Efficient Identification of Maximum Common Induced Subgraphs for Brain-Computer Interface Accelerator Design

Bryan SebaRaj
Yale University
bryan.sebaraj@yale.edu

Advisor: Quanquan Liu
Yale University
quanquan.liu@yale.edu

Advisor: Abhishek Bhattacharjee
Yale University
abhishek@cs.yale.edu

Advisor: Rajit Manohar
Yale University
rajit.manohar@yale.edu

September 11, 2025

Introduction and Background

Brain-Computer Interfaces (BCIs) represent a frontier in medical technology, offering pathways to restore sensory and motor functions for individuals with severe neurological disorders [1]. Clinical applications have demonstrated success in controlling prosthetic limbs, enabling communication, and even restoring a sense of touch [2]. However, the practical, long-term implementation of fully implanted BCIs is severely constrained by the need for low-power, high-efficiency computation. The biocompatibility of implanted devices demands minimal heat dissipation to prevent damage to surrounding neural tissue, placing strict limits on power and energy consumption [3, 4].

Many BCI workloads, such as neural signal processing and data interpretation, can be modeled as computational directed acyclic graphs (CDAGs), a practice well-established in the design of embedded systems and hardware accelerators for signal processing [5]. These workloads often involve a set of canonical algorithms, including the Discrete Wavelet Transform (DWT), Fast Fourier Transforms (FFTs), and Matrix-Vector Multiplication (MVM), each with its unique CDAG structure. Designing a custom hardware accelerator for each specific task is infeasible due to area and power constraints. A more promising approach is to identify and accelerate the computational kernels that are common across multiple BCI workloads.

By finding the largest computational structure shared among the CDAGs of these different algorithms, we can design a single, minimal, and canonical hardware accelerator. This accelerator would efficiently execute the common operations, maximizing hardware reuse and minimizing power consumption. Formally, identifying the largest shared structure is known as the Maximum Common Induced Subgraph (MCIS) problem. This problem is a variant of the subgraph isomorphism problem, a classic computational challenge with applications ranging from pattern recognition to bioinformatics [6]. This thesis proposes to develop a testing framework to efficiently identify the MCIS for BCI-relevant CDAGs, thereby providing a blueprint for the design of next-generation, ultra-low-power BCI hardware.

Problem Description

The central challenge is the computational complexity of the MCIS problem. Finding the maximum common induced subgraph between two graphs is NP-hard [7], and its extension to multiple graphs is at least as difficult. Consequently, brute-force solvers are computationally too inefficient for the complex CDAGs that represent realistic BCI workloads.

However, the CDAGs derived from signal processing algorithms are not arbitrary graphs, often possessing specific structural properties. For example, discrete wavelet transforms can be pruned to reveal tree-like characteristics. These properties may be exploitable, potentially allowing for the development of efficient exact algorithms for these restricted graph classes or highly effective approximation algorithms. This thesis aims to address the lack of a systematic framework to implement, test, and benchmark various MCIS algorithms specifically on BCI-centric workloads. The goal is to implement this framework and utilize it to identify common subgraphs that can serve as the basis for a unified BCI hardware accelerator.

Planned Approach

To tackle this problem, this research will follow a multi-stage approach focused on framework development, algorithm implementation, and rigorous evaluation.

1. **C++ Testing Framework Development:** The primary objective is to design and implement a robust, modular C++ library. This framework will provide data structures for representing and manipulating CDAGs and a clear interface for integrating various MCIS-finding algorithms. Its modularity will be crucial for easily adding, testing, and comparing different solvers.
2. **Ground-Truth Algorithm Implementation:** To establish a baseline for correctness and optimality, an exact MCIS solver will be implemented. This will be based on the well-established reduction of the MCIS problem to the maximum clique problem, which, while computationally expensive, will provide the baseline for evaluating other algorithms on smaller graph instances [8].
3. **Heuristic and Specialized Algorithm Implementation:** More efficient algorithms for finding common subgraphs will be implemented. This will include exploring heuristics that provide approximate solutions for general DAGs and potentially specialized algorithms that exploit the unique structure of BCI-related CDAGs, such as those for trees or level graphs.
4. **Dataset Generation and Rigorous Evaluation:** A comprehensive dataset of CDAGs representing canonical BCI workloads will be generated. This will include graphs for DWT, MVM, and other relevant signal processing tasks. The implemented MCIS algorithms will be rigorously benchmarked against this dataset, with performance measured by runtime, memory usage, and the size of the common subgraph identified.

