Network Analysis Tools: from biological networks to clusters and pathways

Sylvain Brohée, Karoline Faust, Gipsi Lima-Mendez, Gilles Vanderstocken & Jacques van Helden

Laboratoire de Bioinformatique des Génomes et des Réseaux (BiGRe), Université Libre de Bruxelles, Campus Plaine, CP 263, Boulevard du Triomphe, B-1050 Bruxelles, Belgium. Correspondence should be addressed to J.v.H. (Jacques.van.Helden@ulb.ac.be).

Published online 18 September 2008; doi:10.1038/nprot.2008.100

Network Analysis Tools (NeAT) is a suite of computer tools that integrate various algorithms for the analysis of biological networks: comparison between graphs, between clusters, or between graphs and clusters; network randomization; analysis of degree distribution; network-based clustering and path finding. The tools are interconnected to enable a stepwise analysis of the network through a complete analytical workflow. In this protocol, we present a typical case of utilization, where the tasks above are combined to decipher a protein–protein interaction network retrieved from the STRING database. The results returned by NeAT are typically subnetworks, networks enriched with additional information (i.e., clusters or paths) or tables displaying statistics. Typical networks comprising several thousands of nodes and arcs can be analyzed within a few minutes. The complete protocol can be read and executed in ~1 h.

INTRODUCTION

This is the last article in a series of four protocols for the analysis of regulatory sequences with the Regulatory Sequence Analysis Tools[1] (http://rsat.ulb.ac.be/rsat/) and biological networks with the Network Analysis Tools (NeAT)[2] (http://rsat.ulb.ac.be/neat/). The first article[3] presents a protocol to predict the location of binding sites for transcription factors whose specificity is already known (pattern matching). In the second article[4], we describe a protocol for the *ab initio* discovery of biological signals in biological sequences (pattern discovery). The third article[5] shows how to write scripts to automate the analysis on multiple clusters of genes using Web services. In this article, we describe a workflow for deciphering biological networks by combining network comparison, module identification and path finding. This protocol can be executed independently of the three other ones.

Network biology is emerging as a new field in biology, due to the increasing availability of genome-scale data sets of molecular interactions, such as those resulting from high-throughput technologies (e.g., protein interactions, regulatory networks and metabolome). Extracting relevant information from this huge amount of data requires dedicated tools. These data sets are commonly represented as graphs (or networks), where nodes represent molecules, and arcs their interactions. This representation eases data integration and makes it possible to apply well-known network algorithms to analyze the data.

In this protocol, we show how large biological networks can be explored by combining a set of modular tools accessible via a Web interface named NeAT. We describe hereafter the typical questions that can be addressed.

Network topology. It has been observed that some topological properties distinguish biological networks from random networks[6,7]. Noticeably, the distribution of degree (the number of arcs connected per nodes) is often claimed to follow a power-law distribution[6,8]. The tool ***graph-topology*** can be used to analyze the degree distribution of any kind of network.

Network comparison. Given two networks (i.e., protein–protein interaction networks from two different experiments), one would like to analyze their degree of overlap. This question is answered by the tool ***compare-graphs***, which computes the intersection, union and difference between two networks and estimates the statistical significance of the overlap.

Node neighborhood. Given a protein, gene or another node in a biological network, it is of interest to identify its direct or indirect neighbors in this network. This is the task of the tool ***graph-neighbours***, which returns the neighbors of a query node in a network up to a certain distance. This tool can be applied for instance on protein–protein interaction networks to retrieve the interaction partners of a given protein.

Cluster analysis. Various algorithms have been implemented to extract clusters (i.e., groups of densely connected nodes) from biological networks[9–13]. Among those, ***MCL***[14] algorithm has been shown to obtain good performances for extracting protein complexes from protein interaction networks[15–17]. In addition, this algorithm can deal with large graphs and is very efficient in time. For these reasons, we included MCL in the NeAT suite. The clusters resulting from MCL or other methods can be compared to some reference groups (e.g., functional classes) with the program ***compare-classes*** and mapped onto networks with ***graph-get-clusters***. Upon partitioning with MCL, each node belongs to only one cluster. However, sometimes the assignment of a node to a single cluster is an over-simplification of the biological data, for example, a protein may be part of different protein complexes. In those cases, it would be better to describe how much each node is related to the different clusters. The program ***graph-cluster-membership*** postprocesses a clustering result and calculates the membership as the proportion of edges (or weight) linking each node to each cluster. The node–cluster relationships are described as a membership matrix, where each row represents a node and each column a cluster.

Path finding. Given a biological network and two nodes of interest, a common task is to find a biological meaningful path connecting those nodes in the network. For instance, path-finding algorithms are applied to uncover signal transduction or metabolic pathways in protein–protein interaction or metabolic networks, respectively[18–21]. Recently, we evaluated the performance of a *k*-shortest path-finding algorithm for metabolic pathway inference and found high accuracies if appropriate weights are set on the network[22].

Network randomization. Negative controls are essential to estimate the relevance of the results. The tool ***random-graph*** proposes different procedures to randomize a network, which can then be submitted to the same workflows as the original network.

Network alteration. To test the robustness of analytic methods to the presence of noise, or to the incompleteness of information, the tool ***alter-graph*** allows to modify an existing network by random addition or deletion of nodes and/or edges.

Network display. NeAT includes a tool called ***display-graph***, which generates static images of the input networks. Such drawings are convenient for a quick inspection of the results from the Web browser. For more sophisticated layouts and for a dynamical manipulation of the drawing, we recommend graph editors such as yEd (http://www.yworks.com/en/products_yed_about.html) or Cytoscape[23]. The tool ***convert-graph*** permits to export any network analyzed with NeAT into Graph Modeling Language (GML) (http://www.infosun.fim.uni-passau.de/Graphlet/GML/gml-tr.html), a file format supported by both editors.

Figure 1 depicts the way in which the NeAT can be connected. We suggest the reader to follow this flow chart progressively during the execution of the protocol.

Comparison to other graph analysis tool suites

A large variety of graph analysis tools exist. We may classify them in three categories: (i) libraries that can only be used programmatically, for example, Boost (http://www.boost.org/), igraph (http://cneurocvs.rmki.kfki.hu/igraph/index.html) or JUNG (http://jung.sourceforge.net/); (ii) stand-alone tools with graphical user interface (GUI) (Pajek[24], Network Workbench (http://nwb.slis.indiana.edu), BiologicalNetworks[25], VisANT[26], yEd, Cytoscape, etc.) and (iii) tools with GUI available via the Web such as tYNA (http://tyna.gersteinlab.org/tyna/) or CABiNeT (http://mips.gsf.de/genre/proj/CABiNet/).

Usually, the libraries offer generic graph algorithms, whereas stand-alone or Web-based tool suites are often specialized. For instance, VisANT, Cytoscape (with its plugins) and BiologicalNetworks focus on the analysis and display of biological networks, whereas yEd offers a flexible interface for the display, layout and edition of general-purpose graphs, but is equipped with limited analysis functions. Pajek and Network Workbench are stand-alone tools for generic graph analysis. Cytoscape (with plugins) and BiologicalNetworks allow in addition retrieval and integration of biological networks.

We describe hereafter some of the advantages and current limitations of NeAT.

Main advantages. The main advantages of NeAT are

(1) NeAT supports a variety of modular tools, which can either be used separately, or combined in a workflow. These tools include a number of unique features (fuzzy clustering, Web access to MCL and RNSC, *k*-shortest paths with multiple start and end nodes, statistical comparison of classes and clusters, etc.) that are currently not available in other packages.
(2) The programs are designed to enable treating very large graphs (several thousands of nodes) without excessive cost in memory or time.
(3) Although most of the analyses can also be performed in specialized software packages such as R, the NeAT Web site offers a user-friendly access for biologists who are not familiar with programming languages.
(4) NeAT can be run on command line, either by installing it locally or by calling Web services. This is not the case of the other stand-alone and Web-based tools (a notable exception is Pajek). The programmatic access (either as stand-alone application or as Web services) allows one to automate the executions of workflows for multiple data sets, which would require hundreds or thousands of manual operations with conventional GUIs or on a Web site. To our knowledge, there is only one other network tools suite enabling workflows, namely tYNA. NeAT and tYNA are complementary: NeAT supports path-finding, graph-based clustering (MCL, RNSC and fuzzy clustering), network randomization and cluster comparisons, whereas tYNA includes tools to find motifs in networks. Because both tools support Web services, they can be easily combined in workflows, either by programming client scripts or using GUIs such as Taverna[27].
(5) NeAT may be used for any kind of network, but it was developed with biological networks in mind. The tools have been extensively tested on a variety of biological networks (protein–protein interaction networks[17], evolutionary networks[28] and metabolic networks[21,22]). Extensive evaluation is rarely reported for other biological network tools suites.

