Parallel & Distributed Computer Systems 2023

Homework 1

Graph Minor

Deadline: November 18, 2023

The adjacency matrix of a graph $G(V, E)$ is a matrix A such that $A_{ij} = 1$ if $(i, j) \in E$. Otherwise, $A_{ij} = 0$.

A configuration of the nodes of the graph V is a mapping $V \to \{1, 2, \cdots, c\}$ to integers denoting the cluster to which the node belongs.

The graph minor of a graph $G(V, E)$ is the graph $G'(V', E')$ such that $V' = \{1, 2, \cdots, c\}$ and $(i, j) \in E'$ if and only if there exists $(u, v) \in E$ such that u and v belong to the same cluster in the configuration.

For this homework, we relax the definition of the adjacency matrix of the graph minor so that it will be the matrix M that results from the following computation: Let Ω be the $n \times c$ configuration matrix such that $\Omega_{ij} = 1$ if node i belongs to cluster j, and $\Omega_{ij} = 0$ otherwise. Then the adjacency matrix of the graph minor is given by:

$$M = \Omega^T A \Omega.$$

This is not necessarily the most efficient way to compute this result. You are free to use any other method you may find more efficient. Also your code should work with any square matrix A and any configuration. Do not assume that the matrix A is symmetric or that it only has 0-1 entries.

To be able to process large, interesting graphs with millions of nodes and edges (think Facebook mutual friends, for instance) we will use a sparse matrix representation. Matrix A is always stored as a sparse matrix data structure, –in any format you prefer, see for example, Compressed Sparse Row (CSR), Compressed Sparse Columns (CSC), COO, etc–.

The mathematical description of the graph minor, hints to simply form the configuration matrix Ω from the vector of configuration ids and then use a shared memory parallel library to do the sparse matrix-matrix multiplications with the adjacency matrix A. However, this is not the most efficient way to compute the graph minor, and your solution should be faster than this direct approach.

The number of clusters c may be of the same magnitude to the number of nodes n. Therefore, the result should always be stored as a sparse matrix. However, whenever possible, it may be advantageous to use a dense matrix representation to compute (a part of) the graph minor.

Test with square matrices of graph datasets from the web. To load the sparse adjacency matrix use the Matrix Market (MM) format. Examples for parsing

such files are found at https://math.nist.gov/MatrixMarket/mmio-c.html, e.g., `example_read.c` which relies on `mmio.h` and `mmio.c` for reading any MM matrix.

Parallel implementations (C/C++)

Parallelize your implementation with OpenCilk, OpenMP, and Pthreads. Run it in parallel and compare sequential and parallel agreement in results and run times for different large datasets. We will also propose matrices and configuration vectors for you to produce the final diagrams to study the scalability of your code.

If you are interested, please also try to implement and parallelize your approach with Julia, or any other language you prefer.

What to submit

- A 4 page report in PDF format. Report execution times of your implementations with respect to the number of vertices n, the number of edges m, the number of clusters c, and the number of threads p.
- Upload the source code on GitHub, BitBucket, Dropbox, Google Drive, etc. and add a link in your report.
-
- Argue about the validity and effectiveness of your code. You can also include any other information you consider important for the evaluation of your work.
- Cite any external sources you may have used.