



Zürich, 9. September 2026

MSc Thesis (30 points)

Topic: Partitioning Large Directed Acyclic Graphs into Deep and Narrow Parts

Background and Motivation

Directed acyclic graphs (DAGs) are widely used to model dependencies, for example commit histories in version control systems. Two fundamental query types on such graphs are reachability queries, which ask whether one node can reach another, and partial sources queries, which ask, for a given subset S of nodes, for all nodes of S that are not reachable from any other node of S . How hard these queries are depends strongly on the shape of the DAG. We define the depth of a node as the length of the longest path from any source, i.e. any node without incoming edges, to it, and the depth of the DAG as the maximum depth of its nodes. The width of the DAG at depth d is the number of nodes at depth d plus the number of edges passing through depth d , i.e. edges starting at a depth smaller than d and ending at a depth larger than d ; the width of the DAG is the maximum width over all depths. Partial sources and reachability queries can be answered much more efficiently on narrow DAGs than on wide ones. In [3] the ratio of depth to average width is used to measure how narrow a DAG is, and is evaluated on 2,835 real repositories, all GitHub repositories with more than 10,000 stars. Size and depth turn out to be strongly correlated (Pearson $r = 0.99$ on the logarithms), so a growing repository grows into depth rather than width; the median depth is 1,629 while the median average width is just above 2, and in half of the repositories the depth exceeds the average width by a factor of 600 or more. This is no accident: a repository records the evolution of one code base over time, so its commit DAG is essentially a timeline with a few parallel branches that keep converging back. The algorithms of [3] work well precisely because of this shape.

Repositories, however, do not always stay separate. A fork shares the history of its origin and continues it on its own, and merges between related repositories connect their histories further, so the combined commit graph of such a group no longer decomposes into isolated weakly

connected components. Such a combined graph appears wide, but its width is only apparent: the edges connecting the individual histories are sparse, and cutting the graph at these forking and merging points would recover the narrow and deep histories of the single repositories. The idea of this thesis is to capture exactly this situation: partition the combined graph into deep and narrow parts by cutting through the sparse connections, handle each part with the methods above, and treat the remaining edges between parts explicitly. The fewer edges run between parts, the cheaper this last step becomes.

Graph partitioning is a classical problem with a rich body of methods. The multilevel scheme of Karypis and Kumar [5], implemented in the METIS library, is the standard approach and will serve as the baseline of this thesis.

Problem Statement

Classical graph partitioning computes parts of roughly equal size while minimizing the number of edges cut, and it usually treats the graph as undirected. This is not what we want: the direction of the edges matters, the shape of the parts matters, and balance is secondary. Given a DAG $G = (V, E)$, we seek a partition of V into parts $V_1, \dots, V_k$ such that (i) each induced subgraph $G[V_i]$ is deep and narrow, i.e. has a high ratio of depth to width, both measured within $G[V_i]$, and (ii) the number of edges with endpoints in different parts is small. How the shape of the parts and the number of edges between parts are traded off against each other in a single quality measure is part of the problem. It is open how well the established partitioning methods perform under such an objective, and whether the particular structure of commit histories allows a custom method to do better.

Proposed Approach

The goal of this thesis is to systematically evaluate the best-known graph partitioning methods under this new objective and to design a custom method for commit histories. We propose the following main steps:

