Molecular BioSystems

RSC Publishing

PAPER

View Article Online
View Journal | View Issue

Cite this: *Mol. BioSyst.*, 2013, **9**, 1670

MGclus: network clustering employing shared neighbors†

Oliver Frings,[ab] Andrey Alexeyenko[ac] and Erik L. L. Sonnhammer*[abd]

Received 25th October 2012,
Accepted 21st January 2013

DOI: 10.1039/c3mb25473a

www.rsc.org/molecularbiosystems

Network analysis is an important tool for functional annotation of genes and proteins. A common approach to discern structure in a global network is to infer network clusters, or modules, and assume a functional coherence within each module, which may represent a complex or a pathway. It is however not trivial to define optimal modules. Although many methods have been proposed, it is unclear which methods perform best in general. It seems that most methods produce far from optimal results but in different ways. *MGclus* is a new algorithm designed to detect modules with a strongly interconnected neighborhood in large scale biological interaction networks. In our benchmarks we found *MGclus* to outperform other methods when applied to random graphs with varying degree of noise, and to perform equally or better when applied to biological protein interaction networks. *MGclus* is implemented in Java and utilizes the JGraphT graph library. It has an easy to use command-line interface and is available for download from http://sonnhammer.sbc.su.se/download/software/MGclus/.

Introduction

Understanding the relationship between the logical organization of biological networks and their functions has become an important challenge in systems biology.[1,2] The most commonly followed strategy to reduce complexity and extract important information is to partition the data into local components that are densely connected.[1,3,4] The general problem of identifying densely connected components in networks has been studied intensively in recent years and is known under different names such as network module prediction, network clustering, community detection, and graph partitioning.[5]

It has been shown that biological networks have a modular organization,[6] whereas densely connected modules, or network clusters, can often be related to functional gene modules or regulatory pathways.[7–9] Extracting functional modules has not only become a fundamental aspect of systems biology work flows, but also many other areas of bioinformatics such as the analysis of co-expression networks, prediction of protein function, finding functional complexes, and understanding cellular organization.[6,10–14] Large-scale gene interaction networks are constantly growing and have become available for a large number of model organisms.[15–17]

Popular network clustering approaches include *CFinder*,[18] *FastCommunity*,[19] *MCode*,[7] *MCL*,[20] *MINE*,[21] *NEMO*,[5] *SPICi*,[22] and *Cohtop*.[23] Briefly, the *Clique Finder* (*CFinder*) method is based on the Clique Percolation Method (CPM) to locate k-clique percolation clusters. A k-clique is a complete subgraph of size k whereas two k-cliques are said to be adjacent if they share exactly $k - 1$ nodes. The *FastCommunity* method follows a hierarchical agglomerative approach. It starts with each node as its own community and then calculates in each step the expected increase in modularity for the merge of each pair. *Molecular Complex Detection* (*MCode*) has three main steps: vertex weighting, complex prediction, and an optional post-processing step that removes unwanted elements from the final set of clusters. It starts with identifying a set of seed nodes based on local density that are extended in a greedy fashion. The *Markov clustering* (MCL) approach is based on modified random walks on networks. *MINE* (Module Identification in Networks) is a clustering approach similar to *MCode* in that it uses seed nodes that are determined based on the node degree and local density as starting points. Seed nodes are extended in an iterative manner grouping together nodes that improve the modularity score. *NEMO* is based on complete-linkage hierarchical agglomerative clustering. *SPICi* is an extremely speed-efficient local network-clustering method. It builds clusters greedily starting from local seeds that have a high weighted degree and then iteratively adds nodes accounting for the local density around the growing clusters.

[a] *Stockholm Bioinformatics Centre, Science for Life Laboratory, Box 1031, SE-17121 Solna, Sweden. E-mail: Erik.Sonnhammer@sbc.su.se*
[b] *Department of Biochemistry and Biophysics, Stockholm University, Sweden*
[c] *School of Biotechnology, Royal Institute of Technology, Sweden*
[d] *Swedish e-Science Research Center, Sweden*

† Electronic supplementary information (ESI) available. See DOI: 10.1039/c3mb25473a

The *Cohtop* method employs the Kullback–Leibler divergence to assess the gain of merging a pair of clusters. Most methods have in common that they have many user-adjustable parameters which can be a huge complication for inexperienced users.

