Using MCL to extract clusters from networks

Stijn van Dongen[1,*], Cei Abreu-Goodger[1]

[*] corresponding author
[1] European Molecular Biology Laboratory, European Bioinformatics Institute. Wellcome Trust Genome Campus, Hinxton, Cambridge, CB10 1SD, UK. email: stijn@ebi.ac.uk, cei@ebi.ac.uk

Abstract

MCL is a general purpose cluster algorithm for both weighted and unweighted networks. The algorithm utilises network topology as well as edge weights, is highly scalable and has been applied in a wide variety of bioinformatic methods. In this chapter we give protocols and case studies for clustering of networks respectively derived from protein sequence similarities and gene expression profile correlations.

keywords

Network clustering, cluster analysis, protein sequence similarity, gene expression profiles.

1 Introduction

Networks, also known as graphs, are composite structures in which *nodes* are connected by weighted links, called *edges*. An edge always connects two nodes, and a node may be connected to multiple nodes, called its *neighbours*, by as many edges. The nodes usually represent members from a well-defined class of objects, such as genes, proteins, or experimental conditions. An edge may have associated with it a weight that confers a degree of similarity or dissimilarity between the objects connected by the edge. In network analysis in bioinformatics, edge weights predominantly play the role of similarities, and such is the case here.

The MCL algorithm, or MCL in short, stands for Markov CLuster algorithm. It computes stochastic flow through a network by alternating dissipation and reinforcement steps [1–3]. This defines a simple algebraic bootstrapping process in which cluster structure is found as the result. The number of clusters can not and need not be specified, but a single parameter called *inflation* controls the reinforcement step, so that clusterings at different levels of granularity can be found. This is the only parameter that will be varied.

MCL was first published as [1, 2] and then applied to protein sequence networks [4, 5]. It has subsequently been used, among others, for detecting orthologous groups [6], predicting protein complexes from protein interaction networks [7,8], clustering of genes according to their expression profiles [9–11], and clustering of scientific documents [12]. Ensembl families [13] are computed using the same general approach as proposed in [4], and this is detailed in Protocol 1. Protocol 2 provides a recipe for clustering genes according to expression profiles. Both protocols are accompanied by a case study.

The protocols described here are relatively simple and exclusively use command line tools, namely (the program) **mcl** itself, sibling applications that are shipped and installed with **mcl**, and standard UNIX commands. This manner of processing offers many benefits, such as off-line processing, flexible coupling and decoupling of components, adaptability and scalability. The software can be downloaded from `http://micans.org/mcl`, where documentation and installation guidelines are provided as well. Assistance can be sought and bug reports submitted on a mailing list[1]. On Windows **mcl** can run under the freely available Cygwin environment.

No real alternatives exist for Protocol 1, but a more specialised version is available as OrthoMCL [14]. Regarding Protocol 2, BioLayout Express3D is a comprehensive tool for analyzing expression data [10,15], integrating large scale data analysis and network visualisation with the ability to use **mcl**. A recently published approach using MCL is DBF-MCL [11]. Finally, MCL is also integrated in the web services[2] provided through Network Analysis Tools or NeAT (Brohée, this volume; [16]). Another method for clustering networks is Restricted Neighbourhood Search Clustering (King, Pržulj and Jurisica, this volume; [17]). It is a fast and general purpose method, currently utilising absence/presence of edges only without regarding weights.

Protocol 1 describes the analysis and prediction of family relationships between proteins based on their sequence similarity. It starts with a set of files resulting from running **blastall -p blastp** on FASTA files encoding protein sequences, and details the construction, clustering, and analysis of the protein sequence similarity network. The supplementary material gives examples for obtaining FASTA files and running **blastall**.

In Protocol 2 the aim is to group together and analyse genes that show correlated or possibly anti-correlated expression across different conditions. Starting with a table of array expression values for a set of genes measured across different conditions, it describes the necessary steps to create, cluster and analyse a gene expression profile network.

[1] `http://micans.org/mcl/#questions`
[2] `http://rsat.ulb.ac.be/rsat/index_neat.html`

1.1 Constructing appropriate networks

The definition of a network involves several choices. Important among these are the selections of nodes and edges and the similarities assigned to edges. Generally, choices should be based on the appropriateness for the analysis and the methods one intends to apply. We illustrate this in the particular case of MCL and the protocols introduced above.

MCL is affected by differentiation among the edge weights. Often a network has edges between distant objects, in addition to edges between closer objects. Equally often the former are more numerous than the latter, and can play a vital role as third party support in pulling the objects into meaningful groups, i.e. clusters. For what follows it is convenient to use self-identities as a measure for maximally close nodes. As a rule of thumb in the context of MCL, distant nodes, if present, should have similarities between them that are between one and two orders of magnitude smaller than self-identities. If distant similarities otherwise get bigger or smaller they exert too much or too little pull, respectively.

