Final Project

Data Compression – Comparing Haar Algorithms

Liel Berniker, Dvir Segal, Noy Osi

13/10/2021 — 28/07/2022

Contents

1 Vision statement

1.1 Project goal

Our project goal is to compare some data compression algorithms, including Elias Gamma, Elias Delta and Fibonacci codes. The project core is to compare between Elias Gamma and Even-Haar-Transform algorithms.
Generally, the purpose of the project, relates to solving the problem of data compression in the optimal way. For example: an algorithm of data compression in as few bits as possible, so that it can be decoded by UD form, improving existing algorithms, etc.
In our learning process, we will first want to compare the 2 algorithms mentioned above. In order to compare between them, we want to research each of the algorithms deeply. In addition, we read few articles on the subject - such as the article "Adapting the Haar Transform to Natural Number Sequence Compression".
First, we will delve into the examples in Table 1, which compares between many algorithms. As part of our algorithm comparison process, we'll implement certain algorithms in C/ C++ for illustration, so we can compare run times, and also generally illustrate how the algorithm works.

1.2 Project designation

Our project is part of an article prepared by Professor Dana Shapira for an academic journal. This project could help anyone who is involved in data compression and wants to expand horizons in the field.

2 SRD

The main purpose of our project is to compare between two data compression algorithms which mention in the article "New Compression Schemes for Natural Number Sequences". In order to achieve the main purpose, we need to fulfill the following sub-goals:

- Review the article to fully understand the different Haar algorithm examples.
- Observe and understand each example in figure 1 at page 5. This figure, display different types of data compression encoding that will aid us to encode the Haar algorithms.
- Delve into algorithm 1 "Integer-Haar" and algorithm 3" New-Transform" in order to compare between their efficiency later.
- Implement algorithm 1 and 3 in CPP. We'll run the algorithms with different examples and then compare: the number of bits that required for encoding in each algorithm. The encoding in CPP will be in integer and not in bits for our convenient.
- Implement the Inverse algorithms for algorithm 1 and 3 in CPP.
- Collect all the data and summarize it for deep understanding the differences between the two algorithms.

Generally, after we'll carry out all the sub goals, we'll have a better understanding of the main purpose in our research project.

3 Inverse algorithms

We coded the inverse algorithms from the article, which transforms the encoding array into the previous array. We implement both the Inverse-Integer-Haar and the Inverse-New-Transform.

Algorithm 2: *Inverse-Integer-Haar*

INVERSE-INTEGER-HAAR$(k, h_1, \ldots, h_n)$ // n is a power of 2

1 **for** $i \leftarrow 1$ **to** 2^{k-1} **do**

2 $\quad a_{2i-1} \leftarrow h_i + \left\lceil \frac{h_{i+2^{k-1}}}{2} \right\rceil$

3 $\quad a_{2i} \leftarrow h_i - \left\lfloor \frac{h_{i+2^{k-1}}}{2} \right\rfloor$

4 **if** $2^k = n$ **then**

5 $\quad$ **return** $(h_1, \ldots, h_n)$

6 **return** INVERSE-INTEGER-HAAR$(k+1, a_1, \ldots, a_{2^k}, h_{1+2^k}, \ldots, h_n)$

k	1	2	3	4	5	6	7	8
	1640	499	135	61	54	52	6	3
1	1890	1391	135	61	54	52	6	3
2	1958	1823	1422	1361	54	52	6	3
3	1985	1931	1849	1797	1425	1419	1363	1360

Figure 4: *Inverse Integer Haar Example.*

Figure 1: Haar-Integer

Algorithm 4: *Inverse-New-Transform*

INVERSE-NEW-TRANSFORM$(k, h_1, \ldots, h_n)$ // n is a power of 2

1 **for** $i \leftarrow 1$ **to** 2^{k-1} **do**

2 $\quad a_{2i-1} \leftarrow \left\lceil \frac{h_i}{2} \right\rceil + h_{i+2^{k-1}}$

3 $\quad a_{2i} \leftarrow \left\lfloor \frac{h_i}{2} \right\rfloor - h_{i+2^{k-1}}$

4 **if** $2^k = n$ **then**

5 $\quad$ **return** $(h_1, \ldots, h_n)$

6 **return**

INVERSE-NEW-TRANSFORM$(k+1, a_1, \ldots, a_{2^k}, h_{1+2^k}, \ldots, h_n)$

Figure 2: New-Transform

BY: *Liel Berniker, Dvir Segal, Noy Osi*

4 Trial and error

Before we knew we had to do more types of coding, we focused only in Elias Delta and we wanted to check this two cases below.

