Weekly Report 3
Yiduo Ke
ORIE 4999 supervised by Professor Williamson

What I did

Installed and learned Julia (followed a tutorial) and started coding in Julia. These are the topics I learned:

- Printing
- Strings
- Variable assignment
- Comment
- Julia type inferencing
- Data structures in Julia
    - Dictionary
    - Tuple
    - Array
- Loops
    - While loops
    - For loops
    - List comprehension
- Conditionals (if, ifelse, else)
- Function declaration
- Duck-typing in Julia
- Mutating and non-mutating functions
    - Convention to end function name with ! if it's a mutating function
- Broadcasting
- Packages
- Plots
- Multiple dispatch
- Benchmark testing
- Linear algebra in Julia

So, I started coding in Julia. I'm starting simple. I am currently trying to implement (on an undirected graph with unit edge weights) the brute force MAX-CUT solver and the "flip a coin" algorithm. I assume the input is an adjacency matrix (or should I make the input something else? Incidence matrix? Adjacency list? In any case, I can convert between any of these).

What I'll do next week

I'll finish those two functions (brute force and flip a coin) and write test cases. The brute force version (once it's verified correct) will serve as a sanity checker for the approximation algorithms (since it returns the true max cut value).

I will also implement the greedy algorithm that guarantees ½-approximation because it's part of the smallest eigenvalue RECURSIVE-SPECTRAL-CUT algorithm in the Trevisan paper.

After these two are done I'll start implementing the algorithms in the two papers, still on graphs with only unit weights (for now).

I was wondering how to test the approximation algorithms. Because they're approximation algorithms and may return a different partition every time, there's no one set answer. Should I just run them a bunch of times and compare to the brute force answer and plot the benchmark times with respect to input size to make sure my runtime matches with the papers'?