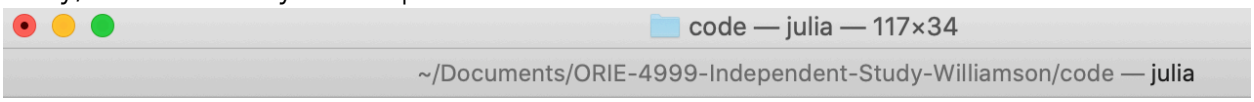
Weekly Report 8
Yiduo Ke
ORIE 4999 Supervised by Professor Williamson

What I did

I fixed my SDP max cut code; it now does better than expected. Initially, Renee and I thought the problem was in Cholesky decomposition and how I used the upper, lower, and permutation matrices. $L * U$ didn't return the original matrix. In this screenshot of my terminal, $matrix$ is the optimized positive semidefinite matrix, P is the permutation matrix after doing Cholesky decomposition, L and U are the lower and upper matrices, respectively, after Cholesky decomposition.




```
julia> P' * matrix * P
8×8 Array{Float64,2}:
  1.00005   -0.496259  -0.500526   1.00005    1.00002   -1.00003    0.498101  -0.496259
 -0.496259   1.00001   -0.503196  -0.496259  -0.49251    0.494598   0.505652   1.00001
 -0.500526  -0.503196   0.999919  -0.500526  -0.504237   0.502161  -0.999955  -0.503196
  1.00005   -0.496259  -0.500526   1.00005    1.00002   -1.00003    0.498101  -0.496259
  1.00002   -0.49251   -0.504237   1.00002    1.00001   -1.00001    0.501818  -0.49251
 -1.00003    0.494598   0.502161  -1.00003   -1.00001    1.00001   -0.49974    0.494598
  0.498101   0.505652  -0.999955   0.498101   0.501818  -0.49974    0.999999   0.505652
 -0.496259   1.00001   -0.503196  -0.496259  -0.49251    0.494598   0.505652   1.00001

julia> L * U
8×8 Array{Float64,2}:
  1.00005   -0.496259  -0.500526   1.00005    1.00002   -1.00003    0.498101  -0.496259
 -0.496259   1.00001   -0.503196  -0.496259  -0.49251    0.494598   0.505652   1.00001
 -0.500526  -0.503196   0.999919  -0.500526  -0.504237   0.502161  -0.999955  -0.503196
  1.00005   -0.496259  -0.500526   1.00005    1.00002   -1.00003    0.498101  -0.496259
  1.00002   -0.49251   -0.504237   1.00002    3.00009   -3.00009    1.50176   -1.4813
 -1.00003    0.494598   0.502161  -1.00003   -3.00009    4.00012   -1.99943    1.97799
  0.498101   0.505652  -0.999955   0.498101   1.50176   -1.99943    2.74966    0.269795
 -0.496259   1.00001   -0.503196  -0.496259  -1.4813     1.97799    0.269795   2.98917

julia> matrix
8×8 Array{Float64,2}:
  1.00001    0.501818   1.00002    1.00002   -0.504237  -1.00001   -0.49251   -0.49251
  0.501818   0.999999   0.498101   0.498101  -0.999955  -0.49974    0.505652   0.505652
  1.00002    0.498101   1.00005    1.00005   -0.500526  -1.00003   -0.496259  -0.496259
  1.00002    0.498101   1.00005    1.00005   -0.500526  -1.00003   -0.496259  -0.496259
 -0.504237  -0.999955  -0.500526  -0.500526   0.999919   0.502161  -0.503196  -0.503196
 -1.00001   -0.49974   -1.00003   -1.00003    0.502161   1.00001    0.494598   0.494598
 -0.49251    0.505652  -0.496259  -0.496259  -0.503196   0.494598   1.00001    1.00001
 -0.49251    0.505652  -0.496259  -0.496259  -0.503196   0.494598   1.00001    1.00001
```

$matrix$ is indeed symmetric and Hermitian:

```
julia> ishermitian(matrix)
true

julia> issymmetric(matrix)
true
```

But it's not positive semidefinite, as some of its eigenvalues are (very slightly) negative:

```
julia> eigvals(matrix)
8-element Array{Float64,1}:
 -4.515259561787817e-16
 -2.734960299678536e-16
 -4.584238334538043e-17
  1.25905729940844e-16
  3.0637514761671636e-16
  4.972909197643032e-16
  3.008782129723279
  4.991279324808953
```

But since the negative eigenvalues are basically 0, I guess $matrix$ is positive semidefinite. I decided to go back to the paper and realized I was too negligent. Turns out I did unnecessary steps after optimizing $matrix$; I didn't need to Cholesky decompose it. Now I got rid of those steps and the the SDP function performs better than expected. Here is a screenshot of test_results.txt:

```
generated brute force test (OPT). max cut: 23
generated coin flip test. mean randomized max cut: 13.64 (should be ≈14
generated greedy test. greedy max cut: 21 (should be ≥14)
generated SDP test. SDP max cut: 22.19 (should be ≈20.194)
generated trevisan test. trevisan max cut: 21 (should be ≥12.213000000000001

generated brute force test (OPT). max cut: 34
generated coin flip test. mean randomized max cut: 23.35 (should be ≈23
generated greedy test. greedy max cut: 29 (should be ≥23)
generated SDP test. SDP max cut: 32.68 (should be ≈29.852)
generated trevisan test. trevisan max cut: 29 (should be ≥18.054000000000002

generated brute force test (OPT). max cut: 16
generated coin flip test. mean randomized max cut: 8.55 (should be ≈8
generated greedy test. greedy max cut: 16 (should be ≥8)
generated SDP test. SDP max cut: 16.0 (should be ≈14.048)
generated trevisan test. trevisan max cut: 16 (should be ≥8.496

generated brute force test (OPT). max cut: 10
generated coin flip test. mean randomized max cut: 5.0 (should be ≈5
generated greedy test. greedy max cut: 8 (should be ≥5)
generated SDP test. SDP max cut: 10.0 (should be ≈8.78)
generated trevisan test. trevisan max cut: 8 (should be ≥5.3100000000000005

generated brute force test (OPT). max cut: 26
generated coin flip test. mean randomized max cut: 17.98 (should be ≈18
generated greedy test. greedy max cut: 25 (should be ≥18)
generated SDP test. SDP max cut: 25.69 (should be ≈22.828)
generated trevisan test. trevisan max cut: 25 (should be ≥13.806000000000001

generated brute force test (OPT). max cut: 6
generated coin flip test. mean randomized max cut: 3.49 (should be ≈3
generated greedy test. greedy max cut: 6 (should be ≥3)
generated SDP test. SDP max cut: 6.0 (should be ≈5.268)
generated trevisan test. trevisan max cut: 6 (should be ≥3.186

generated brute force test (OPT). max cut: 2
generated coin flip test. mean randomized max cut: 1.13 (should be ≈1
generated greedy test. greedy max cut: 2 (should be ≥1)
generated SDP test. SDP max cut: 2.0 (should be ≈1.756)
generated trevisan test. trevisan max cut: 2 (should be ≥1.062
```

What I will do next week

- Clone everything I have now and convert it to weighted graph version