When defining network modules one typically wants to find groups that are tightly connected, *i.e.* having a minimal number of exterior edges and a maximum number of interior edges. However, this can be too simplistic because most interaction networks available today are static and far from complete. Interactions may be missing for example if they only occur in a certain tissue, upon a given stimulus, or in a particular developmental stage that has not been sampled. When assessing the relevance of a set of nodes to each other, it can therefore be of special importance to explicitly consider their network neighborhood.

The *MGclus* method presented here aims to detect modules not only by maximizing their interior edges while minimizing their exterior edges, but also by promoting a common local network neighborhood. This strategy is intended to circumvent problems caused by incompleteness of biological networks.

Using three different benchmark sets, we demonstrate that *MGclus* performs well when compared against other clustering methods popular in the field. First, we performed a benchmark on random clusters with a controlled degree of inter-cluster and intra-cluster connectivity. Second, we benchmarked *MGclus* on two *S. cerevisiae* PPI (protein–protein interaction) networks from BioGRID. Third, we compared the clustering methods on a large scale human interaction network from FunCoup. In all cases *MGclus* outperformed or performed equally well as the other methods.

Methods

The *MGclus* method is designed to find sub-network structures (clusters) that have an uninterrupted path between any two given nodes (cluster members), minimize the number of *exterior* edges, and maximize the number of *interior* edges.

To balance the number of inner and outer edges, at each step of the clustering, for each pair of distinct clusters $\{i,j\}$ the merge gain (MG) is evaluated.

Theory

The MG for two clusters is defined as follows:

$$\mathrm{MG} = 2E_{ij} - (E_i + E_j) \quad (1)$$

where E_i and E_j are the *clustering efficiency* (similar to modularity) of clusters i and j, and E_{ij} is the efficiency for the union of i and j, which is assessed by:

$$E_i = N_{i,\mathrm{internal}}/N_{i,\mathrm{total}} \quad (2)$$

where:

$$N_{\mathrm{total}} = {N_{\mathrm{nodes}}}^2 \quad (3)$$

$$N_{\mathrm{internal}} = \sum_{x=1}^{N_{\mathrm{nodes-cluster}}-1} \sum_{y=x+1}^{N_{\mathrm{nodes-cluster}}} n_{d(x,y)} + \sqrt{n_{s(x,y)}} \quad (4)$$

The degree of internal connectivity of a cluster ((4)) is assessed, by iterating over all pairs $\{x,y\}$ of nodes ($N_{\mathrm{nodes-cluster}}$) in it. If a pair of nodes $\{x,y\}$ has a direct link between them, $n_{d(x,y)}$ is the weight of the edge connecting them (0 otherwise). $n_{s(x,y)}$ is the sum of the edge weights between x and y and their common neighbors. The neighbors shared between x and y might eventually not belong to the formed cluster, but serve as an indirect measure of the mutual relevance of x and y. In the case of an unweighted network the weight of all edges is set to 1, otherwise they should be between 0 and 1.

Algorithm

In each iteration, MG is assessed for all pairs of clusters $\{i,j\}$ where i and j are initially individual nodes. MG is only calculated for cluster pairs that have at least one direct link connecting them. In order to be considered for merging, the MG for a given pair of clusters $\{i,j\}$ has to be higher than the *merge gain cut-off* C_{MG}. The default C_{MG} is 0, *i.e.* the MG needs to be positive and the union of i and j must account better for the network modularity than i and j taken separately. In each iteration I, all cluster pairs are ranked according to their MG and the t top pairs ($t = 100$) are merged. A merge is however only allowed if none of the two clusters was already merged in the same iteration. Therefore, the number of performed merges is in practice normally lower than t.

