## Loss-Less Network Compression

We live in a world, where networks grow exponentially. Not only communication networks, like Internet, but computation networks as well, like Large Language Models (LLM). For example, only Internet of Things (IoT) is responsible for more than 50 Billion devices connected to Internet today, and the latest Large Language Models have exceeded the size of 1 Trillion parameters, with the largest one having 10 Trillion. And these numbers will only grow.

In communication networks this leads to fast growing operation and management costs, configuration complexity, challenges with network-wide optimization, scaling and reliability problems. It also results in inability to provide End-to-End (E2E) Service Level Agreement (SLA) guarantees at the access level. At the same time, growing LLMs' size makes it difficult for real-time applications and deployment in smaller devices, requires more and more Cloud resources & infrastructure for growing LLM inference request, and finally leads to growing companies' expenses.

Though the problems are different, the reason is common – extensively growing network scale. And in both cases a network can be abstracted as a graph, leading us to a seemingly long known, but still not completely resolved graph compression challenge.

It is important to mention, that communication networks are not only important for our daily life, but have become a critical infrastructure for many services. That is why, a potential solution must be *loss-less*, making this problem really challenging. The second key challenge is the requirement to design a general approach, that can be applied to different kinds of networks, not only communication, but also computing. Finally, this general loss-less compression technique must be computational friendly, that is not only reduce the size of a graph, but keeping the ability to run graph algorithms on the compressed graph without loss of accuracy.

We consider 3 key scenarios to use this graph compression technology:

1. Efficient large-scale network topology transmission:
    a. an unweighted or weighted graph has to be compressed to reduce the amount of information being transmitted;
    b. *loss-less compression* here means, that a receiver can reconstruct the original topology exactly;
    c. graph pattern mining or similar techniques can be used to identify similarities;
    d. and the similarity-based coding can be used for the final compression;
2. Efficient shortest path computation in large-scale networks:
    a. a weighted undirected or directed graph has to be compressed to improve the efficiency of graph algorithm execution;
    b. *loss-less compression* here means, that after the compression a graph algorithm, for example (modified) Dijkstra can be executed on the compressed graph to achieve *exact solution* of some graph problem for the original graph;
    c. graph pattern mining or similar techniques can be used to identify similarities;
    d. the similarities can be used for computational friendly graph simplification;
    e. additional pre-processing information about removed parts of the graph can be stored;

    f. modified Dijkstra algorithm can be used on the compressed graph to compute exact shortest path in the original graph;
3. Efficient Large Language Model inference:
    a. a Large Language Model or Deep Neural Network have to be compressed to improve the inference time and resource consumption;
    b. *loss-less compression* here means, that after the compression the compressed model outputs exactly the same values as the original model, and the inference accuracy drop is 0%;
    c. graph pattern mining or similar techniques can be used to identify similarities;
    d. the similarities can be used to simplify the neural network model without affecting the computational output.

Though, the scenarios are quite different, potential approaches to solve these problems share the same critical step – graph analysis to identify similarities. Then, of course, these similarities are used in a different way to achieve the common goal – loss-less network compression.

The challenge presented here seeks to build a general graph analysis, simplification and compression technique, which can provide good enough processed graph for the following coding, graph algorithm execution or computational graph simplification without introducing error into graph restoration process, graph algorithm or computational graph output. Based on this general technique, specific graph compression, graph algorithm execution and neural network simplification solutions have to be proposed too.

Within the scope of this challenge, we are keen to gain comprehensive insights into various facets of the topic. These can encompass, but are not limited to:

1. Literature Review: This component entails a comprehensive review of existing literature, emphasizing mathematical and algorithmic strategies for loss-less, algorithm-friendly graph compression and loss-less neural network compression. The review will identify key methodologies, their strengths, weaknesses, and applications in the context of matrix modifications and updates.
2. Algorithm Development: In this phase, the objective is to devise and implement an innovative algorithm that identifies similarities in unweighted, weighted, undirected and directed graphs, such that they can be efficiently used then for graph coding, algorithm-friendly graph compression, loss-less neural network compression.
3. Performance Evaluation: This segment necessitates a rigorous evaluation of the developed algorithms' performance. It involves benchmarking the computational efficiency, expressed through time complexity metrics, and assessing its numerical values.
4. Loss-less performance guarantees: The final task centers on theoretically proving, that the designed algorithms provide loss-less performance for all three key scenarios.

We would greatly appreciate any insights, recommendations, or innovative approaches that could assist in addressing this challenge. Your expertise and contributions could significantly enhance our collaborative efforts, paving the way for deeper exploration and fruitful future partnerships.

## References


1) A. Francisco, T. Gagie, S. Ladra and G. Navarro, "Exploiting Computation-Friendly Graph Compression Methods for Adjacency-Matrix Multiplication," 2018 Data Compression Conference, Snowbird, UT, USA, 2018, pp. 307-314.
2) Max Bannach, Florian Andreas Marwitz, and Till Tantau. Faster Graph Algorithms Through DAG Compression. In 41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024). Leibniz International Proceedings in Informatics (LIPIcs), Volume 289, pp. 8:1-8:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2024).
3) Yin, H., Clegg, R.G. & Mondragón, R.J. Simplification of networks by conserving path diversity and minimisation of the search information. Sci Rep 10, 19150 (2020).
4) Toivonen, H., Mahler, S., Zhou, F. (2010). A Framework for Path-Oriented Network Simplification. In: Cohen, P.R., Adams, N.M., Berthold, M.R. (eds) Advances in Intelligent Data Analysis IX. IDA 2010. Lecture Notes in Computer Science, vol 6065. Springer, Berlin, Heidelberg.
5) Dennis Mensah, Hui Gao, and Liangwei Yang. Approximation algorithm for shortest path in large social networks. Algorithms, 13:36, 02 2020.
6) Martin Holzer, Frank Schulz, and Dorothea Wagner. Engineering multi-level overlay graphs for shortest-path queries. ACM Journal of Experimental Algorithmics, 13, 09 2008.
7) Navlakha, Saket & Rastogi, Rajeev & Shrivastava, Nisheeth. (2008). Graph summarization with bounded error. SIGMOD. 419-432.
8) Lee H, Shin B, Choi D, Lim J, Bok K, Yoo J. Graph Stream Compression Scheme Based on Pattern Dictionary Using Provenance. Applied Sciences. 2024; 14(11):4553.
9) Navlakha, Saket & Rastogi, Rajeev & Shrivastava, Nisheeth. (2008). Graph summarization with bounded error. SIGMOD. 419-432.
10) Lee H, Shin B, Choi D, Lim J, Bok K, Yoo J. Graph Stream Compression Scheme Based on Pattern Dictionary Using Provenance. Applied Sciences. 2024; 14(11):4553.