Main limitations. NeAT essentially provides facilities for the analysis of networks, clusters and pathways, but is not focused on

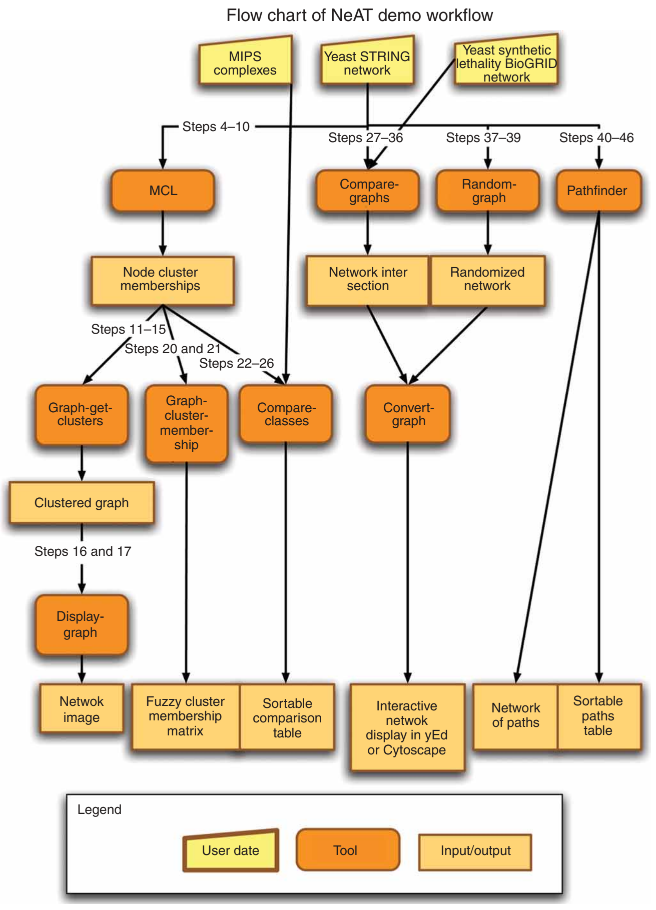


Figure 1 | Flow chart of the data, tools and results described in this protocol. Yellow represents the data set, orange the tool and light brown the results.

the problem of network visualization. This limitation, however, is easily circumvented by installing some specialized visualization software: all graphs generated by NeAT can be exported to several formats, including GML, which can be loaded with Cytoscape, yEd and VisANT, and DOT, which can be loaded with Graphviz.

To summarize, NeAT addresses the needs of researchers interested in the analysis of biological networks. Some tools may require background knowledge (e.g., MCL, fuzzy clustering), whereas others are intuitive and easy to use (e.g., graph conversion and comparison).

For the user with experience in programming, NeAT can be run on command line or within workflow management environments such as Taverna. Otherwise, the user may access NeAT via its Web interface, guided by tutorials and demos. To our knowledge, no other biological network tools suite exists that combines all the features of NeAT.

Other applications of this protocol

For the sake of consistency, the cases treated in this protocol are restricted to protein interaction networks. The tools available in NeAT can also be used to analyze other types of biological networks representing other types of interactions, for example, regulation, signal transduction, metabolic reactions, and ecology. The fuzzy clustering approach was initially conceived to address the problem of classifying phage genomes while taking into account the frequent exchanges of genetic material between them[28]. The *k*-shortest path-finding algorithm has previously been applied to infer relevant pathways in metabolic networks[29,30].

MATERIALS

EQUIPMENT

- This protocol describes an online tool. The only requirement is a computer with Internet connection. Optionally, you can install yEd (http://www.yworks.com/) or Cytoscape[23] for visualization
- Sample interaction networks, which can be obtained from various biological databases. As examples, we cite the following:
  - STRING[31] (http://string.embl.de/), a database integrating seven different types of evidences for physical and/or functional interactions between proteins: experimental evidences, phylogenetic profiles ('co-occurrence'), gene fusion/fission, synteny ('neighborhood'), coexpression, text mining and a data set called 'database', regrouping several criterion selected by the STRING annotation team
  - BioGRID[32] (http://www.thebiogrid.org/), a database of protein and genetic interactions including >116,000 curated interactions from yeast, *Caenorhabditis elegans*, drosophila and human
  - BioCyc[33] (http://www.biocyc.org/) or KEGG[34] (www.genome.jp/kegg/), the two main metabolic pathway databases
- The data required for the study cases treated in this protocol is available in the data repository site: http://rsat.ulb.ac.be/nedt/. All the networks used to illustrate this protocol were taken from the yeast *Saccharomyces cerevisiae*. We selected various networks representing diverse types of interactions between biological molecules (protein interactions, metabolism, protein complexes, genetic interactions, etc.)
  - Protein–protein interactions (physical and functional). From the STRING database[31], we extracted a subset labeled as 'database' by the STRING team. Under this label, they regrouped different types of protein–protein interactions and metabolic relationships (Jensen L., personal communication). This network contains 1,237 nodes representing proteins, and 11,027 edges representing a mixture of physical and functional protein–protein interactions. The interactions between two proteins are considered symmetrical; it is an undirected graph. The network is stored in data repository, a tab-delimited text file named *yeast_string_database_graph_names_undirected.tab*
  - The ***synthetic lethality network*** was extracted from the BioGRID database[32]. It represents genes (2,353 nodes) whose individual deletion is viable, but whose paired deletions (12,419 edges) are lethal
  - Protein complexes. The file *mips_complexes_names.tab* describes the collection of protein complexes annotated in the MIPS database[35]. Complexes detected only by high-throughput experiments were discarded from the data set. The first column of the file gives the gene name, the second column the complex name and the third column the gene identifier. In total, the file contains 1,121 distinct proteins forming 243 distinct complexes. Note that a protein can belong to several complexes
  - Signal transduction pathway. As study case for the path finding, we take a yeast signal transduction pathway mentioned by Scott and colleagues[18]. This pathway, known to regulate filamentous growth in yeast, starts with RAS2 and ends with TEC1. The authors attempt to recover this pathway with a path-finding algorithm based on color coding[18]. We will try to recover it using *Pathfinder*
- Incompatibility between file formats is a constant problem in bioinformatics. To facilitate the use of the Web site, most tools support several among the most popular formats used to describe networks. A description of the supported format is given in **Box 1**

BOX 1 | MAIN GRAPH FORMATS

The ***tab-delimited format*** is a convenient and intuitive way to encode a graph. Each row represents an edge and each column an attribute of this edge. The two column 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 nodes may be swapped. Orphan nodes can be included by specifying a source node without target. Some additional edge attributes (weight, label, color) can be placed in additional columns. The tool *Pathfinder* extends this format by supporting any number of attributes on nodes or edges. Check the *Pathfinder* help page for more details.

The ***GML format*** allows for the specification of additional layout and display attribute, such as node position, as well as the color, the label and the width of nodes and edges. A Graph Modeling Language (GML) file is made up of nested key-value pairs. The most popular graph editors (such as Cytoscape and yEd) support GML as input format.

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. Similar to GML, DOT supports various attributes on nodes (i.e., color, width, label).

Several tools also accept adjacency matrices as input. An ***adjacency matrix*** is an $N \times N$ 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).

The Network Analysis Tools program ***convert-graph*** facilitates the handling of these formats by supporting interconversions between various input (tab, gml, adjacency) and output formats (tab, dot, gml, adjacency).

PROCEDURE

Downloading a sample network

1| We will show on an example workflow how the different tools of NeAT can be combined to analyze a network taken from the STRING database. Open a connection to the data repository for this protocol (see EQUIPMENT).

2| Download the network file *yeast_string_database_graph_names_undirected.tab* on your computer. It is described in a tab-delimited file that contains five columns. Each row represents one interaction between two genes or between their products. As described in **Box 1**, the two first columns indicate the name of the ***Source*** and ***Target*** genes/proteins of the source and target nodes. The third column contains a score ranging from 0 to 900, which reflects the reliability of the indications available for this interaction. Higher scores represent more reliable interactions. In this case, the score is higher if an interaction is found several times in different data set. The columns 4 and 5 contain the gene identifiers corresponding to the gene names in columns 1 and 2.

3| Open a connection to the NeAT Web server: http://rsat.ulb.ac.be/neat/.

Cluster analysis

4| *Extracting clusters from the network with MCL (Steps 4–10)*. We will first apply graph-based clustering to detect groups of highly interconnected nodes in the sample network. For this, we will use the MCL algorithm, a fast unsupervised clustering algorithm based on simulation of flows in graphs[14]. In the menu from the left panel, click on the link ***MCL clustering*** to open the MCL query form.

5| Click on the ***Browse...*** button, and choose the file containing the network (e.g., *yeast_string_database_graph_names_undirected.tab* for the study case discussed here).

6| Specify the columns containing the source, target and (optionally) weight attributes of the tab-delimited file. In our example file, the source and target columns are by default 1 and 2 so we only have to add the weight column: ***Weight column*** = 3. Note that if you want to work with the gene identifiers instead of the gene names, you could have used value 4 and 5 in the source and target column fields. However, this is not recommended in this protocol as in the following we will only work with gene names.
▲ CRITICAL STEP The weighting of edges strongly affects the MCL result, because the principle of the algorithm is to iteratively enforce the weight of the most 'important' edges in the network. The 'importance' of an edge is determined by both its initial weight and its place in the network.

7| Choose an inflation value (between 1.2 and 5). For the study case, select 1.8.
▲ CRITICAL STEP This parameter acts on the granularity of the clustering procedure, that is, the number of clusters (and consequently the number of elements per cluster). The number of clusters increases with the inflation value. This parameter must thus be fine-tuned according to the structure of the network. In a recent evaluation, we found that an inflation value of 1.8 was optimal for protein–protein interaction networks[17].

8| Click on the ***GO*** button. The processing should take a bit less than one 1 min.
? TROUBLESHOOTING

9| The result page displays a figure showing the cluster size distribution, that is, the number of clusters (ordinate) of each size (abscissa). The page also contains a link to the clustering result. This file can be saved by right-clicking on the URL link and selecting ***Save link as...***, and save it on your computer under the name *yeast_string_MCL_clusters.tab*.

10| To inspect the result file, you can either use a text editor to open the file *yeast_string_MCL_clusters.tab* stored on your computer, or click on the URL on the result page. The MCL result is a simple two-column table, where the first column indicates the node names (gene names in our case), and the second column the cluster names. A quick inspection of this table from top to bottom shows that the first clusters contain more nodes than the last ones. MCL sorts the clusters by decreasing order of size.

11| *Extracting the subnetwork defined by the clusters (Steps 11–17)*. We will now map the clusters resulting from MCL onto their original network. For this, there are two alternative ways to proceed: directly load the files stored on the server (option A) or transfer the network and cluster files from your computer (option B).

