Uncovering metabolic cooperation from partial correlation graphs in Neo4j

Janet Matsen

December 16, 2016

1 Introduction

My lab studies interactions between methane oxidizing bacteria and other microorganisms, using microbe-rich sediment at the bottom of Lake Washington as a model ecosystem. Understanding how these communities oxidize methane would provide insight into a major greenhouse gas mitigation system. When a scoop of Lake Washington sediment is incubated in a bottle with methane as the only carbon source, the population becomes enriched in methane-consuming methanotrophs. Non-methanotrophic "methylotrophic" bacteria, which can use single carbon-compounds other than methane, always grow alongside the methanotrophic bacteria. The composition of methanotrophs and methylotrophs in each sample can vary widely. A third group which can only use multi-carbon compounds also appears, further complicating the interactions.

It is suspected that the methanotrophs are providing the methylotrophs with methanol, but are there additional nutrient exchanges? Perhaps the methylotrophs are assisting with nitrogen fixation, or synthesis of key nutrients. What exactly is the third group consuming? Are they eating other secretions or just scavenging the byproducts of cell death? Are they actively killing the methanotrophs and methylotrophs? We aim to discover which organisms tend to pair together in nature, and what metabolic roles each play.

Our lab has gene expression profiles from 83 incubations of Lake Washington sediment. From these, a partial correlation matrix was previously computed. Such a partial correlation matrix can be represented as a graph and mined to identify sub-graph motifs demonstrating metabolic cooperation between microbes. In particular, identification of hubs and cycles that include genes from two or more bacteria can be used to discover metabolic cooperation, such as nutrient exchange. All code for this project is available at github.com/JanetMatsen/Neo4j_meta4.

2 Related Work

Graphs are a useful tool for describing how species interact [1]. Partial correlations are often used to determine the connections in a graph in which the edges represent influence of one node on another [2]. Once a graph is in hand, community structure can be detected

by searching for groups of nodes that are highly connected to each other but have fewer connections to nodes outside the group [3]. Recent work has applied partial correlation analysis to the "phyllosphere" microbiome, the collection of bacteria that live on plants [4].

3 Methods

3.1 Choice of Neo4j

The partial correlation matrix for these 83 samples was represented in Neo4j, which has a user-friendly query language, Cypher. While the queries required to execute this project are straightforward, downstream data analysis and exploration will be greatly accelerated by the ability to perform queries on the graph and display them in Neo4j's browser. In cypher, nodes are represented by parentheses. An example motif we may want to look for could include production, export, import, and consumption of a metabolite:
`(organism:A, gene:metabolite X production)` ↔ `(organism:A, gene:metabolite X export)` ↔ `(organism:B, gene:metabolite X import)` ↔ `(organism:B, gene:metabolite X use )`.

Neo4j does have drawbacks that did not affect this project but should be considered for larger scale projects. First, though multiple clients can run in parallel on the same database, Neo4j cannot scale queries across multiple servers. When the query run time is a key concern, tools such as Grail [5], Giraph [5], or Myria [6] should be considered. Grail provides syntactic layers that allow user-friendly graph-query like APIs to work on traditional RDBMSs. Giraph focuses on parallelizability by using Hadoop. Myria also enhances parallelizability and graph-like queries with the advantage of handling distribution of the database across remote machines. As the results section illustrates, query and algorithm time were not limiting factors in this project.

3.2 Graph construction and connected components

In this Neo4j database, nodes were used to represent genes, and edges connect genes with statistically significant partial correlation with one another. The nodes are labeled with the species, gene id, and gene product. Edges are labeled with the significance weight, and sign (positive or negative). Two gene's nodes have an edge with a large positive value when those two genes tend to be highly expressed together throughout our samples, after controlling for other covarying gene's effects. Conversely, large negative edge weights indicate that the corresponding genes tend to be anticorrelated.

I prepared an information-enriched dataset for import into Neo4j from a file listing only gene loci and a partial correlation values (see this notebook). The result was a 50 million row tsv file enriched with genome names, gene products, magnitudes of partial correlations, and a label for cross-species partial correlations.

