Universidad de la República

Tesis de Maestría

Coding of Multichannel Signals with Irregular Sampling

Autor:
Pablo Cerveñansky

Supervisores:
Álvaro Martín
Gadiel Seroussi

Núcleo de Teoría de la Información
Facultad de Ingeniería

14 de octubre de 2019

Chapter 1

Datasets

1.1 Introduction

Explain why / how every dataset was transformed into a common format.
Show example csv (describe header, data rows, etc.)

Dataset	#Files	#Types	Data Types
IRKIS	7	1	VWC
SST	3	1	SST
ADCP	3	1	Vel
Solar	4	3	GHI, DNI, DHI
ElNino	1	7	Lat, Long, Zonal Winds, Merid. Winds, Humidity, Air Temp., SST
Hail	1	3	Lat, Long, Size
Tornado	1	2	Lat, Long
Wind	1	3	Lat, Long, Speed

Tabla 1.1: Datasets overview.

1.2 IRKIS

1.3 SST

1.4 ADCP

1.5 ElNino

1.6 Solar

1.7 Hail

1.8 Tornado

1.9 Wind

Chapter 2

Coders

2.1 Introduction

- Add two papers as biography
- Explain the Arithmetic Coder (add website as bibliography)

2.2 CoderBase

```
input  : file ∈ Datasets: csv file to be coded
output: CoderBase(file): binary file coded with CoderBase
 1 out = new_binary_file()
 2 out.code_integer(CODER_BASE, 8)
 3 out.code_header(file)
 4 out.code_integer(file.data_rows_count(), 24)
 5 foreach column in file.columns do
 6 |   foreach entry in column.entries do
 7 |   |   if entry = NO_DATA then
 8 |   |   |   value = column.no_data_integer
 9 |   |   else
10 |   |   |   value = entry + column.offset
11 |   |   end
12 |   |   out.code_integer(value, column.total_bits)
13 |   end
14 end
15 out.close_file()
```

FIGURE 2.1: *CoderBase* pseudocode.

```
input  : file: binary file coded with CoderBase
output: DecoderBase(file): csv file decoded with DecoderBase
 1 out = new_csv_file()
 2 coder_value = file.decode_integer(8)
 3 out.decode_header(file)
 4 data_rows_count = file.decode_integer(24)
 5 if coder_value = CODER_BASE then
 6 |   foreach column in out.columns do
 7 |   |   foreach entry in column.entries do
 8 |   |   |   value = file.decode_integer(column.total_bits)
 9 |   |   |   if value = column.no_data_integer then
10 |   |   |   |   out.write_string(NO_DATA)
11 |   |   |   else
12 |   |   |   |   out.write_string(value − column.offset)
13 |   |   |   end
14 |   |   end
15 |   end
16 else
17 |   . . . // if file was coded with a different coder
18 end
19 out.close_file()
```

FIGURE 2.2: *DecoderBase* pseudocode.

2.3 CoderPCA

2.4 CoderAPCA

2.5 CoderCA

2.6 CoderPWLH and CoderPWLHInt

2.7 CoderGAMPS and CoderGAMPSLimit

2.8 CoderFR

2.9 CoderSF

Chapter 3

Experimental Results

In this chapter we present the experimental results of our project. The main goal of our experiments is to analyze the performance of every one of the coding algorithms implemented in Chapter 2. To achieve that, we use them to code the different data types of the datasets introduced in Chapter 1. In Section 3.1 we give an overview of the performed experiments. In Section 3.2 we compare the compression performance of the masking and non-masking modes. In Section 3.3 we study how the window size parameter affects the performance of the coding algorithms. In Section 3.4 we compare the performance of the algorithms among each other, while in Section 3.5 we compare them with the gzip algorithm.

3.1 Experimental Setting

We evaluate the compression performance of all the coding algorithms presented in Chapter 2 on the datasets described in Chapter 1. For each algorithm we test both the masking and the non-masking mode (except for *CoderBase*, *CoderFR* and *CoderSF*, which only operate in non-masking mode).