(A) Directly load the files stored on the server

(i) The MCL result page displays a '***Next step***' box, allowing you to send the MCL output to several alternative tools. Click the button ***Map those cluster on the network***. This will call a form *graph-get-clusters*, with prefilled values for the parameters ***Graph*** and ***Clusters***.

(B) Transfer the network and cluster files from your computer

(i) An alternative way to enter data in the tool *graph-get-clusters* is to click on the link '***Map clusters onto network***' in the left menu. This will open an empty form, in which you will have to enter the data (for our study case, the graph is in the file *yeast_string_database_graph_names_undirected.tab* and the clusters in the file *yeast_string_MCL_clusters.tab*). However, this would require to transfer those two files to the server, albeit it already contains a copy of both in the temporary directory. Whenever possible, you should thus use the '***Next step***' buttons rather than transferring the files back and forth between your computer and the server.

12| The main choice for the tool *graph-get-clusters* is the ***output type***. The program supports two types of operations between the network and the clusters. (i) The option ***annotated graph*** labels each edge according to its intracluster or intercluster nature. (ii) ***intracluster edges*** selects a subnetwork restricted to the intracluster edges (intercluster edges are simply deleted from the network). You can experiment the three options. In this section of the protocol, we will extract the subnetwork defined by the MCL clusters. For this, select the option ***intracluster edges***.

13| Several output formats are proposed, but for the visualization purpose, select the intracluster edges output in the ***GML format***.

14| Click on the button ***GO***.

15| The result page should appear after <1 min, displaying a set of buttons for postprocessing the graph-get-cluster result, and a link toward the result file. You can store the resulting GML graph on your computer for later use by right-clicking on the URL in the *graph-get-clusters* result page. Save the result in a file named *yeast_string_MCL_intra_cluster.gml*.

16| A quick way to visualize the result is to fetch it to the NeAT visualization tool. However, beware that this tool offers limited functionalities: it returns a static image, with a simplistic layout. The main function of this tool is to provide a quick view of the result, before visualizing it with specialized tools. To visualize the result network with NeAT, click on the button ***Display the graph***. A new form will then be displayed. Select the desired output format. If the network is weighted (e.g., our study case), you can activate the option ***Edge width proportional to the weight***. To obtain the figure, click on the button ***GO***. This process may be slow (>1 min) depending on the size of the graph.

17| For a better visualization of the network, open the GML formatted file obtained in Step 15 with Cytoscape, yEd or any other visualization program of your choice. After having opened the GML file, you need to apply some layout to display nodes and edges harmoniously. For yEd and Cytoscape we recommend the option ***Organic layout***. After this, you should see a set of well-separated components, each corresponding to a MCL cluster. Each cluster is displayed with a specific color for the edges (**Fig. 2a**).
? TROUBLESHOOTING

18| *Mapping the clusters onto the network (Steps 18 and 19).* In the previous section, we used ***graph-get-clusters*** to separate the MCL clusters by deleting intercluster edges from the original network. Alternatively, the same tool can be used to label all the edges according to the cluster composition. Come back to Step 12, but this time, select '***annotated graph (all edges)***' as ***output type***.

19| Repeat Steps 13–17, and compare visually the result with that obtained in the previous section (**Fig. 2b**).

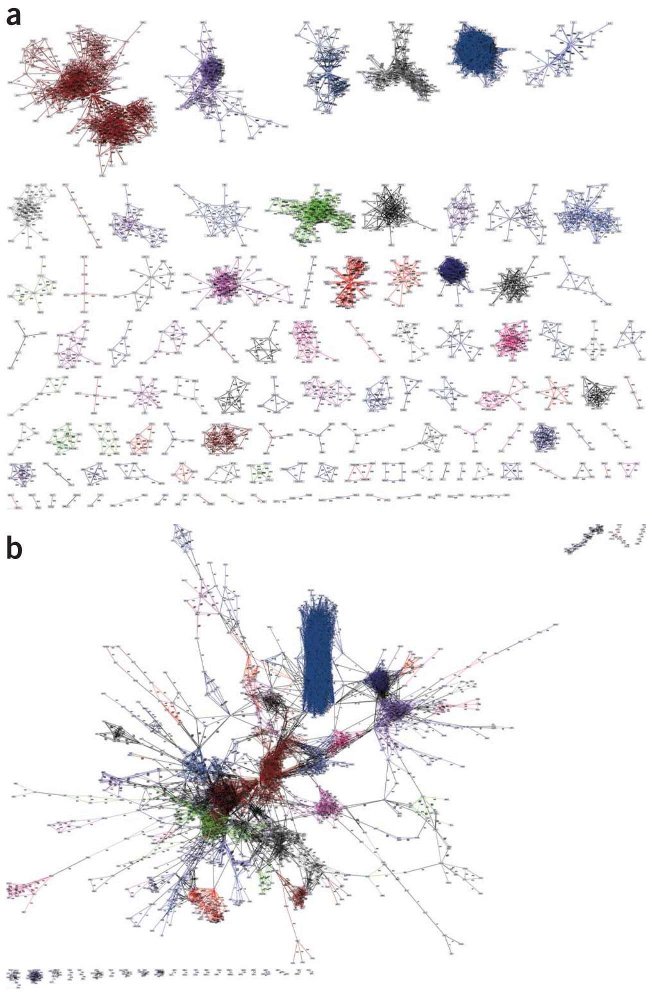


Figure 2 | Mapping of the clusters obtained with the MCL algorithm on the STRING database data set. (**a**) Only the intracluster edges were conserved and each cluster is highlighted with a different color. (**b**) Intercluster (black) and intracluster (colored) edges are both displayed. The layout and display were obtained with yEd.

Fuzzy clustering

20| We will now use the tool ***graph-cluster-membership*** to compute the degree of membership of each node to each of the cluster obtained from MCL. This can be done in either of two ways. (i) Come back to the page with the MCL output and click on '***Cluster membership***' to open ***graph-cluster-membership***. (ii) Alternatively, you can click on the link ***Cluster membership*** in the left panel, and specify the graph parameters as in Steps 5 and 6. To upload the MCL output, click on the button besides '***Upload clusters from file***': and specify the location of the file *yeast_string_MCL_clusters.tab* on your disk.

21| Search for the membership matrix and select weight as stat.
▲ CRITICAL STEP For weighted graphs, weight or relative weight may be chosen. Otherwise, the strength of the links is not considered for calculating the membership of a node to a cluster. When relative weight or relative edge is selected, the weights or number of edges of a node to a cluster are divided by the number of nodes of that cluster.

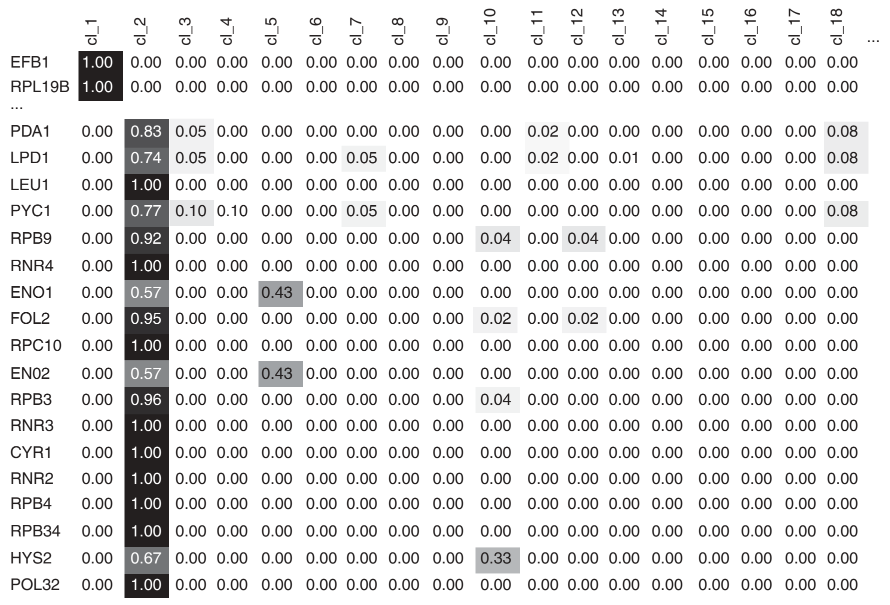
22| Click on the ***GO*** button. After a minute, a page appears with links to three files: a tab-delimited text file, and two image files providing, respectively, low- and high-resolution heatmaps. In all cases, the output displays the membership matrix, where entries correspond to the membership degree of the node given by the row to the cluster given by the column. The text-formatted table contains the numeric values of the memberships coefficient associating each node (row) to each cluster (column). This is a tab-delimited file that can be loaded in various programs (e.g., Excel, R) for further processing. The heatmap is a graphical representation of the same data, where the gray level is proportional to the degree of membership (**Fig. 3**).
? TROUBLESHOOTING

23| *Comparing the clusters with reference classes (Steps 23–27)*. To evaluate the biological relevance of the clusters discovered with MCL, we can compare them with some reference classification, for example the Gene Ontology[36] or the collection of protein complexes from the MIPS database. To illustrate this, we will compare the MCL clusters obtained above with the complexes stored in the MIPS database. Each MCL cluster will be compared to each complex of the database. Cluster/complex comparisons will then be scored with different statistics described in the manual page of the tool and in **Box 2**. Come back to the page with the MCL result (Step 9). On in the ***Next step*** box, click on the button '***Compare these clusters with other clusters***'. Alternatively, in case you saved the MCL result in a file, you can directly click on the link ***Compare clusters/classes*** in the left panel and upload the MCL result file with the ***Browse*** button under ***query class***.

24| As reference classes, we will use the collection of MIPS complexes. For this, first download the file *mips_complexes_names.tab* from the data repository (see EQUIPMENT).