1. **Baseline and Survey:** Implement an evaluation pipeline and measure the partitions produced by the multilevel scheme of Karypis and Kumar [5] on real commit graphs under the new quality measures. The real commit graphs are the commit histories of the Linux kernel, Kubernetes, and Magento used in [3], and the combined commit graphs of fork networks, for example the forks of Git, as recorded in the Software Heritage archive [6]. Extend the comparison to the most prominent alternatives, including multilevel partitioners with strong refinement [7], streaming partitioners [8], and acyclic partitioning methods designed specifically for DAGs [4]; the survey of Buluç et al. [2] serves as the map of the field. Since these methods optimize balance and cut rather than part shape, the outcome of this study is a genuine open question and a contribution in its own right.
2. **Quality Measures:** Formally define the quality of a partition in terms of the depth and width of its parts and the number of edges between parts, and establish elementary relationships between these quantities, for example how the width of the input graph limits what any partition can achieve.
3. **Custom Method:** Commit histories have a distinctive structure: the vast majority of nodes have exactly one predecessor and one successor, so the graph decomposes into long straight segments, while branching and merging events are comparatively rare and local [1]. A custom method can exploit this, for example by growing parts along these segments up to a size or width threshold instead of coarsening arbitrary matchings, or

by replacing the refinement objective in a multilevel scheme with a width-based one. The thesis will design, implement, and analyze at least one such method.

Primary Goals and Tasks

The project will focus on the following core objectives:

1. **Survey and Baselines:** Systematically evaluate METIS and the most prominent alternative partitioning methods on real commit graphs under the new measures.
2. **Formalization:** Define the partition quality measures based on the depth and width of the parts and the number of edges between parts, and prove elementary bounds where possible.
3. **Algorithm Design:** Develop a custom partitioning method that exploits the structural properties of commit histories.
4. **Implementation:** Implement the custom method and the evaluation pipeline.
5. **Evaluation:** Compare the custom method against the baselines on commit histories of individual repositories and of fork networks, analyzing part shape, cut size, running time, and the effect on the size and query performance of the indexes built per part.

Project Schedule (26 Weeks)

The thesis is planned to be completed over a 26-week period with the following tentative schedule:

- **Weeks 1–3:** Literature review, familiarization with the supervisor's previous work, implementation and understanding of the multilevel partitioning scheme.
- **Weeks 4–8:** Setup of the evaluation pipeline and baseline study: evaluation of METIS and the alternative methods on real commit graphs.
- **Weeks 9–11:** Formalization of the quality measures.
- **Weeks 12–17:** Design and implementation of the custom partitioning method.
- **Weeks 18–21:** Empirical evaluation and comparison against the baselines.
- **Weeks 22–26:** Finalizing the analysis, writing the thesis, and preparing the final presentation.

University of Zurich
Department of Informatics

Prof. Dr. Michael Böhlen
Professor

Supervisor: Mateo Dujić (dujic@ifi.uzh.ch)

Start date: September 7, 2026

End date: March 7, 2027

References

- [1] Christian Bird, Peter C. Rigby, Earl T. Barr, David J. Hamilton, Daniel M. German, and Prem Devanbu. The promises and perils of mining git. In *Proceedings of the 6th International Working Conference on Mining Software Repositories (MSR)*, pages 1–10, 2009.
- [2] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. In *Algorithm Engineering: Selected Results and Surveys*, volume 9220 of *Lecture Notes in Computer Science*, pages 117–158. Springer, 2016.
- [3] Mateo Dujic, Michael Böhlen, Sven Helmer, and Oded Lachish. Determining file provenance in directed acyclic graphs using partial sources, 2026. Under submission.
- [4] Julien Herrmann, M. Yusuf Özkaya, Bora Uçar, Kamer Kaya, and Ümit V. Çatalyürek. Multilevel algorithms for acyclic partitioning of directed acyclic graphs. *SIAM Journal on Scientific Computing*, 41(4):A2117–A2145, 2019.
- [5] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
- [6] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. The Software Heritage graph dataset: Public software development under one roof. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR)*, pages 138–142, 2019.
- [7] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In *Proceedings of the 19th European Symposium on Algorithms (ESA)*, volume 6942 of *Lecture Notes in Computer Science*, pages 469–480. Springer, 2011.
- [8] Charalampos Tsourakakis, Christos Gkantsidis, Bozidar Radunovic, and Milan Vojnovic. FENNEL: Streaming graph partitioning for massive scale graphs. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 333–342, 2014.