Different databases were built for varying partial correlation values in the set `[0.005, 0.0075, 0.01, 0.0125, 0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05, 0.055, 0.06]` The following Cypher query was used to iterate over the 50 million edge dataset and build databases:

```
LOAD CSV WITH HEADERS FROM
    'file: \%s'
AS line FIELDTERMINATOR '\t'
WITH toFloat(line.pcor) AS pcor,
     abs(toFloat(line.pcor)) AS abs_pcor,
     line.cross_species AS cross_species,
     line.association AS association,
     line.source_locus_tag AS source_locus_tag,
     line.target_locus_tag AS target_locus_tag,
     line.source_organism_name AS source_organism_name,
     line.target_organism_name AS target_organism_name,
     line.source_gene AS source_gene,
     line.target_gene AS target_gene,
     line.source_gene_product AS source_gene_product,
     line.target_gene_product AS target_gene_product
WHERE abs_pcor > %f
MERGE (g1:Gene {locus_tag:source_locus_tag,
                organism:source_organism_name,
                gene:source_gene,
                gene_product:source_gene_product})
MERGE (g2:Gene {locus_tag:target_locus_tag,
                organism:target_organism_name,
                gene:target_gene,
                gene_product:target_gene_product})
MERGE (g1) -[:X {pcor:pcor,
                 pcor_abs:abs_pcor,
                 cross_species:cross_species,
                 association:association}]-> (g2)
```

The `MERGE` operator ensures the pattern exists in the Neo4j database instance. Pieces of the pattern that are not found are added. The `%s` at the top and `%f` in the `WHERE` clause allowed string substitution in Java to specify which source file and cutoff to use. I used the Neo4j Java API to call this query from Eclipse (see ConstructNetwork50M.java), and count the number of nodes added to the database. The time to build the network, and network statistics such as the number of nodes, number of edges, and graph density were reported to standard out. After my script worked well, I added command line arguments and saved the script to a JAR to be run on a remote machine with $\sim$ 65 GB of memory.

I made a second JAR (see ConnectedComponentsFinder50M.java) that runs Neo4jSNA's[1] connected components algorithm. Neo4jSNA was most recently updated for Neo4j 2.3.2, which is much older than the current version of Neo4j, which is 3.0.7. Even the latest 2-series version of Neo4j, 2.3.7, is not compatible with Neo4jSNA, but the JARs for 2.3.2 was found via a link from the Chocolatey package manager. The connected components algorithm labeled the nodes in my Neo4j database with the corresponding connected components identification number. The labels enumerated by iterating over nodes using the Java API.

[1] https://github.com/besil/Neo4jSNA

The connected components run time, and key statistics about the connected components were printed to standard out so they could later be parsed by Python.

Both jars were called from a wrapper in Python (see database_comparisons.py and the corresponding ipython notebook) using the subprocess module. Other python objects held information pertinent for each level of the analysis. The Database class could submit jobs to create single databases and parse statistics about the product from Java's standard output. The DatabaseComparison class contained many database objects, a summary dataframe with a row of statistics per database, and methods to plot these statistics. Database objects each contained a `connected_components` attribute, which was an instance of the ConnectedComponentsDB class. Each ConnectedComponentsDB instance included one or more ConnectedComponent instances. These Python structures allowed me to explore results across different granularities in my jupyter notebook sessions.

4 Results

4.1 Database construction

Figure 1 shows the distribution of partial correlation values for the partial correlation matrix.

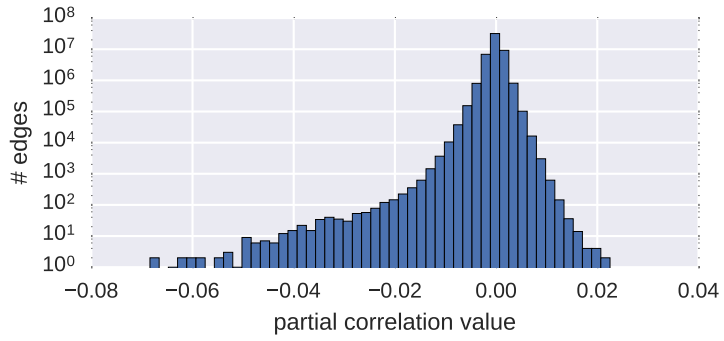

Figure 1: Distribution of partial correlation values each database was built from.

Database construction times scaled with the number of edges (Figure 2). No database took more than 50 minutes to build.