25| We will now specify the ***reference classes*** in the *compare-classes* form. To indicate that MIPS complexes will serve as reference classes, click on the ***Browse...*** button below ***Reference classes***, and select the file ***mips_complexes_names.tab*** that you downloaded on your computer at Step 24. In NeAT, classes are described with the same tab-delimited format as clusters: each row describes the membership of one element (first column) to a class (second column). Optionally, an additional column can be specified with the option '***Score column***', to indicate a score that will be used to compute some similarity metrics (e.g., dot product). For this study case, the MIPS complexes are described in a three-column file indicating the protein name (first column), the complex (second column) and the gene ID (third column). There is no score associated to the protein-complex membership, and we will thus leave empty the option ***score column***.

26| The options ***Thresholds on return fields*** (see **Box 2**) allow one to combine various constraints to select the most significant intersections between query and reference clusters/classes. The default threshold on significance ($sig \geq 0$) usually gives satisfactory results. For our study case, the other thresholds do not need to be changed.



	cl_1	cl_2	cl_3	cl_4	cl_5	cl_6	cl_7	cl_8	cl_9	cl_10	cl_11	cl_12	cl_13	cl_14	cl_15	cl_16	cl_17	cl_18	...
EFB1	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
RPL19B	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
...																			
PDA1	0.00	0.83	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.08	
LPD1	0.00	0.74	0.05	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.08	
LEU1	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
PYC1	0.00	0.77	0.10	0.10	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	
RPB9	0.00	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	
RNR4	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
ENO1	0.00	0.57	0.00	0.00	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
FOL2	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	
RPC10	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
EN02	0.00	0.57	0.00	0.00	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
RPB3	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
RNR3	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
CYR1	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
RNR2	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
RPB4	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
RPB34	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
HYS2	0.00	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
POL32	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

Figure 3 | Fuzzy-clusters obtained by combining MCL and the graph-cluster-membership tools. The section of the heatmap shows that several proteins have cluster membership percentages larger than zero for several clusters; for example, ENO1 belongs to both clusters cl_2 (57%) and cl_5 (43%).

BOX 2 | METRICS FOR COMPARISONS BETWEEN CLASSES OR BETWEEN GRAPHS

In several sections of this protocol, we try to detect significant intersections between two classifications (e.g., MCL clusters, MIPS complexes, etc.) or between two graphs (e.g., interactome). The Network Analysis Tools suite includes specialized programs to compare classes/clusters (*compare-classes*), or graphs (*compare-graphs*), using various comparison statistics.

In both cases (classes or graphs), we can consider that we have a finite set of N elements. For *compare-classes*, N is the total number of elements that can be a member of any reference or query class (e.g., all the yeast genes). For *compare-graphs*, the N elements are all the edges that could possibly be traced between any pair of nodes of the input graph (e.g., all possible intersections between any pair of proteins).

Let us then define

N	the total number of elements in the universe (cluster/class members for *compare-classes*, graph edges for *compare-graphs*);
R	a reference set (one class/cluster, or one graph), containing Nr elements;
Q	a query set (one class/cluster, or one graph), containing Nq elements;
C	the intersection between a query and the reference set;
Nc	the number of elements in this intersection.

Maximal number of edges in a graph

The maximal number of arcs between a set of X nodes depends on whether this graph is directed or not and on whether it does or does not admit self-loops. We can easily compute the value in the four possible cases.

Directed	Self-loops	Number of edges
Yes	Yes	$N = X^2$
Yes	No	$N = X^2 - X = X(X-1)$
No	No	$N = \frac{X(X-1)}{2}$
No	Yes	$N = \frac{X+X(X-1)}{2} = \frac{X(X+1)}{2}$

The column "Number of edges" corresponds to the N used for the statistics on graph comparisons.

Jaccard coefficient

The Jaccard coefficient is defined as the ratio between the intersection and the union between two sets.

$$J = \frac{R \cap Q}{R \cup Q} = \frac{Nc}{Nr + Nq - Nc}.$$

The advantage of the Jaccard coefficient is that it gives us an intuitive perception about the mutual coverage of the query and the reference. However, it presents the weakness to be independent of the absolute sizes of the union and intersection. For example, an intersection of 1 element between a set of 3 and a set of 2 elements will give the same Jaccard coefficient as an intersection of 100 between a set of 300 and a set of 100 elements, whereas the random expectation for these two events is very different. A more reliable statistics is the hypergeometric coefficient, as discussed below.

The hypergeometric probability

The hypergeometric distribution is often used to estimate the significance of the intersection between two random selections in a set. The classical example of application of the hypergeometric distribution is the random selection without replacement in an urn containing a set of white and black balls.

The reference set (classes or graph) can be assimilated to the black balls of the urn example. The query set corresponds to the selection without replacement (indeed, a member cannot appear several times in the same class and an edge cannot appear several times in the same graph). The hypergeometric P value indicates the probability to observe by chance at least x elements at the intersection between the query set and the reference set.

$$P_{\text{val}} = P(X \geq Nc) = \sum_{i=Nc}^{Nq} \frac{C_{Nr}^{i} C_{N-Nr}^{Nq-i}}{C_{N}^{Nq}}.$$

The P value can be interpreted as the probability for one comparison to return a false positive.

In the case of *compare-classes*, an important correction has to be done for multitesting. Indeed, each class of the query set (e.g., MCL clusters) will be compared to each class of the reference set (e.g., MIPS complexes). The number of comparisons is thus the product between the number of classes in the query set, and in the reference set, respectively. Thus, the nominal P value can be misleading because even a low P value is expected to emerge by chance when the number of comparisons is very high. A classical correction for this multitesting is to compute the E value.

$$E_{\text{val}} = T \cdot P_{\text{val}},$$

where T is the number of tests. The E value represents the number of false positives to be expected in a battery of *t*-tests.

To give a realistic order of magnitude, in our study case, we compared 243 clusters obtained from MCL with 107 complexes annotated in the MIPS. The number of comparisons is thus $T = 243 \times 107 = 26{,}001$. Consequently, with an upper threshold of 1% on the P value, we would expect at least 260 false positives!

It is also usual to perform a minus log conversion of the E value, which gives the 'significance score'.

$$\text{sig} = -\log_{10}(E_{\text{val}}).$$

The *sig* score gives an intuitive perception of the exceptionality of the intersection between sets: a negative significance indicates that an intersection of at least that size is likely to occur by chance, a positive value means that it is significant.

27| Click on the ***GO*** button. After a few seconds, a result page appears with links pointing toward two alternative output formats: tab-delimited text file or hypertext markup language (HTML) page (**Fig. 4**). The tab-delimited text file can be downloaded to your computer and then imported to various applications for further analysis. The HTML format is useful for inspecting and handling the result on the Web browser. The NeAT HTML tables support dynamic reordering of the rows according to the values of any column, by clicking on the header of this column. The first line of the result file indicates the parameters used for the analysis and documents the content of the columns. This is followed by a result table, where each line reports the comparison between one MCL cluster and one complex.

Network comparison

28| In this section, we will use the tool ***compare-graphs*** to compare the interactions annotated for the yeast *Saccharomyces cerevisiae* in the STRING database with the synthetic lethality relationships obtained from the BioGRID database. Download on your computer the file *Saccharomyces_cerevisiae_biogrid_synthetic_lethality_names.tab* from the data repository (see EQUIPMENT).

29| In the NeAT menu of the left panel, click on the link ***Network comparison***. For our study case, select as ***Query graph*** the previously downloaded file *yeast_string_database_graph_names_undirected.tab*, and as ***Reference graph*** the file *Saccharomyces_cerevisiae_biogrid_synthetic_lethality_names.tab*, by clicking on the corresponding ***Browse...*** buttons. For each file, you need to specify the columns containing source and target nodes, respectively. In both files, the first column contains the source and second column the target. We thus just have to fill the weight column for the query graph (weight = 3) as done previously. The reference network (synthetic lethality) does not contain edge weights.

