Algebraic techniques for parallelising algorithms

When designing algorithms, often we are challenged with a following scenario: we know an efficient algorithm solving particular task, but we need to compute the same solution for many correlated instances simultaneously. This type of problems arise naturally in plenty of settings:

- computation of shortest paths between ALL pair of vertices in a given network (e.g. network is a road network, and we want to be able to perform efficient routing),
- computation of a similarity between given pattern sequence and EVERY possible subsequence of some longer input sequence,
- computation of a similarity score between ALL pairs of input points from some high-dimensional space (representing e.g. interests of users of a social network).

Such problems occur in many computational settings, both centralized algorithms and parallel algorithms (computation happens over many communicating machines in parallel). We can further distinguish exact version of the problems, where the answer is computed precisely, and approximate versions, where some slack in solutions is allowed. Regardless of the setting, in the past years there has been a constant stream of innovative techniques and approaches to this setting, such as:

- There are two fundamental linear algebra operations: vector convolution and matrix multiplication. Both admit faster algorithms than a naive brute-force approaches. Both encode a form of primitive computational parallelism, where batching similar operations together makes them effectively cheaper than performing them separately.
- Error correcting codes are a tool for succinctly encoding objects that allows for representation of objects that is resilient to small changes. This makes them very useful when we want to detect objects that are almost identical (up to few changes, that is errors).
- Sketching is a form of succinctly representing an object in a form of a much shorter sketch. Such sketch usually contains only a portion of original object representation, but appropriately designed can contain useful structural information. A popular form of sketching is e.g. fingerprinting, where based on equality of fingerprints we can test equality of original objects (up to negligible errors).

The goal of this project is to explore mentioned techniques of making parallel computation effective. We especially want to focus on: exploring graph algorithms in the topic of flows and cut-structure, which is still underrepresented avenue of graph research in the context of all-to-all computation; exploring algorithms for integer sequences, that is searching for occurrences or almost-occurrences of patterns in the input text and exploring the rich world of fine-grained complexity of problems, especially in the context of approximate computation.