Network Analysis Tools (NeAT)
Tutorial

Sylvain Brohée
sbrohee@ulb.ac.be

Karoline Faust
kfaust@ulb.ac.be

Jacques van Helden
jvhelden@ulb.ac.be

Laboratoire de Bioinformatique des Génomes et des Réseaux (BiGRe)
Laboratory of Genome and Network Biology
Université Libre de Bruxelles, Belgium
http://www.bigre.ulb.ac.be

January 22, 2008

Contents

0.1 Warning

This tutorial is in construction. The current version only covers a very small fraction of the ***NeAT***tools. For the tools not covered yet by the tutorial, the `DEMO` buttons already give some hints about typical cases of utilization. We intend to develop further those tutorials very soon.

Chapter 1

Introduction

Since a few years, large scale biological studies produced huge amounts of data about networks of molecular interactions (protein interactions, gene regulation, metabolic reactions, signal transduction). The integration of these data sets can be combined to acquire a global view of the pieces that, altogether, contribute to the complexity of biological processes. High-throughput data is however notoriously noisy and incomplete, and it is important to evaluate the quality of the different pieces of information that are taken in consideration for building higher views of biological networks.

An important effort will be required to extract reliable information from the ever-increasing ocean of high-throughput data. This will require the utilization of powerful tools that enable us to apply statistical analysis on large graphs. For this purpose, we developed the **Network Analysis Tools** (***NeAT***), as set of tools performing basic operations on networks and clusters.

The tools can be used in three ways:

1. **Web server interface**

   http://rsat.scmbb.ulb.ac.be/neat/

   The Web interface gives a convenient and intuitive access to the tools, and allows you to bring your data sets through some typical analysis work flows in order to extract the best of it.

2. **Stand-alone application**

   http://rsat.scmbb.ulb.ac.be/rsat/distrib/

   Most of the tools are freely available to academic users, according to a licence for non-commercial and non-military usage.

   The license covers both the Regulatory Sequence Analysis Tools (***RSAT***) and the Network Analysis Tools (***NeAT***). It can be downloaded from the RSAT Web site.

3. **Web services**

In addition, people having computer skills can also use be same tools via a Web services interface, in order to integrate them in automatic work-flows. To obtain information on the Web services, connect the ***NeAT***web server, and in the left menu, select **Information - Web services**.

Chapter 2

Network visualization and format conversion

2.1 Introduction

2.1.1 Network visualization

To help the scientists apprehending their interest network, it is sometimes very useful to visualize them. Networks are generally represented by a set of dots (or of boxes) which represents its nodes that are linked via lines (the edges) or arows (arcs in the case of directed graphs). The nodes and the edges may present a label and / or a weight. The node label is generally indicated in the node box and the edge label is often placed on the line.

NeAT contains some facilities to represent networks. It contains its own visualization software (display-graph) that will be described in the following. Moreover, it allows the conversion of the graph into formats that may be used by some visualisation tools like ***Cytoscape*** ([14], *http://www.cytoscape.org*), ***yED*** (*http://www.yworks.com/products/yed/*)or ***VisANT*** ([7], *http://visant.bu.edu/*).

Hereafter, we describe briefly some of the major formats used for graph description.

2.1.2 Graph formats

Incompatibility between file formats is a constant problem in bioinformatics. In order to facilitate the use of the NeAT website, most of our tools support several among the most popular formats used to describe networks.

- The tab-delimited format is a convenient and intuitive way to encode a graph. Each row represents an arc, and each column an attribute of this arc. The two columns fields are the source and target nodes. If the graph is directed, the source node is the node from which the arc leaves and the target node is the node to which the arc arrives. Logically, in undirected graph, the columns containing the source and the target node may be inverted. Some additional arc attributes (weight, label, color) can be placed in pre-defined columns. Orphan nodes can be included by specifying a source node without target. The tool ***Pathfinder*** extends this format by supporting any number of attributes on nodes or edges as well as the color, the label and the width of nodes and edges.

- A *GML* file is made up of nested key-value pairs. The most popular graph editors support GML as input format (like Cytoscape and yED). More information on this format can be found at *http://www.infosun.fim.uni-passau.de/Graphlet/GML/*.
- The *DOT* format is a plain text graph description language. DOT files can be loaded in the programs of the suite GraphViz (*http://www.graphviz.org/*). It is a simple way of describing graphs in a human- and computer-readable format. Similarly to GML, DOT supports various attributes on nodes (i.e. color, width, label).
- VisML is the XML format required by VisANT, a very light but powerful visualisation tool.
- Several tools also accept adjacency matrices as input. An adjacency matrix is a N x N table (with N the number of nodes), where a cell $A[i,j]$ indicates the weight of the edge between nodes i and j (or 1 if the graph is unweighted).

2.2 Visualisation of a co-expression network

2.2.1 Study case

In this demonstration, we will show you how to visualize a network using some popular network visualization tools. This network we will study consist in the top scoring edges of the yeast co-expression network included in the integrative database String [19]. This undirected weighted networks contains 537 nodes representing genes and 4801 edges. An edge between two nodes means that they are co-expressed. The weight expresses at which level both genes are co-expressed. We will explain how to display this network with NeAT, Cytoscape, yED and VisANT. As Cytoscape and yED are not online tools, we will only describe their utilization in the command-line section.

2.2.2 Protocol for the web server

Format conversion and layout calculation

1. In the ***NeAT***menu, select the command ***format conversion / layout calculation***.

   In the right panel, you should now see a form entitled “convert-graph”.

2. Click on the link `DEMO`.

   The form is now filled with a graph in the tab-delimited format, and the parameters have been set up to their appropriate value for the demonstration, i.e., the network will be converted from tab-delimited to GML format, the source node column is 1, the target column is 2 and the weight column is 3.

The option `Calculate the layout of the nodes (only relevant for GML output` may also be chosen, otherwise the nodes will all be in diagonal and the resulting graphic will not be very instructive.

If the edges present a weight, ***convert-graph*** is able to represent the weight of the edges by computing a color gradient proportional to edge weights and coloring the edges according to it. There are five different color gradients : blue, red, green, grey and yellow to red. The darker (or the more colored) it is, the higher the weight. Moreover ***convert-graph*** can also change the width of the edge proportionnally to its weight. To this, we must choose a color gradient for the `Edge color intensity proportional to the weight` and the option `Edge width proportional to the weight of the edge` must be checked (which is automatically the case with the demonstration).

3. Click on the button `GO`.

   The resulting graph in GML format is available as an HTML link. Right clink on the link and save it with name *string_coexpression.gml*.

Visualization using NeAT

1. In the `Next Step` pannel, click on `Display the graph`.

   The form of ***display-graph*** is displayed. By default, the figure output format is jpeg, change it to png which gives a better resolution. NeAT also allow the postscript format.

2. Uncheck `Calculate the layout of the nodes (mandatory for all input format except GML)` as ***convert-graph*** already computed it.

3. Check `Edge width proportional to the weight of the edges`

4. Click on the `GO` button.

   The figure is available by clicking on the HTML link. Clicking a the link leads to a static figure representing the network.

Visualization using VisANT

1. After the step *Format conversion and layout calculation*, click on the `Load in VisANT`

   A page is displayed. Three links are available

   - A link to the graph in the format you obtained it from ***convert-graph*** (here GML).
   - A link (VisANT logo) to the VisANT applet
   - A link to the graph in VisML (the input format of VisANT)

2. Click on the logo of VisANT

   The VisANT applet is loaded.