30| Since in our example, only the edges of the STRING network are weighted, select ***weight/label of the query*** for the option ***Weight/label on the edges of the output graph***.

```
; compare-classes  -v 1 -r /home/rsat/rsa-tools/public_html/tmp/compare-ref-classes.4BlvXAMVHb -q /home/rsat/rsa-tools/public_html/tmp/compare-query-classes.rw4INKwoRT -return rank,
  occ,proba -lth Q 1 -lth R 1 -lth QR 1 -lth sig 0 -sort sig
; Input files
;                               query_classes                /home/rsat/rsa-tools/public_html/tmp/compare-query-classes.rw4INKwoRT
;                               ref_classes                  /home/rsat/rsa-tools/public_html/tmp/compare-ref-classes.4BlvXAMVHb
; Query classes (nq)                                      106
; Reference classes (nr)                                  243
; Elements  in ref classes (nr)                          1121
; Elements in query classes (nq)                         1240
; Elements in query+ref classes                          1861
; Population size                                        1861
; Comparisons (rn*nq)                                   25758
; Multi-testing correction (nc)                         25758
; Sort key                      sig
; Thresholds                    lower                        upper
;                               Q                                     1 NA
;                               QR                                    1 NA
;                               R                                     1 NA
;                               sig                                   0 NA
; Column contents
;                               1                            rank      Rank of the comparison
;                               2                            ref       Name of the first class (called class Q hereafter)
;                               3                            query     Name of the second class (called class R hereafter)
;                               4                            R         Number of elements in class R
;                               5                            Q         Number of elements in class Q
;                               6                            QR        Number of elements found in the intersecion between classes R and Q
;                               7                            QvR       Number of elements found in the union of classes R and Q. This is R or Q.
;                               8                            R!Q       Number of elements found in class R but not class Q
;                               9                            Q!R       Number of elements found in the class Q but not in class R
;                               10                           !Q!R      Number of elements of the population (P) found neither in class Q nor in the class R
;                               11                           P_val     P-value of the intersection, calculated witht he hypergeometric function. Pval = P(X >= QR).
;                               12                           E_val     E-value of the intersection. E_val = P_val * nb_tests
;                               13                           sig       Significance of the intersection. sig = -log10(E_val)
              #rank             ref                          query             R         Q        QR       QvR       R!Q       Q!R      !Q!R   P_val      E_val        sig
                1               Cytoplasmic-ribosomes        cl_1            138       151       123       166        15        28      1695 4.00E-146 1.00E-141     140.99
                2               cytoplasmic-ribosomal-large-scl_1             81       151        74       158         7        77      1703  7.30E-81  1.90E-76      75.73
                3               26S-proteasome               cl_8             36        36        32        40         4         4      1821  2.80E-60  7.10E-56      55.15
                4               cytoplasmic-ribosomal-small-scl_1             57       151        49       159         8       102      1702  1.10E-48  2.80E-44      43.56
                5               19-22S-regulator             cl_8             18        36        18        36         0        18      1825  8.80E-34  2.30E-29      28.64
                6               Pre-replication-complex      cl_17            14        16        14        16         0         2      1845  1.80E-33  4.70E-29      28.32
                7               Replication-complexes        cl_17            49        16        16        49        33         0      1812  3.60E-27  9.30E-23      22.03
                8               Replication-complex          cl_17            19        16        13        22         6         3      1839  3.00E-26  7.80E-22      21.11
                9               H+-transporting-ATPase-vacuolcl_9             15        34        14        35         1        20      1826  3.20E-25  8.20E-21      20.09
               10               20S-proteasome               cl_8             15        36        14        37         1        22      1824  8.60E-25  2.20E-20      19.65
               11               F0-F1-ATP-synthase           cl_9             15        34        13        36         2        21      1825  1.90E-22  5.00E-18       17.3
               12               Cytochrome-bc1-complex       cl_38             9         7         7         9         2         0      1852  2.40E-18  6.10E-14      13.21
               13               Oligosaccharyltransferase    cl_21             9        13         8        14         1         5      1847  3.30E-18  8.50E-14      13.07
               14               Replication-initiation-complecl_17             8        16         8        16         0         8      1845  3.70E-18  9.40E-14      13.03
               15               Replication-fork-complexes   cl_10            30        27        13        44        17        14      1817  4.30E-18  1.10E-13      12.95
               16               Anaphase-promoting-complex   cl_4             11        58        11        58         0        47      1803  1.00E-17  2.60E-13      12.58
               17               Cytochrome-c-oxidase         cl_27             8        10         7        11         1         3      1850  6.30E-17  1.60E-12      11.79
               18               RNA-polymerase-I             cl_2             14       149        14       149         0       135      1712  2.50E-16  6.40E-12      11.19
               19               Cdc28p-complexes             cl_4             10        58        10        58         0        48      1803  3.90E-16  1.00E-11         11
```

Figure 4 | Most significant associations between MCL clusters versus MIPS complexes. This figure shows only the top of the table returned by the program *compare-classes*. Each row represents the comparison between one complex (reference) and one MCL cluster (query).

31| Several alternatives are possible for the option ***Output type***, corresponding to various combinations of the query and reference graphs (union and difference). The arcs of the resulting graph will be labeled and colored differently depending on whether they belong to the query graph only, the reference graph only or to their intersection. For the study case, to only return the arcs that are in common to both graphs, select ***Intersection*** as ***Output type***.

32| If you want to visualize the resulting network with yEd or Cytoscape, select ***GML format*** as ***Output format***.

33| In case your graph is directed, check the option ***Graphs must be considered as directed***, so that an edge from node A to node B is considered as distinct from an edge from B to A. In our study case, protein interactions are undirected, so this option must remain unchecked.

34| Finally, you can indicate whether or not your graph may admit ***self-loops*** (edges having the same node as source and target). In our study case (synthetic lethality versus STRING interactions), the graph is undirected and has no self-loop.
▲ CRITICAL STEP The intersection statistics will be strongly affected by the nature of the graph (directed or not, with or without self-loops), as described in **Box 2**.

35| When this form is filled, click on the ***GO*** button. The computation of the comparison may take some time (between 10 s and a few minutes) depending on the size of the input networks.

36| The result page (**Fig. 5a**) shows statistics about the sizes of the input graphs, their union, intersection and differences (see **Box 2**) and a link pointing to a separate file corresponding to the comparison network. To save this network on your computer, right click on its URL and select ***Save link as***.... The resulting network can be visualized as described above (**Fig. 5b**).

37| The '***Next step***' box permits to use the network resulting from *compare-graphs* as input for some other NeAT programs (clustering, display, randomization, etc.).

Negative controls

38| To check that the results described previously were not obtained by chance only, we can run random negative controls by applying the processes described previously to random graphs. The program ***random-graph*** can be used to generate random graphs according to various random models. Click on the link ***Network randomization*** in the left menu. Upload a graph (e.g., *yeast_string_database_graph_names_undirected.tab*) and select the output format of your choice.
▲ CRITICAL STEP The most important parameter is the choice of the type of randomization. In general, we would recommend to select the option ***node degree conservation*** that consists in shuffling the edges, each node keeping the same number of neighbors as in the original graph. This procedure is specially designed to avoid duplicating edges, unless you check the option '***Allow duplicated edges***' (this should usually not be done). Another randomization type is the ***node degree distribution conservation*** where the global distribution of the node degree is conserved but each node presents a different degree than in the original graph. Finally, the program also supports ***Erdös-Renyi randomization***, where edges are distributed between pairs of nodes with equal probability.

39| To obtain the randomized network, click on the ***GO*** button.

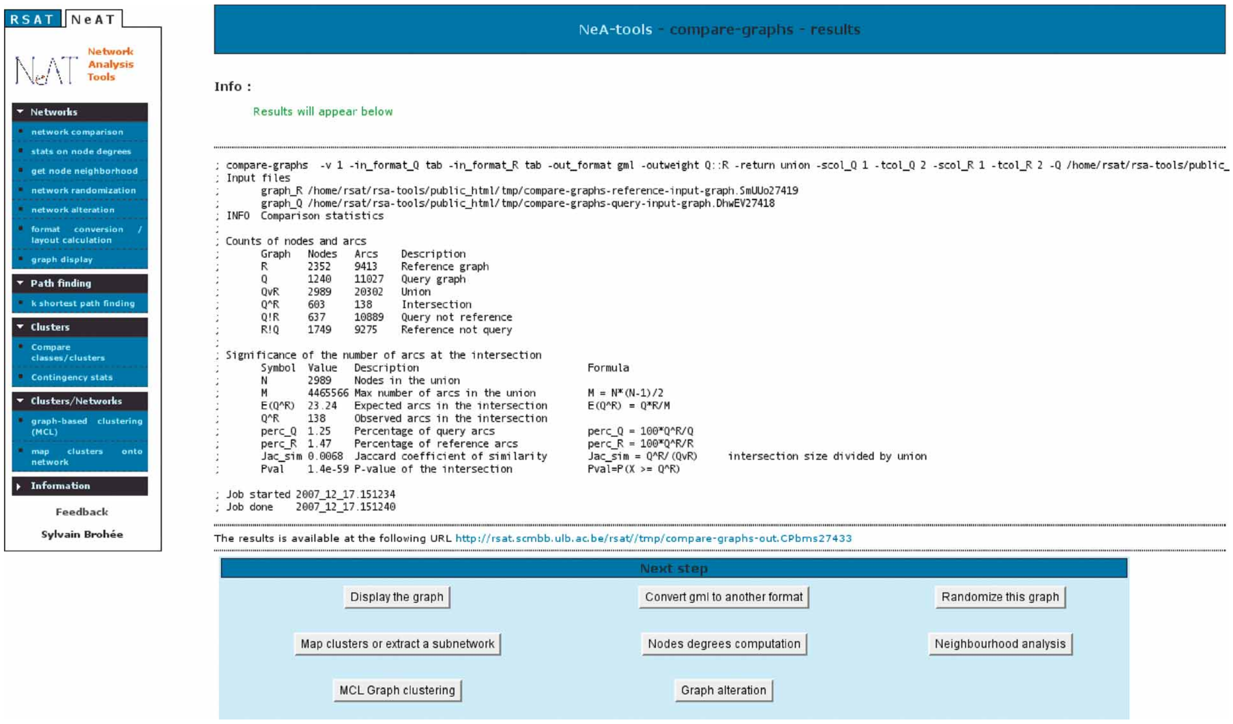
40| You can now apply to this randomized network all the steps described in the previous paragraphs (clustering, subnetwork extraction, comparison with reference graph, etc.). In principle, the results obtained with the randomized graph should be clearly less convincing than those obtained with the real STRING interaction network.

Path finding

41| Given an interaction network (e.g., the STRING database network) and two query proteins, we can ask which intermediate proteins connect them. This question can be answered using *Pathfinder*, a tool that retrieves the *k*-shortest paths in a network for given source and target nodes (see **Box 3** for more information on *k*-shortest paths finding). The STRING network with converted weights is available in the data repository (see EQUIPMENT), in the file *string_database_graph_converted_weights.tab*. Download this file to your computer.