The raw data in Protocol 1 are E-values for BLAST scores between two sequences. A naive transformation would take the inverse of these E-values to arrive at a similarity. E-values of 1e-160 and 1e-20 are thus mapped to similarities 1e160 and 1e20, with 140 orders of magnitude between them, and the second similarity might as well be zero. The normal, widely used transformation step uses a log-transformation followed by a sign reversal, leading to similarities 160 and 20, with approximately one order of magnitude between them.

The raw data in Protocol 2 are (absolute values of) Pearson correlations between different probes on the array across series of experiments. A cutoff is chosen in order to keep the data manageable and meaningful. In our case study 1 we use a cutoff of 0.6. If left at that, similarities vary in the interval $[0.6 - 1]$. However, this falls short if we desire to attach much greater importance to correlation values close to one compared to values close to 0.6. We note that correlations in the lower ranges are much more common, yet they are in the same order of magnitude as high correlations and have a comparable impact on MCL. The approach chosen is to shift the interval $[0.6-1]$ to the interval $[0-0.4]$,which results in a gradual fading out of the similarities between more distant probes.

A few concluding remarks are in order. Often a threshold is chosen so that similarities below it are excluded. Examples are a transformed E-value cutoff and the correlation cutoff. Ideally the data are transformed in such a manner that the network changes gradually as the cutoff is varied. The transformations given here satisfy this property. Second, other choices can be made in order to tune the network for a given task. A good example is the prediction of orthology and paralogy from simple sequence similarity networks such as performed by OrthoMCL[1]. In this case one must be careful to choose species that evenly sample the species subtree in which one is interested. Additionally, OrthoMCL submits the input network to a reduction step in which only putative orthologs and paralogs are considered by using best reciprocal hit derived criteria, followed by an augmentation step adding so called co-orthologs. Finally, the network construction step should be seen as separate from the clustering method itself.

Sometimes these are lumped together, complicating the comparison between different clustering methods.

1.2 Case study 1: Clustering based on sequence similarity

Sequence similarity is widely used to transfer functional annotation between genes. This process leads to errors, given the entangled nature of gene evolution. Duplications give rise to paralogs, which can then functionally diverge while maintaining high levels of sequence similarity. Many proteins are composed of more than one domain, each with different functional properties and even different evolutionary histories. Complex phylogenetic analysis can sometimes be required to identify the correct evolutionary relationships between genes. A simple but powerful approach involves clustering a network based on sequence similarity. If there is detectably more sequence similarity between orthologs than between paralogs, a good clustering algorithm should be able to separate these two groups.

Orthologs may be difficult to find, particularly if recent duplications have occurred, or if the network contains sequences from more than one group of closely related organisms. In these cases the cluster structure will reflect this, some clusters will mostly contain sequences from closely related organisms and others will contain paralogs. This structure can still be useful for understanding the phylogenetic relationship between organisms, but shall not be further discussed in this chapter.

The first protocol focuses on clustering a protein sequence similarity network. We use the proteomes of twenty eight fully-sequenced genomes of the order Rickettsiales. These intracellular alphaproteobacteria have complex life cycles, involving arthropod and mammalian hosts, and their highly streamlined genomes have become the paradigm of reductive evolution (for a review, see [18]). They also represent the closest extant relatives of the ancestor to mitochondria. Beyond having small genomes, Rickettsiales can still be quite different, reflecting several hundred million years of evolution. In this case study, we will show examples of how orthologs are separated from paralogs with MCL.

1.3 Case study 2: Clustering based on expression profiles

The use of microarrays for analysing gene expression is widespread. After measuring the expression of thousands of genes over a variety of conditions, clustering the results can help to visualise the data and gain further functional insights [19]. The cluster structure can then be used to predict functional relationships amongst each set of co-expressed genes, assuming that co-expression implies co-regulation.

In bacteria, many genes are organised in operons, which are transcribed in a coordinated manner. Co-regulation of multiple operons can be caused by transcription factors, leading to the formation of so-called regulons. In optimised bacterial genomes, genes that belong to the same functional pathway (e.g. the biosynthesis of an amino acid) are expected to be co-regulated: when the amino acid is present, they can all be turned off and when the amino acid is absent, they can all be turned on. The same logic can be applied to protein complexes, where the level of the protein subunits should be regulated in a coordinated manner. If these kinds of functional relationships demand co-expression, co-expression should then be able to be used to predict them.