3. Accept the authentification certifate.

2.2.3 Protocol for the command-line tools

Format conversion and layout calculation

If you have installed a stand-alone version of the NeAT distribution, you can use the programs ***convert-graph*** and ***display-graph*** on the command-line. This requires to be familiar with the Unix shell interface. If you don't have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results). To visualize the networks with yED, VisANT or Cytoscape, you must of course install them on your computer.

1. First let us download the network file *string_coex_simple.tab* from the NeAT tutorial download page : *http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/*

2. In this first step, we will convert the tab delimited String network that we just downloaded into a GML file by using this command. We compute the layout of the nodes. Moreover, we compute an edge width and an color proportional to the weight on the edge.

   ```
   convert-graph  -from tab -to gml -wcol 3 -i string_coex_simple.tab
   -o string_coex_simple.gml -layout -ewidth -wcol 3 -ecolors fire
   ```

Visualization using NeAT

Use the following command to create a graph using the NeAT ***display-graph*** program.

```
display-graph  -in_format gml -out_format png -i string_coex_simple.gml
-o string_coex_simple.png -ewidth
```

Visualization using Cytoscape (version 2.3)

1. Open Cytoscape

2. Click on `File` > `Import` > `Network...` > `Select`

3. Select the file *string_coex_simple.gml* If the graph contains more than 500 nodes, it will not be displayed immediately. Right click on the name of the graph file in the *Cytopanel 1* and select `Create view...`.

Visualization using yED (version 3)

1. Open yED
2. Click on `File` > `Import`
3. Select the file *string_coex_simple.gml*

   As NeAT GML converter add edge labels of the type *nodeName1_nodeName2* for unweighted or unlabeled graph, you may need to remove the edge label for a better visibility.
4. Click on one edge (random)

   The edge you clicked on is now selected.
5. Press *Ctrl+A*

   All edges are now selected.
6. In the *Property view* (Right of the screen), in the *label* part, uncheck the `visible` option.

Chapter 3

Comparisons between networks

3.1 Introduction

Protein interaction networks have deserved a special attention for molecular biologists, and several high-throughput methods have been developed during the last years, to reveal either pairwise interactions between proteins (two-hybrid technology) or protein complexes (methods relying on mass-spectrometry). The term *interactome* has been defined to denote the complete set of interactions between proteins of a given organism.

Interactome data is typically represented by an un-directed graph, where each node represents a polypeptide, and each edge an interaction between two polypeptides.

The yeast interactome was characterized by the two-hybrid method by two independent groups, Uetz and co-workers [17], and Ito and co-workers [8], respectively. Surprisingly, the two graphs resulting from these experiments showed a very small intersection.

In this tutorial, we will use the program ***compare-graphs*** to analyze the interactome graphs published by from Uetz and Ito, respectively.

We will first perform a detailed comparison, by merging the two graphs, and labelling each node according to the fact that it was found in Ito's network, in Uetz' network, or in both. We will then compute some statistics to estimate the significance of the intersection between the two interactome graphs.

3.2 Computing the intersection, union and differences between two graphs

3.2.1 Study case

In this demonstration, we will compare the networks resulting from the two first publications reporting a complete characterization of the yeast interactome, obtained using the two-hybrid method.The first network [17] contains 865 interactions between 926 proteins.The second network [8] contains 786 interactions between 779 proteins. We will merge the two networks (i.e. compute their union), and label each edge according to the fact that it is found in Ito's network, Uetz' network, or both. We will also compute the statistical significance of the intersection between the two networks.

3.2.2 Protocol for the web server

1. In the ***NeAT***menu, select the command ***network comparison***.

   In the right panel, you should now see a form entitled "compare-graphs".

2. Click on the button `DEMO`.

   The form is now filled with two graphs, and the parameters have been set up to their appropriate value for the demonstration. At the top of the form, you can read some information about the goal of the demo, and the source of the data.

3. Click on the button `GO`.

   The computation should take a few seconds only. The result page shows you some statistics about the comparison (see interpretation below), and a link pointing to the full result file.

4. Click on the link to see the full result file.

3.2.3 Protocol for the command-line tools

If you have installed a stand-alone version of the NeAT distribution, you can use the program ***compare-graphs*** on the command-line. This requires to be familiar with the Unix shell interface. If you don't have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results).

We will now describe the use of ***compare-graphs*** as a command line tool. The two two-hybrid datasets described in the previous section may be downloaded at the following address *http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/*. These are the files *uetz_2001.tab* and *ito_2002.tab*.

1. Go in the directory where the files containing the graphs to compare are located.

2. Type the following command

   ```
   compare-graphs -v 1 -Q ito_2002.tab -R uetz_2001.tab -return union \
   -o uetz_2001_union_ito_2002.tab
   ```

Using these options, some comparaison statistics are displayed and the results are stored in the tab-delimited file *uetz_2001_union_ito_2002.tab*.

In order to compute the difference or the intersection, you must change the `-return` option. For example, to compute the intersection, you shoud type.

```
compare-graphs -v 1 -Q ito_2002.tab -R uetz_2001.tab -return intersection \
-o uetz_2001_inter_ito_2002.tab
```

3.2.4 Interpretation of the results

The program ***compare-graphs*** uses symbols R and Q respectively, to denote the two graphs to be compared. Usually, R stands for reference, and Q for query.

In our case, R indicates Ito's network, whereas Q indicates Uetz' network. The two input graphs are considered equivalent, there is no reason to consider one of them as reference, but this does not really matter, because the statistics used for the comparison are symmetrical,as we will see below.

Union, intersection and differences

The result file contains the union graph, in tab-delimited format. This format is very convenient for inspecting the result, and for importing it into statistical packages (R, Excel, ...).

The rows starting with a semicolon (;) are comment lines. They provide you with some information (e.g. statistics about the intersection), but they will be ignored by graph-reading programs. The description of the result graph comes immediately after these comment lines.

Each row corresponds to one arc, and each column specifies one attribute of the arc.

1. **source**: the ID of the source node
2. **target**: the ID of the target node
3. **label**: the label of the arc. As labels, we selected the option "Weights on the query and reference". Since the input graphs were un-weighted, edge labels will be used instead of weights. The label `<NULL>` indicates that an edge is absent from one input network.
4. **color** and **status**: the status of the arc indicates whether it is found at the intersection, or in one graph only. A color code reflects this status, as indicated below.
   - *R.and.Q*: arcs found at the intersection between graphs R and Q. Default color: green.
   - *R.not.Q*: arcs found in graph R but not in graph Q. Default color: violet.
   - *Q.not.R*: arcs found in graph Q but not in graph R. Default color: red.

   The result file contains several thousands of arcs, and we will of course not inspect them by reading each row of this file. Instead, we can generate a drawing in order to obtain an intuitive perception of the graph.

