CSCI-4974/6971: Homework 3
`<v1.1>` updated March 25, 2026

Community Detection Algorithms and Benchmarking

Due Date: Friday 3 April 2026, 11:59pm via Submitty

For this assignment, we're going to be implemnting a multi-level community detection algorithm based on min-cut optimization and compare it to a multi-level modularity-based algorithm using an LFR-style benchmarking experiment.

We will be using Submitty for collecting homeworks. Upload a single *.py file that outputs responses for all of the below. Pay careful attention to output formatting. Your code should be runable as a script on the command line (via `bash$ python3 hw03.py`). **You can use any NetworkX/NumPy/SciPy/PyTorch functionality you wish, but do not use any other external libraries**.

1. The first part of the assignment will be to build a multi-level community detection algorithm around Label Propgation. Recall how the multi-level Louvain Algorithm works at a high level:

   - For each vertex, find the community that maximizes modularity.
   - Coarsen each community into a single vertex.
   - Repeat the above until we can no longer improve modularity.

   We will be implementing the same process for label propagation and min-cut optimization. Pay careful attention to the differences from our baseline algorithm.

   - **Vertex Assignments:** For each vertex, find the community that this vertex has the most neighbors to, with ties broken randomly. *Do not place this vertex into that community until all vertices' assignments have been found.* We are using a level-synchronous approach here, to avoid immediate convergence and maintain an explicit community hierarchy.
   - **Coarsen Communities:** Create a new graph, where all vertices in a given community are merged into a single new vertex. Add any (u, v) edge where $u \in C_1$ and $v \in C_2$. Create self loops and multi-edges as needed.
   - **Iterate:** ~~Do the above until convergence, where no more updates are processed~~. (**v1.1**) Instead, do the above until modularity no longer improves.

- **Return:** Return the community assignments on a per-vertex basis with respect to the original graph. Note that you might need to maintain vertex→community assignments at each level in order to do this.

2. We will next be performing an LFR-style experiment to evaluate the differences in community detection performance between the Louvain algorithm, the asynchronous baseline Label Propagation algorithm, and our new synchronous multi-level algorithm. Using the parameters below, generate 5 instances of all combinations of parameters for all mixing parameter (μ) values, and run all 3 algorithms on each instance. For each set of parameters and varying μ, calculate the area under the NMI curve as the sum of NMI values across the varying μ. Output for the number of tests that each algorithm 'won'.

   - **Num vertices:** 500, 1000, 2000
   - **Avg degree:** 4, 10
   - **PL exp. for degrees (τ_1):** 2.5
   - **PL exp. for comms. (τ_2):** 1.5
   - **Min Comm:** 2×(average degree)
   - **Mixing parameter (μ):** 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7

   (**v1.1**) Note: NetworkX's implementation of the LFR benchmark fails quite regularly. A simple way to get around this is to put the function call in a try-except block within a "while not successful" loop.