4.1 Examples that Haar Integer is better than Haar New Transform

Input	Bit's Number - Integer	Bit's Number - New Tranform
8, 0	11	13
5, 3	10	12
7, 1	11	13
68, 60	17	19
120,60,50,38	37	39
140,60,31,25	39	41
37,33,29,25	26	27
1930, 1926, 1922, 1918	33	35
2110, 2046, 1982, 1918	51	53
157,123,68,52,33,29,28,22	69	72

4.2 Examples that Haar New Transform is better than Haar Integer

Input	Bit's Number - Integer	Bit's Number - New Tranform
1849,1797,1425,1419	51	48
150, 43	25	24
3100,1797,1400,1157	69	63
3100,1797,400,157	70	64
1995,1950,1934,118	63	60

BY: *Liel Berniker, Dvir Segal, Noy Osi*

5 Results

5.1 Unary coding

We want to encode series of 2^1 length, on Haar Integer and Haar New-Transform for Unary encoding. So we'll run different series in length 2 by raffle two numbers each time, 100,000 times in total (each time we select 2 numbers because we are in series of 2^1 length). We'll take the average because there are many numbers. As we increase the exponents, it'll takes a lot more bits to encode Unary for the New-Transform, than the Haar-Integer.

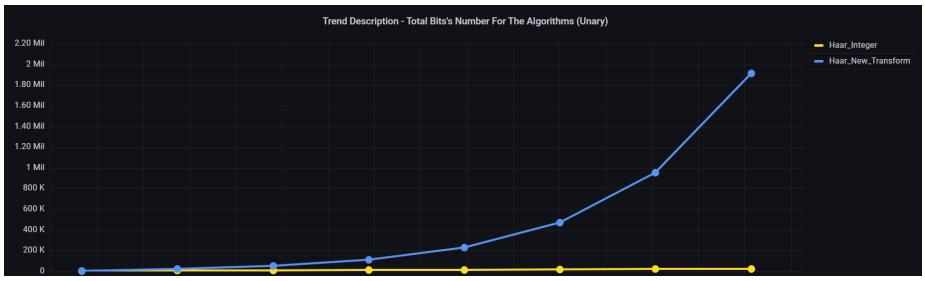


Figure 3: Unary - trend description

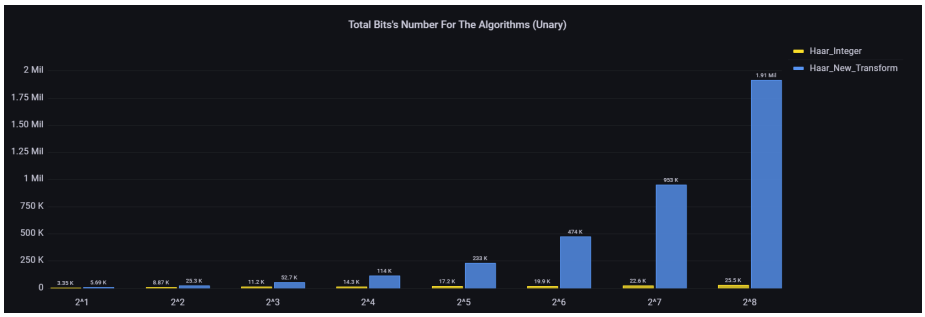


Figure 4: more precisely

The main conclusion: when we encode in Unary, we'll prefer to use the Haar Integer because we can see that it's much better when we use the various exponents. In addition, we can see that in the Haar Integer, the growth trend is linear, while the growth trend in the Haar New-Transform is exponential.

BY: *Liel Berniker, Dvir Segal, Noy Osi*

5.2 Binary coding

We can see that up to 2^6 there is almost a full equality in bits.
There may be slight changes but they are negligible on the orders of magnitude shown here.