Sizes of the union, intersection and differences

The beginning of the result file gives us some information about the size of the two input files, their union, intersection, and differences.

```
; Counts of nodes and arcs
;    Graph    Nodes    Arcs     Description
;    R        779      786      Reference graph
;    Q        926      865      Query graph
;    QvR      1359     1529     Union
;    Q^R      346      122      Intersection
;    Q!R      580      743      Query not reference
;    R!Q      433      664      Reference not query
```

Statistical significance of the intersection between two graphs

The next lines of the result file give some statistics about the intersection between the two graphs. These statistics are computed in terms of arcs.

```
; Significance of the number of arcs at the intersection
;  Symbol   Value     Description                         Formula
;  N        1359      Nodes in the union
;  M        922761    Max number of arcs in the union     M = N*(N-1)/2
;  E(Q^R)   0.74      Expected arcs in the intersection  E(Q^R) = Q*R/M
;  Q^R      122       Observed arcs in the intersection
;  perc_Q   14.10     Percentage of query arcs            perc_Q = 100*Q^R/Q
;  perc_R   15.52     Percentage of reference arcs        perc_R = 100*Q^R/R
;  Jac_sim  0.0798    Jaccard coefficient of similarity  Jac_sim = Q^R/(QvR)
;  Pval     2.5e-228  P-value of the intersection         Pval=P(X >= Q^R)
```

A first interesting point is the maximal number of arcs (M) that can be traced between any two nodes of the union graph. In our study case, the graph obtained by merging Ito's and Uetz' data contains $N = 1359$ nodes. This graph is undirected, and there are no self-loops. The maximal number of arcs is thus $M = N * (N-1)/2 = 922,761$. This number seems huge, compared to the number of arcs observed in either Uetz' ($A_Q = 865$) or Ito's ($A_R = 786$) graphs. This means that these two graphs are sparse: only a very small fraction of the node pairs are linked by an arc.

The next question is to evaluate the statistical significance of the intersection between the two graphs. For this, we can already compute the size that would be expected if we select two random sets of arcs of the same sizes as above ($A_Q = 865$, $A_R = 4,038$).

If the same numbers of arcs were picked up at random in the union graph, we could estimate the probability for an arc to be found in the network R as follows: $P(R) = A_R/M = 0.000852$. Similarly, the probability for an arc of the union graph to be part of the network Q is $P(Q) = A_Q/M = 0.000937$. The probability for an arc to be found independently in two random networks of the same sizes as R and Q is the product of these probabilities.

$$P(QR) = P(Q) * P(R) = A_R/M \cdot A_Q/M = 7.98e - 07$$

The number of arcs expected by chance in the intersection is the probability multiplied by the maximal number of arcs.

$$\begin{aligned} E(QR) &= P(QR) \cdot M \\ &= (A_Q \cdot A_R)/M \\ &= 7.98e-07 \cdot 922761 = 0.74 \end{aligned}$$

Thus, at the intersection between two random sets of interaction, we would expect on the average a bit less than one interaction. It seems thus clear that the 122 interactions found at the intersection between he two published experiments is much higher than the random expectation.

We can even go one step further, and compute the *P-value* of this intersection, i.e. the probability to select at least that many interactions by chance.

The probability to observe *exactly* x arcs at the intersection is given by the hypergeometrical distribution.

$$P(QR = x) = \frac{C_R^x C_{M-R}^{Q-x}}{C_M^Q} \tag{3.1}$$

where

- R is the number of arcs in the reference graph;
- Q i the number of arcs in the query graph;
- M is the maximal number of arcs;
- x is the number of arcs at the intersection between the two graphs.

By summing this formula, we obtain the P-value of the intersection, i.e. the probability to observe *at least* x arcs at the intersection.

$$Pval = P(QR >= x) = \sum_{i=x}^{min(Q,R)} P(X = i) = \sum_{i=x}^{min(Q,R)} \frac{C_R^i C_{M-R}^{Q-i}}{C_M^Q}$$

We can replace the symbols by the numbers of our study case.

$$\begin{aligned} Pval &= P(QR >= 122) \\ &= \sum_{i=x}^{min(865,786)} \frac{C_{786}^i C_{922761-786}^{865-i}}{C_{922761}^{865}} \\ &= 2.5e-228 \end{aligned}$$

This probabilty is so small that it comes close to the limit of precision of our program ($\approx 10^{-321}$).

Summary

In summary, the comparison revealed that the number of arcs found in common between the two datasets (Ito and Uetz) is highly significant, despite the apparently small percentage of the respective graphs it represents (14.10% of Ito, and 15.52% of Uetz).

3.3 Strengths and weaknesses of the approach

3.4 Exercises

1. Using the tool the tool ***network randomization***, generate two random graphs of 1000 nodes and 1000 arcs each (you will need to store these random networks on your hard drive). Use the tool ***network comparison*** to compare the two random graphs. Discuss the result, including the following questions:
   (a) What is the size of the intersection ? Does it correspond to the expected value ?
   (b) Which P-value do you obtain ? How do you interpret this P-value ?
2. Randomize Ito's network with the tool ***network randomization***, and compare this randomized graph with Uetz' network. Discuss the result in the same way as for the previous exercise.

3.5 Troubleshooting

1. The P-value of the intersection between two graphs is 0. Does it mean that it is impossible to have such an intersection by chance alone ?

   No. Any intersection that you observe in practice might occur by chance, but the limit of precision for the hypergeometric P-value is $\approx 10^{-321}$. Thus, a value of 0 can be interpreted as $Pval < 10^{-321}$.

2. The web server indicates that the result will appear, and after a few minutes my browser displays a message "No response the server".

   How big are the two graphs that you are comparing ? In principle, compare-graphs can treat large graphs in a short time, but if your graphs are very large (e.g. several hundreds of thousands of arcs), the processing time may exceed the patience of your browser. In such case, you should consider either to install the stand-alone version of ***NeAT***on your computer, or write a script that uses ***NeAT***via their Web services interface.

Chapter 4

Node degree statistics

4.1 Introduction

In a graph, the degree k of a node is the number of edges connected to this node. If the graph is directed, we can make a distinction between the in-degree (the number input arcs) and the out-degree (number of output arcs). In this case, the degree of the node consists in the sum of the in-degree and of the out-degree of this node.

Different nodes having different degrees, this variability is characterized by the degree distribution function $P(k)$, which gives the probability that a node has exactly k edges, or, in other words gives the observed frequency of a node of degree k.

Scale-free graphs were first described by Barabasi based on the study of the web connectivity, followed by several different biological networks [9].

A graph is scale-free if the distribution of the vertex degree (k) follows a power-law distribution of the form $P(k)\ k^{-\gamma}$.

The main property of such graphs is that it should have on one hand some highly connected nodes, called hubs, which are central to the network topology, and *keep the network together* and on the other hand a lot of poorly connected nodes linked to the hubs.

In the following, we will check if this scale free property also applies to the two-hybrid network described by Uetz *et al* [17] by computing the degree of each node and plotting the node degree distribution of the graph.

4.2 Analysis of the node degree distribution of a biological network

4.2.1 Study case

In this demonstration, we will analyze the node degree distribution of the first published yeast protein interaction network. This network is the first attemp to study the yeast interactome using the two-hybrid method and contains 865 interactions between 926 proteins [17].

4.2.2 Protocol for the web server

1. In the ***NeAT***menu, select the command ***stats on node degrees***.

   In the right panel, you should now see a form entitled “graph-node-degree”.

2. Click on the button `DEMO`.

   The form is now filled with a graph in the tab-delimited format, and the parameters have been set up to their appropriate value for the demonstration, i.e., the degree of all nodes will be computed. At the top of the form, you can read some information about the goal of the demo, and the source of the data.

3. Click on the button `GO`.

   The computation should take less than one minute. On one hand, the result page displays a link to the result file and on the other hand the graphics and raw data of the node degree distribution are also available. These will be discussed in the *Interpretation of the results* section.

4.2.3 Protocol for the command-line tools

If you have installed a stand-alone version of the NeAT distribution, you can use the program ***graph-node-degree*** on the command-line. This requires to be familiar with the Unix shell interface. If you don’t have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results).

We will now describe the use of ***graph-node-degree*** as a command line tool. The two two-hybrid dataset described in the previous section may be downloaded at the following address *http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/*. This is the file *uetz_2001.tab*.