We also test several combinations of algorithm parameters. Specifically, for the algorithms that admit a window size parameter w (every algorithm except *CoderBase* and *CoderSF*), we test all the values of w in the set $W = \{4, 8, 16, 32, 64, 128, 256\}$. For the encoders that admit a lossy compression mode with a threshold parameter e (every encoder except *CoderBase*), we test all the values of e in the set $E = \{1, 3, 5, 10, 15, 20, 30\}$, where each threshold is expressed as a percentage fraction of the standard deviation of the data type being coded. For example, for a certain data type with a standard deviation of 20, taking $e = 10$ implies that the lossy compression allows for a maximum sample distortion of 2.

Definition 3.1.1. We refer to a specific combination of a coding algorithm and its parameter values as a *coding algorithm instance (CAI)*. We define *CI* as the set of all the CAIs obtained by combining each of the algorithms presented in Chapter 2 with the parameter values (from W and E) which are suitable for that algorithm.

We assess the compression performance of a CAI mainly through the compression ratio, which we define next. For that definition, we recall that *CoderBase* is a trivial encoder that serves as a base ground for compression performance comparison.

Definition 3.1.2. Let f be a file and let z be a data type of a certain dataset introduced in Chapter 1. We define f_z as the subset of data of type z from file f.

Definition 3.1.3. The *compression ratio (CR)* of a CAI $c \in CI$ for the data of type z of a certain file f is given by

$$CR(c, f_z) = 100 \times \frac{|c(f_z)|}{|CoderBase(f_z)|}, \tag{3.1}$$

where $|c(f_z)|$ and $|CoderBase(f_z)|$ are the sizes of the resulting files obtained when coding f_z with c and *CoderBase*, respectively.

The performance of c improves as $|c(f_z)|$ decreases. Thus, our main goals are to analyze which CAIs minimize equation (3.1) for the different data types, and to study how do the coding algorithms and the values of their parameters influence the result of this equation.

To compare the compression performance between a pair of CAIs we calculate the relative difference, which we define next. In general, it only makes sense to compare CAIs that have the same threshold parameter e.

Definition 3.1.4. The *relative difference (RD)* between a pair of CAIs $c_1, c_2 \in CI$ for the data of type z of a certain file f is given by

$$RD(c_1, c_2, f_z) = 100 \times \frac{|c_2(f_z)| - |c_1(f_z)|}{|c_2(f_z)|}, \tag{3.2}$$

where $|c_1(f_z)|$ and $|c_2(f_z)|$ are the sizes of the resulting files obtained when coding f_z with c_1 and c_2, respectively.

c_1 has a better performance than c_2 when the result of equation (3.2) is a positive value. As said value increases, the relative performance of c_1 with respect to c_2 improves.

3.2 Comparison of Masking and Non-Masking Modes

In this section we compare the compression performance of the masking and non-masking modes of each of the evaluated coding algorithms that admit both modes. Specifically, we compare:

- *CoderPCA-M* and *CoderPCA-NM*
- *CoderAPCA-M* and *CoderAPCA-NM*
- *CoderCA-M* and *CoderCA-NM*
- *CoderPWLH-M* and *CoderPWLH-NM*
- *CoderPWLHInt-M* and *CoderPWLHInt-NM*
- *CoderGampsLimit-M* and *CoderGampsLimit-NM.*

Definition 3.2.1. Let A be the set of coding algorithms listed above. We define a_M and a_{NM} as the masking and non-masking modes for any given coding algorithm $a \in A$.

For comparing the performance of a_M and a_{NM} when compressing certain data with a given threshold parameter e, we calculate the relative difference. We only consider the window size parameters w_M and w_{NM} that obtain the best results (i.e. minimize the compression ratios). We refer to w_M and w_{NM} as the optimal window sizes and formally define them next.

Definition 3.2.2. The *optimal window size (OWS)* of a coding algorithm $a \in A$ and a threshold parameter $e \in E$, for the data of type z of a certain file f is given by

$$OWS(a, e, f_z) = w^* \in W \text{ such that } CR((a, w^*, e), f_z) = \min_{w \in W} \Big\{ CR((a, w, e), f_z) \Big\}, \tag{3.3}$$

where we take the smallest window in the event more than one value satisfies the equation[1]. Note that the triplet (a, w, e) defines a CAI in CI, hence we are correctly applying equation (3.1).

