Network/Graph Visualization of Connectivity Map Query Results

**Abstract**

The Connectivity Map (CMap) is a database of gene expression signatures resulting from perturbation experiments performed on a variety of cell types. Users query the database to find signatures that are very similar or dissimilar to a signature of interest. Due to the current size of the database, such queries yield long and unwieldy hit lists that are challenging to prioritize. The goal of this work is to provide a web-based tool for visualizing CMap query results in a graph or network layout and to refine query results into more actionable hit lists.

**Background**

The Connectivity Map (CMap) is a database of gene expression signatures resulting from perturbation experiments performed on a variety of cell types. The perturbations are diverse in nature, but the three main categories are small molecules, short hairpin RNAs (shRNAs) or over-expression constructs (OEs). These three categories make up the vast majority of profiles in the CMap database, which contains over 400,000 signatures in total.

The utility of the database is that of a gene expression search engine. Researchers may use CMap to find connections between signatures within or external to the database. The goal of CMap is to facilitate hypothesis generation. Hypotheses may be in the form of “Knockdown of gene X connects to knockdowns of pathway Y members, so X is probably a member of Y.” Or perhaps “compound Z connects to knockdown of gene X, so perhaps X is the target of Z”. **Examples (mTOR-rapaymcin, BRAF-vemurafenib).**

A CMap query is a focused question in which a user comes with an input query, called the query signature, and computes the similarity, or connectivity, between his/her query and other signatures in the database. There are various methods for computing connectivity, such as Spearman or Pearson correlations, but a novel metric called the Connectivity Score (CS) has also been implemented. The weighted connectivity score (WTCS) is a variant that weights the step size of traditional CS. **Add more detail about scores**. Positive connectivity indicates that two signatures’ expression changes were similar and vice versa.

Because of the large size of the CMap database, interpreting and prioritizing query results has become a difficult task. Accepting on the top 1% of connections yields nearly 4,000 signatures. The goal of this work is to assist in further refining these query results and in reducing the number of potential follow-ups to something more actionable.

**Other approaches to query result refinement and/or network visualization**

Summly

A large number of CMap perturbagens have been profiled across multiple cell lines. Summly is an algorithm that leverages these multiple contexts to distil the core biological functions of a pertubagen. It does so by rewarding connections that occur in multiple cell lines, thus yielding a more pertubagen-centric view of query results.

PageRank

View the Internet as graphs and find the nodes that are most important.

**My approach**

Leverage the existing all-by-all matrix to do lookups instead of connectivity computation.

1. Pre-compute a graph of all N signatures (possibly only in Summly space), where each node corresponds to a CMap signature. The nodes in the graph are gene expression signatures from the CMap database and the edges are connectivity scores between the signatures. The non-existence of an edge indicates a connectivity score of 0.
2. Insert edges between all pairs of nodes having nonnegative connectivity scores between them. Could also replace connectivity scores with Summly scores to further reduce the size of graph.
3. For a given query, retain the top M results. (*Aside: might be able to compute M by plotting median connectivity vs. number of results retained. Once this starts to drop off, we know we’re no longer seeing consistent results*). Highlight these nodes in the graph. Their proximity should be indicative of whether there exist a consistent “theme” in the connections.
4. Filter out non-specific hits (those that connect to many other signatures).
5. Cluster the M results by using k-means. Use 1 – WTCS as the distance metric. This will yield k distinct clusters, and may give insight into the biological themes present. Could potentially use consensus clustering, but this might be overkill.
6. Compute within cluster sum of squares (WCSS) to get at cluster tightness. Also use GAP statistic?
7. Additionally, compute a word cloud for all existing annotations for the M connection signatures and display this alongside. Possibly split this by cluster.

**How to test/validate**

To validate clustering, I will make a test dataset with random signatures and then some number of groups of well-characterized signatures spiked in. I should see that these signatures cluster together. Ex: random signatures with a spike-in of HDAC inhibitors.

To validate utility of network visualization and highlighting might simply need to get biologist feedback. Ask them whether results are usable.

**Technical Details**

Back end

All-by-all connectivity matrix is stored as HDF5 binary file.

Convert to JSON and store in MongoDB.

MongoDB made accessible via NodeJS RESTful API.

May need to do clustering on server.

Front end

HTML5 / JavaScript application

D3 for drawing nodes

No computation done here, simply lookups from database.

The final app should be a webpage where someone inputs a list of signature IDs corresponding to their query results. They then see:

1. The top M results highlighted on the connectivity graph
2. The M results clustered
3. Word cloud(s) for the M results and/or for each cluster
4. Click or mouseover reveals additional info about a particular signature

**Potential challenges and solutions**

1. Visualizing a massive network in the browser. Will this be cumbersome?
   1. Could use something like edge bundling to show only a high-level view of the network outside of the hits.
   2. Could use a zooming feature where the entire graph is shown but obscured and only the local area is highlighted.
   3. Could also restrict the size of the graph in a few ways.
      1. Show only gold signatures
      2. Only use Summly space (core cell lines)
      3. Stratify by perturbation type (cp, oe, sh)
2. Computing top M most relevant hits in the browser. Will this be slow?
   1. Could offload to a server such as AWS
   2. Could require this to be pre-computed in a separate app
3. Running the clustering in the browser. This may be slow for long hit lists.
   1. Could offload to a server such as AWS
   2. Could require this to be pre-computed in a separate app