1. The first step consist in applying ***graph-node-degree*** on the two-hybrid dataset. To this, go into the directory where you downloaded the file *uetz_2001.tab* and use this command.

   ```
   graph-node-degree -v 1 -i uetz_2001.tab -all -o uetz_2001_degrees.tab
   ```

   The file *uetz_2001_degrees.tab* is created and contains the in-, out- and global degree of each node of the Uetz *et al* data set.

2. In the second step, we will study the degree distribution of the nodes. To this, we use the program ***classfreq*** from the RSAT suite that compute the distribution of a set of number. As the graph we are working with is undirected, we will only compute this degree distribution for the global degree of the nodes which is the fourth column of the file *uetz_2001_degrees.tab* obtained at the previous step.

   ```
   classfreq -i uetz_2001_degrees.tab -v 1 -col 4 -ci 1 -o uetz_2001_degrees_freq.tab
   ```

3. Finally, we will display the distribution graph in the PNG format in order to visualize the degree distribution and determine if it has a scale free behaviour. The program XYgraph from RSA-tools will be used for this purpose. Note that we could use other tools like ***Microsoft Excel*** or ***R***. The results will be stored in the file *uetz_2001_degrees_freq.png* that you can open with any visualization tool.

```
XYgraph -i uetz_2001_degrees_freq.tab \
-title ’Global node degree distribution for Uetz et al (2001) interaction graph’ \
-xcol 2 -ycol 4,6 -xleg1 Degree -lines \
-yleg1 ’Number of nodes’ -legend -header -format png \
-o uetz_2001_degrees_freq.png
```

4.2.4 Interpretation of the results

graph-node-degree result file

Open the resulting file produced by ***graph-node-degree***. According to the requested level of verbosity (`-v #` option), the file begins with some lines starting with the ’#’ or ’;’ symbols that contains some information about the graph and the description of the columns.

The results consists in a five column data set.

1. Node name
2. Number of ingoing edges
3. Number of outgoing edges
4. Global degree (sum of the second and third columns).
5. Indicate whether the node only contains outgoing edges (source node) or ingoing edges (target node) or both (intermediate).

Node degree distribution

Let us first have a look at the node degree distribution data file produced by the ***class-freq*** program (raw data). This file is a tab-delimited file containing 9 columns. Each line consists in a value interval. In our case, the value is the degree of the nodes.

1. Minimal value of the interval
2. Maximal value of the interval
3. Central value of the interval
4. Frequency : Number of elements in this class interval (number of nodes having a degree comprised betwee the minimal and the maximal values.
5. Cumulative frequency.

6. Inverse cumulative frequency
7. Relative frequency : number of elements in this class over the total number of elements
8. Relative cumulative frequency
9. Inverse relative cumulative frequency

The first result line contains the distribution results for the nodes having only one neighbour (i.e. degree comprised between 1 and 2), from it we can see that 577 over 926, i.e., 62% of the nodes have a degree of one. Moreover, about 90% of the nodes have a degree lower than 4. This is indicative of the scale-free nature of the interaction network.

The figure best illustrates the scale-freeness of the graph. When looking at the graphical representation of this distribution, we can see two curves. The blue curve represents the absolute frequency and the green curve the inverse cumulative frequency. The exponential decrease of both curves shows that there are a lot more nodes poorly connected than highly connected (hubs). The Uetz graph thus presents a scale free behaviour.

Chapter 5

Study of the neighborhood of the nodes

5.1 Introduction

In a graph, the neighbours of a node consist in the set of nodes that are connected to this node up to a certain distance, i.e., the number of steps between the source node and its neighbours. In weighted graphs, one can also consider the neighbours up to a certain maximal weight.

In the following, we will refer to the node for which we search the neighbours the *seed node*.

According to the type of graph, it might be interesting to retrieve the neighbours of the nodes in a graph.

For example, in protein-protein interaction network, the function of the neighbours of a protein whose biological role is unknown might give insights in the function of the protein. Moreover, in interaction graphs, if a group of neighbours have similar biological functions, they are likely to form a structural complex.

In co-regulation networks, where each node is a gene and an edge between two genes means that those genes are co-regulated (i.e. co-repressed and co-expressed), exploring the neighbours of the nodes may help in the discovery of new regulons.

In the following, we will illustrate the study of nodes neighborhood by looking for neighbours of some orphan proteins (i.e. protein of unknown function) in a protein protein interaction network. We will then look if the neighbours of these proteins present similar functions.

5.2 Analysis of the neighbours of orphan nodes in an interaction protein network

5.2.1 Study case

In this demonstration, we will analyze the neighbours of the orphan nodes of the Gavin *et al* (2006) interaction data set. These interaction data were obtained by co-immunoprecipitation followed by a mass spectrometry experiment in order to discover structural protein complexes. [6]. This network contains 6531 interactions between 1430 proteins.

We will then compare these groups of neighbours with functionnal classes of proteins annotated in the MIPS [12] in order to detect if the groups of neighbours present a significatively high number of co-regulated proteins.

5.2.2 Protocol for the web server

1. In the ***NeAT***menu, select the command ***get node neighborhood***.

   In the right panel, you should now see a form entitled "graph-neighbours".

2. Go on the demo dataset download web page.*http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/* and download the files *gavin_2006_names.tab*, *orphan_gavin.tab* and *mips_name_class_description.tab* on your computer.

3. In the *Upload graph from file* text area, load the file *gavin_2006_names.tab* you just downloaded.

4. Uncheck `Include each node in its neighborhood (with a distance of zero)`

5. Check the radio-button *List of nodes* in the seed node part of the form

6. In the *Upload seed nodes from file* text area, load the file *orphan_gavin.tab.tab* you just downloaded.

7. Click on the button `GO`.

   The computation should take less than one minute.

   The result page should display the results in the tab-delimited or HTML format. These files will be described in the section *Interpretation of the results*

8. We will now see if the different groups of neighbours contain a significantly high number of proteins of similar function. To this, we will compare the groups of neighbours we just obtained with annotated groups of proteins, e.g., the genes annotated according to the gene ontology [2] or, in this example, according to the functionnal classes of the MIPS [12]. In the `Next step` pannel, click on the button `Compare the groups of neighbours`.

   You are redirected to the form of another program ***compare-classes*** that allows to compare two class files (the query file and the reference file). Each class of a query file is compared to each class of a reference file. The number of common elements is reported, as well as the probability to observe at least this number of common elements by chance alone. The query classes are already loaded and consist in the different groups of neighbours we discovered previously with ***graph-neighbours***.

9. In the *Upload reference classes from file* text area, load the file *mips_name_class_description.tab* downloaded in the first part of this tutorial. The classes files are two column files, the first column contains the elements and the second column the class to which the elements belong. Elements may belong to more than one class.

10. The default paramaters are sufficient. We will only keep the comparison presenting a significance higher than 0.

11. Click on the button `GO`.

12. You obtain the links to the result file in the tab-delimited format or in the HTML format. The obtained results will be described in the next section.

5.2.3 Protocol for the command-line tools

If you have installed a stand-alone version of the NeAT distribution, you can use the program ***graph-neighbours*** on the command-line. This requires to be familiar with the Unix shell interface. If you don't have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results).

We will now describe the use of ***graph-neighbours*** as a command line tool. The Gavin *et al* (2006) [6] co-immunoprecipitation dataset described in the previous section and the other files necessary for this tutorial may be downloaded at the following address *http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/* (*gavin_2006_names.tab*, *orphan_gavin.tab* and *mips_name_class_description.tab*).