Expression data can be noisy: two genes can show apparent co-expression for technical reasons such as probe cross-hybridisation. Given the scale of microarray experiments, where the number of genes is usually much higher than the number of conditions, some level of co-expression can be expected simply by chance. Clustering can help overcome some of these problems, by finding groups of consistently co-expressed genes across multiple conditions. It is worthwhile noting that more elaborate filtering steps can enhance the results. Conserved co-expression, either using different species or comparing paralogous pairs of genes in a single species, has been shown to improve functional prediction derived from co-expression analysis [20].

For this case study we use Affymetrix *Escherichia coli* expression data from the Many Microbe Microarray Database [21]. The dataset comprises 466 different experimental conditions, enabling us to extract clusters of diverse functions. The advantage of working with *E. coli* is the wealth of available annotation, both at the level of gene function and regulation [22, 23]. We rely on these annotations to exemplify how different levels of clustering can yield important functional insights.

2 Material

2.1 Software

The software required consists of programs provided in the MCL distribution, available from `http://micans.org/mcl/`. A version at least as recent as 09−320 should be used. Here the version number is of the form `YY-DDD`, encoding the year and the day-in-year. The programs used are **mcl**, **mcxload**, **mcxdump**, **mcxarray**, **clm**, and **mcx**. These are installed by default. The **clm** program implements various analysis modes for clusterings and networks, and is invoked as **clm** *mode*. The **mcx** program similarly supplies analysis modes for networks. An example is **mcx query**, used to gauge graph properties as a graph is subjected to increased levels of edge weight thresholding. Additionally the script **clxdo** is found in the `scripts/` subdirectory of the unpacked MCL source, and the standard UNIX commands **cut** and **sort** are used.

2.2 File formats

The **mcl** program allows two types of input. The first type is called ABC-format, useful for simple protocols, and triggered by the `--abc` option to **mcl**. Input is read line by line, each line encoding an edge as two labels separated by whitespace, optionally followed by a numeric similarity. The clustering is output in another line-based format, where a single line encodes a clustering as a list of white-space separated labels.

For more complex protocols such as described here it is useful to have a more abstract representation of a network. This format is obtained by feeding the ABC-format to **mcxload**. It outputs a network in *native mcl format* as well as a *dictionary file* allowing the mapping of the native format onto the original label format. By default **mcl** writes output clusterings in native format as well. The program **clm** has various analysis modes such as **order**, **close** and **dist**, all of which understand this format. The program **mcxdump** maps results back to the original labels using a dictionary file, which has a simple two-column format consisting of an **mcl** identifier and the label it refers to. **mcxdump** can output ABC-format, line-based cluster dumps, as well as structured tables.

Protocol 1 uses as a starting point files resulting from running **blastall -p blastp** with parameter `-m8` on FASTA files encoding protein sequences. The `-m8` options causes **blastall** to output a concise tabular output. The information that is relevant to **mcl** is present in columns 1, 2, and 11, namely the identifiers of the two sequences being compared and the E-value.

Protocol 2 uses a table of normalised intensity values from Affymetrix microarrays. This is the standard way of representing processed microarray results for single-channel experiments.

2.3 Sample files

Case study 1 uses FASTA files downloaded[3] from Uniprot [24] for twenty eight genomes of the order Rickettsiales, containing sequences for a total of 31374 proteins. Case study 2 uses *E. coli* expression data downloaded[4] from the Many Microbe Microarrays Database [21]. It contains expression values for 4297 genes measured across 466 conditions.

3 Methods

Both protocols generate clusterings at different levels of granularity, and we consider it useful to analyse the data in this wider context. A rigid strategy for choosing a single clustering is not given. Criteria for such strategies differ from case to case and depend on the research objectives. Clusterings at different levels of granularity may summarise the data equally well, albeit from different points of view. A fine-grained clustering implies a conservative view of family relationships, whereas a coarse-grained clustering implies a more permissive view. MCL can generate meaningful clusters at very different levels of granularity, as demonstrated in case study 2.

Three simple means to compare different clusterings are supplied, namely by granularity, by distance, and by edge weight capture, but it is suggested to

[3]`http://www.uniprot.org/`
[4]`http://m3d.bu.edu/norm/`

use these as measurement tools rather than decision criteria.

For simplicity, both protocols put all generated files in the same directory. A more organised way of working would use subdirectories such as `fasta/`, `blast/` and `clusterings/`, but we leave these improvements to the practitioner. A general guideline for the commands given below is that after the program name any word that starts with a hyphen is an *option* to the program. If it starts with a single hyphen the option takes an argument, which is the next word. If it starts with two hyphens the option takes no argument and sets a modality.

The first protocol is elaborate in that it shows many ways to assess clusterings. The majority of these are not repeated in the second protocol but are nevertheless applicable there as well.