The clustering terminates when no further cluster pairs with an MG higher than C_{MG} are found. Depending on the network size and sparseness, a typical run requires between 50 and 100 iterations. In this way, the agglomeration becomes relatively well balanced, *i.e.* no particular size configurations, such as addition of single genes to large clusters, or merging of equally sized clusters are favored. To control the number and compactness of the found clusters the user can specify C_{MG}. A higher cut-off will typically result in more clusters of smaller size and higher density.

Implementation

MGclus is implemented in Java and utilizes the JGraphT graph library for performance oriented handling of large data sets.

As input, the method takes a tab-separated network file, with node names in the first and second column and an optional edge weight in the third column. *Via* the command line interface, the user can further specify various parameters including the merge-gain cut-off.

Clustering methods

Besides *MGclus* we considered eight widely used clustering methods, namely *CFinder*,[18] *FastCommunity*,[19] *MCode*,[7] *MCL*,[20] *MINE*,[21] *NEMO*,[5] *SPICi*,[22] and *Cohtop*.[23] Note that *MCode* and *NEMO* are only available as Cytoscape plugins and could therefore not be included in all benchmarks. To apply *MINE* to the random clusters, we used a Perl implementation provided by the authors. The parameter settings used to run the individual methods are given in the ESI.† We stopped runs that ran longer than 24 hours. All benchmarks were conducted on an Intel Core i7 with 16 GB of memory.

Interaction and gold standard datasets

To evaluate the performance of all methods quantitatively, a set of random clusters with varying degree of inter-cluster and intra-cluster connectivity was assembled. Each set contained a defined number of random clusters each of size 10. Assembly started with a set of unconnected nodes whereas each node was assigned to a particular cluster. The nodes were then randomly connected to each other until a certain degree of inter-cluster and intra-cluster connectivity was reached. Inter-cluster connectivity was iteratively chosen between 0.01 and 0.15 and intra-cluster connectivity between 0.1 and 1.0. The average accuracy of each method was assessed after 50 randomization runs.

We utilized *S. cerevisiae* protein interaction data from BioGRID (version 3.1.77).[24] Similar to the work of Song *et al.*,[4] two different networks were assembled. We retrieved all protein interaction data available for *S. cerevisiae* from BioGRID and then filtered the link set so that the first network contained all high-throughput physical interactions whereas the second network was further limited to data from experiments utilizing the yeast two-hybrid technique. The first network contained in total 4152 proteins and 15 177 links and the second smaller network contained 2413 proteins and 4157 links. In the following we will refer to those two networks as the small and the large *S. cerevisiae* network. As a reference for functional groups in *S. cerevisiae* we used functional complex annotations from the CYC2008 database.[25] The data set contained in total 401 complexes comprising 1914 genes. For benchmarking the different clustering methods, we filtered for genes present in both the CYC2008 database and the *S. cerevisiae* networks. In the case of the smaller network 991 genes were also present in the CYC2008 database and 1544 in the case of the bigger network.

A more comprehensive network was derived from the FunCoup database.[15,16] The FunCoup database provides global interaction networks for a variety of species and combines different types of evidence: protein–protein interactions, mRNA co-expression, sub-cellular co-localization, phylogenetic profile similarity, co-targeting by either miRNA or transcription factors, protein co-expression, and domain–domain interactions. It further transfers evidence from other eukaryotic species *via* orthologs. In this study we used the human FunCoup v2.0 network. The link set was limited to the subset that was trained on the PPI gold standard and had a confidence score of 0.9 or higher. The whole network included 9831 unique genes with 671 267 links between them. All references to the FunCoup human network refer to this subnetwork. As a reference to benchmark different clustering methods on the FunCoup network we collected human pathway annotations from the KEGG database[26] and considered each pathway as a cluster. Our pathway data set contained 198 pathways spanning 5043 genes. The intersection of the human FunCoup network and the pathway data set contained 3298 genes.

Cluster evaluation

The accuracy of the different clustering methods was assessed based on the overlap between clusters and reference groups. As a measure of the overlap between two sets of proteins we used the Jaccard similarity coefficient. It is defined as the size of the intersection divided by the size of the union. We determined the average Jaccard similarity coefficient for each method on each benchmark set. Comparing different clustering methods is complicated by the fact that some methods attempt to cluster the whole network while other methods leave big parts of the network unclustered. To correct for this fact we included unclustered nodes as singletons, as this makes results more comparable between methods. Furthermore, only genes present in both the network and the reference groups were included in the score calculation. This means that clusters are filtered for annotated genes and that the size of clusters might potentially be changed.