1. The first step consist in applying ***graph-neighbours*** on the co-immunoprecipitation dataset. To this, go into the directory where you downloaded the files *gavin_2006_names.tab*, *orphan_gavin.tab* and use this command.

   ```
   graph-neighbours -v 1 -i gavin_2006_names.tab \
    -seedf orphan_gavin.tab \
    -o gavin_2006_orphan_neighbours_1.tab
   ```

   The file *gavin_2006_orphan_neighbours_1.tab* is created and contains for each node of the seed file the list of its direct neighbours, i.e., for each protein, the list of proteins that co-precipitated with it.

2. In the second step, we will compare these groups of neighbours to different groups of annotated proteins in order to discover if the groups of neighbours do contain a significatively high number of proteins of similar functions. This will give insights into the function of the orphans proteins used as seed nodes in the first step. To this, we will use the RSAT ***compare-classes*** program that allows to

compare two class files (the query file and the reference file) (see previous section or the RSAT tutorial for a more complete description of ***compare-classes***). Use the following command to compare the two files.

```
compare-classes -v 1 \
-q gavin_2006_orphan_neighbours_1.tab -r mips_name_class_description.tab \
-sort sig -return proba,occ,jac_sim \
-o gavin_2006_orphan_neighbours_1_cc_mips_functionnal_classes.tab -lth sig 0
```

We obtain a file *gavin_2006_orphan_neighbours_1_cc_mips_functionnal_classes.tab* containing the significant comparaison results. We will discuss it in the following section (interpretation of the results).

5.2.4 Interpretation of the results

graph-neighbours result file

According to the requested level of verbosity, the result file may first contain several lines (starting with "#" or ";"). These deliver some information about the analysed graph (number of nodes, edges, seed nodes, ...). The results are then displayed in four columns.

1. Name of the neighbour.
2. Name of the seed node (for which the neighbours are seeked in the graph).
3. Distance between the seed node and its neighbour (number of steps).
4. The last column, only relevant for directed graph, indicate whether the arc between the seed node and its neighbour is an out- or an in-going arc.

This file can be considered as a class file (see above for a more complete description) with the name of the neighbour being the member (first column) and the name of the seed node, the name of the class (second column).

compare-classes result file

The result of the comparaison between the groups of neighbours and the MIPS annotated classes are displayed in a multi-column file sorted by decreasing order of significance. When looking at the HTML version of the file, you may click on the header on the column to sort the table according to this field.

Each line displays the comparaison between a MIPS annotated class (reference class) and a group of neighbours (query class). What we want to know is if there is a significatively high number of members of the same MIPS class in a given group of neighours.

- ref : Name of the MIPS functionnal class.
- query : Name of the group of neighbours (seed node).

- R : Size of the reference class (number of members in this MIPS class).
- Q : Size of the query class (number of neighbour for this seed node).
- QR : Intersection size between the group of neighbours and the functionnal class.
- QvR : Union size between the group of neighbours and the functionnal class.
- R!Q : Elements that are in the functionnal class but not in the groups of neighbours.
- Q!R : Elements that are not in the functionnal class but are in the groups of neighbours.
- !Q!R : Elements that are not in the functionnal class nor in the groups of neighbours.
- P-val : P-value of the comparaison, propability (according to the hypergeometric law) to be wrong when claimin that there is a significatively high number of proteins of the same class in the group of neighbours.
- E-val : E-value of the comparaison. P-value multiplied by the total number of comparaisons. This value corresponds to the estimated number of false positives for a given P-value threshold.
- sig : Significance of the comparaison. This correpsonds to $-log_{10}(E - val)$. This index gives an intuitive perception of the exceptionality of the common elements : a negative significance indicates that the common matches are likely to come by chance alone, a positive value that they are significant.

Considering the file, we can observe that 7 seed nodes (on the 46) have a group of neighbours presenting a similar function. For example, 9 out of the 10 neighbours of the Yil161w protein (interacting with this protein) have their function related to ribosome biogenesis and 8 out of 10 neigbours are located in the cytoplasm. This may indicate that this protein may also be implied in ribosome biogenesis

Chapter 6

Influence of graph alteration and randomization on clustering

6.1 Introduction

Although negative controls and method evaluation are crucial points to the experimental biologist, this is far from being the same in bioinformatics where, too often, no negative control is associated to the predictions, so that one cannot estimate the probability of these predictions to biogically valid.

For this reason, in NeAT we developped programs allowing to randomize and to add some specified levels of noise to networks. This allows the user to apply the techniques used to find relevant results on networks where there is less or no signal and thus were no interesting result should emerge.

NeAT programs are able to generate randomized networks according to three methods.

1. *Node degree conservation* : this approach consists in shuffling the edges, each node keeping the same number of neighbors as in the original graph.

2. *Node degree distribution conservation* : in which the global distribution of the node degree is conserved but each node presents a different degree than in the original graph.

3. *Erdos-Renyi randomization* : where edges are distributed between pairs of nodes with equal probability.

6.2 Quantitative assessment of a clustering algorithm

6.2.1 Study case

In this demonstration, we will use the approach developped in [1] where we evaluated the performances of different graph clustering algorithms. Graph clustering algorithms allow to retrieve in a graph the groups of nodes that contain more connections between them than with the rest of the nodes of the graph. Clustering algorithms are often used in biology in order to extract coherent groups of nodes from networks (complexes detection (e.g. see [15, 10, 1, 13]), protein families detection [5], co-expressed genes detection in co-expression networks (e.g see [11]), . . .). The NeAT web server proposes the ***MCL*** (*Markov Cluster algorithm*) clustering algorithm developped by Stijn van Dongen [18, 5]. To follow the command-line tools instructions, you should have MCL installed on your computer (available at *http://micans.org/mcl/*).

MCL simulates a flow on the graph by calculating successive powers of the associated adjacency matrix. At each iteration, an *inflation step* is applied to enhance the contrast between regions of strong or weak flow in the graph. The process converges towards a partition of the graph, with a set of high-flow regions (the clusters) separated by boundaries with no flow. The value of the *inflation parameter* strongly influences the number of clusters. According to [1], the optimal inflation value for clustering protein interaction networks is 1.8.

We will use an artificial interaction network created from the complexes annotated in the MIPS database by creating an edge between all the nodes belonging to the same complex [12]. This network contains 12262 edges between 1095 nodes. We will then use the MCL clustering algorithm on this network, on a little altered network, on a highly altered network and finally on a randomized network.

We will then compare these clusters to the MIPS complexes and estimate how well MCL can retrieve protein complexes from a protein-protein interaction and the influence of the noise on the results.

In this example, we will only use random alteration, i.e., the edges that are removed are randomly chosen. This is done to mimick what happens really in biological experiments where some inter-relationships between the nodes (genes, proteins, metabolites, . . .) may not be discovered (false negatives) or are erroneously discovered (false positives). However the ***alter-graph*** program also allows to alterate the network with targeted attack on user-selected nodes. In their study, Spirin and Mirny [16] showed the affect of node targeted attacks on clustering results.

6.2.2 Protocol for the web server

Dataset download

Go on the demo dataset web page.*http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/* and download the MIPS complex network file (*complexes_rm_00_ad_00.tab*) and the complexes (*mips_complexes.tab*).

Network alteration

1. In the ***NeAT***menu, select the command ***network alteration***.
2. In the *Upload graph from file* text area, load the file *complexes_rm_00_ad_00.tab* containing the MIPS complexes network that you just downloaded.
3. In the `edges to add` text area, enter 10%.

4. In the `edges to remove` text area, enter 10%.
5. Click on the button `GO`.
6. Right click on the resulting file and save it with name *complexes_rm_10_ad_10.tab*.

