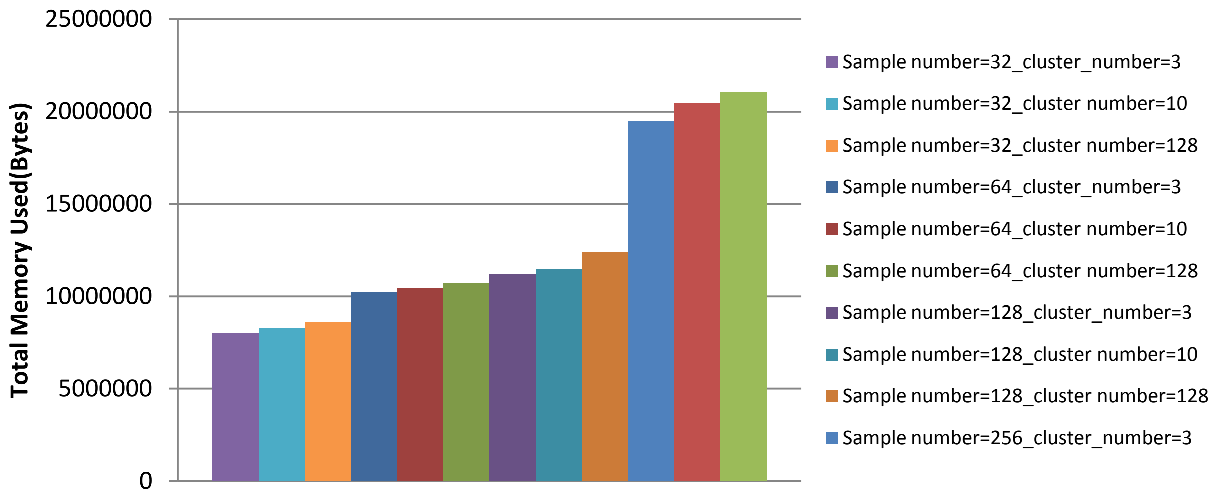


Figure 4.22 Total Memory Used for different values of sample and cluster number

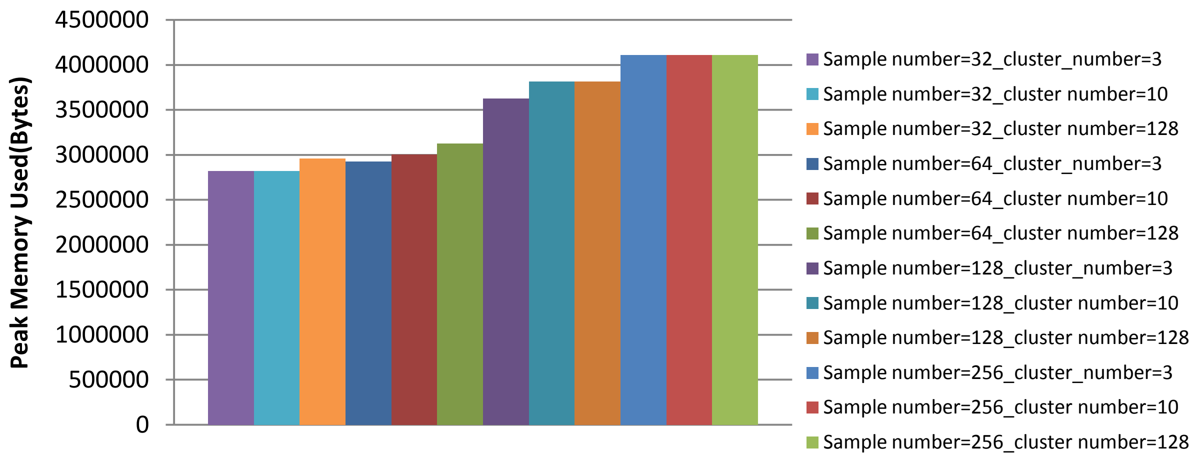


Figure 4.23 Peak Memory used for different values of sample and cluster number

The above figures illustrate that the best compression ratio is achieved for 64 sample number and 3 clusters. It is remarkable that these results do not agree with the SPIHT results on the same method, because now the compression procedure is different and the encoding with LZO algorithm is done per fragment, comparing to the SPIHT where the encoding part came at the end of the pre-procession method. We see that the computational overhead of the algorithm increase proportional as the sample number and the cluster number increase. More specifically the biggest growth is observed when the sample number increases. From 32 to 64 sample number there is a 30 % increase to the execution time and total memory used and an about 20 % increase as for the Peak memory. From 64 to 128 sample number there is a 18 % increase on the Peak Memory metric ,an 85 % increase on the execution time and more or less 15 % on the Total Memory that is used. More significant growth is illustrated between 128 and 256 sample number. The execution time and total memory over double their previous values and the Peak memory is increased by 20 %.