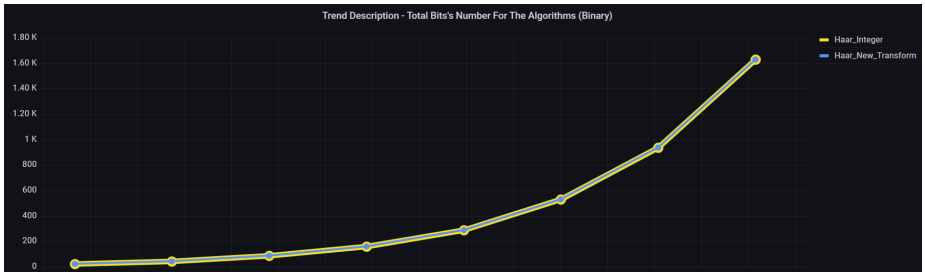


Figure 5: Binary - trend description

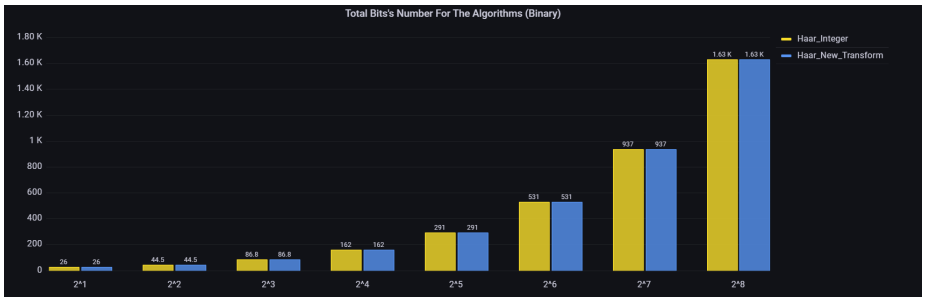


Figure 6: more precisely

5.3 Elias Gamma coding

The differences between the algorithms are negligible but exist.
Ostensibly, there is equality but as we increase the exponents, there is a slight advantage to the Haar New-Transform.

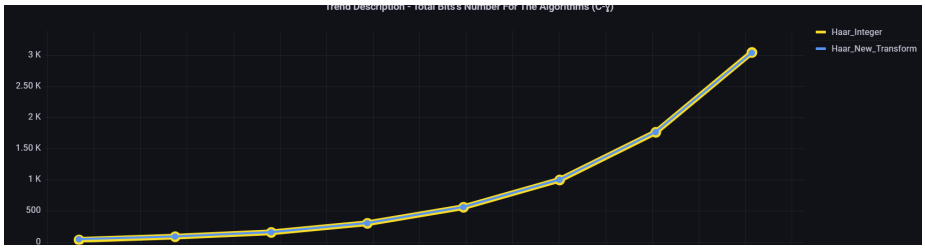


Figure 7: Elias Gamma - trend description

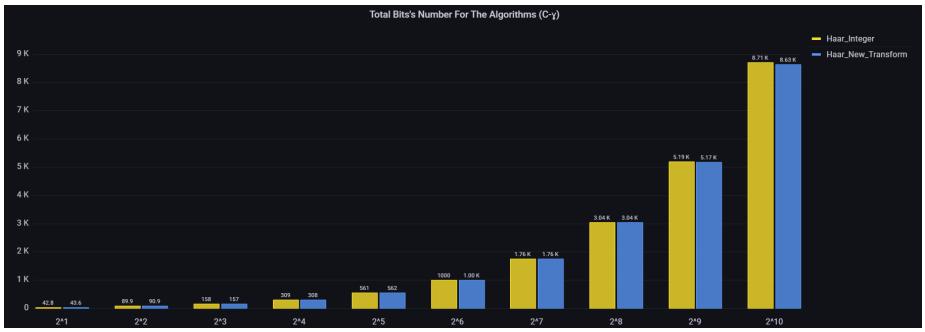


Figure 8: more precisely

5.4 Elias Delta coding

It can be seen that the trend is as we increase the exponents, the trend in both of them is exponential, when the Haar Integer is greater.

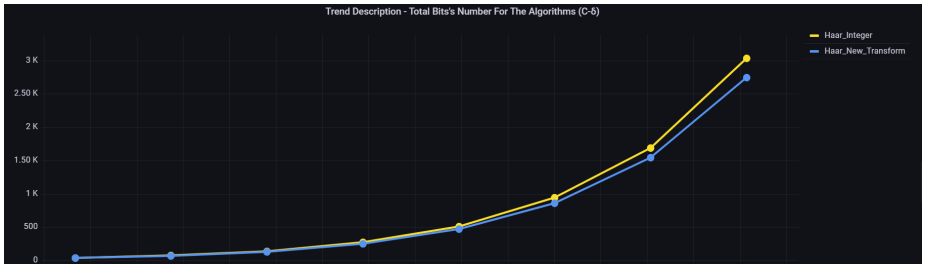


Figure 9: Elias Delta - trend description

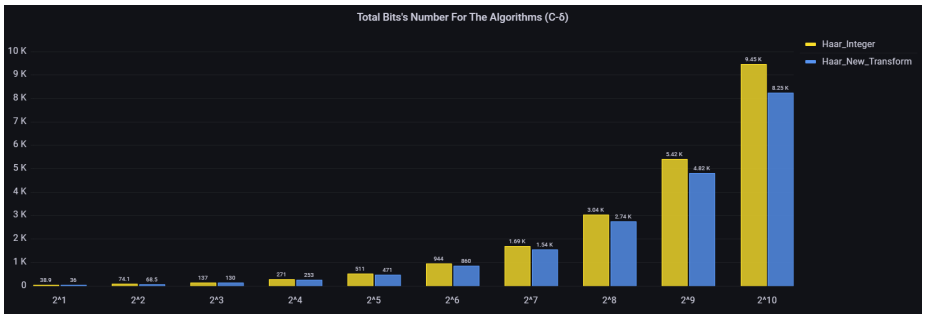


Figure 10: more precisely

5.5 Elias Gamma VS Delta coding

The differences are shown. The orders of magnitude in Unary and Binary are much bigger in relation to Elias Gamma and Delta. The Unary would reach to millions while Elias Gamma and Delta reach to big size but not that huge.