   Re-do the this alteration procedure using 50% of edges removal and 100% of edges addition. Save the resulting file with name *complexes_rm_50_ad_100.tab*.

Network randomization

1. In the ***NeAT***menu, select the command ***network randomization***.
2. In the *Upload graph from file* text area, load the file *complexes_rm_00_ad_00.tab*.
3. Select the `Node degree conservation` randomization type.
4. Click on the button `GO`.
5. Right click on the resulting file and save with name *complexes_rm_00_ad_00_random.tab*.

Networks clustering and clustering assessment

1. In the ***NeAT***menu, select the command ***graph-based clustering MCL***.
2. In the *Upload graph from file* text area, load the file *complexes_rm_00_ad_00.tab*.
3. Click on the button `GO`. You should now obtain a link to the clustering results and the distribution of the sizes of the different clusters.
4. In the *Next step* pannel, click on the button *Compare these clusters to other clusters*.
5. In the *Upload reference classes from file* text area, load the *mips_complexes.tab* file.
6. Choose the *matrix file* output format
7. Click on the button `GO`. You now obtain a contingency table, i.e, a table with N rows and M columns (N being the number of MIPS complexes and M, the number of clusters). Each cell contains the number of protein common to one complex and one cluster.
8. To calculate some statistics on this contingency table, click on the `contingency-table statistics` button in the `Next step` pannel.
9. The ***contingency-stats*** form appears. As the contingency table is already uploaded, just lick on the `GO` button.
10. Save the resulting file under name *contigency_stats_rm_00_ad_00.tab*

Repeat these steps for *complexes_rm_10_ad_10.tab*, *complexes_rm_50_ad_100.tab* and *complexes_rm_00_ad_00_random.tab* and save the resulting files under the names *contigency_stats_ad_10_rm_10.tab*, *contigency_stats_ad_50_rm_100.tab*, *contigency_stats_ad_00_rm_00_random.tab*, respectively.

6.2.3 Protocol for the command-line tools

If you have installed a stand-alone version of the NeAT distribution, you can use the programs ***random-graph*** and ***alter-graph*** on the command-line. This requires to be familiar with the Unix shell interface. If you don't have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results).

We will now describe the use of ***random-graph***, ***alter-graph***, ***compare-classes*** and ***contingency-stats*** as command line tools. For this tutorial, you need to have the MCL program installed.

Start by going on the demo dataset download web page.*http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data* and downloading the MIPS complex network file (*complexes_rm_00_ad_00.tab*) and the complexes (*mips_complexes.tab*).

Network alteration

1. Go in the directory where you downloaded the file.
2. Use the following commands to alter the graph. Note that MCL is not an RSAT / NeAT program and thus cannot treat RSAT comments lines (starting with "#" or with ";"). We thus have to suppress them in the command.

   ```
   alter-graph -v 1 -i complexes_rm_00_ad_00.tab \
   -rm_edges 10% -add_edges 10% \
   | cut -f 1,2 | grep -v ';' > complexes_rm_10_ad_10.tab
   ```

   Re-use this command, but modify the percentage of removed (-rm_edges 50%) and added edges (-add_edges 100%). Save the resulting file with name *complexes_rm_50_ad_100.tab*.

Network randomization

1. Use the following commands to randomize the graph by shuffling the edges. The node degrees will be conserved.

   ```
   random-graph  -v 1 -i complexes_rm_00_ad_00.tab \
   -random_type node_degree \
   | cut -f 1,2 | grep -v ';' > complexes_rm_00_ad_00_random.tab
   ```

Networks clustering and clustering assessment

1. Use the following commands to apply MCL on the network

   ```
      mcl  complexes_rm_00_ad_00.tab \
      --abc -I 1.8 -o complexes_rm_00_ad_00_clusters.mcl
   ```

2. Convert the cluster file obtained with MCL with the program ***convert-classes*** into a file that is readable by NeAT / RSAT (two column cluster file).

   ```
      convert-classes -i complexes_rm_00_ad_00_clusters.mcl
      -from mcl -to tab -o complexes_rm_00_ad_00_clusters.tab
   ```

3. Compare the obtained clusters to the MIPS complexes with the program ***compare-classes***

   ```
   compare-classes -q complexes_rm_00_ad_00_clusters.tab \
   -r mips_complexes.tab \
   -matrix QR \
   -o complexes_rm_00_ad_00_clusters_cc_complexes_matrix.tab
   ```

4. Study the obtained matrix with the ***contingency-stats*** program

```
contingency-stats -i complexes_rm_00_ad_00_clusters_cc_complexes_matrix.tab \
  -o contigency_stats_ad_00_rm_00.tab
```

Repeat these steps for *complexes_rm_10_ad_10.tab*, *complexes_rm_50_ad_100.tab* and *complexes_rm_00_ad_00_random.tab* and save the resulting files under the names *contigency_stats_ad_10_rm_10.tab*, *contigency_stats_ad_50_rm_100.tab*, *contigency_stats_ad_00_rm_00_random.tab*, respectively.

6.2.4 Interpretation of the results

We will now compare the performances of MCL when applied to networks containing an increasing proportion of noise or no signal at all.

Files description

Randomized network As the real MIPS complexes network, this randomized network contains 12262 edges between 1095 nodes. With our parameter choice, no edge should be duplicated. However, as in ***random-graph*** the iterative process designed to avoid duplicated edges may not be totally efficient, some duplicated edges may subsist in the randomized network.

Altered networks This file is a classical NeAT tab-delimited edge list. However, there is a fifth column that indicates whether the edge comes from the original graph (*original*) or was added randomly (*random*).

- As the MIPS complex newtwork, the network with 10% of added and removed edges contains 12262 edges between 1095 nodes, which is logical as we removed and added the same number of edge (in this case 1226).

- The network with 100% of added edges (+ 12262 edges) and 50% of removed edges (-6131 edges) contains 18393 edges between 1095 nodes. This graph contains thus more noisy than relevant edges.

Contingency table As already explained in a previous section, having n MIPS complexes and m MCL clusters, the contingency table T is a $n \cdot m$ matrix where row i corresponds to the i^{th} annotated complex, and column j to the j^{th} cluster. The value of a cell $T_{i,j}$ indicates the number of proteins found in common between complex i and cluster j.

The clustering quality will be evaluated from this table by calculating the Sensitivity (Sn), the Positive predictive value (PPV), the row wise separation (Sep_r) and the cluster separation (Sep_c).

Contingency table metrics A list of metrics and their value. These will be described in the next section.

Metrics description

Sensitivity, Positive predictive value and geometric accuracy For each complex, we can calculate an sensitivity value. This corresponds to the maximal fraction of protein of a complex that are attributed by MCL to the same cluster. Sn measures how well proteins belonging to the same complex are grouped within the same cluster.

$$Sn_{i.} = \frac{max_{i.}(T_{ij})}{N_i}$$

where N_i corresponds to the size of the complex.

Moreover, for each cluster j, we calculated the Positive Predictive Value (PPV) which corresponds to the maximal fraction of a cluster belonging to the same complex. This reflects the ability of this cluster to detect one complex.

$$PPV_{.j} = \frac{max_{.j}(T_{ij})}{M_j}$$

where M_j corresponds to the cluster size.