For each data type z of each dataset file f, and each threshold parameter $e \in E$ and coding algorithm $a \in A$, we calculate the equation (3.2) as

$$RD((a_M, w^*_M, e), (a_{NM}, w^*_{NM}, e), f_z), \tag{3.4}$$

where $w^*_M = OWS(a_M, e, f_z)$ and $w^*_{NM} = OWS(a_{NM}, e, f_z)$. Note that both triplets (a_M, w^*_M, e) and (a_{NM}, w^*_{NM}, e) define CAIs in CI, hence we are correctly applying the relative difference equation.

As an example, in Figure 3.1 and Figure 3.2 we display the compression ratio and relative difference plots obtained for two data types of different datasets. We selected these two plots because they include the maximum and minimum values returned by equation (3.4).

Table 3.1 summarizes the results obtained for each dataset when comparing the relative performance of a_M and a_{NM} for each $a \in A$. The second column outlines the amount of gaps in the dataset. The third column displays the percentage of combinations in which a_M performs better than a_{NM} (i.e. cases in which equation (3.4) returns a positive value). The last column shows the range for the values of equation (3.4) for said combinations.

Dataset	**Dataset Characterstic**	**Cases where masking supersedes non-masking mode (%)**	**Range of equation (3.4)**
IRKIS	Many gaps	100	(0; 36.88]
SST	Many gaps	100	(0; 50.60]
ADCP	Many gaps	100	(0; 17.35]
ElNino	Many gaps	100	(0; 50.52]
Solar	Few gaps	51	[-0.25; 1.77]
Hail	No gaps	0	[-0.04; 0)
Tornado	No gaps	0	[-0.29; 0)
Wind	No gaps	0	[-0.12; 0)

TABLA 3.1: Relative performance of a_M and a_{NM} for each $a \in A$. In the last column we highlight the maximum (blue) and minimum (red) values of equation (3.4).

For every coding algorithm, on datasets with many gaps the masking mode always produces the best result, while on gapless datasets the non-masking mode always achieves the best result. On the dataset with few gaps, on each half of the combinations the best results are obtained with different modes.

Observing the last column of Table 3.1, we notice that in every case in which the non-masking mode a_{NM} performs best, the relative difference is close to zero. In Figure 3.2 we show the case in which a_{NM} obtains the most significative relative difference. This occurs for the Tornado dataset and in the table we can verify that in such case the relative difference is -0.29%.

[1] This was never the case on our experiments.