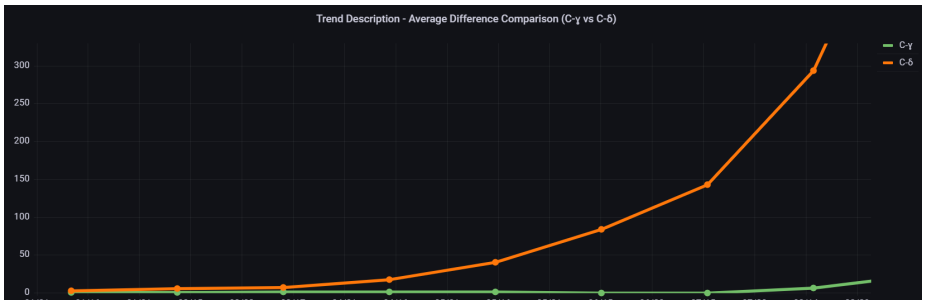


Figure 11: the saving trend between Elias Gamma and Delta

In Elias Delta, the trend is rising exponentially, while in Elias Gamma the trend is much more moderate.

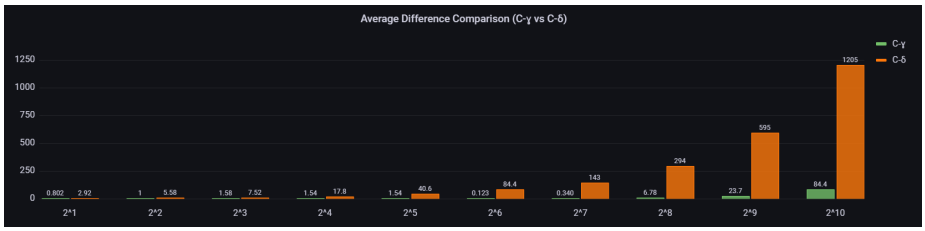


Figure 12: more precisely

5.6 Power of 2

The encoding are shown. Usually the Elias Delta takes the least bits (the green part). We have unequivocally seen that the trend in Elias Delta is repeating and that Elias Delta is always the most efficient. In addition, we can notice that while the power is increase, the ratio stay the same, and when we look at the 10 pictures, we can notice the trend that for 10 powers (from a series of lengths 2 to 1024 numbers) we can see the distribution of how many bits more or less need in each encoding.