To summarize these values at the level of the confusion table, we calculated the average of these values. First, we calculated their classical mean by averaging all the $PPV_{.j}$ and $Sn_{i.}$ values. We also calculated a weighted mean where the clusters and complexes have a weight proportional to their relative size on the the calculation of the mean.

$$Sn = \frac{\sum_{i=1}^{n} Sn_{i.}}{n}$$

$$PPV = \frac{\sum_{j=1}^{m} PPV_{.j}}{m}$$

$$Sn_w = \frac{\sum_{i=1}^{n} N_i Sn_{i.}}{\sum_{i=1}^{n} N_i}$$

$$PPV_w = \frac{\sum_{j=1}^{m} M_j PPV_{.j}}{\sum_{j=1}^{m} M_j}$$

Sensitivity and PPV reflect two contradictory tendencies of the clustering. Sn increases when all the proteins of the same complex are grouped in the same cluster and PPV decreases when proteins coming from different complexes are grouped in the same cluster. If all the proteins of the network are grouped in the same cluster, we maximize the Sn but the PPV is almost 0. On the other hand, if each protein is placed in a different cluster, the PPV is maximized but the sensitivity is very low. A compromise must be found between these two cases by using another statistics. We defined the geometric accuracy as the geometrical mean of the PPV and the Sn.

$$Acc_g = \sqrt{PPV \cdot Sn}$$

Separation We also defined another metrics called *Separation* (Sep). High Sep values indicated a high bidirectionnal correspondance between a cluster and a complex.

The row-wise separation estimates how a complex is isolated from the others. Its maximal value is 1 if this correspondance is perfect, i.e., when all the protein of a complex are grouped in one cluster and if this cluster does not contain any other protein. This maximal value may also be reached when the complex is separated between many clusters containing only members of the complex.

$$Sep_{r_{i.}} = \sum_{j=1}^{m} \left(\frac{T_{i,j}}{\sum_{j=1}^{m} T_{i,j}} \cdot \frac{T_{i,j}}{\sum_{i=1}^{n} T_{i,j}} \right)$$

The column-wise separation indicates how well a cluster isolates one or more complex from the other clusters. The maximal value 1 indicates that a cluster contains all the elements of one or more complexes.

$$Sep_{c_{.j}} = \sum_{i=1}^{n} \left(\frac{T_{i,j}}{\sum_{j=1}^{m} T_{i,j}} \cdot \frac{T_{i,j}}{\sum_{i=1}^{n} T_{i,j}} \right)$$

As for the sensitivity and the PPV, for each clustering result, all values of $Sep_{c_{.j}}$ and $Sep_{r_{i.}}$ are averaged over all clusters and all complexes. We then calculate a global separation value by calculating the geometrical mean of the average row wise separation and of the average column wise separation.

$$Sep = \sqrt{Sep_c \cdot Sep_r}$$

Score comparaison

The table summarizes the kind of values that should be obtained for the metrics described in the previous section. As the alteration and the randomization procedure are random processes, you should not obtain exactly the same results.

#	**true**	**ad10 / rm10**	**ad100 / rm50**	**random**
ncol	125	114	713	361
nrow	220	220	220	220
mean	0.0569	0.0624	0.00998	0.0197
Sn	0.998	0.985	0.418	0.291
PPV	0.884	0.836	0.867	0.459
acc	0.941	0.91	0.642	0.375
acc_g	0.939	0.907	0.602	0.365
Sn_w	0.997	0.992	0.502	0.157
PPV_w	0.621	0.62	0.688	0.244
acc_g_w	0.787	0.785	0.588	0.196
sep_r	0.567	0.507	0.676	0.192
sep_c	0.998	0.979	0.208	0.117
sep	0.752	0.704	0.375	0.15

As expected, the value of the global parameters, the geometric accuracy (row acc_g), the weighted geometric accuracy (row acc_g_w) and the separation (row sep) decrease drastically as the network contain less and less relevant information.

We can observe that the sensitivity is more affected than the *PPV* and that the complex wise separation (sep_r) is more affected than the cluster wise separation. This is due to the fact that by increasing the noise, MCL increases the number of small sized clusters (ncol) too and, as we saw in previous section, this has an impact on the sensitivity.

Note that with a random graph, we would have a separation of 0.15 but an unweighted geometric accuracy of 0.365 which is far from being 0. The relatively good performances of MCL on the highly altered graph must thus be taken with caution as the gain in performances is only of 23%. This illustrates the interest of using negative controls.

Chapter 7

Path finding

7.1 Introduction

Given a biological network and two nodes of interest, the aim of k shortest path finding is to enumerate the requested number of shortest paths connecting these nodes ordered according to their weight. For instance, we might look for all shortest paths between a receptor and a DNA binding protein to predict a signal transduction pathway from a protein protein interaction network. Another example is the prediction of a metabolic pathway given two reactions or compounds of interest and a metabolic network.
A problem encountered in many biological networks is the presence of so-called hub nodes, that is nodes with a large number of connections. For example, in bacterial protein-protein interaction networks, CRP has the role of a hub node because it interacts with many targets. Likewise, in metabolic networks, compounds such as ADP or water are hubs, since they are generated and consumed by thousands of reactions.
The shortest path very likely traverses the hub nodes of a network. It depends on the biological context, whether this behaviour is desired or not. In metabolic networks, we are less interested in paths going through water or ADP, since those paths are often not biological relevant. For instance, we can bypass the glycolysis pathway by connecting glucose via ADP to 3-Phosphoglycerate. To avoid finding irrelevant pathways like this one, we tested different strategies and concluded that using a weighted network gave the best results [3],[4]. In a weighted network, not the shortest, but the lightest paths are searched. Hub nodes receive large weights, making them less likely to appear in a solution path.
Whether weights are used and how they are set has to be decided depending on the biological network of interest.

In this chapter, we will demonstrate path finding on the example of metabolic networks. We will work on a network assembled from all metabolic pathways annotated for the yeast *S. cerevisiae* in BioCyc (Release 10.6). We will also show the influence of the weighting scheme on path finding results.

7.2 Computing the k shortest paths in weighted networks

7.2.1 Study case

The yeast network constructed from BioCyc data consists of 1,185 nodes and 2,656 edges. It has been obtained by unifying 171 metabolic pathways. Note that this network is bipartite, which means that it is made up of two different node types: reactions and compounds. An edge never connects two nodes of the same type. For the tutorial, we choose to represent the metabolic data as undirected network.

We will recover the heme biosynthesis II pathway given its start and end compound, namely glycine and protoheme. First, we will use the ”degree” weighting scheme, which penalizes hub nodes. Second, we will infer the path using the ”unit” weighting scheme and compare the results.

7.2.2 Protocol for the web server

1. In the ***NeAT***menu, select the command ***k shortest path finding***.

   In the right panel, you should now see a form entitled “Pathfinder”.

2. Click on the button `DEMO`.

   The form is now filled with the BioCyc demo network, and the parameters have been set up to their appropriate value for the demonstration. At the top of the form, you can read some information about the goal of the demo, and the source of the data.

3. Click on the button `GO`.

   The computation should take no more than two minutes. When it is finished, a link to the results should appear.

4. Click on the link to see the full result file.

   It lists a table of all paths found for the requested rank number (5 by default). You can also specify another type of output, for instance a network made up of all paths found. Vary the parameter `Output type` for this.

To see how results change with modified weight, you can repeat steps 1 and 2. Before clicking on `GO`, choose “unit weight” as `Weighting scheme` and set the `Rank` to 1. Continue as described above. You will obtain another paths table than before.

7.2.3 Protocol for the command-line tools

If you have installed a stand-alone version of the NeAT distribution (Graphtools), you can use the program Pathfinder on the command-line. This requires to be familiar with the Unix shell interface. See the Readme of Graphtools for more details. If you do not have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results).