42| In the NeAT main menu, click on the menu ***Path finding***, then on ***k shortest path finding*** to open the *Pathfinder* query form. Upload your network by clicking on the ***Browse...*** button in the section ***Network***. Alternatively, you can copy–paste the network into the text field. For the case you would like to store this network on the server for later use, click the

a



Info :

Results will appear below

```
; compare-graphs  -v 1 -in_format_Q tab -in_format_R tab -out_format gml -outweight Q::R -return union -scol_Q 1 -tcol_Q 2 -scol_R 1 -tcol_R 2 -Q /home/rsat/rsa-tools/public_
; Input files
;       graph_R /home/rsat/rsa-tools/public_html/tmp/compare-graphs-reference-input-graph.SmUUo27419
;       graph_Q /home/rsat/rsa-tools/public_html/tmp/compare-graphs-query-input-graph.DhwEV27418
; INFO  Comparison statistics
;
; Counts of nodes and arcs
;       Graph   Nodes   Arcs    Description
;       R       2352    9413    Reference graph
;       Q       1240    11027   Query graph
;       QvR     2989    20302   Union
;       Q^R     603     138     Intersection
;       Q!R     637     10889   Query not reference
;       R!Q     1749    9275    Reference not query
;
; Significance of the number of arcs at the intersection
;       Symbol  Value   Description                         Formula
;       N       2989    Nodes in the union
;       M       4465566 Max number of arcs in the union     M = N*(N-1)/2
;       E(Q^R)  23.24   Expected arcs in the intersection   E(Q^R) = Q*R/M
;       Q^R     138     Observed arcs in the intersection
;       perc_Q  1.25    Percentage of query arcs            perc_Q = 100*Q^R/Q
;       perc_R  1.47    Percentage of reference arcs        perc_R = 100*Q^R/R
;       Jac_sim 0.0068  Jaccard coefficient of similarity   Jac_sim = Q^R/(QvR)     intersection size divided by union
;       Pval    1.4e-59 P-value of the intersection         Pval=P(X >= Q^R)

; Job started 2007_12_17.151234
; Job done    2007_12_17.151240
```

The results is available at the following URL http://rsat.scmbb.ulb.ac.be/rsat//tmp/compare-graphs-out.CPbms27433

b

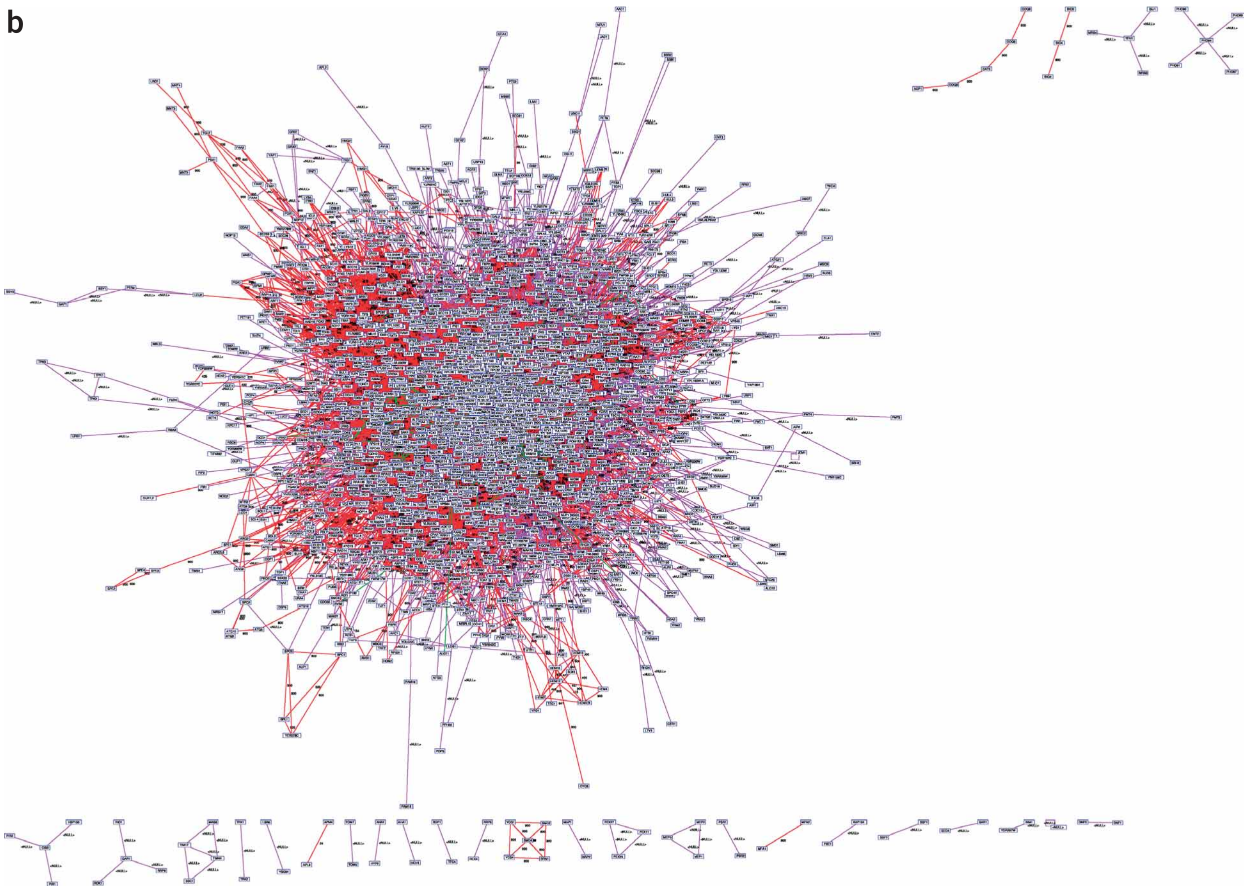


Figure 5 | Result of the fusion between the BioGRID synthetic lethality data set (reference graph) and the yeast–protein interaction data set annotated in the STRING database (query graph). (**a**) Comparison statistics (see **Box 2**). (**b**) Drawing of the union graph (with yEd). Color code: red edges, false positives (edges found in the query graph but not in the reference graph); blue edges, false negatives (edges found in the reference graph but not in the query); green edges, true positives (edges present in both networks, in this case, only 138 among the ~20,000).

BOX 3 | *k*-SHORTEST PATHS FINDING IN WEIGHTED NETWORKS

Path finding attempts to find the shortest path between a given start node and a given end node in a network (graph). If several such paths exist, they should be all returned as equally valid solutions. Sometimes, we are not only interested in the shortest path, but also in the second shortest, third shortest or, in general, the *k*-shortest paths. The task of a *k*-shortest paths algorithm is to enumerate all paths up to the requested rank (*k*) in the order of their length. This is accomplished for example by the recursive enumeration algorithm[37] or by Eppstein's algorithm[38]. Often, the edges in biological networks are not equally relevant. For example, experimentally validated protein–protein interactions are more trustable than those observed in only one high-throughput experiment. To express such differential reliabilities, a higher *cost* (*weight*) can be placed on edges representing less trustable protein–protein interactions. When costs, or weights, have been set on the nodes or edges of a network, we no longer search for the *shortest* but for the *lightest* (that is less costly) path. Consequently, the *k*-shortest paths algorithm returns paths ranked according to their *weight* with the lightest path on top.

The weights have to be selected in a relevant way for the biological network of interest. The choice of a relevant weighting criterion clearly depends on your experience about this network and about the quality of the data available.

In a previous study[21,22], we evaluated the accuracy of *k*-shortest path finding for inferring metabolic pathways from compound/reaction networks, and showed that a graph where each node is weighted according to its degree (number of incoming + outgoing edges) achieves an accuracy of 83%.

In our study case with the yeast interaction network, we will use the scores provided by STRING as weights. In this case, the score assigned to an edge is a measurement for the reliability of the protein–protein interaction represented by this edge. In contrast, for Pathfinder, an edge weight is the cost of this edge. Therefore, we converted the scores into costs using the following formula:

$$W_e = \frac{1{,}000}{S_e},$$

where S_e is the score of an edge as defined in STRING (from 0 to 1,000), and W_e is the weight assigned to that edge for path finding.

Store network on server check box. This will allow you to perform further analyses on the same network, without having to transfer it repeatedly from your computer to the server.

43| Enter the IDs of the source and target nodes. For this study case, type RAS2 in the ***Source nodes*** field, and *TEC1* into the ***Target nodes*** field.

44| For the option ***Weighting scheme***, select '*as given in input graph*'. This will specify that weights should be taken from the third column of the input file and not calculated according to a predefined weighting scheme.

45| The result can be exported in various formats, depending on what you want to do with the resulting paths. (i) If you want to display the path in a tabular format, select ***Table*** as ***Output format***. (ii) The result can also be exported to GML format, to visualize the resulting paths as a subset of the original network (this can be done with visualization Cytoscape or yEd). For this, select ***Graph*** as ***Output format***, set the ***Graph format*** to ***GML format*** and set the ***Graph output type*** to ***paths unified into one graph***.

46| Click ***GO*** to start the computation.
? TROUBLESHOOTING

47| The result will be displayed according to the option chosen at Step 45. (i) Output format 'Table'. After a few seconds (or minutes, depending on the size of your graph), the results should appear in the form of two links. The first link points to the table of paths in simple text format, the second to the same table in HTML format (**Fig. 6**). If the checkbox ***Store network on server*** has been clicked, Pathfinder returns the identifier of the submitted network in addition. Submitting this identifier instead of the network itself speeds up the next path-finding job performed on it, because the previously transferred network is used, thereby avoiding to upload it again. (ii) The result form will contain a link to the resulting network in a GML file, which can be downloaded on your computer and displayed with Cytoscape or yEd. This link is followed by a '***Next steps***' box, which will permit you to fetch the result network into another NeAT tool.