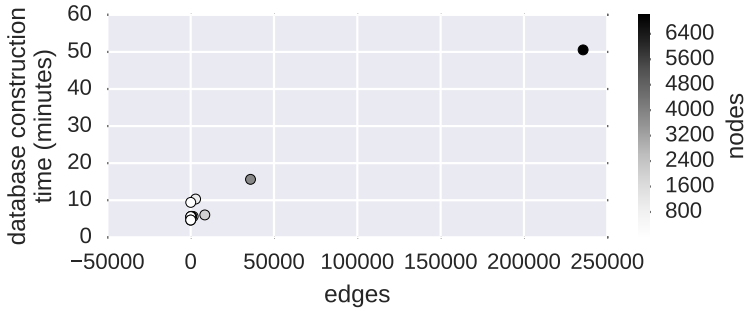

Figure 2: Database construction times scaled approximately linear with the number of edges.

4.2 Connected components run time

Queries ran fast (Figure 3). Even for the largest database (6,400 nodes and 250,000 edges), the connected components algorithm took only 12 seconds. This was a relief, as long algorithm and query wait times were potentially the biggest drawback of choosing Neo4j.

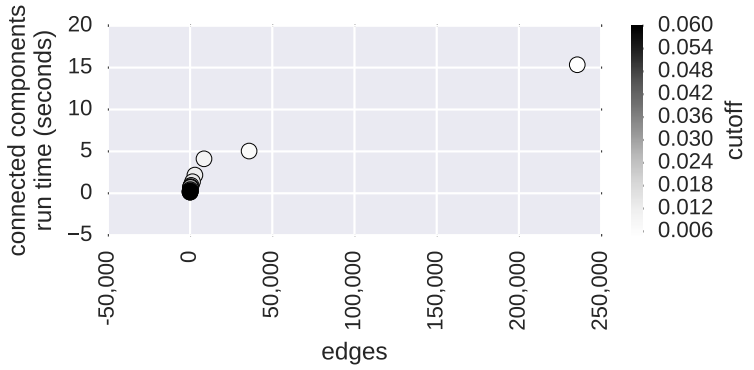

Figure 3: Fast run times for connected components.

4.3 Biological analysis of connected components

This project has a particular focus on edges that link genes between species. Figure 4 shows the number of cross-species connected components as the partial correlation cutoff is varied. With low partial correlation magnitude cutoffs (minimal trimming), the graph has a high density and there is just one large connected component. With high cutoff values (e.g. partial correlation cutoff = 0.06), the majority of the network has been trimmed out and only a few small connected components are left. For moderate cutoff values, the number of distinct connected components peaks.

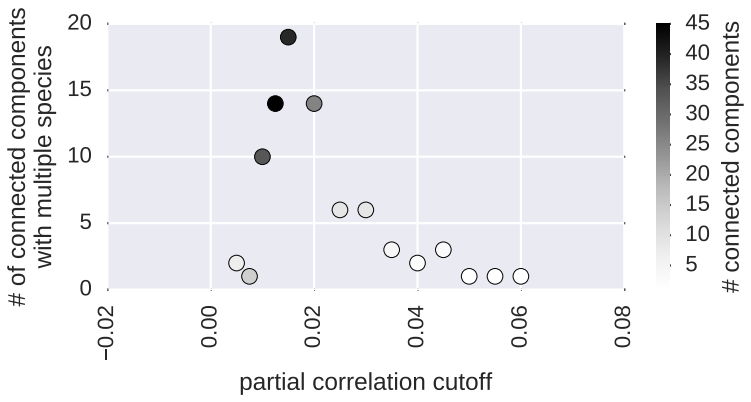

Figure 4: Count of the number of connected components containing genes from multiple species for each database constructed.

Deeper analysis of the database corresponding to partial correlation cutoff = 0.02 is shown in Figures 5, 6. In addition to having an interesting quantity of connected components containing multiple species, this cutoff also corresponds with a suitable trimming parameter to correct for multiple hypothesis testing. For these reasons, I inspected the connected components results for this dataset in greater depth.

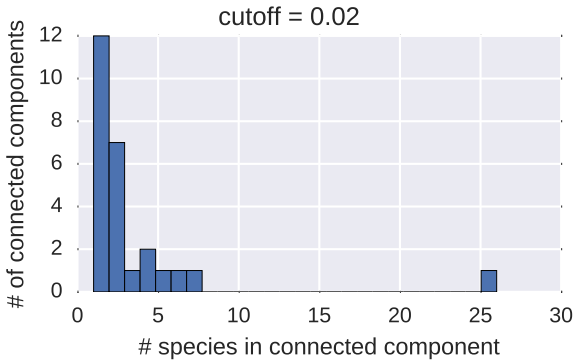

Figure 5: Distribution of the number of different species in the connected components of the database built using the cutoff abs(partial correlation) > 0.02.