4.2.3. Comparison Overview between the aforementioned methods.

In this section we could try to clarify which of the aforementioned methods can be of some importance to our goal and also to clear up the differences of each algorithm and its individual characteristics.
We would present the top-achieving algorithms as the Compression ratio and the PRD is concerned.

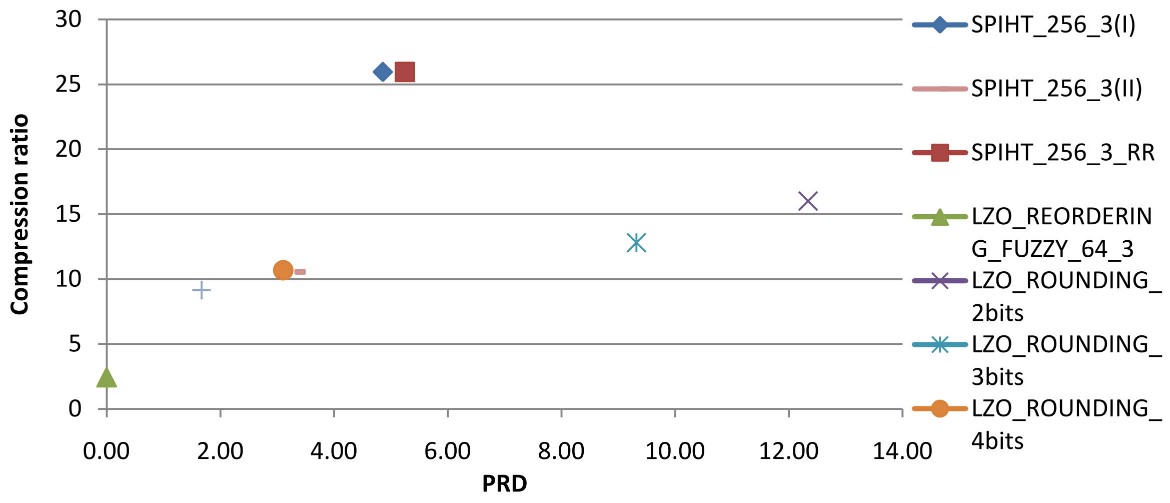


Figure 4.24 Compression ration over PRD results from the top-achieving algorithms.

Firstly, we have to define the reason that we illustrate the 2 different SPIHT methods with the same sample and cluster number. As we have already mentioned, in the SPIHT algorithm we can define the size of the compressed file by changing the PRD value of the compressed file. So, we present 2 different versions of SPIHT algorithm with 256 samples and 3 clusters. Also we should state here that as it is described in the 4.2.1.1 the 256 sample number and the 3 clusters is the top achieving algorithm as the compression ratio and PRD is concerned. From Figure 4.24 we can make out that the best compression ratio is achieved with SPIHT algorithm with 256 samples and 3 clusters. The best lossless algorithm (PRD is equal to zero) is the LZO algorithm when we use the Reordering with Fuzzy means clustering with 64 samples and 3 clusters. On the other hand, the best trade-off between compression ratio and PRD is done from LZO Rounding algorithm. As we can see from Figure 4.24 the LZO Rounding with 4 bits given to the fractional part has 0.22 smaller PRD value and 0.2 compression ratio bigger than the SPIHT with 256 sample number and 3 clusters(version II).

