Compression of large dynamic graphs

Research proposal - 4 to 6 months

November 9, 2024

Supervisors	Binh-Minh Bui-Xuan[1], Researcher (LIP6, NPA)
	Mehdi Naima[2], Assistant professor (LIP6, ComplexNetworks)
Mail	mehdi.naima@lip6.fr
	Binh-Minh.Bui-Xuan@lip6.fr
Location	Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
Address	Campus Pierre et Marie Curie, 4 Place Jussieu, 75005 Paris

Keywords	Graph algorithms, Graph compression, Streaming algorithms

1 Context

Graph compression refers to the application of data compression techniques specifically adapted to graph structures.

Lossless graph compression aims to represent a graph more compactly, while preserving all its information. This can be particularly important in areas such as the storage and transmission of large graphs.

On the other hand, *link streams* [5, 3] refer to the manipulation, analysis and representation of edges when data arrives continuously and in real time, rather than being stored in its entirety before being processed. This is particularly relevant in areas where data is generated and updated continuously, such as social networks, IoT sensors, surveillance systems, etc. See Figure 1 for an example.

Figure 1: A link stream. The x-axis represents time, while the y-axis represents nodes. Edges arrive between nodes over time.

The Huffman algorithm is a lossless data compression algorithm invented by David A. Huffman in 1952 [2, 6]. This algorithm is widely used to compress

[1] www-npa.lip6.fr/~buixuan
[2] busyweaver.github.io

files, and is notably used in popular file formats such as GIF, ZIP, and MP3 (for lossless compression).

Huffman algorithm has two main variants a **static** one and an **adaptive** one. Both the static and adaptive versions rely on constructing a tree that assigns a code to each symbol and is such that symbols with higher frequencies are placed near the root of the tree, while those with lower frequencies are placed further away. This makes it possible to give shorter codes to the most frequent symbols.

The static algorithm begins by analyzing the input file and determines the frequency of appearance of each symbol and then build an optimal tree of the overall file. See Figure 2 for an example. Therefore the compressor needs to send both the tree and the encoded text to the decompressor.

On the other hand, the adaptive version does not need to read the whole file and can process data from a flow and send the compressed data to the decompressor on the fly. This is interesting especially for dynamic graphs where edges come in a stream and sometimes it is unknown when the stream ends but we still want to send the data in a compressed way especially to reduce communications costs.

(a)

Character		a	b	d	e	f	h	i	k	n	o	r	s	t	u	v
Frequency	9	5	1	3	7	3	1	1	1	4	1	5	1	2	1	1

(b)

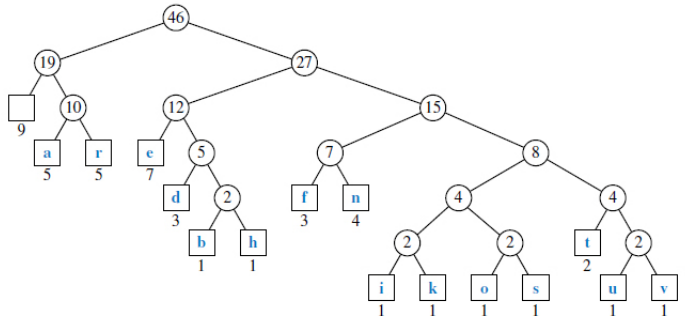

Figure 2: Symbols and their frequencies are shown at the top. Bottom: the Huffman tree built from symbols and frequencies. Figure taken from [1].

2 Objectives

We want to adapt the Dynamic Huffman algorithm for the compression of link streams. First, we want to adapt the Dynamic Huffman algorithm to graphs and see what changes need to be made to the classic text compression algorithm.

Assess and evaluate the gain in doing Huffman graph compression versus the classical Huffman versions of data compression.

Secondly, we aim to build an efficient implementation of this new algorithm so that we can test it on real-world dynamic graphs or synthetic dynamic graphs

and evaluate its performance in comparison with other algorithms of the literature [4].

Finally, we want to have a theoretical analysis of the performance of our proposed algorithms in terms of average time and space complexities.

3 Research Plan

- Literature review on dynamic graph compression.
- Algorithm design of specialized version of Huffman adaptive to dynamic graphs.
- Show the advantage of this specialized version over the classical one.
- Compare our results with known algorithms on dynamic graphs.

References

[1] massivealgorithms.blogspot. `https://massivealgorithms.blogspot.com/2014/06/greedy-algorithms-set-3-huffman-coding.html`, 2008.

[2] Donald E Knuth. Dynamic huffman coding. *Journal of algorithms*, 6(2):163–180, 1985.

[3] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8:1–29, 2018.

[4] Panagiotis Liakos, Katia Papakonstantinopoulou, Theodore Stefou, and Alex Delis. On compressing temporal graphs. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 1301–1313. IEEE, 2022.

[5] Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.

[6] Jeffrey Scott Vitter. Design and analysis of dynamic huffman codes. *Journal of the ACM (JACM)*, 34(4):825–845, 1987.