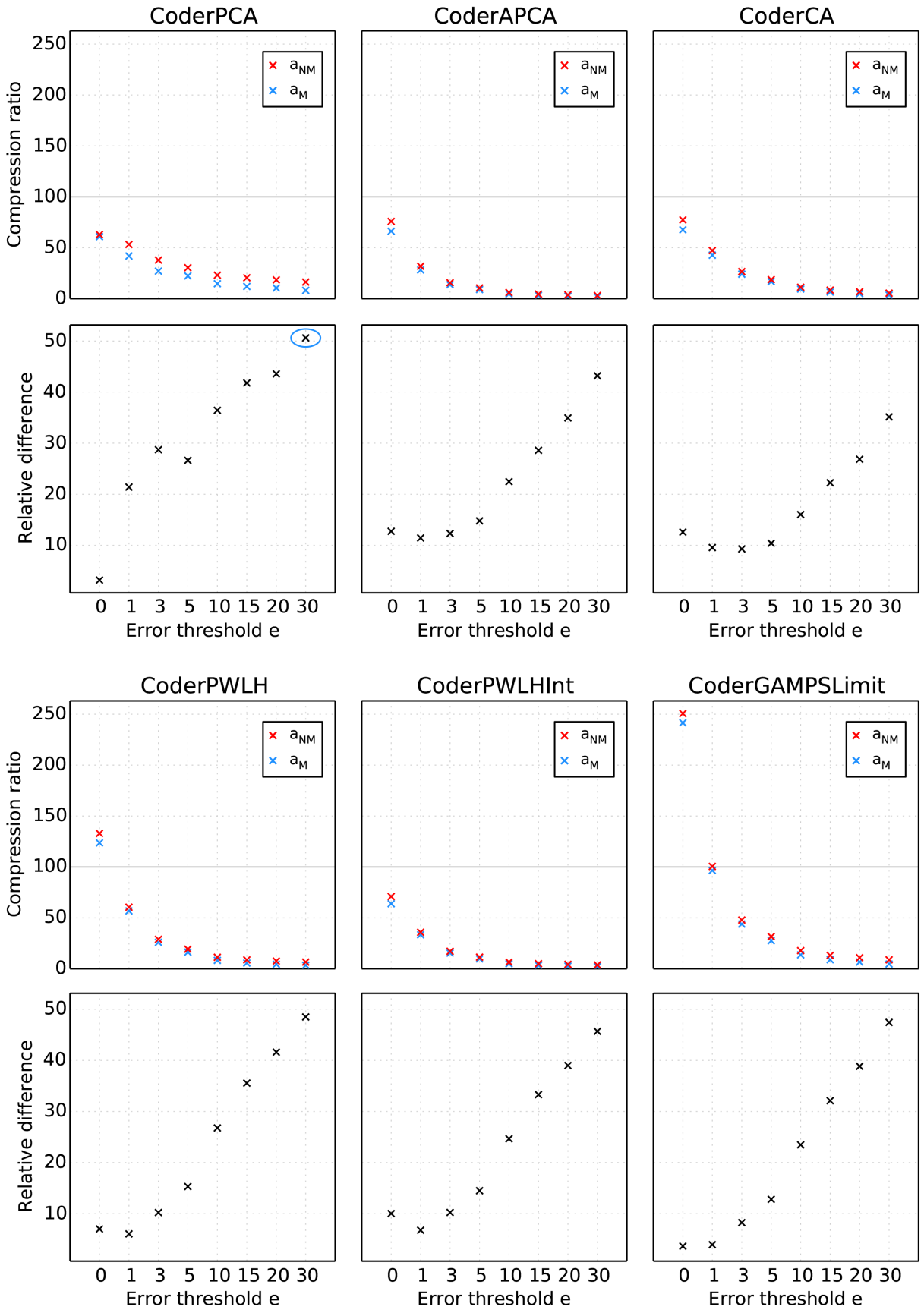

FIGURE 3.1: Compression ratio and relative difference plots for a_M and a_{NM} for every coding algorithm $a \in A$, for the "VWC" data type of the SST dataset. In the relative difference plot for CoderPCA we marked with a blue circle the case in which a_M obtains the most significant relative difference (50.60%).