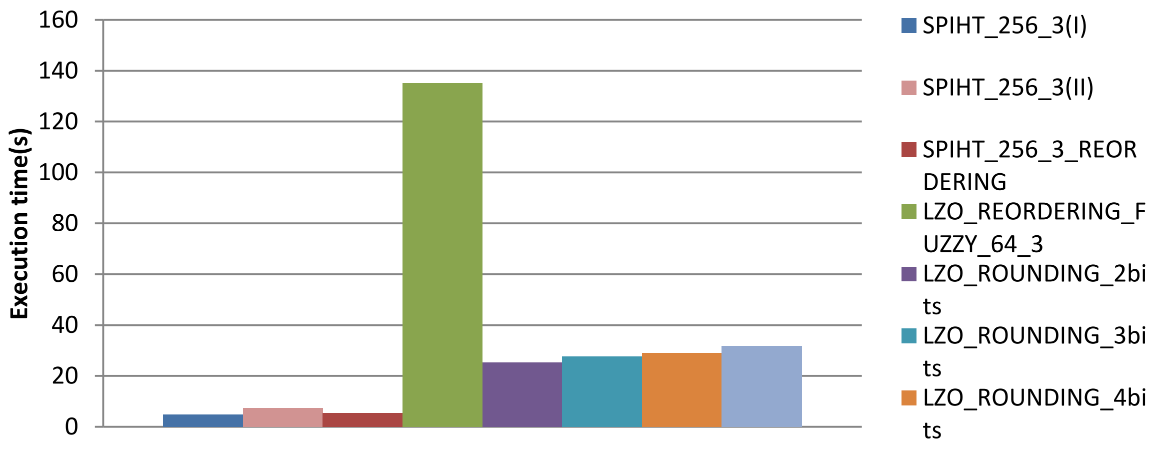


Figure 4.25 Execution time for the top-achieving algorithms

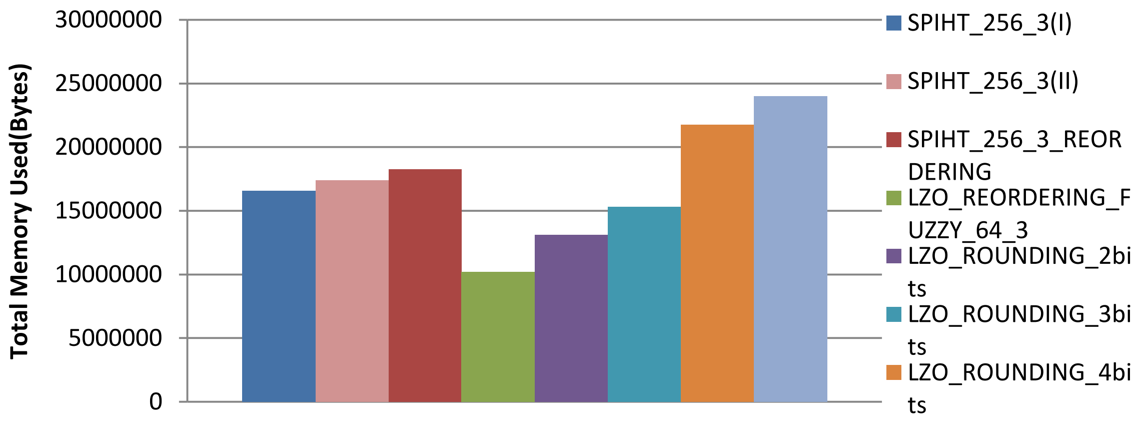


Figure 4.26 Total memory used for the top-achieving algorithms

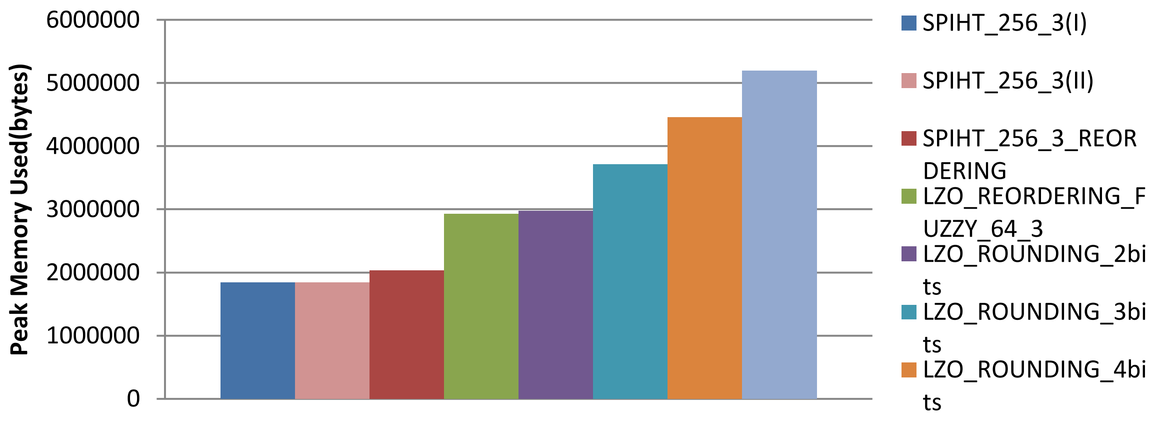


Figure 4.27 Peak Memory used time for the top-achieving algorithms

From the above figures that illustrate the computational overhead of the top-achieving algorithms we can see that the lossless method needs more time to efficient finish the compression. Most of the time is spent in the transforming the coefficients which are double precision values, in order to be properly saved in binary files. Moreover we can see that now the SPIHT with 256 sample number and 3 clusters (version II) performs better, as far as the performance metrics it concerns, than the LZO Reordering with 4 bits for the fractional part. More specifically, it consumes 15 seconds and uses 500000 bytes less.