3.1 Protocol 1: Clustering protein sequence similarity networks

The first command to run requires that **blastall -p blastp** results were obtained using the **-m8** parameter. The blastall output files are assumed to reside in the current directory and have names with suffix `blast`. The supplementary material has examples of running **blastall**. Below the ↩ symbol indicates that the example continues on the following line and that <**ENTER**> or new-line should not be pressed.

1. The following command reads the BLAST results and creates a file in ABC-format.

   ```
   cut -f 1,2,11 *blast > seq.abc
   ```

2. The ABC-format file `seq.abc` is read by **mcxload**, which creates a network file called `seq.mci` and stores the label information in a separate file `seq.dict`.

   ```
   mcxload -abc seq.abc -write-tab seq.dict -o seq.mci --stream-mirror ↩
      --stream-neg-log10 -stream-tf 'ceil(200)'
   ```

   The first part to the command, on the first (continued) line, represents the standard way of loading networks in ABC-format. The `--stream-mirror` option ensures that an undirected network is obtained, which is preferable when using **mcl**. The last continued line is *specific to this protocol.* It specifies a negative log-10 transformation and additionally edge weights are capped to a maximum value of 200. The `-tf` infix in the `-stream-tf` option stands for 'transformation', and several other programs in the MCL suite accept such an option.

3. Now run **mcl** with different inflation parameters. The first argument to **mcl** is always the input file. By default it is assumed to be a network file.

   ```
   mcl seq.mci -I 1.4
   mcl seq.mci -I 2
   mcl seq.mci -I 6
   ```

The inflation values here represent a good first set of parameters. Lower and higher inflation values yield respectively coarser and finer clusterings. File names `out.seq.mci.I14`, `out.seq.mci.I20`, and `out.seq.mci.I60` are automatically created by **mcl**. The option `-o` can be used to override this.

4. The output can be transformed to label output as follows.

   ```
   mcxdump -icl out.seq.mci.I14 -o dump.seq.mci.I14 -tabr seq.dict
   ```

   This generates a format with all labels for a cluster put together on a single line separated by tab characters.

5. Different clusterings output by **mcl** are not guaranteed to be strictly nested, although usually they are *almost fully* nesting, as seen further below. A strictly nesting set of clusterings is created as follows.

   ```
   clm order -prefix P out.seq.mci.I{14,20,60}
   ```

   This generates from the list of clusterings specified a new set of clusterings `P1`, `P2`, and `3`, each of which is a strict superclustering of all the following.

6. It is possible to encode the nested clusterings as a tree, in this case of depth just three.

   ```
   clm order -o seq.mcltree out.seq.mci.I{14,20,60}
   mcxdump -imx-tree seq.mcltree -tab seq.dict --newick -o tree.nwk
   ```

   The file `seq.mcltree` is a concatenation of networks in in **mcl**'s native format. It is transformed to Newick format by **mcxdump**. The branch lengths simply indicate the level at which nodes and clusters join, and should not be given any further meaning.

7. Small clusters are to some extent explained by the input graph already consisting of different components. Therefore it is useful to obtain the baseline coarse clustering consisting of the natural maximum components of the input graph (also known as *connected components*). The program **clm close** does just this.

   ```
   clm close -imx seq.mci --write-cc -o seq.base
   ```

8. The granularity of a clustering is gauged from the list of cluster sizes. Smaller clusters are sometimes not so interesting, so the second command below specifies a minimum cluster size. We include the previously obtained connected components in this analysis.

   ```
   clxdo granularity seq.base out.seq.mci.I{14,20,60}
   clxdo granularity_gq 5 seq.base out.seq.mci.I{14,20,60}
   ```

9. It is very useful to assess how close two clusterings are. To this end, issue the following.

   ```
   clm dist --chain out.seq.mci.I{14,20,60}
   ```

   The `--chain` option causes only consecutive clusterings to be compared, whereas the default behaviour is to do all pairwise comparisons. The first number reported is the distance between two clusterings, which may be used to gauge quantitative differences between them. It should be interpreted as the number of nodes required to change location in one clustering in order to obtain the other [25].

   In our example we obtain distances 2977 and 3949 respectively, translating to node fractions of approximately 9 and 13 percent. This indicates that the three clusterings are genuinely very different in terms of granularity. In contrast, the clustering obtained with inflation 7 (using `-I 7`) has a distance of 364 to the result at inflation 6. This translates to a fraction of approximately one percent of nodes that needs relocation between the two clusters, indicating that in this example the clusterings obtained at high inflation levels are very stable. This likely means that the clusters represent tight sets of highly conserved sequences that do not easily split further.

   The *dist* mode to **clm** can also be used to judge how far the projections computed by **clm order** are from the original MCL clusterings.