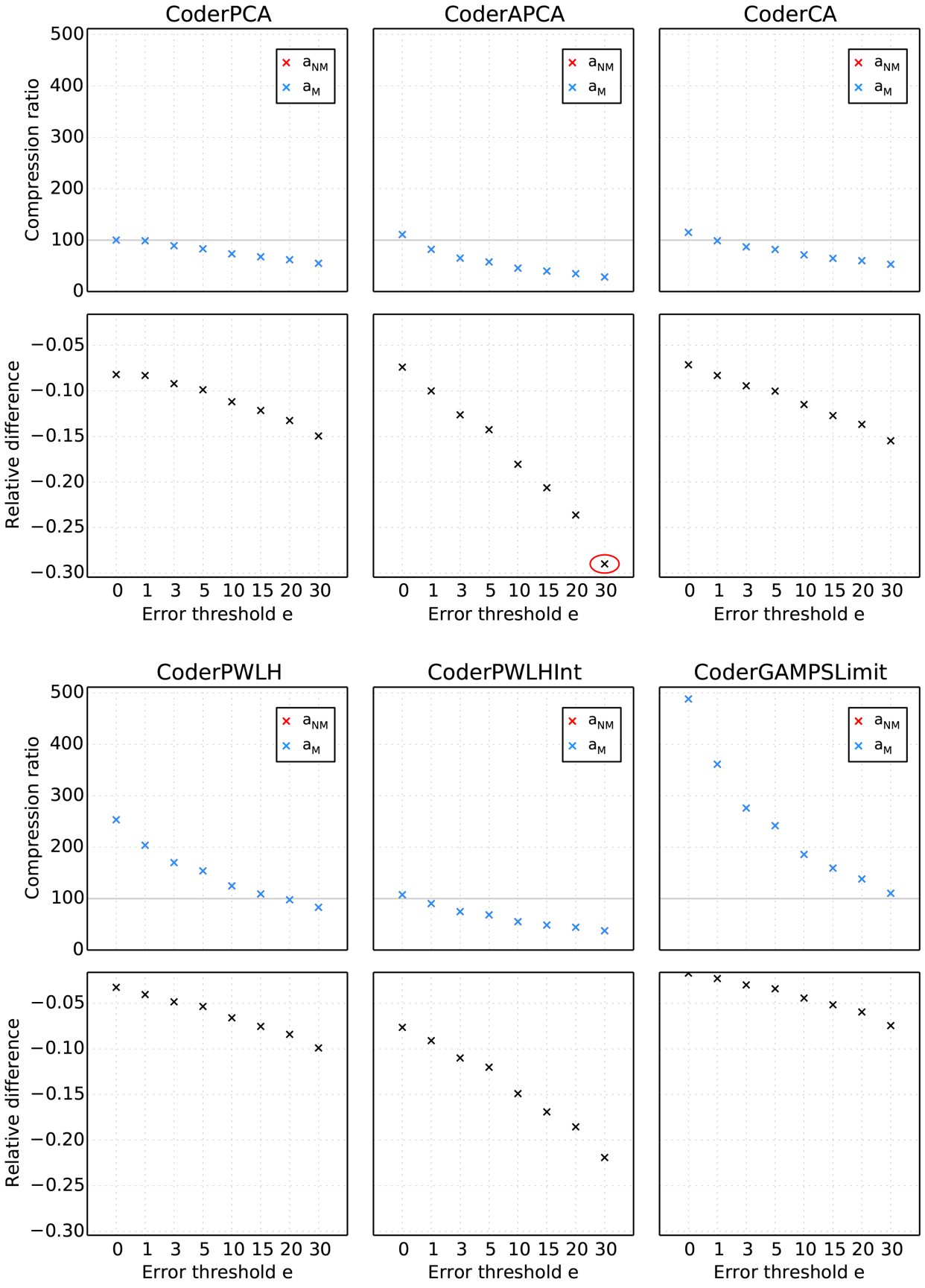

FIGURE 3.2: Compression ratio and relative difference plots for a_M and a_{NM} for every coding algorithm $a \in A$, for the "Longitude" data type of the Tornado dataset. In the compression ratio plot the red and blue markers are almost overlapping. In the relative difference plot for CoderAPCA we marked with a red circle the case in which a_{NM} obtains the most significant relative difference (-0.29%).

On the other hand, when the masking mode a_M performs best, the relative difference reaches high absolute values. The maximum, which amounts to 50.60%, is achieved for the SST dataset. We show that particular case in Figure 3.1.

The experimental results presented in this section suggest that if we were interested in compressing a dataset with many gaps, we would benefit by using a masking coding algorithm a_M. However, even if the dataset didn't have any gaps, the performance gain obtained by using a non-masking algorithm a_{NM} instead would be negligible. Since the a_M algorithm is more robust and performs better in general, in the next sections we will focus on its study.

3.3 Window Size Parameter

TODO

3.4 Mask Coders Performance

In this section we analyze the performance of every one of the mask coders implemented in Chapter 2. Once again, the compression rate (equation (3.1)) and the relative difference (equation (3.2)) will be the metrics we use for comparing the coders between each other.

We considered the results obtained when coding the different data types of the datasets introduced in Chapter 1. For example, in Figure 3.3 we can see the graphs obtained for the "VWC" data type of the IRKIS dataset. For each $<c \in C,\ e \in E>$ combination we plot two values: the window size which minimizes the compression rate and said compression rate.

Easily, after observing the plots we noticed that in general the compression rate for coders *CoderPWLH-M*, *CoderGAMPSLimit-M* and *CoderSF-M* was worst than the rest.

