PROJECT: COMMUNITY DETECTION

Overview: Community detection is a ubiquitous graph data mining task. Depending on the type of graphs (e.g., directed, undirected, static, dynamic, weighted, unweighted, labeled, unlabeled, unipartite, bipartite or multipartite) and depending on the goal of the graph mining task, community detection can be differently defined. Moreover, for a given community definition, various community detection algorithms could be designed and implemented that target either dense or sparse graphs and meet particular performance characteristics. The performance could be measured along various dimensions: time and/or space efficiency, accuracy of identifying the communities with the known ground truth, etc.

Goal: To implement the given community detection algorithm for real-world graphs.

To do that, your team will need:

a) To implement the algorithm assigned to your team with an "obsession" with the highest performance possible.
b) To choose 3 implementations of different algorithms from other teams.
c) To provide a detailed comparative analysis of the performance of your team's implementation with respect to the 3 chosen algorithms.

Project Teams: For this project, you will be working in a team assigned by the TA/instructor. You will not be allowed to choose the team members. The reason is that on a real job you often do not choose your colleagues, but have to work on solving the problem given the colleagues assigned to the project.

Input: You will be provided with the following materials in advance:

- Graph datasets with ground truth communities:
  - Graph 1: Amazon.
  - Graph 2: DBLP.
  - Graph 3: YouTube.
- Scientific publication that describes the algorithm to be implemented.
- Benchmarking codes to measure the performance of your team's community detection algorithm with respect to various performance characteristics.

Output: The implementation of the algorithm and a report (see details below).

Project Details:

1. Read and understand your scientific publication.
2. Implement the method described in the publication.
3. Implementation can use any programming language (C, C++, Java, Matlab, SAS, R, etc.).
4. Implementation can be serial or parallel (using Hadoop, openMP, MPI, etc.).
5. Measure performance of the algorithm.
   a) The notion of "good" communities will be tested using the following performance metrics:
      i. **Separability** captures the intuition that good communities are well-separated from the rest of the network.

ii. **Density** builds on the intuition that good communities are well connected.
iii. **Cohesiveness** characterizes the internal structure of the community. Intuitively, a good community should be internally well and evenly connected, i.e., it should be relatively hard to split a community into two sub communities.
iv. **Clustering coefficient** is a measure of the degree to which nodes in a graph tend to cluster together.

Required Reading: *Defining and Evaluating Network Communities based on Ground-truth by J. Yang, J. Leskovec. IEEE International Conference On Data Mining (ICDM), 2012.*

b) The conformance of the identified communities to the ground truth communities will be tested using the following performance metrics:
   i. **Precision** is the proportion of vertex pairs in the same identified community that are also in the same ground truth community.
   ii. **Recall** is the proportion of vertex pairs in the same ground truth community that are also in the same identified community.
   iii. **F-measure** is the harmonic mean of precision and recall.
   iv. **Normalized Mutual Information** considers all the possible community matching between the ground truth communities and the identified communities. For more details, see: http://arxiv.org/pdf/1110.2515v2.pdf
   v. **Rand Index** or simple matching coefficient, accounts for both the specificity and sensitivity of the identified communities. For more details, see: http://www3.nd.edu/~dial/papers/ASONAM11c.pdf

Required Reading: *Evaluating Community Detection for Networks with Ground Truth Information*.

c) The algorithm "runtime" is an additional performance criterion that will be taken into account.

Bonus Points

Project teams that are assigned the same scientific publication will compete for the most effective algorithm implementation. **The top ranked team based on runtime will get two BONUS points added to their total course score**. **The second ranked team based on runtime will get one BONUS point added to their total course score**.

Project Plan

a) The project duration is two weeks.
b) At the end of the first week, each team is required to submit parts (i) and (ii) of the "Submission Requirements."
c) The implementations of all the groups will be posted on the course page.
d) Select the implementations of 3 algorithms that are different from the one assigned to your team.
e) At the end of the second week, submit parts (iii) and (iv) of the "Submission Requirements."

Submission Requirements

i. Algorithm code with detailed comments.

ii. README file with detailed instructions. It should include the following information:
   a. Software that needs to be installed (if any) with URL's to download and instructions to install them.
   b. Environment variable settings (if any) and OS it should/could run on.
   c. Instructions on how to run the program.
   d. Instructions on how to interpret the results.
   e. Sample input and output files.
   f. Citations to any software you may have used or any dataset you may have tested your code on.

   In short, the TA and any other team that chooses your implementation should be able to install any required software, set up the environment, execute your program, and obtain results without any prior knowledge about your project.

iii. Moodle form to enter performance metrics

Report the performance metrics of you algorithm using the dataset(s) provided.

iv. Project Report - The report is divided into two sections:

Section 1 (0.5 - 1 page)

This section should detail the contributions made by each member of the team. Divide the contributions into research, coding, result collection, and documentation. Write down who was involved in each part and what they did. If required, you may add more subsections to the contributions.

Section 2 (2 – 3 pages)

This section of the report should describe, in your own words, the algorithm discussed in the scientific publication assigned to your team. This section should also discuss the performance of this algorithm. If the algorithm is parameterized, then include some discussion and empirical results on the effect of the parameters on the identified communities. Provide a detailed comparative discussion of your team's algorithm with the 3 other algorithms chosen.

Grading Rubric

Criteria	Percentage
Implementation	10%
Code executes and produces required results	40%
Moodle form to report performance metrics	5%
Project Report – Section 1	5%
Project Report – Section 2	40%
Bonus Points	1 or 2 to the total course score

Project was created by Kanchana Padmanabhan and Nagiza Samatova