   ```
   clm dist P1 out.seq.mci.I14
   clm dist P2 out.seq.mci.I20
   clm dist P3 out.seq.mci.I60
   ```

   The answers obtained in our case study are $\{0, 32, 102\}$, meaning for example, that only 102 nodes need relocation between `P3` and `out.seq.mci.I60`. To recap, the projected clusterings are strictly nesting, and additionally very close to the MCL clusterings from which they were derived. This implies that the latter are *almost* fully nesting, and hence represent compatible results at different levels of granularity. To obtain a strictly nested set of clusterings the program **clm order** can be used.

10. Finally, **clm** provides an **info** mode that gives a simple numerical criterion for a clustering. It rewards clusterings both for being granular and for capturing many edges in the input graph. Its criterion lies in the range $[0-1]$ and achieves 1 only for the natural clustering of a graph that falls apart into node sets (components) such that no edges between different components exist and edges are present for all node pairs that fall within the same component. In addition, it is affected by differentiation among the edge weight. It is not intended as an optimisation criterion, but can be used to detect trends and optionally to find bad apples (clusterings).

```
clm info seq.mci out.seq.mci.I{14,20,60}
```

In our example the results are $\{0.76, 0.82, 0.86\}$, compared with an efficiency of 0.47 for the baseline clustering of maximal components in the input graph. This suggests satisfactory consistency and performance. As a general guideline, as graphs include more distant relationships (lower similarities) and as inflation is lowered (leading to coarser clusterings), this criterion will degrade. This does not mean those results are invalid. However, a sharp drop in this criterion could be reason to distrust the pertinent clustering.

3.2 Protocol 2: Clustering gene expression profiles

The protocol uses as example a tabular file `ecoli.exprs` containing expression values for genes across a series of experiments. The program **mcxarray** is used to create a network file and a dictionary file.

1.
   ```
   mcxarray -data ecoli.exprs -co 0.2 -skipr 1 -skipc 1 -tf 'abs()' ↩
      -o ecoli20.mci -write-tab ecoli.dict
   ```

   In this step we create a test network for the purpose of finding a reasonable correlation cutoff. Hence a very low correlation cutoff of 0.20 is used. The `-tf 'abs()'` instruction reverses the sign of negative values, implying that we consider correlations and anti-correlations alike. The first line, containing column names, is skipped according to `-skipr 1`. The `-skipc 1` instruction causes the first column in the tabular input file to be skipped, and numerical input will be expected in all other columns. The program is asked to write labels for the rows with `-write-tab ecoli.dict`, and it deduces that these labels must be found in the first (skipped) column.

2.
   ```
   mcx query -imx ecoli20.mci --vary-correlation
   ```

   The correlation cutoff is varied from 0.20 to 0.95 in increments of 0.05, and at each cutoff the graph resulting from removing lower correlations is analysed. Reported attributes of the resulting graph include the size of its largest component, the number of singletons (isolated nodes), and node degree and edge weight statistics (mean, median, and inter-quartile range). In this example we are prepared to choose a relatively low threshold as the genes were measured across many experiments. We wish to have a small number of singletons (in the initial graph) while keeping the average node degree in check. In this example we choose a cutoff of 0.60, at which seven percent of the nodes are singletons (not connected to any other node), and at which the node degree mean and median are 98 and 36 respectively.

3.
   ```
   mcx alter -imx ecoli20.mci -tf 'gq(0.6), add(-0.6)' -o ecoli60.mci
   ```

   After choosing a correlation cutoff in the previous step, we create a more selective network by applying this cutoff, as instructed by the `gq(0.6)` part of the argument to `-tf`. The correlations are shifted to the interval $[0-0.4]$

by subtracting 0.6 from all the values, increasing the contrast between lower and higher correlations, as required for MCL. It is suggested to set the shift value to the cutoff value.

4. We apply **mcl** with the same set of inflation values as before. After this, the programs from the first protocol can be used as before.

```
mcl ecoli60.mci -I 1.4
mcl ecoli60.mci -I 2
mcl ecoli60.mci -I 6
```

3.3 Application of the method to study case 1: Clustering based on sequence similarity

In Figure 1 we have plotted the cumulative size distribution of all clusters obtained at different inflation values. For a given cluster size N represented along the X-axis, the Y-axis represents the fraction of all genes contained in a cluster of size at most N. Notice that at any inflation value there is a large number of clusters of size 28. These represent well defined groups of orthologs, containing one protein from each species. In order to visualise what is happening during the clustering process, we highlight three examples and track them across different inflation values (protein annotation was obtained from Uniprot [24]).

[Figure 1 about here.]