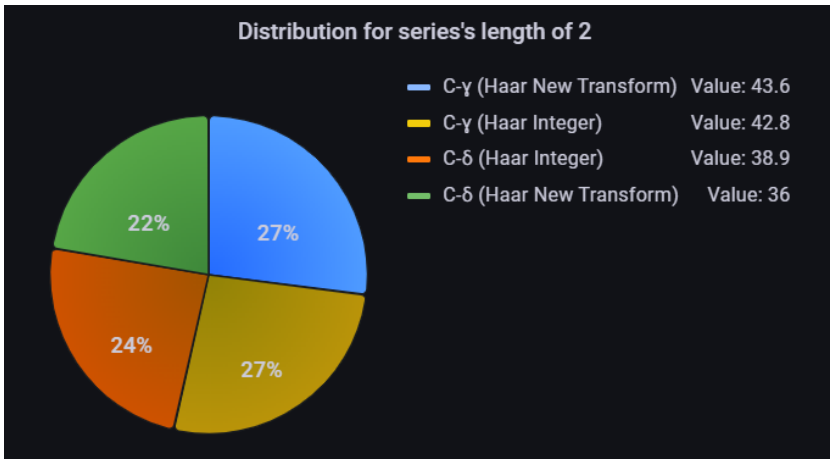


Figure 13: 2^1

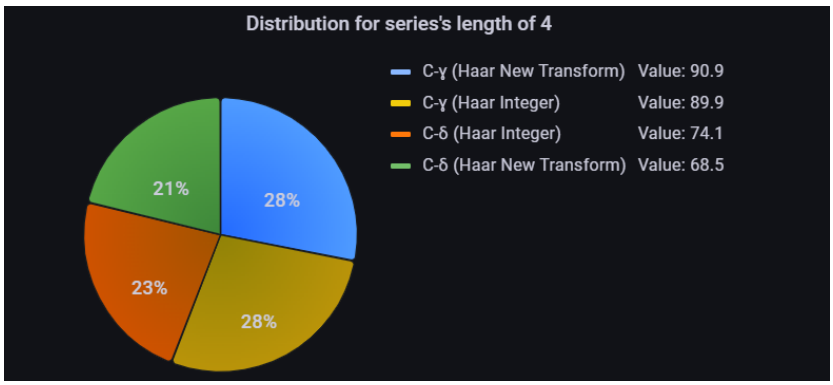


Figure 14: 2^2

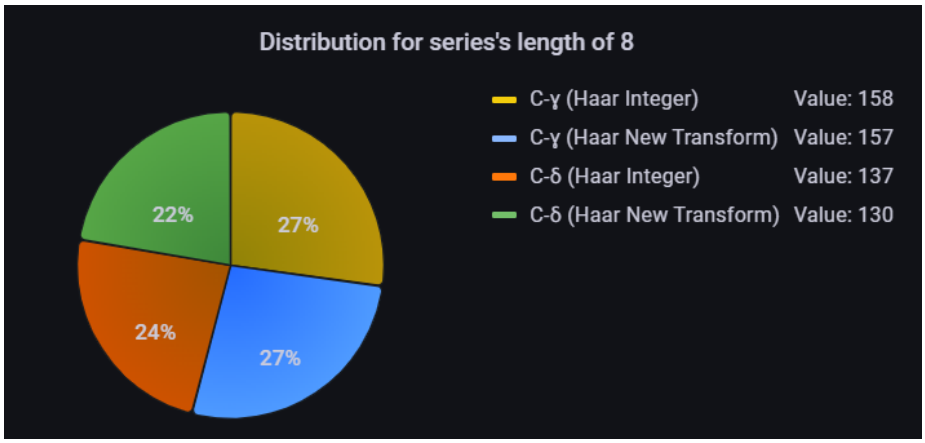


Figure 15: 2^3

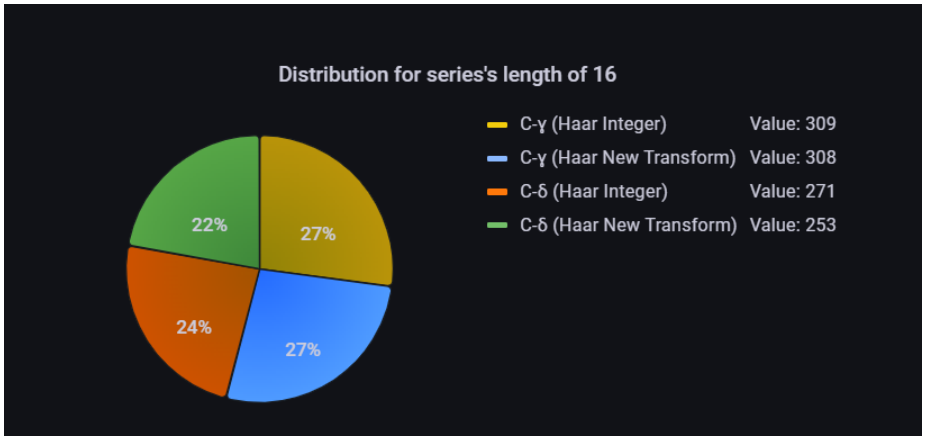


Figure 16: 2^4

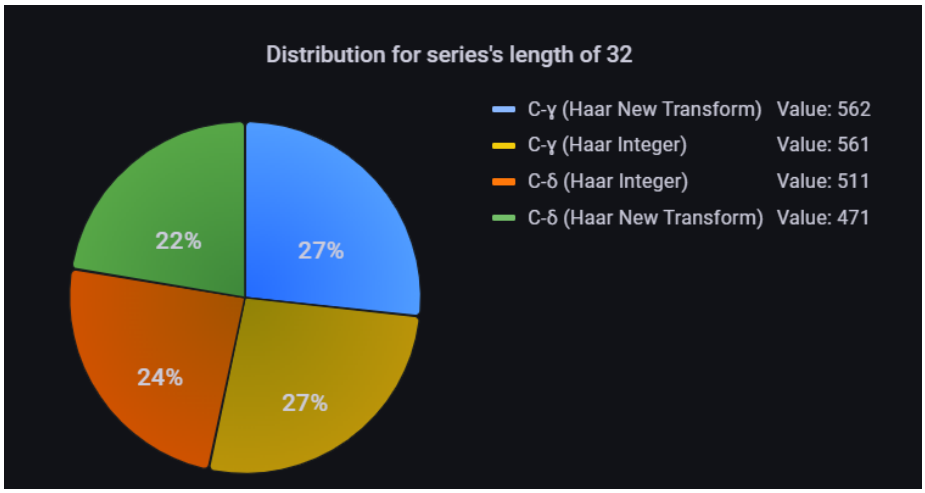


Figure 17: 2^5

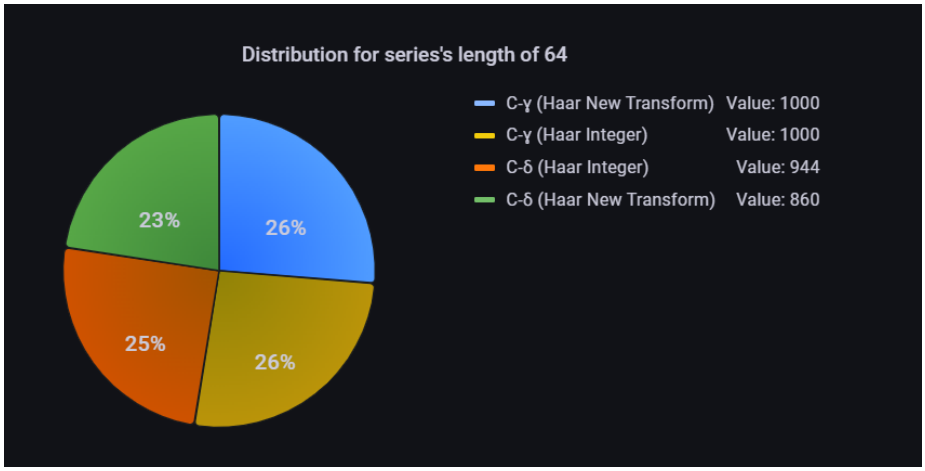


Figure 18: 2^6

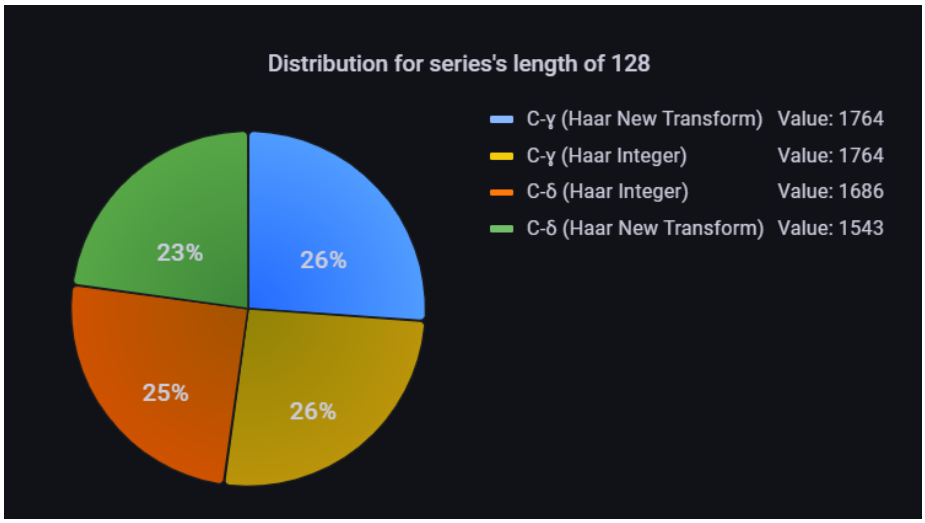


Figure 19: 2^7