● TIMING

The timings listed below depend on the server load (the number of jobs currently running on the server). However, for the study case we expect the tools to finish within 5 min.
Compare graph: $<$30 s; MCL: $<$20 s; graph-get-clusters: $<$40 s; display-graph: $<$1 min; compare-classes: $<$20 s; Pathfinder: $<$1 min; Fuzzy clustering: $<$2 min

? TROUBLESHOOTING

Troubleshooting advice can be found in **Table 1**.

```
; Experiment exp_0
; Pathfinding results
; Date=Thu Jun 26 16:03:53 CEST 2008
; ===============================
; INPUT
; Source=RAS2
; Target=TEC1
; Graph=Pathfinder_tmpGraph_d597cd86-3095-475f-99e7-d70a542d072a.tab
; Undirected=true
; Metabolic standard format=false
; REA format=false
; Temporary directory=Temp
; CONFIGURATION
; Algorithm=rea
; Weight Policy=
; Weights given on arcs=true
; Maximal weight=1000000
; Maximal length=1000000
; Minimal length=0
; Exclusion attribute=ReferencedObject.PublicId
; Rank=5
; REA timeout=10
; EXPLANATION OF COLUMNS
; Start node=given start node identifier
; End node=given end node identifier
; K=path index
; Rank=rank of path (paths having same distance have the same rank, though their step number might differ)
; Distance=weight of path (sum of edge weights)
; Steps=number of nodes in path
; Path=sequence of nodes from start to end node that forms the path
; ===============================
#start node   end node   path index   rank   distance   steps   path
RAS2          TEC1                1      1       6.25       6 RAS2->CDC42->STE20->FUS3->STE12->TEC1
RAS2          TEC1                2      2        7.5       7 RAS2->CDC42->STE20->STE11->FUS3->STE12->TEC1
RAS2          TEC1                3      2        7.5       7 RAS2->CDC42->SHO1->STE20->FUS3->STE12->TEC1
RAS2          TEC1                4      2        7.5       7 RAS2->CDC42->STE20->FUS3->DIG1->STE12->TEC1
RAS2          TEC1                5      2        7.5       7 RAS2->CDC42->BEM1->STE20->FUS3->STE12->TEC1
RAS2          TEC1                6      2        7.5       7 RAS2->CDC42->STE20->STE5->FUS3->STE12->TEC1
RAS2          TEC1                7      2        7.5       7 RAS2->CDC42->STE20->FUS3->DIG2->STE12->TEC1
RAS2          TEC1                8      2        7.5       7 RAS2->CDC42->STE20->STE7->FUS3->STE12->TEC1
RAS2          TEC1                9      9       8.75       8 RAS2->CDC42->STE20->STE5->STE7->FUS3->STE12->TEC1
RAS2          TEC1               10      9       8.75       8 RAS2->CDC42->BEM1->STE20->STE11->FUS3->STE12->TEC1
RAS2          TEC1               11      9       8.75       8 RAS2->CDC42->STE20->STE7->STE11->FUS3->STE12->TEC1
RAS2          TEC1               12      9       8.75       8 RAS2->CDC42->STE20->STE5->STE11->FUS3->STE12->TEC1
RAS2          TEC1               13      9       8.75       8 RAS2->CDC42->SHO1->STE20->FUS3->DIG1->STE12->TEC1
RAS2          TEC1               14      9       8.75       8 RAS2->CDC42->BEM1->STE20->STE7->FUS3->STE12->TEC1
RAS2          TEC1               15      9       8.75       8 RAS2->CDC42->STE20->STE5->FUS3->DIG1->STE12->TEC1
RAS2          TEC1               16      9       8.75       8 RAS2->CDC42->STE20->STE7->FUS3->DIG1->STE12->TEC1
RAS2          TEC1               17      9       8.75       8 RAS2->CDC42->STE20->STE11->STE7->FUS3->STE12->TEC1
RAS2          TEC1               18      9       8.75       8 RAS2->CDC42->STE20->STE11->STE5->FUS3->STE12->TEC1
RAS2          TEC1               19      9       8.75       8 RAS2->CDC42->STE20->STE7->KSS1->DIG1->STE12->TEC1
RAS2          TEC1               20      9       8.75       8 RAS2->CDC42->SHO1->STE20->STE5->FUS3->STE12->TEC1
RAS2          TEC1               21      9       8.75       8 RAS2->CDC42->BEM1->STE20->FUS3->DIG1->STE12->TEC1
RAS2          TEC1               22      9       8.75       8 RAS2->CDC42->STE20->STE11->FUS3->DIG2->STE12->TEC1
RAS2          TEC1               23      9       8.75       8 RAS2->CDC42->STE20->STE7->FUS3->DIG2->STE12->TEC1
RAS2          TEC1               24      9       8.75       8 RAS2->CDC42->BEM1->STE20->FUS3->DIG2->STE12->TEC1
RAS2          TEC1               25      9       8.75       8 RAS2->CDC42->STE20->FUS3->DIG1->DIG2->STE12->TEC1
RAS2          TEC1               26      9       8.75       8 RAS2->CDC42->STE20->FUS3->DIG2->DIG1->STE12->TEC1
RAS2          TEC1               27      9       8.75       8 RAS2->CDC42->SHO1->STE20->STE11->FUS3->STE12->TEC1
RAS2          TEC1               28      9       8.75       8 RAS2->CDC42->STE20->STE7->KSS1->DIG2->STE12->TEC1
RAS2          TEC1               29      9       8.75       8 RAS2->CDC42->STE20->STE7->STE5->FUS3->STE12->TEC1
RAS2          TEC1               30      9       8.75       8 RAS2->CDC42->STE20->STE5->FUS3->DIG2->STE12->TEC1
RAS2          TEC1               31      9       8.75       8 RAS2->CDC42->SHO1->STE20->STE7->FUS3->STE12->TEC1
RAS2          TEC1               32      9       8.75       8 RAS2->CDC42->STE20->STE11->FUS3->DIG1->STE12->TEC1
RAS2          TEC1               33      9       8.75       8 RAS2->CDC42->BEM1->STE20->STE5->FUS3->STE12->TEC1
RAS2          TEC1               34      9       8.75       8 RAS2->CDC42->SHO1->STE20->FUS3->DIG2->STE12->TEC1
```

Figure 6 | Result obtained with *Pathfinder* upon execution of protocol with the study case. The table lists the paths found between RAS2 (source node) and TEC1 (target node), ranked by increasing value of weight (distance). RAS2 and TEC1 are the start and end node of the filamentous growth pathway in yeast.

TABLE 1 | Troubleshooting table.

Step	Problem	Possible reason	Solution
8	After a few minutes, I still do not have any answer and the browser displays "Server is not responding"	If you submitted a heavy task, the processing may exceed 5 min. After that delay, Internet browser programs stop waiting for the server and display the error message	For heavy tasks, it is preferable either to install the stand-alone version of the command-line tools on your machine or to write a client script for the RSAT Web services
		Another possibility (if the task you submitted is not heavy) is that there is a problem with your Internet connection	
17	No graph layout after having loaded a GML file into yEd or Cytoscape	When a graph is loaded in yEd or CytoScape, it is initially displayed with a trivial layout (all nodes on a diagonal)	In yEd: select **Layout** from the menu, then select the submenu **Organic** and choose the option **Classic**, then click **OK**
			In Cytoscape: select **Layout** from the menu, then select the submenu **yFiles** and choose **Organic**
			Both editors offer other layouts that you may try

(continued)

TABLE 1 | Troubleshooting table (continued).

Step	Problem	Possible reason	Solution
22	You obtain the message: "Error Incongruence between graph and cluster files"	Cluster and graph files do not correspond to the same network, or the specified format of the graph is not correct	Check that the clusters correspond to the graph. If so, check the format of your graph file is the one you entered in the relevant field
	The low-resolution heatmap does not display properly	The image is scaled so it fits on the window	Click on the image to zoom. You may inspect the whole map by using the scrolling bar
46	You obtain the message: "PATHFINDER ERROR: One of your seed nodes is not part of the input graph"	You provided seed node identifiers that do not match any of the node identifiers of the input graph	Check the spelling of your seed node identifiers In general, all tools require an exact match between input node identifiers and those of nodes in the network

ANTICIPATED RESULTS
Clustering
Figure 2 shows the results that should be obtained by applying the MCL graph clustering algorithm on the STRING database interaction network. Each cluster is highlighted with a specific color. **Figure 2a** only displays intracluster edges, so that each cluster appears as a separate component. This representation highlights the intracluster structure and edge density, and could give indication about possible improvement of the clusters by further subdivision. For example, the top-left cluster seems to be composed of several various connected regions, which could be explored in more detail, taking into account some biological knowledge. On **Figure 2b**, both intra- and interclusters are displayed. Intracluster edges are highlighted by cluster-specific colors, whereas intercluster edges are displayed in black, thereby revealing the interactions that were discarded during the clustering procedure. These two representations thus provide complementary indication for the interpretation of the clustering result.

The heatmap in **Figure 3** represents a section of the node–cluster membership matrix, with cells colored according to the membership degree (the darker the cell, the higher the membership value). Within each cell, the membership degree values are displayed, indicating how strongly each node (row) is connected to each cluster (column). This strength (node–cluster membership) is defined as the sum of weights of the edges connecting the considered node to the considered cluster, divided by the sum of weights of all edges starting from this node. The figure shows only a fragment of the table, but it already appears that some genes have similar membership profiles, thereby suggesting their involvement in common functions. This is the case of the genes *LPD1*, *PDA1* and *PYC1*, which are involved in pyruvate metabolism.

Figure 4 shows the results of the comparison between the dense clusters of the STRING graph and the complexes annotated in the MIPS database. The header gives a short explanation for the content of each column of the result table. In this case, results are sorted by decreasing values of the hypergeometric significance (last column) calculated as described in **Box 2**. Each row describes the comparison between one MCL cluster and one MIPS complex. For example, the first row compares the MCL cluster '*cl_2*', which contains 151 proteins, with a set of 138 proteins involved in cytoplasmic ribosomes. The intersection contains 123 proteins, which represents a very high fraction of both the MCL cluster, and the annotated complex. The probability to observe such an intersection by chance is 4E–146. The E value, obtained with the correction for multitesting, indicates that the number of false positives expected with such a *P* value would be 1E–141. In other terms, the correspondence between this MCL cluster and the cytoplasmic ribosome is too high to be explained by chance.