RNA polymerase σ^{70} (RpoD) is required for the transcription initiation of housekeeping genes. Rickettsiales also encode a heat-shock sigma factor σ^{32}. The connected components of the network, as well as low inflation clusterings, group all the sigma factors together. At inflation 2 and above, they correctly split into two clusters. The σ^{70} cluster contains 30 proteins, one from 27 of the species and 3 from *Rickettsia bellii* OSU 85-389 whose σ^{70} is annotated as being split into three separate proteins. The σ^{32} cluster contains the remaining 27 sigma factors, with *Rickettsia bellii* OSU 85-389 lacking a σ^{32} in its annotated proteome.

Bacterial ATP synthase is a multimeric enzyme comprised of 8 subunits. The α and β subunits are homologous, sharing a highly conserved nucleotide-binding domain which is also present in the transcription termination factor Rho. Initially, 91 of these proteins are connected, but different levels of inflation can split them apart. As seen in the inset of Figure 1, low inflation generates two clusters, one with 57 ATP synthase subunits and one with 34 Rho terminators (several *Anaplasma* and *Ehrlichia* encode two copies of Rho). At inflation values of 2 and above, the ATP synthases correctly split into two clusters. The first one contains 29 α subunits, with *Rickettsia bellii* RML369-C encoding a split protein. The second cluster contains the 28 ATP synthase subunit β proteins.

ATP-binding cassette (ABC) transporters represent one of the largest protein families in bacteria. In the unclustered network, ABC transporters lie in the single largest connected component of 10,012 proteins. Many other families of proteins are also included in this component, for instance DNA polymerase I. This protein in *R. felis* contains a *Rickettsia* palindromic element [26], giving rise to low-score BLAST hits to many unrelated protein families. These low-weight connections are amongst the first to be split by MCL, leading to the separation of DNA polymerase I and other orthologous groups at the lowest levels of inflation. Increasing inflation leads to further splitting of the largest cluster, with high values leading to many ABC transporters being clustered with consistent functional annotation, such as the Zinc import ATP-binding protein ZnuC, the Ribonucleotide ABC transporter and the Glutamine ABC transporter. At inflation of 6 the largest remaining cluster contains exclusively ADP/ATP carrier proteins, a family of proteins required by Rickettsiales to obtain ATP from the host cell in exchange for ADP.

3.4 Application of the method to study case 2: Clustering based on expression profiles

The correlation of gene expression values yields a very different network, as seen in Figure 2. Due to the low correlation cutoff (0.6), almost all the genes are initially part of a single connected component. Increasing inflation continuously breaks the network down, with a tendency at the end towards clusters with 2 or 3 genes, mostly representing operons. Unless cited, the information discussed in this section was obtained from EcoCyc [23].

[Figure 2 about here.]

One of the most consistent clusters comprises a set of genes related to motility. This cluster includes the flagellum operons: *flgAMN*, *flgBCDEFGHIJ*, *flgKL*, *flhBAE*, *flhDC*, *fliAZY*, *fliC*, *fliDST*, *fliE*, *fliFGHIJK*, *fliLMNOPQR* and *motAB*; chemotaxis genes: *cheAW*, *tar-tap-cheRBYZ*, *trg* and *tsr*; the aerotaxis gene *aer* and several others. This suggests that other members of this cluster are likely to be functionally connected to motility, giving us novel predictions for genes with no known function such as *ymdA*, *ynjH*, *yjcZ* and *yecR*.

Another interesting cluster contains genes related to the SOS response, which is induced by DNA damage [27]. At lower inflation values it also contains many prophage genes. At high inflation levels, all of the prophage genes are split off, leaving a core set of genes all of which are involved in the SOS response and DNA repair: *lexA-dinF*, *dinB*, *dinD*, *dinG*, *dinI*, *recAX*, *recN*, *sulA*, *symE*, *umuDC*, *uvrB*, *ydjM* and *yebG*. The set of prophage genes which were initially in the cluster could also merit further analysis, since silent bacteriophage genes are known to be induced by the SOS response [27].

Many groups of genes with related functions are less tightly co-regulated than the previous examples. For instance, one of the largest clusters at low inflation contains the genes for all three nitrate reductases encoded by *E. coli* as well as many other reductase enzymes. By using any GO enrichment tool such as DAVID [28] we can see that this cluster is highly enriched for the Biological

Process *anaerobic respiration.* At higher inflation values, many of the complexes start to split off into new clusters, and the enrichment score for *anaerobic respiration* falls. Some of the first genes to form a new cluster are *ccmBCDEFGH*, components of the CcmCDE protoheme IX reservoir complex and the CcmEFGH holocytochrome c synthase. At higher inflation these complexes separate into two distinct clusters. Meanwhile the nitrate reductases are also split into different clusters, containing the structural genes for each complex as well as some of its accessory proteins. An interesting point is that a single operon consists of *napFDAGHBC-ccmABCDEFGH*, with alternative promoters allowing distinct transcription of *napFDAGHBC* and *ccmABCDEFGH.* Nevertheless, the clustering process is able to separate all these genes into three groups that are consistent with the complexes they are known to form.