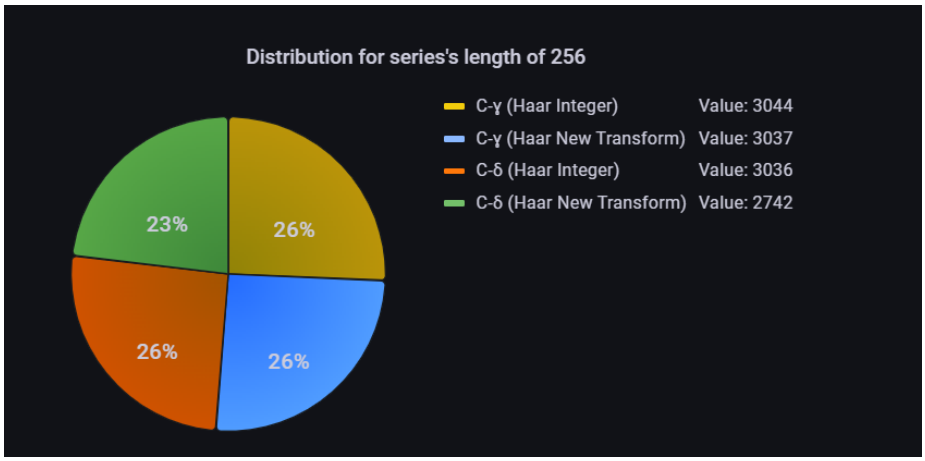


Figure 20: 2^8

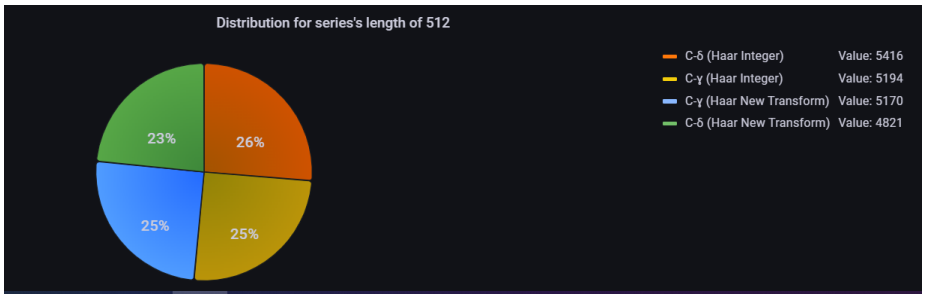


Figure 21: 2^9

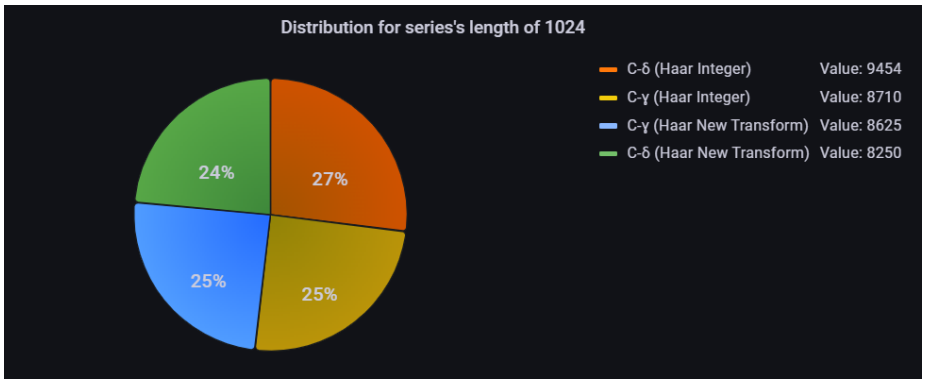


Figure 22: 2^{10}

In conclusion: as we increase the power, elias delta will be the most efficient encoding, even though the relativity is still maintained because pretty much it's always around 25%.

5.7 Fibonacci coding