Network comparison
Figure 5a shows the statistics of comparison between the STRING 'database' network and the BioGRID synthetic lethality data set. The 'synthetic lethality' network used as reference contains 2,352 proteins linked by 9,413 edges, and the query graph contained 1,240 nodes and 11,027 edges. The intersection between those graphs is apparently weak: 138 edges only. The Jaccard coefficient indicates that this intersection represents no $>0.68\%$ of the union. However, the number of edges expected by chance at the intersection is even smaller: $E(Q \wedge R) = 23.24$. The hypergeometric *P* value (**Box 2**) indicates the probability to observe at least 138 edges at the intersection when 23.24 are expected by chance. In this study case, we observe that even with no $>0.68\%$ of edges at the intersection, the *P* value is very low (1.4E−59). In other terms, the number of edges at the intersection is too much high to be explained by chance, and is more likely to result from the biological relevance of both datasets.

Pathfinder
The known signal transduction path connecting RAS2 and TEC1 consists of the following steps[18]:
RAS2 - CDC42 - STE20 - STE11 - STE7 - KSS1 - DIG1/2 - TEC1

Pathfinder reports the following path of first rank (the matching parts are underlined, and the nonseed matching part are highlighted in bold):

RAS2 - **CDC42 - STE20** - FUS3 - STE12 - TEC1

This path connects STE20 to TEC1 via FUS3 and STE12, bypassing STE11, STE7, KSS1 and DIG1/2.

Among the paths of length 8 (third rank paths), we find paths closer to the annotated pathway, such as

RAS2 - **CDC42 - STE20 - STE11** - FUS3 - **DIG1** - STE12 - TEC1

Scott and colleagues applied their path-finding algorithm to another yeast protein–protein interaction network of similar size (4,500 nodes and 14,500 edges) taken from MIPS. For RAS2 and TEC1, they obtain the following as best path of length 8: RAS2 - CDC25 - HSP82 - **STE11** - STE5 - **STE7** - **KSS1** - TEC1

Although Pathfinder has not been designed in particular for protein interaction networks, it can be used to predict signal transduction pathways if appropriate weights have been set on the network under investigation. The accuracy of the prediction depends also on the data quality. For example, the STRING interaction network does not contain any edge between DIG2/DIG1 and TEC1, making it impossible to reach a prediction accuracy of 100%. When predicting pathways from real-world interaction networks, one must always keep in mind that these data might be incomplete or contain false positive interactions.

ACKNOWLEDGMENTS S.B. is the recipient of a PhD grant from the Fonds pour la Formation à la Recherche dans l'Industrie et dans l'Agriculture (FRIA). K.F. is supported by the Actions de Recherches Concertées de la Communauté Française de Belgique (ARC grant number 04/09-307). G.L.-M. was funded by a PhD grant from the Fonds Xenophilia [Université Libre de Bruxelles (ULB)] and by a postdoctoral fellowship from the Région Wallonne de Belgique (TransMaze project 415925). The BiGRe laboratory is supported by the BioSapiens Network of Excellence funded under the sixth Framework program of the European Communities (LSHG-CT-2003-503265) and by the Belgian Program on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office, project P6/25 (BioMaGNet). We acknowledge the students of the Master in Bioinformatics and Modeling (ULB, Belgium) for their useful corrections and suggestions. S.B. and K.E. equally contributed to this article.

Published online at http://www.natureprotocols.com/
Reprints and permissions information is available online at http://npg.nature.com/reprintsandpermissions/

1. Thomas-Chollier, M. *et al.* RSAT: regulatory sequence analysis tools. *Nucleic Acids Res.* **36**, W119–W127 (2008).
2. Brohée, S. *et al.* NeAT: a toolbox for the analysis of biological networks, clusters, classes and pathways. *Nucleic Acids Res.* **36**, W444–W451 (2008).
3. Turatsinze, J.-V., Thomas-Chollier, M., Defrance, M. & van Helden, J. Using RSAT to scan genome sequences for transcription factor binding sites and *cis*-regulatory modules. *Nat. Protoc.* doi:10.1038/nprot.2008.97 (2008).
4. Defrance, M., Janky, R., Sand, O. & van Helden, J. Using RSAT oligo-analysis and dyad-analysis tools to discover regulatory signals in nucleic sequences. *Nat. Protoc.* doi:10.1038/nprot.2008.98 (2008).
5. Sand, O., Thomas-Chollier, M., Vervisch, E. & van Helden, J. Analyzing multiple data sets by interconnecting RSAT programs via SOAP Web services–an example with ChIP-chip data. *Nat. Protoc.* doi:10.1038/nprot.2008.99 (2008).
6. Jeong, H., Tombor, B., Albert, R., Oltvai, Z.N. & Barabási, A.L The large-scale organization of metabolic networks. *Nature* **407**, 651–654 (2000).
7. Jeong, H., Mason, S.P., Barabási, A.L. & Oltvai, Z.N. Lethality and centrality in protein networks. *Nature* **411**, 41–42 (2001).
8. Fell, D.A. & Wagner, A. The small world of metabolism. *Nat. Biotechnol.* **18**, 1121–1122 (2000).
9. Blatt, M., Wiseman, S. & Domany, E. Superparamagnetic clustering of data. *Phys. Rev. Lett.* **76**, 3251–3254 (1996).
10. Bader, G.D. & Hogue, C.W.V. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics* **4**, 2 (2003).
11. Gagneur, J., Jackson, D.B. & Casari, G. Hierarchical analysis of dependency in metabolic networks. *Bioinformatics* **19**, 1027–1034 (2003).
12. Spirin, V. & Mirny, L.A. Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. USA* **100**, 12123–12128 (2003).
13. King, A.D., Przulj, N. & Jurisica, I. Protein complex prediction via cost-based clustering. *Bioinformatics* **20**, 3013–3020 (2004).
14. Van Dongen, S. *Graph Clustering by Flow Simulation*. PhD Thesis (Centers for Mathematics and Computer Science (CWI), University of Utrecht, 2000).
15. Pereira-Leal, J.B., Enright, A.J. & Ouzounis, C.A. Detection of functional modules from protein interaction networks. *Proteins* **54**, 49–57 (2004).
16. Enright, A.J., Van Dongen, S. & Ouzounis, C.A. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.* **30**, 1575–1584 (2002).
17. Brohée, S. & van Helden, J. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics* **7**, 488 (2006).
18. Scott, J., Ideker, T., Karp, R.M. & Sharan, R. Efficient algorithms for detecting signaling pathways in protein interaction networks. *J. Comput. Biol.* **13**, 133–144 (2005).
19. Bebek, G. & Yang, J. PathFinder: mining signal transduction pathway segments from protein-protein interaction networks. *BMC Bioinformatics* **8**, 335 (2007).
20. Rahman, S.A., Advani, P., Schunk, R., Schrader, R. & Schomburg, D Metabolic pathway analysis web service (Pathway Hunter Tool at CUBIC). *Bioinformatics* **21**, 1189–1193 (2004).
21. Croes, D., Couche, F., Wodak, S. & van Helden, J. Metabolic PathFinding: inferring relevant pathways in biochemical networks. *Nucleic Acids Res.* **33**, W326–W330 (2005).
22. Croes, D., Couche, F., Wodak, S. & van Helden, J. Inferring meaningful pathways in weighted metabolic networks. *J. Mol. Biol.* **356**, 222–236 (2006).
23. Shannon, P. *et al.* Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**, 2498–2504 (2003).
24. de Nooy, W., Mrvar, A. & Batagelj, V. *Exploratory Social Network Analysis with Pajek* Series: Structural Analysis in the Social Sciences (No. 27) (Cambridge University Press, Cambridge, 2005).
25. Baitaluk, M., Sedova, M., Ray, A. & Gupta, A. BiologicalNetworks: visualization and analysis tool for systems biology. *Nucleic Acids Res.* **34**, W466–W471 (2006).
26. Hu, Z. *et al.* VisANT 3.0: new modules for pathway visualization, editing, prediction and construction. *Nucleic Acids Res.* **35**, W625–W632 (2007).
27. Hull, D. *et al.* Taverna: a tool for building and running workflows of services. *Nucleic Acids Res.* **34**, W729–W732 (2006).
28. Lima-Mendez, G., van Helden, J., Toussaint, A. & Leplae, R. Reticulate representation of evolutionary and functional relationships between phage genomes. *Mol. Biol. Evol.* **25**, 762–777 (2008).
29. Croes, D., Couche, F., Wodak, S.J. & van Helden, J. Inferring meaningful pathways in weighted metabolic networks. *J. Mol. Biol.* **356**, 222–236 (2006).
30. Croes, D., Couche, F., Wodak, S.J. & van Helden, J. Metabolic PathFinding: inferring relevant pathways in biochemical networks. *Nucleic Acids Res.* **33**, W326–W330 (2005).
31. von Mering, C. *et al.* STRING 7-recent developments in the integration and prediction of protein interactions. *Nucleic Acids Res.* **35**, D358–D362 (2007).
32. Breitkreutz, B.J. *et al.* The BioGRID Interaction Database: 2008 update. *Nucleic Acids Res.* **36**, D637–D640 (2008).
33. Keseler, I.M. *et al.* EcoCyc: a comprehensive database resource for *Escherichia coli*. *Nucleic Acids Res.* **33**, D334–D337 (2005).
34. Kanehisa, M., Goto, S., Kawashima, S. & Nakaya, A. The KEGG databases at GenomeNet. *Nucleic Acids Res.* **30**, 42–46 (2002).
35. Mewes, H.W. *et al.* MIPS: analysis and annotation of proteins from whole genomes. *Nucleic Acids Res.* **32**, D41–D44 (2004).
36. Ashburner, M. *et al.* Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.* **25**, 25–29 (2000).
37. Jimenez, V.M. & Marzal, A. Computing the K shortest paths: a new algorithm and an experimental comparison. In *Proceeding of the 3rd International Workshop on Algorithm Engineering (WAE 1999)* Vol. **1668**, 15–29 (Springer-Verlag, London, 1999).
38. Eppstein, D. Finding the k shortest paths. *SIAM J. Comput.* **28**, 652–673 (1998).