As shown in 5, about half of the connected components contain genes from a single species. This is not surprising given that there are many co-regulated genes in individual species. While most connected components have less than 8 species represented, one connected component contained genes from 25 different species. The distribution of source organisms was explored in Figure 6.

number of genes for each organism across connected components (cutoff = 0.02)

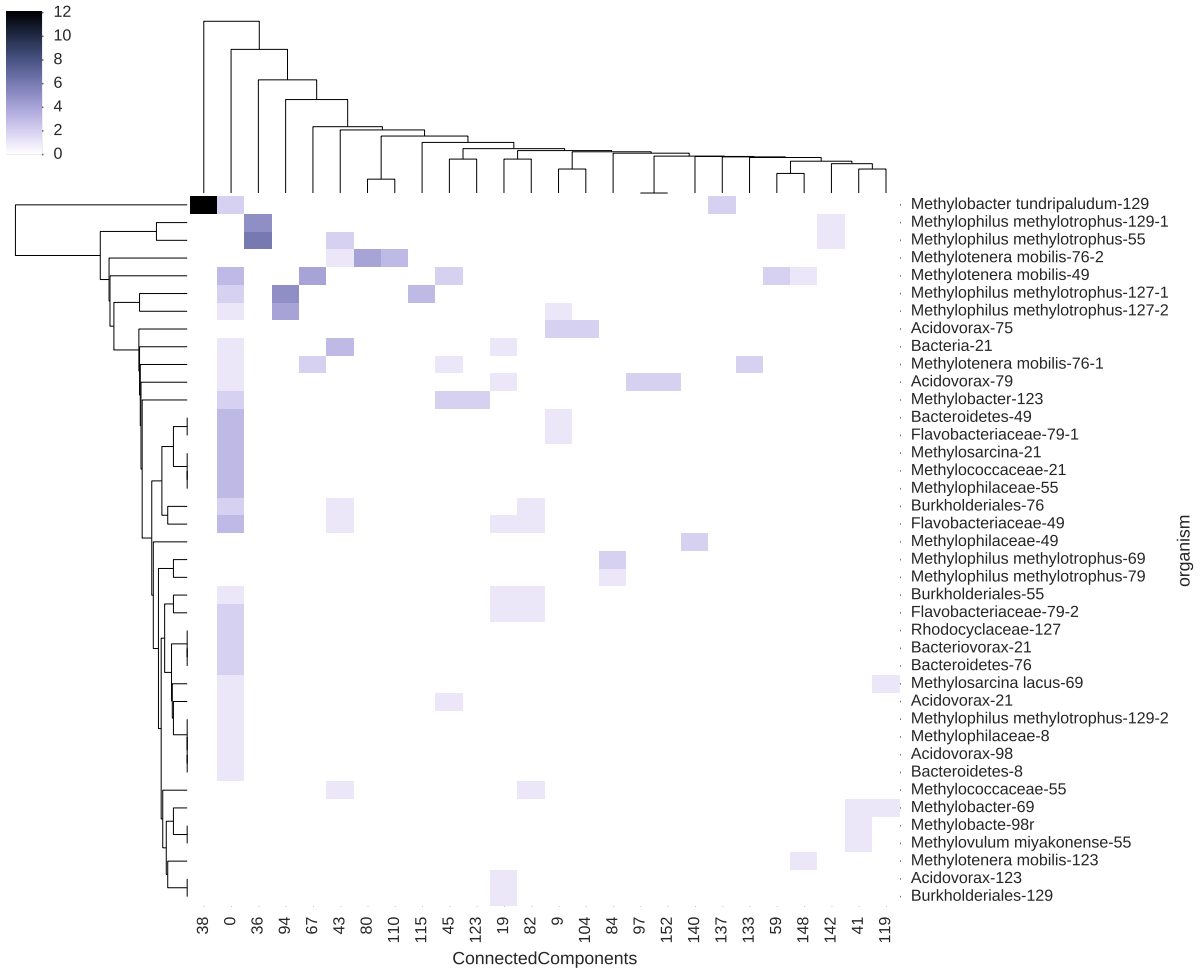

Figure 6: Host genome for genes in each connected component built from edges with partial correlation magnitudes ≥ 0.02.