Analyze the data and discard these

PONER OTRA GRAFICA DE OTRO TIPO DE DATO

		e = 0		e = 1		e = 3		e = 5		e = 10		e = 15		e = 20		e = 30	
Dataset	Data Type	OWS	CR	OWS	CR	OWS	CR	OWS	CR	OWS	CR	OWS	CR	OWS	CR	OWS	CR
IRKIS	VWC	4	20.32	4	18.35	5	12.37	6	6.77	7	3.07	8	2.22	8	1.71	8	1.21
SST	SST	8	60.84	3	28.12	5	13.64	6	8.88	7	4.63	8	3.15	8	2.39	8	1.72
ADCP	Vel	8	68.22	8	68.22	2	66.8	2	61.07	2	48.44	2	40.9	3	34.9	3	25.93
Solar	GHI	2	77.65	3	76.1	4	71.39	4	67.2	4	58.52	4	52.41	4	47.03	4	37.78
	DNI	2	75.93	4	72.22	4	65.75	4	61.37	4	53.98	4	48.55	4	43.36	4	35.66
	DHI	2	77.66	2	77.43	4	71.62	4	67.6	4	60.12	4	53.62	4	47.86	4	38.71
ElNino	Lat	4	15.96			4	15.82	4	15.11	4	12.34	5	9.89	5	8.61	6	5.76
	Long	3	17.36	4	17.05	4	13.04	5	11.75	6	8.65	6	6.56	7	4.93	8	2.37
	Zonal Winds	8	31.46			8	31.46	8	31.46	2	27.36	2	23.5	2	20.54	3	16.44
	Merid. Winds	8	31.46			8	31.46	8	31.46	2	29.16	2	25.86	2	23.33	2	19.15
	Humidity	8	23.1	8	23.1	8	23.1	8	23.1	2	20.51	2	18.14	2	16.01	2	12.94
	AirTemp	8	32.68	8	32.68	2	30.33	2	27.39	2	22.42	3	19.24	3	16.76	4	13.31
	SST	8	32.91	2	30.96	2	24.6	2	20.61	3	14.17	4	10.66	4	8.21	5	5.42
Hail	Lat	8	100.04	8	100.04	2	89.83	2	82.62	2	71.49	3	64.62	3	57.49	3	46.75
	Long	8	100.03	8	100.03	2	85.91	2	77.5	2	65.06	3	55.38	3	48.72	4	38.74
	Size	2	80.61	2	80.59	2	80.59	2	80.58	2	80.56	2	80.53	2	80.52	3	64.35
Tornado	Lat	8	100.05	2	85.43	2	70.63	2	65.17	3	54.17	3	46.78	4	41.95	4	33.48
	Long	8	100.11	2	82.12	2	65.09	3	57.66	3	45.55	4	39.88	4	34.84	4	28.41
Wind	Lat	8	100.03	8	100.03	2	88.74	2	81.29	2	69.82	3	62.44	3	56.18	3	47.15
	Long	8	100.03	2	95.41	2	80.29	2	73.21	3	62.06	3	54.33	3	48.52	4	39.73
	Speed	4	65.49	3	43.82	6	25.9	7	16.79	5	15.71	6	12.29	6	10.33	6	8.21

Tabla 3.2: Mask results overview (1).