Results

We developed a new algorithm *MGclus* to detect modules, or clusters, in a network. It specifically searches for clusters with a strongly interconnected neighborhood, which is often the case in large scale biological interaction networks. *MGclus* was compared to a selection of widely used clustering methods including *MCL*,[20] *CFinder*,[18] *FastCommunity*,[19] *MCode*,[7] *MINE*,[21] *NEMO*,[5] *SPICi*,[22] and *Cohtop*.[23] Three different benchmarks were conducted. We started by comparing the clustering methods using simulated random clusters. The second benchmark was based on two *S. cerevisiae* PPI networks derived from BioGRID. In the last test we compared the clustering methods on a large scale human network from the FunCoup database.

The comparison of different clustering methods is complicated by the fact that some methods only report clusters for small parts of the network, yielding high accuracy but low coverage. To correct for this we included unclustered genes as singletons, as this makes results more comparable between methods. Some methods such as *MINE* and *MCode* reach a higher score when not counting singletons, but then suffer from very low coverage (see Table S1, ESI†).

To quantitatively evaluate the performance of all methods considered, they were tested on a set of random clusters with varying degrees of inter-cluster connectivity and fixed intra-cluster connectivity. In each run the number of random clusters was set to 15 or 30, and each cluster contained ten nodes. Nodes were randomly connected to each other until a certain degree of inter-cluster connectivity was reached. After 50 iterations the average Jaccard similarity coefficient for each method was determined. Furthermore, we repeated the analysis with 5, 10, 15, and 30 random clusters with a wider range of inter-cluster and intra-cluster connectivities (see ESI†).

Fig. 1 shows the results for 15 and 30 random clusters with fixed intra-cluster connectivity (0.8) and varying inter-cluster connectivity. We found *MGclus* to perform clearly better than other methods in both tests. For 15 random clusters, all methods except for *MGclus*, *MCL*, and *SPICi* had an average Jaccard score below 0.5 at all inter-cluster connectivities above 0.04. This became even more apparent for 30 random clusters, where only *MGclus* and *SPICi* reached a Jaccard score above 0.5