The left-most connected component in Figure 6, (#38), is a cluster containing only Methylobacter, a dominant organism in the samples. The next connected component corresponds to the 25-organism bar of Figure 5. Each of these connected components was explored in the Neo4j browser; they can easily be picked out using the query `MATCH (n) where n.ConnectedComponents=0 RETURN n`. Different connected components can be viewed in Neo4j's browser by substituting a different number for the `0`. More sophisticated queries are also possible. For example,

```
MATCH (n)<- [e {cross_species:'True'}] -> (m)
    WHERE n.gene_product =~ '.*transport.*'
        AND e.pcor_abs > 0.045
    RETURN n, m
```

demonstrates restriction to cross-species edges where the first node's gene product contains

the substring"transport" and the partial correlation magnitude is above 0.045. Data exploration so far revealed a mixture of expected genes (such as genes involved in single-carbon compound metabolism), as well as more surprising genes that will be followed up with literature searches.

5 Future work

5.1 Social network detection algorithms

This work provided a foundation for exploration of our lab's partial correlation matrices as graphs using systematic methods. Though this study was restricted to identification of connected components, subsequent analyses will use other community detection tools to pick out subgraphs such as hubs and cycles. The experience gained here provides a jumping point for implementing other community detection algorithms in Java. One promising algorithm, described in [3], finds edges that are least central to communities in the graph and removes them. Such "between" edges can be found by looking for edges that are often used in shortest paths between pairs of nodes.

5.2 Improve input data and loading strategy

Future work will also be done on a larger partial correlation matrix, including 5-10x more species. This motivates more efficient storage of input information for Neo4j. My text-rich input tsv file was 11.5GB, but occupied only 0.18 GB once in the Neo4j database. This is due to the repetitive nature of my edge-oriented input files. Descriptors like the parent organism and the gene product for a particular node was repeated each time that node appeared.

The next iteration of this project will avoid redundancy by keeping separate tables for nodes and edges. Nodes will be entered first, using only unique node IDs. Edges will be added next by matching on the unique node IDs and adding only the partial correlation value and a `T` or `F` label indicating whether it is cross-species. After that scaffold is prepared, I will iterate over the nodes and add information about each node. This may increase the database build times but prevents needing to prepare such large tsv files, which Pandas was already slow to produce for this 50-million edge network.

5.3 Alter schema to enhance browser-based data exploration

It is also worth considering a different schema for the database that would allow coloring by organism type in the Neo4j browser. This would accelerate data exploration by drawing the eye to cross-species edges. The current version's nodes are all of the same type, namely gene. While there are labels indicating what species each node corresponds to, Neo4j's browser only allows node coloring based on node type, not property values. Consequently, all nodes are the same color when I run the browser. The edges are all of the same type as well, preventing coloring by the sign or magnitude of the partial correlation for that edge. A different strategy is to make the node type a name that represents the different species represented in the dataset. Edges could also have two types indicating positive or negative partial correlations.

6 Conclusion

This project was a pilot study for mining interactions between different species in a cultured version of a natural environment. The software developed and insights gained during the course of this project will accelerate future iterations of this project that will have larger partial correlation matrices due to an increase in the number of organisms represented. A similar analysis will likely become a section in my thesis when I defend in April.

References

[1] Ana I Borthagaray, Matías Arim, and Pablo A Marquet. Inferring species roles in metacommunity structure from species co-occurrence networks. *Proceedings of the Royal Society of London B: Biological Sciences*, 281(1792):20141425, 2014.

[2] Alfred Hero and Bala Rajaratnam. Hub discovery in partial correlation graphs. *IEEE Transactions on Information Theory*, 58(9):6064–6078, 2012.

[3] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[4] Matthew T Agler, Jonas Ruhe, Samuel Kroll, Constanze Morhenn, Sang-Tae Kim, Detlef Weigel, and Eric M Kemen. Microbial hub taxa link host and abiotic factors to plant microbiome variation. *PLoS Biol*, 14(1):e1002352, 2016.

[5] Jing Fan, Adalbert Gerald Soosai Raj, and Jignesh M Patel. The case against specialized graph analytics engines. In *CIDR*, 2015.

[6] Daniel Halperin, Victor Teixeira de Almeida, Lee Lee Choo, Shumo Chu, Paraschos Koutris, Dominik Moritz, Jennifer Ortiz, Vaspol Ruamviboonsuk, Jingjing Wang, Andrew Whitaker, et al. Demonstration of the myria big data management service. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 881–884. ACM, 2014.