Parallel Execution Strategies for Causal Structure Learning in Heterogeneous Computing Systems

Motivation
Learning causal structures (CSL) in high-dimensional data promises huge potential in real-world applications. CSL supports humans in understanding highly complex systems with abundant data and allows for a causal interpretation. For example, CSL provides data-driven decision support for effective troubleshooting in manufacturing\(^4\), or CSL supports genetic research when constructing gene regulatory networks to understand disease mechanisms. However, algorithms for constraint-based CSL have high computational complexity and are currently limited by their long runtimes.

Enabling the efficient execution in heterogeneous computing systems, leveraging the parallel computing capabilities of Graphics Processing Units (GPUs) constitutes the required speed-up for application in practice.

Background
One approach to CSL is constraint-based methods, which apply conditional independence (CI) tests suitable to the underlying data distribution of the observational data. Our work focuses on the PC algorithm\(^5\) to learn the causal graphical model. The PC algorithm consists of two phases, see Fig. 1, with the adjacency search defining the overall computational complexity; thereby, it is the focus of our work.

A GPU’s parallel structure provides massive computational power and is well-suited for processing tasks exposing data parallelism. A GPU follows the SIMT execution model, meaning that multiple threads execute the same instruction on different data. The data resides in the GPU’s on-chip memory, which requires prior transfer from DRAM.

Research Questions (RQ)
(RQ1) How can we efficiently execute constraint-based CSL on a GPU, considering the SIMT execution model? In this context, a parallel execution strategy and a definition of tasks for parallel execution are required, which need to reflect a suitable granularity. Further, the characteristics of various CI tests for different data distributions have to be considered.

(RQ2) How can we scale constraint-based CSL on a GPU to very large datasets, which exceed the on-chip memory? A data processing model is required in this context, which copes with the limited on-chip memory capacity and balances data transfer overhead.

(RQ3) How can we fully utilize all processing units, e.g., CPUs and GPUs, in a heterogeneous computing system to jointly learn the causal structures? In this context, load balancing mechanisms are required, which handle tasks for parallel execution with different granularities suitable for each processing unit.

Contributions
Development of gpupcalg – an R-package with C++/CUDA extensions gpupcalg will include GPU-accelerated implementations of CI tests (addressing RQ1) for:
- multivariate normal distributed data\(^3\);
- discrete data\(^1\);
- and mixed data.

gpupcalg implements strategies (addressing RQ2 & RQ3) to:
- overcome single GPU memory limitations\(^4\);
- execute on multiple GPUs in parallel;
- and execute cooperatively on CPU-GPU.

Preliminary Results
- On multivariate normal distributed data with limitation to conditioning sizes of up to 1 GPU-accelerated execution achieves speed-up over CPU-based execution of factors up to 700\(^3\).
- A block-based approach to overcome GPU memory limitations shows improved scalability over an implicit memory managed version Fig. 2\(^4\).
- On discrete data, GPU-accelerated execution achieves speed-up over CPU-based systems of factors up to 62 Fig. 3\(^1\).
- Extension to multi-GPU promises reasonable scaling with the number of devices for multivariate normal distributed data.

Conclusion
This thesis investigates parallel execution strategies within heterogeneous computing systems to efficiently execute CSL. While we achieve speed-up over conventional CPU-based approaches in several settings, it remains to define a generalized model for parallel execution of CSL based on CI test characteristics. Hence, our current effort is to extend our findings to other currently existing and future CI tests.

References:

- Note, some publications are published under my birth name Schmidt.

Fig. 1: Sketch of the PC algorithm with its two phases to learn the causal graphical model from observational data
Fig. 2: Comparing explicit (Block-Based) with implicit (UM) memory managed versions to scale CSL beyond GPU memory limitations
Fig. 3: Execution times of parallel adjacency searches within PC algorithm on discrete data on multi-core CPU (parallelPC, bnlearn) or GPU (disc-cupc, gpupc)