Proposal for the Master's Thesis Extended grammar-based graph compression

Matthias Dürksen

November 23, 2018

1 Motivation

Nowadays, there is more and more data, some of it is stored in graphs. The graphs often become very large and therefore require a lot of memory. To solve this problem, you can compress the graph in order to reduce the memory requirements. In this thesis a grammar-based graph compression is used. An approach to grammer-based compression for graphs was already invented by me in a previous work [1].
This concept [1] has already been extended in several bachelor theses. There is a recompression for already compressed graphs [5] and also an efficient coding [3].
Since there are only two digram types so far and the further work only supports one type, it limits the coding efficiency. Therefore, the aim of this paper is to examine how other digram types can improve compression.

2 Fundamentals

Graph compression supports graphs with nodes and edge labels. With my approach, the graph is first converted. The result of the conversion is a graph with only node labels and no edge labels, which facilitates compression.
After the conversion comes the transformation. Hereby, patterns (which are called digrams) are searched and the most frequent pattern/digram is replaced by a nonterminal symbol. So far there are two types of digrams we can distinguish between, the basic and the adjacency digrams. A basic digram (Fig. 1) consists of two adjacent nodes and the edge that connects these nodes. Thus, two adjacent nodes and the incident edge can be replaced by one nonterminal symbol.
The adjacency digram (Fig. 2) is formed by two nodes which both have a common adjacency node x. Thus, the adjacency digram contains the two nodes and the two edges that are incident to the node x. This digram can be replaced by a nonterminal

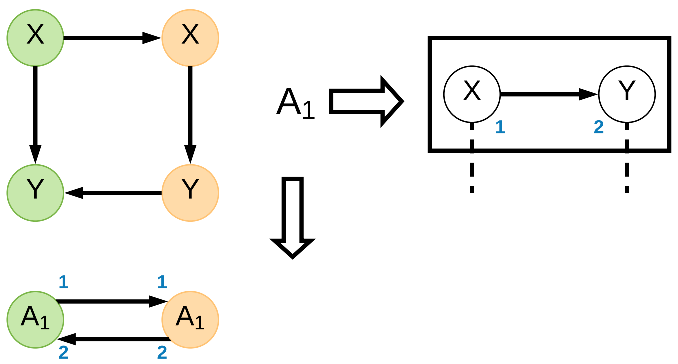

Figure 1: Application of a replacement of a basic digam

and an edge from the nonterminal to the node x. Thus, the possibilities of replacement by the adjacency digram are significantly increased.

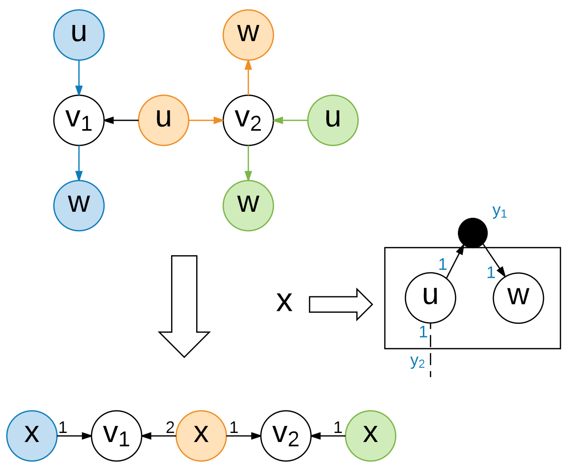

Figure 2: Application of a replacement of an adjacency digam

After the search for the most frequent digram, all non-overlapping occurrences are replaced by the nonterminal. Then, the search for the most frequent digram and the replacement of the occurrences is continued until there are no more digrams which occur frequently enough.

3 This Thesis

The research will investigate which other types of digrams can be created that have a positive effect on compression. One goal is to test whether the compression approach of Maneth [4] can be simulated by this graph compression. Furthermore it shall be analyzed whether digram types, which

become edges when replacing the occurrences, are compatible with the other types of digrams and how such digram types can look like. If there is still time for more research and development, these digrams could be integrated into the further work. One of these works is the efficient coding of Georgi [3], so that the compressed graph can be stored efficiently in memory. These compressed graphs and the corresponding grammar rules can be coded on the basis of Georgi's work [3] and thus save the graph in a memory efficient way. Furthermore, the works of Frerk [2] and Werneke [5] have developed an extension for the recompression of graphs. This method can be used if, for example, two subgraphs were compressed individually and these graphs are compressed together. This method combines the two graphs and checks whether the grammar rules can be chosen more efficiently in the combination of the two graphs than to combine the previous rules. This method can also be used if an existing graph has been modified and thus has new nodes or old nodes have been removed. Thus, it is checked again whether there is now an efficient compression. Frerk's approach is based on the graph model and Werneke's methods work on memory level, so that Georgi's coding is considered here. The work of Georgi [3] and Werneke [5] only supports graph compression, which does not replace adjacency digrams. Thus, the adjacency digrams and the newly defined digram types must be integrated into the work.

References

[1] Matthias Duerksen. “Grammatik-basierte Graphkompression”. Bachelor’s Thesis. University of Paderborn, 2016.

[2] Philip Frerk. “Grammatik-basierte Rekompression angewendet auf Graphen”. Bachelor’s Thesis. University of Paderborn, 2017.

[3] David Georgi. “Kompakte Kodierung für grammatikbasierte Graphenkompression”. Bachelor’s Thesis. University of Paderborn, 2017.

[4] Sebastian Maneth and Fabian Peternek. “Compressing Graphs by Grammars”. English. In: *Proceedings of the 32nd International Conference on Data Engineering – ICDE 2016*. IEEE, June 2016. DOI: `10.1109/ICDE.2016.7498233`.

[5] Lars Werneke. “Grammatikbasierte Rekompression von Graphen”. Bachelor’s Thesis. University of Paderborn, 2018.