3.5 Conclusions

With these study cases we have illustrated the power of MCL to cluster networks of different complexity, even without any prior preprocessing. We have shown the relevance of varying the inflation parameter to obtain different clusterings, all of which can be interesting and biologically relevant. We remind the practitioner that different problems require different considerations, and preprocessing the underlying network may be desirable to accommodate focused research questions.

References

1. van Dongen S: **A cluster algorithm for graphs**. Tech. rep., National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam 2000.

2. van Dongen S: **Graph Clustering by Flow Simulation**. *PhD thesis*, University of Utrecht 2000.

3. van Dongen S: **Graph Clustering Via a Discrete Uncoupling Process**. *SIAM Journal on Matrix Analysis and Applications* 2008, **30**:121–141.

4. Enright A, van Dongen S, Ouzounis C: **An efficient algorithm for the large-scale detection of protein families**. *Nucleic Acids Research* 2002, **7**:1575–1584.

5. Enright AJ, Kunin V, Ouzounis CA: **Protein families and TRIBES in genome sequence space**. *Nucleic Acids Res.* 2003, **31**:4632–4638.

6. Li L, Stoeckert C, Roos D: **OrthoMCL: Identification of ortholog groups for eukaryotic genomes**. *Genome Research* 2003, **13**:2178–2189.

7. Pereira-Leal JB, Enright AJ, Ouzounis CA: **Detection of Functional Modules From Protein Interaction Networks**. *PROTEINS: Structure, Function, and Bioinformatics* 2004, **54**:49–57.

8. Brohée S, van Helden J: **Evaluation of clustering algorithms for protein-protein interaction networks**. *BMC Bioinformatics* 2006, **7**:488.

9. Samuel Lattimore B, van Dongen S, Crabbe MJ: **GeneMCL in microarray analysis**. *Comput Biol Chem* 2005, **29**:354–359.

10. Freeman TC, et al.: **Construction, visualisation, and clustering of transcription networks from microarray expression data**. *PLoS Comput. Biol.* 2007, **3**:2032–2042.

11. Lopez F, et al.: **TranscriptomeBrowser: a powerful and flexible toolbox to explore productively the transcriptional landscape of the Gene Expression Omnibus database**. *PLoS ONE* 2008, **3**:e4001.

12. Theodosiou T, et al.: **PuReD-MCL: a graph-based PubMed document clustering methodology**. *Bioinformatics* 2008, **24**:1935–1941.

13. Hubbard TJ, et al.: **Ensembl 2009**. *Nucleic Acids Res.* 2009, **37**:D690–697.

14. Chen F, et al.: **Assessing performance of orthology detection strategies applied to eukaryotic genomes**. *PLoS ONE* 2007, **2**:e383.

15. Theocharidis A, et al.: **Network visualization and analysis of gene expression data using BioLayout Express(3D)**. *Nat Protoc* 2009, **4**:1535–1550.

16. Brohee S, Faust K, Lima-Mendez G, Sand O, Janky R, Vanderstocken G, Deville Y, van Helden J: **NeAT: a toolbox for the analysis of biological networks, clusters, classes and pathways**. *Nucleic Acids Res.* 2008, **36**:W444–451.

17. King AD, Przulj N, Jurisica I: **Protein complex prediction via cost-based clustering**. *Bioinformatics* 2004, **20**:3013–3020.

18. Darby AC, et al.: **Intracellular pathogens go extreme: genome evolution in the Rickettsiales**. *Trends Genet.* 2007, **23**:511–520.

19. d'Haeseleer P: **How does gene expression clustering work?** *Nat. Biotechnol.* 2005, **23**:1499–1501.

20. van Noort V, Snel B, Huynen MA: **Predicting gene function by conserved co-expression**. *Trends Genet.* 2003, **19**:238–242.

21. Faith JJ, et al.: **Many Microbe Microarrays Database: uniformly normalized Affymetrix compendia with structured experimental metadata**. *Nucleic Acids Res.* 2008, **36**:D866–870.

22. Gama-Castro S, et al.: **RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation**. *Nucleic Acids Res.* 2008, **36**:D120–124.

23. Keseler IM, et al.: **EcoCyc: a comprehensive view of Escherichia coli biology**. *Nucleic Acids Res.* 2009, **37**:D464–470.