5. Conclusions

In this thesis, we wanted to find and compare algorithms that can compress Biodata (mainly ECG) and more specifically algorithms tailored to the types of data typically used by IMDs in an energy efficient way with minimal loss of signal quality.
In Chapter 2, we stated the Background information about the algorithms that we would use and the algorithms that we would compare. Moreover, we mentioned some related work that has being done in that field of research and more generally based on ECG data compression or data compression on data stemmed from IMDs. Based on the 2 algorithms that have proposed significant results during our academic research the SPIHT and the LZO algorithm, we did a comparison between these algorithms evaluating their compression ratio, PRD results and also the computational overhead of their performance. Before, Evaluation chapter we stated clearly all the implementations needed in order to undergo a fair and efficient comparison. In Chapter 3, Implementation Chapter we mentioned all the methods and process techniques that we would use. Also, we presented the tools that we used for this Implementation and the Evaluation of the algorithms. Following, Chapter 4 the Evaluation Chapter we evaluated the pre-stated methods that we described in the previous chapter. More specifically, we concluded that as it is stated at [10] the most appropriate sample number is 256 and cluster number is 3 to use the SPIHT encoding algorithm. With these variables the SPIHT algorithms achieves its best values as far as compression ratio and PRD concerns. Moreover, we have pointed out that as we increase the sample number and cluster number in SPIHT encoding algorithm the performance getting worst. We should also mention that the reordering does not offer anything at all in the SPIHT algorithm with fuzzy C means clustering, as it produces bigger PRD values and the same compression ratio and performance metrics than the SPIHT algorithm with fuzzy C means clustering without rounding.
As for the LZO encoding algorithm, which according to [8] is ideal for implemented devices, we can see that combined to reordering with fuzzy C means algorithm achieves a decent compression ratio with the disadvantage of execution time that is too much ,but using a small proportion of the Memory and all the aforementioned losslessy. On the other hand, the reordering with the sorting method performs really badly, as the size of the indices that have to store is really big and at the end it provides negative compression ratio percentage. Moreover, the most promising data algorithm is the Rounding method combined with LZO algorithm. We have used this method in order to create a lossy algorithm with the use of lossless encoding algorithm as the LZO, in order to be better comparable to the SPIHT algorithm with fuzzy C means clustering. After the comparison between these methods we have

figured out that the LZO-Rounding algorithm cannot achieve the compression ratio of the SPIHT-Reordering algorithm, but for a little bigger compression ratio it can achieve better PRD values, with a little loss as far as its performance metrics it concerns.
Finally, we have ended up to more partial conclusions. There is any significant difference at the compression ability of LZO compressor either the input of the algorithm is with the original data or the data are transposed.
At the end of the day, we could conclude that low-overhead compression of ECG recordings for Implantable medical devices is a very crucial and interesting subject that has a lot of parameters ,which one should highly take into consideration before chooses the most appropriate algorithm.

5.1. Future work

Based on our current work we proposed some of the future work that can be done in order to increase the contribution of this diploma thesis to this field of research. Firstly, we could try the same algorithms and methods on different types of biomedical data, such as Electromyogram, Electroencephalogram, Blood pressure, Pulmonary function ,Respiratory Cycle etc etc.In order to create a more generic algorithm that could include the majority of the data that are transmitted from an IMD
Secondly, we could study different algorithms that have presented promising results on the Biodata compression tailored for IMDs. Algorithms like these can be found in the Related work section of our study.
Finally, we could create a new compressor tailored to ECGs, based on the lessons of this thesis. This algorithm could be a combination of the aforementioned algorithms or creation of a new one based on the conclusions that extracted from the thesis.

6. Bibliography

[1] Yeun-Ho Joung, "Development of Implantable Medical Devices: From an Engineering Perspective," *International Neurourology Journal*, no. 13, pp. 98-106, September 2013.

[2] Brown P, Nissen Zuckerman DM, "Medical device recalls and the FDA approval," *Arch Intern Med*, no. 171, pp. 1006–1018.

[3] Lav Gupta, "Security in Low Energy Body Area Networks for Healthcare," 2014.

[4] Sarbari Gupta, "Implantable Medical Devices - Cyber Risks and Mitigation Approaches," 2012.

[5] Khalid Sayood, *Introduction to Data Compression*, THIRD EDITION ed., 2006.