Deliverables and Timelines

The project will culminate in a set of deliverables that document the research process and its outcomes, including the final thesis report, C++ library, and CDAG dataset.

1. **Thesis Proposal**
   - From: September 4, 2025; To: September 11, 2025
2. **Testing Framework**
   - Task: Implement the outline of a C++ library to support CDAG implementation and MCIS finding testing. Specifically implement, nodes/graphs and a robust testing/benchmarking framework using Google Test.
   - From: September 8, 2025; To: September 14, 2025
3. **Brute Force MCIS Algorithm**
   - Task: Implement a brute-force MCIS solver using maximum clique reduction.
   - From: September 15, 2025; To: September 21, 2025
4. **CDAG Dataset**
   - Task: Algorithmically construct a set of computational directed acyclic graphs which represent the optimal schedules of canonical BCI workloads, such as DWT and MVM.
   - From: September 22, 2025; To: October 10, 2025
5. **Other MCIS Algorithms**
   - Task: Implement other, more efficient MCIS solvers in C++ and evaluate for correctness.
   - From: October 11, 2025; To: October 31, 2025

6. **Midterm Report & Presentation**
    - Task: Submit a progress report detailing the C++ library and tests.
    - From: October 27, 2025; To: November 3, 2025
7. **MCIS Algorithm Benchmarks**
    - Task: Rigorously benchmark all MCIS algorithms, measuring the aforementioned data points.
    - From: November 4, 2025; To: November 24, 2025
8. **Final Deliverables Package**
    - Task: Prepare the final package including the C++ library with its respective CDAG MCIS evaluation and performance tests.
    - From: November 24, 2025; To: December 1, 2025
9. **Final Report and Poster**
    - Final Report to Advisors by December 1, 2025
    - Final Poster by December 2, 2025
    - Final Poster Presentation on December 5, 2025
    - Final Report Submission by December 11, 2025

Future Work

The completion of this thesis and its core deliverables will lay the groundwork to identify the most promising MCIS candidates and use high-level synthesis (HLS) tools to translate these CDAGs into a hardware description language like Verilog. This would enable pre-fabrication simulation and evaluation of the accelerator's power, performance, and area (PPA).

Future work can extend this to the physical realization of fabricating the designed accelerator and testing it in a real-world BCI system. This could ultimately involve integration into experimental setups for animal testing, providing definitive proof of the design's efficacy. Furthermore, the framework itself can be extended to incorporate a larger library of MCIS algorithms and a broader range of BCI workloads, establishing a comprehensive toolchain for algorithm-hardware co-design in the neurotechnology space.

References

[1] Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., & Vaughan, T. M. (2002). Brain-computer interfaces for communication and control. *Clinical neurophysiology*, *113*(6), 767-791.

[2] Hochberg, L. R., et al. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, *485*(7398), 372-375.

[3] Nurmikko, A., et al. (2010). Listening to brain microcircuits: a review of methods for chronically implanted neural interfaces. *Proceedings of the IEEE*, *98*(3), 375-388.

[4] Bhattacharjee, A., Liu, Q. C., Manohar, R., Pothukuchi, R. P., & Ugur, M. (2025). Dataflow-Specific Algorithms for Resource-Constrained Scheduling and Memory Design. In *37th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 25).*

[5] Bhattacharyya, S. S., Murthy, P. K., Lee, E. A. (2013). *Software synthesis from dataflow graphs.* Springer Science Business Media.

[6] Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, *23*(1), 31-42.

[7] Garey, M. R., Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

[8] Barrow, H. G., Burstall, R. M. (1976). Subgraph isomorphism, matching relational structures and maximal cliques. *Information Processing Letters, 4*(4), 83-84.