Weekly Report 2

Yiduo Ke

ORIE 4999 supervised by Professor Williamson

# What I did

I did the rest of the assigned reading: Lectures 10 and 11 of ORIE 6334 from Fall 2019, and Trevisan 2012 from the beginning to section 4.

# My Thoughts

This was a much harder topic to understand than last week’s (semidefinite programming). I had to learn some topics mentioned in the paper on my own (such as spectral partitioning, incidence matrix of a graph, induced subgraphs, etc.).

From my understanding after reading the materials several times, this is my pseudocode of the algorithm from the beginning of section 4 (page 9 of the PDF), and thus the meat of the paper:

(on next page)

**function** Recursive-Spectral-Cut(V,E,δ){

y = 2-Thresholds Spectral Cut with δ;

M = number of weighted number of edges (i,j) such that at least one of y\_i and y\_j is not 0 (i.e**.** weighted number of edges incident to S⊆V);

C = the weighted number of cut edges (i,j) such that y\_i and y\_j are both not 0 and have opposite signs (i.e**.** one vertex is in L and the other is in R);

X = weighted number of cross edges (i,j) such that exactly one of y\_i and y\_j is 0 (i.e**.** one vertex is in S and the other is not)

**if** (C+X/2 <= M/2){

**use** a greedy algorithm to find a partition of V that cuts at least |E|/2 edges (coin flipping could work?), and **return** it

}

**else** {

L = {i such that y\_i == -1};

R = {i such that y\_i == +1};

V\_prime = {i such that y\_i == 0};

G\_prime(V\_prime, E\_prime) = induced subgraph of V\_prime;

(V\_1, V\_2) = Recursive-Spectral-Cut(V\_prime, E\_prime, δ);

**return** bigger\_cut((V\_1 ∪ L, V\_2 ∪ R), (V\_1 ∪ R, V\_2 ∪ L));

}

}

When the paper said, “Run the algorithm of Theorem 1 with accuracy parameter δ” (section 4), I’m assuming it’s the algorithm given 2-Thresholds Spectral Cut (abbreviated 2TSC) given on pages 1775-1776 (pages 7-8 in the PDF). 2TSC seems mostly straightforward, but **I’m having trouble understanding how to get the and where the comes into play**. To the best of my understanding, is not an input to any program, but is rather a means of measuring if a solution satisfies – when plugged into (3) – an upper bound. I took note of “the optimization problem in (4) is the problem of computing the smallest eigenvalue of “(1775). So, does this mean is the eigenvector corresponding to the smallest eigenvalue of the matrix ? I really can’t tell how to get used in 2TSC.

# What I will do next

I will start learning Julia and hopefully start implementing the algorithms, starting with naïve random algorithms and simple, greedy algorithms, then semidefinite programming, then Trevisan’s (though I have to understand his algorithm completely first).