[6] M. F. X. J Oberhumer. "LZO source code". [Online]. www.oberhumer.com/opensource/lzo

[7] L. Erdődi, "File compression with LZO algorithm using NVIDIA CUDA architecture," *4th IEEE International Symposium on Logistics and Industrial Informatics*, September 2012.

[8] Georgi N. Gaydadjiev Christos Strydis, "Profiling of Lossless-Compression Algorithms for a Novel Biomedical-Implant Architecture," *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis*, pp. 109-114, 2008.

[9] A. Materka, M. Strzelecki P. Szczypińsk M. Kociołek, "Discrete wavelet transform –derived features for digital image texture analysis," *Proceedings of Interational Conference on Signals and Electronic Systems*, pp. 163-168, September 2001.

[10] Wisnu Jatmiko, Aniati Murni Arymurthy Sani M. Isa, "Beat Reordering for Optimal Electrocardiogram Signal Compression using SPIHT," *IEEE International Conference on Systems, Man, and Cybernetics*, October 2012.

[11] Ziya Arnavut, "ECG Signal Compression Based on Burrows-Wheeler Transformation and Inversion Ranks of Linear Prediction," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, March 2007.

[12] Pawel Turcza, Tomasz P. Zielinski' Krzysztof Duda, "Lossless ECG Compression with Lifting Wavelet Transform ," May 2001.

[13] Antti Koski*, "Lossless ECG encoding ," *Computer Methods and Programs in Biomedicine* , pp. 23-33 , 1997.

[14] Elmar Uwe ,Kurt Melcher Leonardo Vidal Batista, "Compression of ECG signals by optimized quantization of discrete cosine transform coefficients," *Medical Engineering & Physic*, no. 23, pp. 127–134, 2001.

[15] Michael L. Hilton, "Wavelet and Wavelet Packet Compression of

Electrocardiograms," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, vol. 44, no. 5, MAY 1997.

[16] LA Koyrakh, "Data Compression for Implantable Medical Devices," *Computers in Cardiology*, vol. 35, p. 417−420, 2008.

[17] I. S. Member, K. Faez, I. Member, and S. Sargolzaei A. Sargolzaei, "A New Robust Wavelet Based Algorithm for Baseline Wandering Cancellation in ECG Signals," *Electrical Engineering*, pp. 33-38, 2009.

[18] G. Ramakrishnan and S. Saha, "ECG coding by wavelet-based linear prediction," *IEEE transactions on bio-medical engineering*, vol. 44, no. 12, pp. 1253-61, Dec. 1997.

[19] ROBERT EHRLICH ,WILLIAM FULL JAMES C. BEZDEK, "FCM: THE FUZZY c-MEANS CLUSTERING ALGORITHM," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191-203, 1984.

[20] N. Sriraam, "Correlation dimension based lossless compression of EEG signals," *Biomedical Signal Processing and Control* , no. 7, pp. 379– 388, 2012.

[21] Eameema Muntimadugu , Michael Jaffe and Abraham J. Domb Wahid Khan, "Implantable Medical Devices ," in *Focal Controlled Drug Delivery.*, ch. 2.

[22] J. T. Rubinstein, "How Cochlear Implants Encode Speech," *Current Opinion in Otolaryngology & Head and Neck Surgery Journal*, no. 12, pp. 444-448, 2004.

[23] L. Wentai, "Image Processing and Interface for Retinal Visual Prostheses," *Circuits and Systems*, no. 3, pp. 2937-40, 2005.

[24] M. Burrows and D.J. Wheeler, "A Block-sorting Lossless Data compression algortihm," Systems Research Center, Palo Alto, May 10 1994.

[25] DANIEL D. SLEATOR, ROBERT E. TARJAN,and VICTOR K. WEI JON LOUIS BENTLEY, "A locally adaptive data compression scheme," *Communications of the ACM*, vol. 4, no. 29, pp. 320-30, April 1984.

[26] C. E. Shannon, "A mathematical theory of communication," no. 27, pp. 379-423., July 1948..

[27] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, no. 40, pp. 1098-1 101, September 1952.

[28] PAUL G. HOWARD and JEFFREY SCOTT VITTER, "Arithmetic Coding for Data Compression," 1994.

[29] T. Natarajan, and K. R. Rao N. Ahmed, "Discrete cosine transform," *IEEE Trans. Comput*, no. 23, pp. 90-93, 1974.

[30] Chun-Hee Lee and Chin-Wan Chung, "Compression Schemes with Data Reordering for Ordered Data," December 2009.

[31] WB Pennebaker, *JPEG: Still image data compression standard*. USA: Kluwer, 1993.