		e = 0		e = 1		e = 3		e = 5		e = 10		e = 15		e = 20		e = 30	
Dataset	Data Type	OWS	CR	OWS	CR	OWS	CR	OWS	CR	OWS	CR	OWS	CR	OWS	CR	OWS	CR
IRKIS	VWC		13.44		13.44	5	12.37	6	6.77	7	3.07	8	2.22	8	1.71	8	1.21
SST	SST		52.06	3	28.12	5	13.64	6	8.88	7	4.63	8	3.15	8	2.39	8	1.72
ADCP	Vel		61.38		61.38		61.38	2	61.07	2	48.44	2	40.9	3	34.9	3	25.93
Solar	GHI		69.01		69.01		69.01	4	67.2	4	58.52	4	52.41	4	47.03	4	37.78
	DNI		66.88		66.88	4	65.75	4	61.37	4	53.98	4	48.55	4	43.36	4	35.66
	DHI		61.01		61.01		61.01		61.01	4	60.12	4	53.62	4	47.86	4	38.71
ElNino	Lat		7.89				7.89		7.89		7.89		7.89		7.89	6	5.76
	Long		7.1		7.1		7.1		7.1		7.1	6	6.56	7	4.93	8	2.37
	Zonal Winds	8	31.46			8	31.46	8	31.46	2	27.36	2	23.5	2	20.54	3	16.44
	Merid. Winds	8	31.46			8	31.46	8	31.46	2	29.16	2	25.86	2	23.33	2	19.15
	Humidity	8	23.1	8	23.1	8	23.1	8	23.1	2	20.51	2	18.14	2	16.01	2	12.94
	AirTemp	8	32.68	8	32.68	2	30.33	2	27.39	2	22.42	3	19.24	3	16.76	4	13.31
	SST		32.43	2	30.96	2	24.6	2	20.61	3	14.17	4	10.66	4	8.21	5	5.42
Hail	Lat	8	100.04	8	100.04	2	89.83	2	82.62	2	71.49	3	64.62	3	57.49	3	46.75
	Long	8	100.03	8	100.03	2	85.91	2	77.5	2	65.06	3	55.38	3	48.72	4	38.74
	Size		36.73		36.73		36.73		36.73		36.73		36.73		36.73		36.73
Tornado	Lat	8	100.05	2	85.43	2	70.63	2	65.17	3	54.17	3	46.78	4	41.95	4	33.48
	Long	8	100.11	2	82.12	2	65.09	3	57.66	3	45.55	4	39.88	4	34.84	4	28.41
Wind	Lat	8	100.03	8	100.03	2	88.74	2	81.29	2	69.82	3	62.44	3	56.18	3	47.15
	Long	8	100.03	2	95.41	2	80.29	2	73.21	3	62.06	3	54.33	3	48.52	4	39.73
	Speed	4	65.49	3	43.82	6	25.9	7	16.79	5	15.71	6	12.29	6	10.33	6	8.21

TABLA 3.3: Mask results overview (2).