7.2.4 Interpretation of the results

Degree weighting scheme

First, we run Pathfinder with degree weighting scheme, which is the default weighting scheme of the demo. This weighting scheme sets the weights of compound nodes to their degree and of reaction nodes to one. The first ranked path obtained should look like this:

GLY 5-AMINOLEVULINIC-ACID-SYNTHASE-RXN 5-AMINO-LEVULINATE PORPHOBILSYNTH-RXN PORPHOBILINOGEN OHMETHYLBILANESYN-RXN HYDROXYMETHYLBILANE UROGENIIISYN-RXN UROPORPHYRINOGEN-III UROGENDECARBOX-RXN COPROPORPHYRINOGEN_III RXN0-1461 PROTOPORPHYRINOGEN PROTOPORGENOXI-RXN PROTOPORPHYRIN_IX PROTOHEMEFERROCHELAT-RXN **PROTOHEME**

This path recovers very well the annotated heme biosynthesis II pathway.

Unit weighting scheme

We repeated path finding on the same network but used the unit weighting scheme, which sets all node weights to one. This is equivalent to path finding in an unweighted network. We obtain a large number of paths of first rank, among them this one:

GLY GLUTATHIONE-SYN-RXN ADP PEPDEPHOS-RXN PROTON PROTOHEMEFERROCHELAT-RXN **PROTOHEME**

This path deviates strongly from the heme biosynthesis II pathway annotated in BioCyc. It contains two hub nodes: ADP and PROTON.

7.2.5 Summary

To sum up: path finding can predict pathways with high accuracy if an appropriate weighting scheme is applied to the network of interest. Our metabolic example shows that the heme biosynthesis II pathway is accurately predicted when using a weighted network and not found at all when using an unweighted network. The take home message is that in order to use Pathfinder on biological networks, weights have to be carefully adjusted.

7.2.6 Strengths and Weaknesses of the approach

Strengths

The strength of the approach is that for a given network and appropriate weighting scheme, pathways can be discovered with high accuracy. These pathways may be known or novel pathways. Other methods such as pathway mapping are unable to recover entirely novel pathways or pathways which are combinations of known pathways.

Weaknesses

The weakness is that the weighting scheme has to be optimized for the biological network of interest.

7.2.7 Troubleshooting

1. No path could be found.

   Make sure that your start and end nodes are present in your network of interest. If no path could be found, none of the end nodes is reachable from the start nodes, thus no path exists. For big graphs and long waiting time, there is the possibility that the pre-processing step of REA, namely to compute the shortest paths from the source to all nodes with Dijkstra, was not finished before the server timeout. In this case, a path might exist but could not be detected due to the timeout.

2. An out of memory error occurred.

   When searching for paths with the ”unit” weighting scheme in large networks, there might be a large number of possible paths for each requested rank. Although REA has a memory-efficient way to store paths with pointers, there is a limit for the number of paths that can be hold in memory. Reduce the number of requested paths or the size of the graph or use another weighting scheme.

Chapter 8

Recapitulative exercises

Bibliography

[1] Sylvain Brohee and Jacques van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7:488, 2006.

[2] The Gene Ontology Consortium. The gene ontology project in 2008. *Nucleic Acids Res*, Nov 2007.

[3] D. Croes, F. Couche, S. Wodak, and J. v. Helden. Metabolic pathfinding: inferring relevant pathways in biochemical networks. *Nucleic Acids Research*, 33:W326–W330, 2005.

[4] D. Croes, F. Couche, S. Wodak, and J. v. Helden. Inferring meaningful pathways in weighted metabolic networks. *J. Mol. Biol.*, 356:222–236, 2006.

[5] A J Enright, S Van Dongen, and C A Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res*, 30(7):1575–84, 2002.

[6] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dmpelfeld, Angela Edelmann, Marie-Anne Heurtier, Verena Hoffman, Christian Hoefert, Karin Klein, Manuela Hudak, Anne-Marie Michon, Malgorzata Schelder, Markus Schirle, Marita Remor, Tatjana Rudi, Sean Hooper, Andreas Bauer, Tewis Bouwmeester, Georg Casari, Gerard Drewes, Gitte Neubauer, Jens M Rick, Bernhard Kuster, Peer Bork, Robert B Russell, and Giulio Superti-Furga. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, Mar 2006.

[7] Zhenjun Hu, David M Ng, Takuji Yamada, Chunnuan Chen, Shuichi Kawashima, Joe Mellor, Bolan Linghu, Minoru Kanehisa, Joshua M Stuart, and Charles DeLisi. Visant 3.0: new modules for pathway visualization, editing, prediction and construction. *Nucleic Acids Res*, 35(Web Server issue):W625–W632, Jul 2007.

[8] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc Natl Acad Sci U S A*, 98(8):4569–74., 2001.

[9] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabsi. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, Oct 2000.

[10] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis, Thanuja Punna, Jos M Peregrn-Alvarez, Michael Shales, Xin Zhang, Michael Davey, Mark D Robinson, Alberto Paccanaro, James E Bray, Anthony Sheung, Bryan Beattie, Dawn P Richards, Veronica Canadien, Atanas Lalev, Frank Mena, Peter Wong, Andrei Starostine, Myra M Canete, James Vlasblom, Samuel Wu, Chris Orsi, Sean R Collins, Shamanta Chandran, Robin Haw, Jennifer J Rilstone, Kiran Gandi, Natalie J Thompson, Gabe Musso, Peter St Onge, Shaun Ghanny, Mandy H Y Lam, Gareth Butland, Amin M Altaf-Ul, Shigehiko Kanaya, Ali Shilatifard, Erin O'Shea, Jonathan S Weissman, C. James Ingles, Timothy R Hughes, John Parkinson, Mark Gerstein, Shoshana J Wodak, Andrew Emili, and Jack F Greenblatt. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637–643, Mar 2006.

[11] B. Samuel Lattimore, Stijn van Dongen, and M. James C Crabbe. Genemcl in microarray analysis. *Comput Biol Chem*, 29(5):354–359, Oct 2005.

[12] H. W. Mewes, S. Dietmann, D. Frishman, R. Gregory, G. Mannhaupt, K. F X Mayer, M. Mnsterktter, A. Ruepp, M. Spannagl, V. Stmpflen, and T. Rattei. Mips: analysis and annotation of genome information in 2007. *Nucleic Acids Res*, Dec 2007.

[13] Jose B Pereira-Leal, Anton J Enright, and Christos A Ouzounis. Detection of functional modules from protein interaction networks. *Proteins*, 54(1):49–57, Jan 2004.

[14] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res*, 13(11):2498–2504, Nov 2003.

[15] Roded Sharan, Igor Ulitsky, and Ron Shamir. Network-based prediction of protein function. *Mol Syst Biol*, 3:88, 2007.

[16] Victor Spirin and Leonid A Mirny. Protein complexes and functional modules in molecular networks. *Proc Natl Acad Sci U S A*, 100(21):12123–12128, Oct 2003.

[17] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J. M. Rothberg. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature*, 403(6770):623–7., 2000.

[18] Stijn Van Dongen. *Graph clustering by flow simulation*. PhD thesis, Centers for mathematics and computer science (CWI), University of Utrecht, 2000.

[19] Christian von Mering, Lars J Jensen, Michael Kuhn, Samuel Chaffron, Tobias Doerks, Beate Krger, Berend Snel, and Peer Bork. String 7–recent developments in the integration and prediction of protein interactions. *Nucleic Acids Res*, 35(Database issue):D358–D362, Jan 2007.