It can be seen that as we increase the exponents, the Haar New-Transform is much better than the Haar Integer.

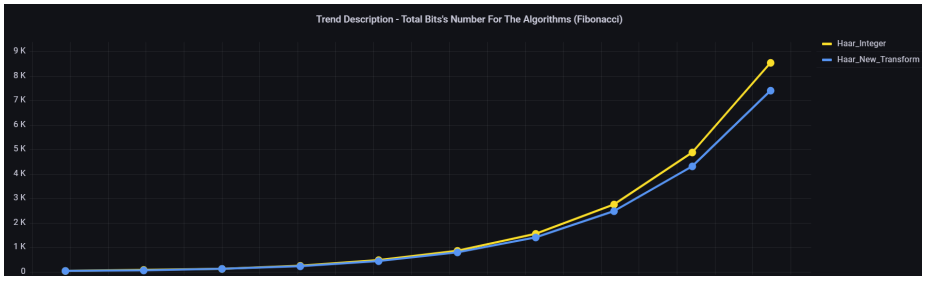


Figure 23: Fibonacci - trend description

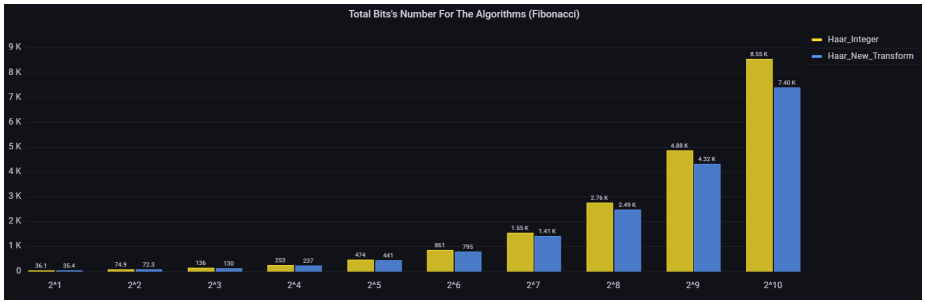


Figure 24: more precisely

5.8 Fibonacci VS Elias Gamma VS Delta coding

We can notice that Fibonacci coding with Haar New-Transform is the most efficient coding and we get the best results (it's also discussed in the article).