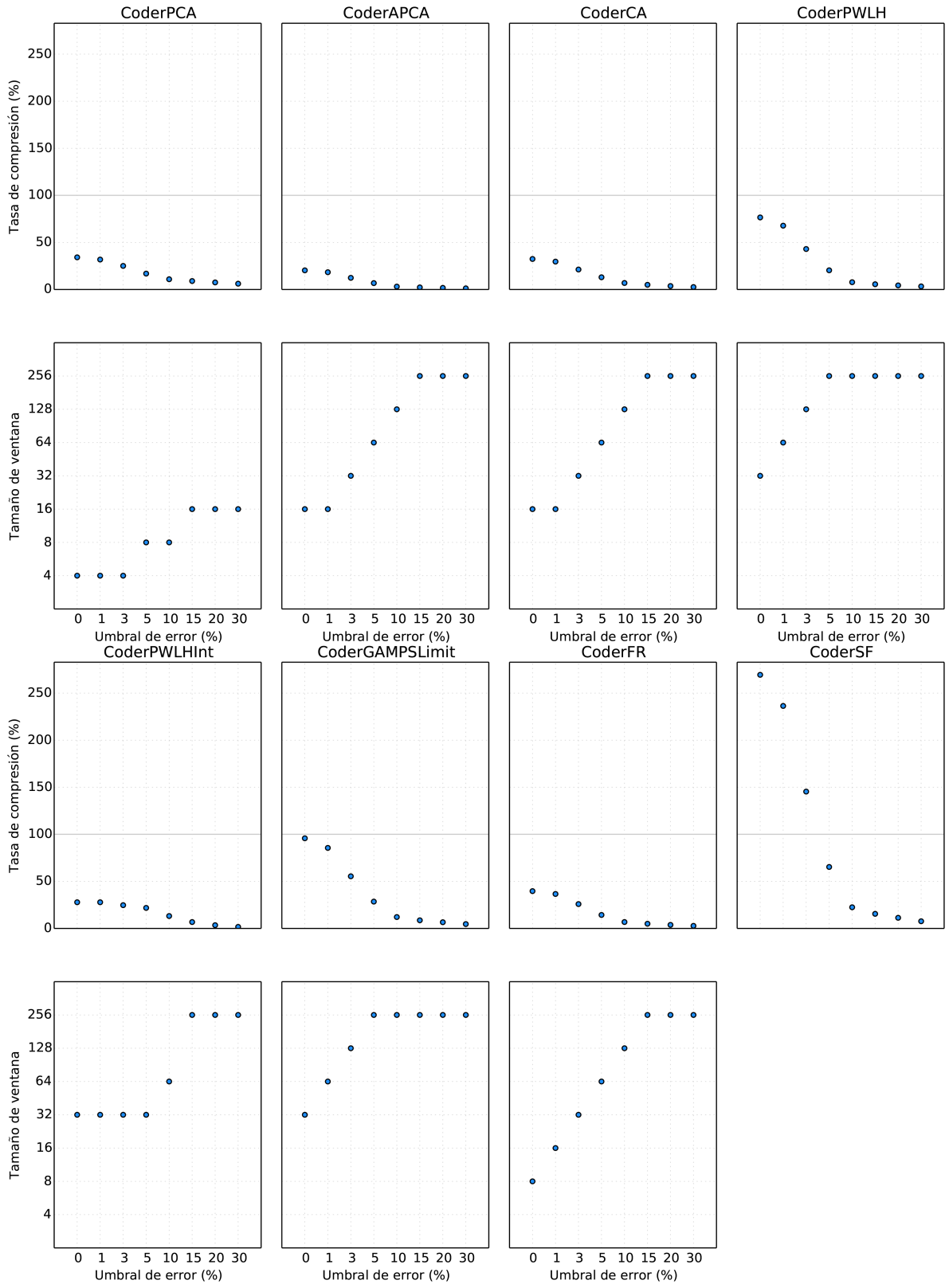

FIGURE 3.3: Compression rate and Window size graphs for the different combinations $<c \in C,\ w \in W, e \in E>$ for the "VWC" data type of the IRKIS dataset.

3.5 Comparison with the gzip Algorithm

HECHO INFORME:

- Elegir nomenclatura para los dos distintos modos de ejecución => CoderPCA-NM (sin máscara) y CoderPCA-M (con máscara).

- Realizar un análisis cuantitativo para saber qué tanto mejor comprime el modo MM=0 en los pocos casos en los que funciona mejor que el modo MM=3. Vimos que esos casos se dan en los datasets con pocos o ningún gap, y la diferencia en las tasas de compresión es mínima. En cambio, cuando hay gaps en los datasets, la diferencia relativa de rendimiento a favor del modo MM=3 es mayor. Escribir un párrafo con dicho análisis, incluyendo alguna gráfica como ejemplo.

- Agregar tabla con resumen de los datasets - ver AVANCES / DUDAS (13)

- Poner las gráficas horizontales, 3 arriba y 3 abajo.

- Mencionar que CoderSlideFilter no tiene en cuenta el parámetro con el tamaño máximo de la ventana.

TODO INFORME:

- Vimos que en los datasets sin gaps, en general para todas las combinaciones <tipo de dato, algoritmo> la diferencia relativa no crece al aumentar el umbral de error. Escribir un párrafo explicando el por qué de este comportamiento.

- Mencionar experimentos ventana local vs ventana global. (ver minuta de la reunión del lunes 10/06/2019).

- Mencionar relación de compromiso entre el umbral y la tasa de compresión: al aumentar el umbral mejor la tasa de compresión (lógico).

- Agregar tabla con resumen de los algoritmos.

- Subir todo el material complementario en un link (después referirlo en el informe)

TODO CÓDIGO:

- Para los experimentos sin máscara no se están considerando los datos para los algoritmos CoderFractalRestampling y CoderSlideFilter.

- Agregar tests para MM=3.

- Al ejecutar los algoritmos GAMPS/GAMPSLimit sobre el dataset de "El Niño" (546 columnas) tengo problemas de memoria en Ubuntu, pero no en la Mac.

- Universalizar algoritmo

- Modificar GAMPS/GAMPSLimit para que utilice floats (4 bytes) en vez de doubles (8 bytes). De todas maneras, no creo que esto cambie los resultados de manera significativa, ya que aun si la cantidad de bits utilizados al codificar con GAMPS/GAMPSLimit fuera la mitad, en ningún caso superaría la tasa obtenida con el mejor codificador.