CSCI-4974/6971: Parallel Graph Analysis

www.cs.rpi.edu/~slotag/classes/FA17/index.html

Prof. George M. Slota
gmslota@gmail.com
Office Hours: 14:00-16:00 Wednesday in 317 Lally
(and by appointment)

Class Hours: 16:00-17:50 Monday&Thursday in 106 Carnegie

1 Course Description

This class is an introduction to parallel graph processing and mining. Students will learn about research challenges related to the study of large graphs on modern parallel systems. Course topics will include, but are not limited to, the following:

- General computational graph processing
- Social, web, and biological network analysis and mining
- Graph processing frameworks (Giraph, Pregel, NetworkX, etc.)
- Graph analysis algorithms
  - Graph connectivity, centrality measures, community detection, subgraph matching, graph alignment
- Computational optimizations for parallel graph analysis
  - Parallel processing methods, graph partitioning and ordering, system hardware considerations
- Random graph models
- Graph computations as matrix operations
- Graph visualization

1.1 Prerequisites

This course has no listed formal prerequisite classes. However, it is recommended that students have, in addition to basic knowledge of data structures and algorithms and experience in C/C++ and Java, experience in parallel computing and some knowledge of graphs and graph algorithms. Loose prerequisites to satisfy this knowledge would be CSCI-4320 or CSCI-6320, Parallel Programming or Parallel Computing and CSCI-4260 or MATH-4150, Graph Theory. More specifically, it is expected students have some basic knowledge of:

- **Graphs and Graph Algorithms**: Basic definitions, traversal algorithms, and graph computation algorithms (e.g. PageRank or min/max-flow).
- **Parallel Programming**: Popular shared and distributed libraries and APIs. We'll be using OpenMP for shared-memory parallelization, but experience with Pthreads, OpenCL, etc. is fine, since the concepts and usage are all similar. A working knowledge of MPI and experience debugging parallel application will also be extremely useful.
- **High Performance Computing**: Developing, submitting, and executing applications on large *nix-based clusters. Knowing how to access clusters, write code and job scripts, and submit them for execution.

We will review the required material during the course, but you might get behind if you've never seen any of the above terms at all!

1.2 Course Resources

There is no official textbook for this course. Instead, we will be utilizing a wide variety of textbooks and papers found online. We will also extensively utilize a number of available repositories housing real-world graphical data. Check the website for an updated list of resources.

2 Course Schedule

Classes will meet every Monday and Thursday at 16:00 in 235 Darrin with the following exceptions:

September 4: No class - Labor Day
October 9: No class - Columbus Day
October 10: **Yes class** - following Monday schedule
November 13 and 16: No Class - Instructor on travel
November 23: No class - Thanksgiving

The above may change with little to no notice. For an up-to-date schedule with class notes and content, check the website.

3 Homeworks

There will be several programming-based assignments due approximately every 3 weeks during the semester. These assignments will follow the course content and can be worked on during hands-on time following class lectures.

4 Project

As one of the main goals of this course is to teach end-to-end graph analysis (identifying an interesting problem, implementing and optimizing a solution, analyzing results), a semester-long project will be performed with this goal in mind. This project will be done on an individual basis, but exceptions will be made for small groups with justification – but note that expectations will scale with group size.

Projects can either focus on the analytic aspect or implementation/optimization aspect of computational graph analysis. E.g. for analytic: performing some novel community structure analysis on some real-world date. E.g. for implementation/optimization: implement a known parallel algorithm and optimize it to improve scalability on highly irregular data. For graduate students, solving problems or using data from your own research is encouraged, as long as it fits the general "graph analysis" theme and doesn't overlap with project work from another course.. We'll discuss some possible project ideas in class.

There's five main parts to the project. Check the website for dates.

1. Project proposal: One page report summarizing the problem you want to solve, the proposed methodology, and expected outcomes.

2. First update presentation: 5 minute presentation detailing progress so far. It is expected that you introduce your project to the class and explain your work so far. This will be done during class about a month after the proposal, so some progress into the methodology should be accomplished.

3. Second update presentation: 5 minute presentation giving ongoing progress. This will be another month after your first update, so some results are expected.

4. Final presentation: 15 minute presentation giving a full overview of your project, including an introduction to the topic, a detailed explanation of methodology, and final results.

5. Final report: Report submitted with a standard research paper organization: intro, background and prior work, methodology, experimental setup (data and systems used), and results. All supporting code and data must be submitted with the report. Reproducibility is a key component of science – I should be able to recreate all your results!

5 Grading Policies

Homework policy: Homeworks will comprise 60% of the final grade. For homeworks, the lowest graded assignment will be dropped. Homeworks will be due on Thursdays via email to `gmslota@gmail.com` before the start of class. No late assignments will be accepted, except under verifiable extenuating circumstances. Assignments will consist of both programming and short response questions. Specific grading criteria for each assignment will be given.

Project policy: The project will comprise 40% of the final grade. The project grade will be broken down with 5% of the final class grade given to the project proposal, 5% given to the first update presentation, 5% given to the second update presentation, 5% for the final presentation, and 20% for the final results and report.

Grade Modifiers Policy: Grade modifiers will be used in this class. You can expect to earn a B- if your score is greater than 79.5 and less than 83, B if your score is greater than 83 and less than 86, B+ if your score is greater than 86 and less than 89.5. The similar modifier points occur for the A, C and D ranges except that there is no A+ nor is a D- allowed under the RPI Grade Modifier Policy.

Attendance Policy: Attendance is not required during normal lectures and class times. However, **attendance is required on project presentation days**. A student will receive a zero for that portion of their project grade if they are absent from their presentation without a verifiable excuse.

6 Academic Integrity

While it is expected and encouraged for students will form study groups and collaborate, assignments should all be done individually. What this means is: no sharing of any question responses or code is allowed. If it is determined during grading that students shared or duplicated work, these students will receive a zero on that assignment. On a second offense, any student involved will fail the course. For both offenses, the academic integrity violations will be reported to the school.

7 Learning Outcomes

At the end of this course, you will:

- Be able to implement and run **parallel graph algorithms** on real-world data
- Understand basic approaches for **social, web, and biological network analysis**
- Know the various **random graph models** and how the relate or don't relate to real-world data
- Recognize the representation of **graphs as matrices** and how to perform equivalent computations on both

- Know various **optimization techniques and considerations** for achieving parallel scalability when processing irregular (graph) data