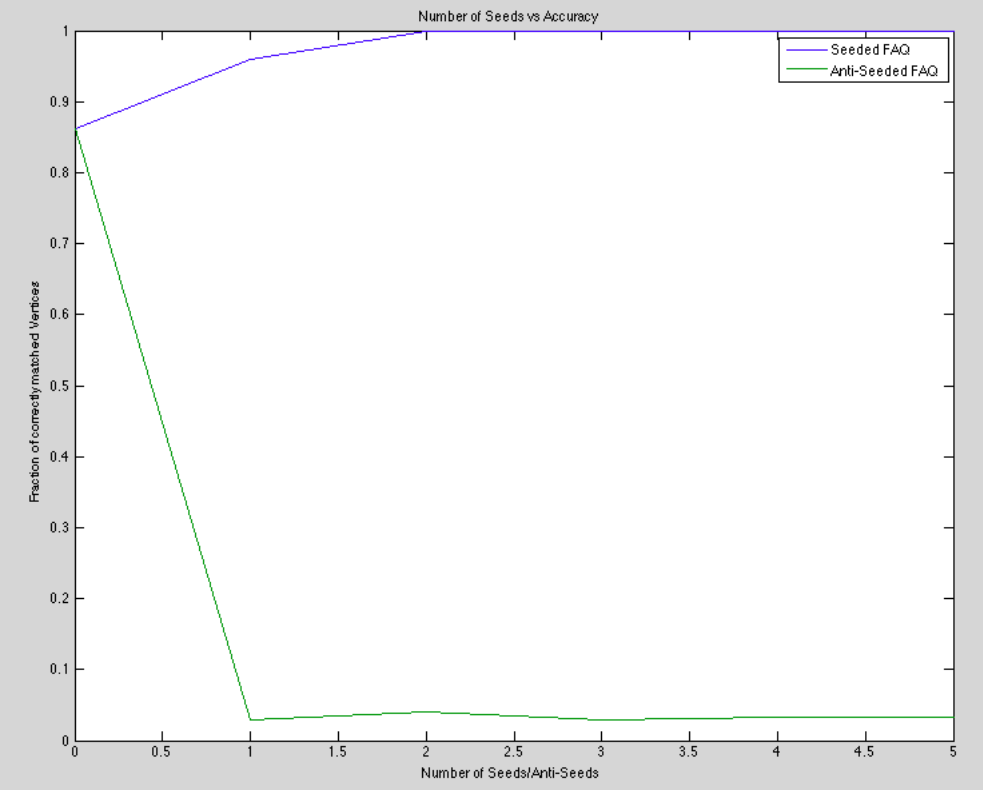
EN.580.694: Statistical Connectomics
Final Project Report

Elan Hourticolon-Retzler · May 14, 2015

Extending FAQ Graph Matching to use Anti-Seed Informating



Opportunity Fast Approximate Quadratic Programming for Graph Matching has proven itself to be an effective method for graph matching, finding the best permutation matrix to map the vertices of one graph to another. This is done by relaxing the objective function (of minimizing edge differences between graphs) and using an iterative approach to find an optimal permutation matrix.

Fishkind, Adali, and Prieb have additionally demonstrated that first seeding even just a few vertex matches greatly improves performance. This was done by breaking our Permutation matrix into blocks and leaving us with a smaller subset of vertices that we need to match.

Challenge Since we know that adding information about positive matches to our algorithm is useful it is conceivable that adding information about which vertices would similarly be helpful since it has the same effect of reducing our search space - albeit each anti seed being largely less informative since each Seed immediately implies NumVertices-1 AntiSeeds . Another impetus is that finding and annotation positive matches can be a costly process however it may be much

Action In a similar fashion of Fishkind, Adali, and Prieb, I broke up the permutation matrix into blocks however instead of marking known matches with the identity matrix sized according to the number of seeds I used a block of 0's sized based on the number of anti-seeds.

From this I proceeded with the same relaxation of the constraints requiring P to be permutation matrix - from each element being $\{0, 1\}$ to each element lying in the range $[0, 1]$.

Since the both the authors of the FAQ paper and the Seeded variant (Priebe and Vogelstein) paper open sourced their code I was able to build on it changing the Permutation matrix and associated gradients.

Resolution Despite my efforts at changing my formulation to include Anti-Seeds, performance has significantly suffered. My original thoughts let me to suspect a bug in my code or a error in my calculations when calculating the different block gradients, however reviewing both has turned up nothing. I now suspect there is a fundamental inconsistency in how I'm approaching the problem (i.e. how I'm partitioning the permutation matrix) since adding even a single Anti-Seed is drastically affecting my results.

Future Work This project is very much a work in progress in that I believe it to be a worthwhile and achievable endeavor however my current efforts have turned out to be fruitless.

Immediate future work includes discovering why introducing Anti-Seeds is so significantly derailing performance.

After that issue is solved I'd like to move away from large block Anti-Seeds towards finer grain resolution (ie. breaking up Anti-Seeds into many disjoin blocks instead of a sing large one). The reason I did not initially start with this is that automatically partitioning the

permutation matrix in this way seems to be a rather nontrivial programatic undertaking, one that can be pursued once I have the simplest case working).

References

[1] Carey E. Priebe Donniell E. Fishkind, Sancar Adali. Seeded graph matching. *ArXiv*, 2012.

[2] Michael Syskind Pedersen Kaare Brandt Petersen. *The Matrix Cookbook*. Technical University of Denmark, 2012.

[3] C.E. Priebe. Seeded graph matching.

[4] Joshua Vogelstein John Conroy Vince Lyzinski Louis Podrazik Steven Kratzer Eric Harley Donniell Fishkind Jacob Vogelstein Carey Priebe. Fast approximate quadratic programming for large (brain) graph matching. *ArXiv*, 2014.

[5] Joshua Vogelstein. Fastapproximateqap.