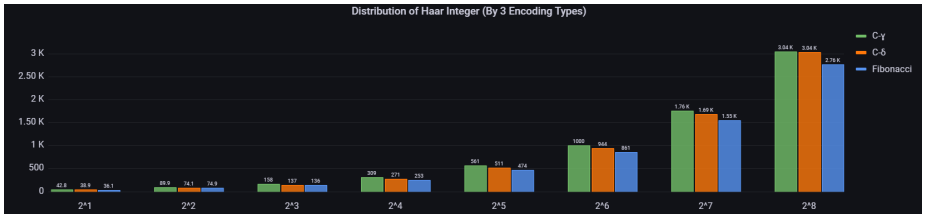


Figure 25: Haar Integer

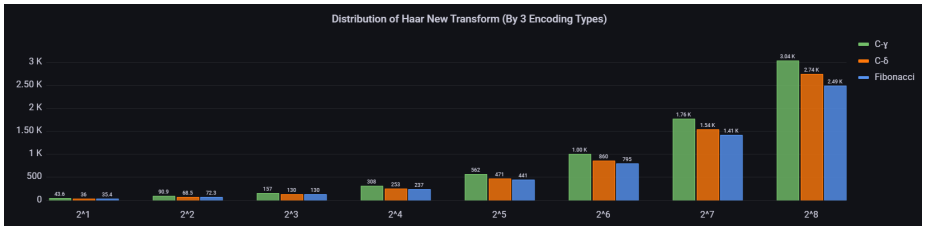
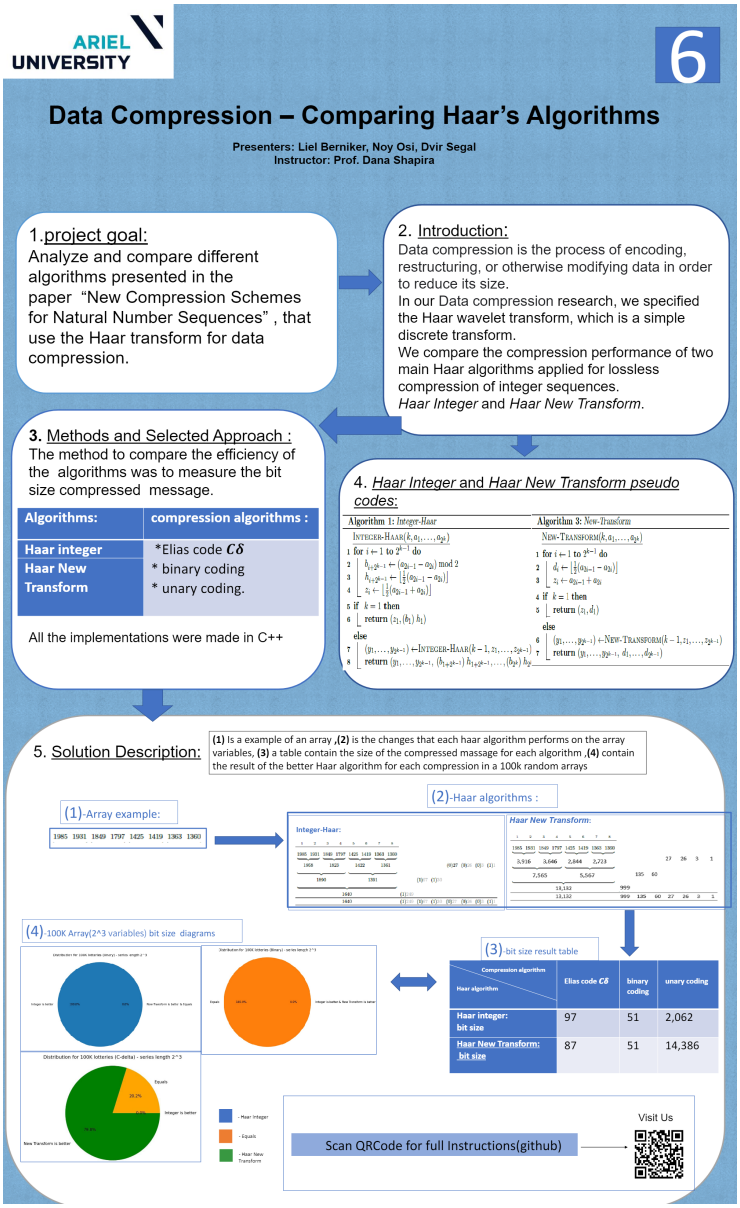


Figure 26: Haar New Transform

6 Our Poster

On June 9th, we had projects presentation day. During this day, we showed this following poster and we presented our project to all the faculty members of the computer science department and the students.



BY: *Liel Berniker, Dvir Segal, Noy Osi*