24. Bairoch A, et al.: **The Universal Protein Resource (UniProt) 2009**. *Nucleic Acids Res.* 2009, **37**:D169–174.

25. van Dongen S: **Performance criteria for graph clustering and Markov cluster experiments**. Tech. rep., National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam 2000. [URL: http://www.cwi.nl/static/publications/reports/INS-2000.html.].

26. Ogata H, Audic S, Barbe V, Artiguenave F, Fournier PE, Raoult D, Claverie JM: **Selfish DNA in protein-coding genes of Rickettsia**. *Science* 2000, **290**:347–350.

27. Neidhardt FC, Curtiss R: *Escherichia Coli and Salmonella: Cellular and Molecular Biology.* ASM Press, Washington, 2nd edition 1996. [Walker GC. The SOS response of Escherichia coli. pp. 1400-16].

28. Huang DW, Sherman BT, Lempicki RA: **Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources**. *Nat Protoc* 2009, **4**:44–57.

List of Figures

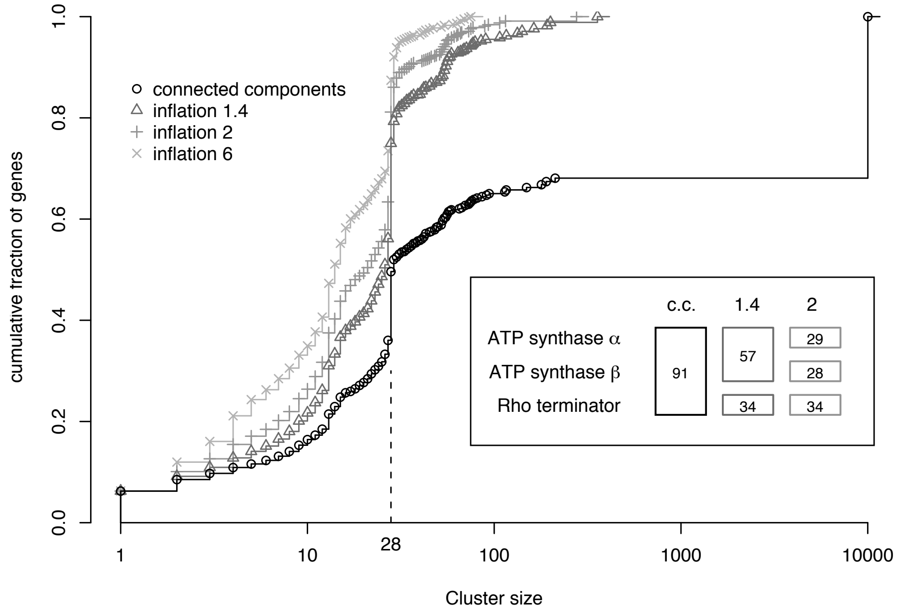

Figure 1: Cluster size distributions of the Uniprot sequence similarity network

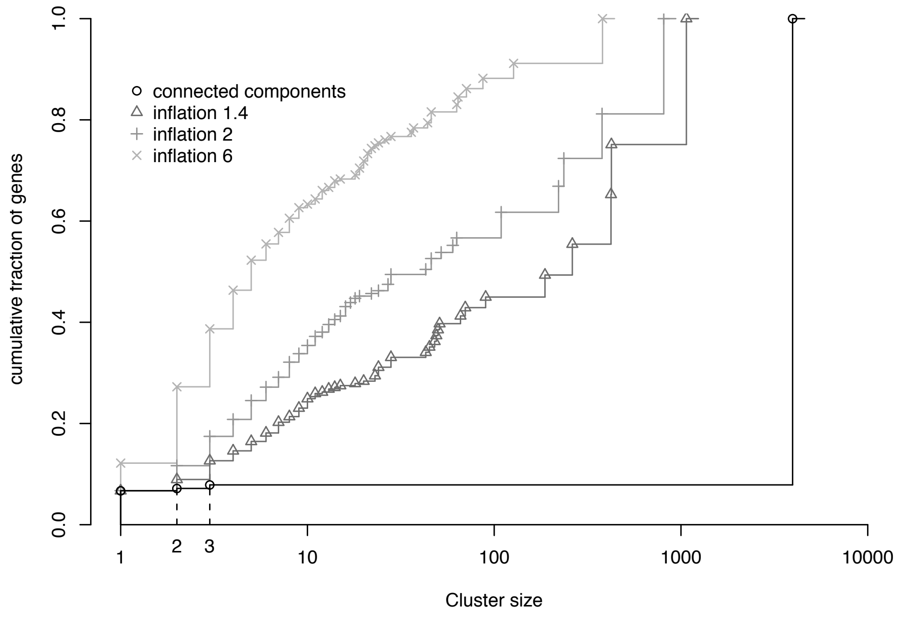

Figure 2: Cluster size distributions of the *E. coli* gene expression network