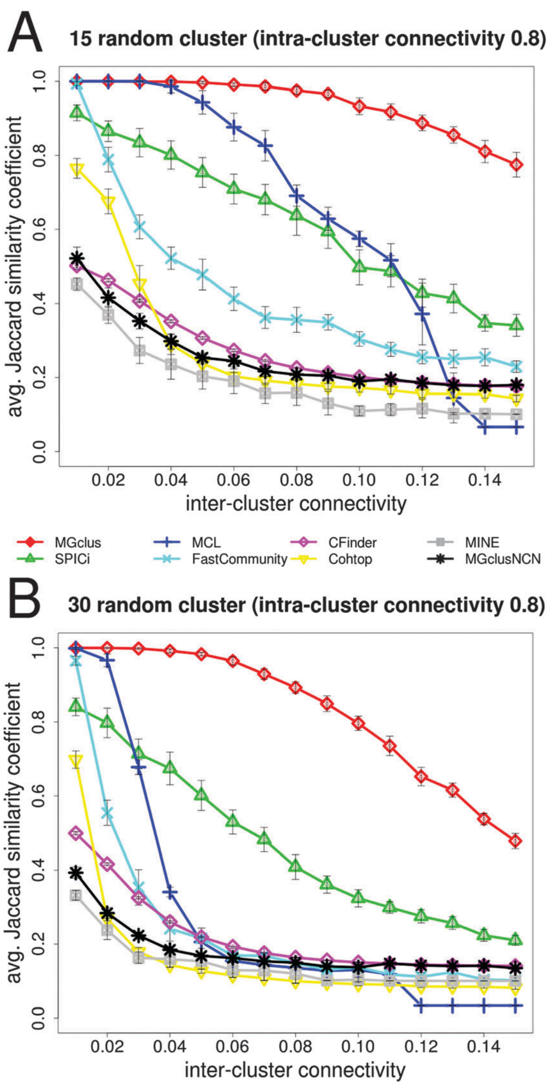

Fig. 1 Accuracy benchmark using simulated data. Various clustering methods were benchmarked on random graphs with (a) 15 random clusters or (b) 30 random clusters of size 10. The intra-cluster connectivity was fixed (0.8) and the inter-cluster connectivity was varied between 0.01 and 0.15. The shown Jaccard similarity coefficient between predicted and true clusters is the average of 50 runs. Vertical error bars on data points represent the standard deviation for each method. We also included a version of *MGclus* that does not use shared neighbors to calculate the clustering score (*MGclusNCN*). Without shared neighbors *MGclus* performed very poorly, thus demonstrating the usefulnesses of incorporating shared neighbor counts in the cluster finding process. Only methods that can be run standalone were included in this benchmark.

at inter-cluster connectivities above 0.04. In both benchmarks, *MGclus* substantially outperformed the other methods at all levels of inter-cluster connectivity except for the lowest. This was generally observed also for a wider range of parameters (see ESI†).

In the second benchmark we utilized *S. cerevisiae* protein interaction data from BioGRID.[24] Two different networks were derived: a smaller network that was limited to data from yeast two-hybrid experiments, and a larger network that contained all high-throughput physical interaction data. Accuracy of the clustering methods was assessed using reference groups from the CYC2008 database.[25]

Fig. 2A and B shows the results of this benchmark. The overall outcome was the same on both networks. *MGclus* scored the best, closely followed by *MCL* and *FastCommunity.* It is notable that many methods left huge parts of the network unclustered, limiting their practical usefulness for many users. For example, *MCode* clustered only 9 percent of the small network, *CFinder* only 19 percent, and *NEMO* only 16 percent (Table S1a, ESI†). This low coverage does not impact the benchmark results much however, because most genes are unannotated. All methods except for *MINE*, that took 35 minutes to cluster the bigger network, ran in a reasonable time (Table 1).

In the last benchmark we used a human large scale protein interaction network derived from high-confidence links in the FunCoup database, using KEGG pathways as reference clusters. Results are shown in Fig. 2C. Overall the recovery of reference groups was much lower than that for the *S. cerevisiae* networks. This is not unexpected as the network contains about 44 times more links and about 2.4 times more nodes than the large *S. cerevisiae* network. *MGclus* performed best followed by *Cohtop*. The two methods *CFinder* and *NEMO* could not be included into the comparison, as *CFinder* failed to cluster the network in a reasonable time and *NEMO* crashed after a while. It remained unclear why *NEMO* failed on the human network. However, since it performed successfully on the two smaller *S. cerevisiae* networks, it is likely related to the bigger network size.

An important feature of *MGclus* is that it accounts for shared network neighbors. To assess the value of this, we compared *MGclus* with and without the use of shared neighbors (*MGclusNCN*). We found that *MGclus* benefits particularly in early steps of the clustering from accounting for shared network neighbors as it helps to accurately group nodes with high mutual relevance. Among the random clusters *MGclusNCN* performed very poorly and noticeably worse on the other three networks, demonstrating the usefulness of incorporating the network context into the clustering procedure.

Discussion

As large-scale gene interaction networks keep growing and become available for a large number of organisms, there is an increasing need for tools to accurately extract informative patterns from these networks. However, although many methods have been proposed there is no obvious optimal solution, and most methods provide rather different results. *MGclus* is designed to promote the shared local network neighborhood during the clustering. We found that accounting for shared network neighbors is especially useful at early steps of the clustering, as it helps to reliably group nodes with high mutual relevance.

As a benchmark we quantitatively assessed the ability of different clustering methods to recover the random cluster with a controlled degree of inter-cluster connectivity and intra-cluster connectivity. While differences in performance were less apparent for a smaller number of random clusters, *MGclus* performed notably better than other popular methods when increasing the number of clusters.

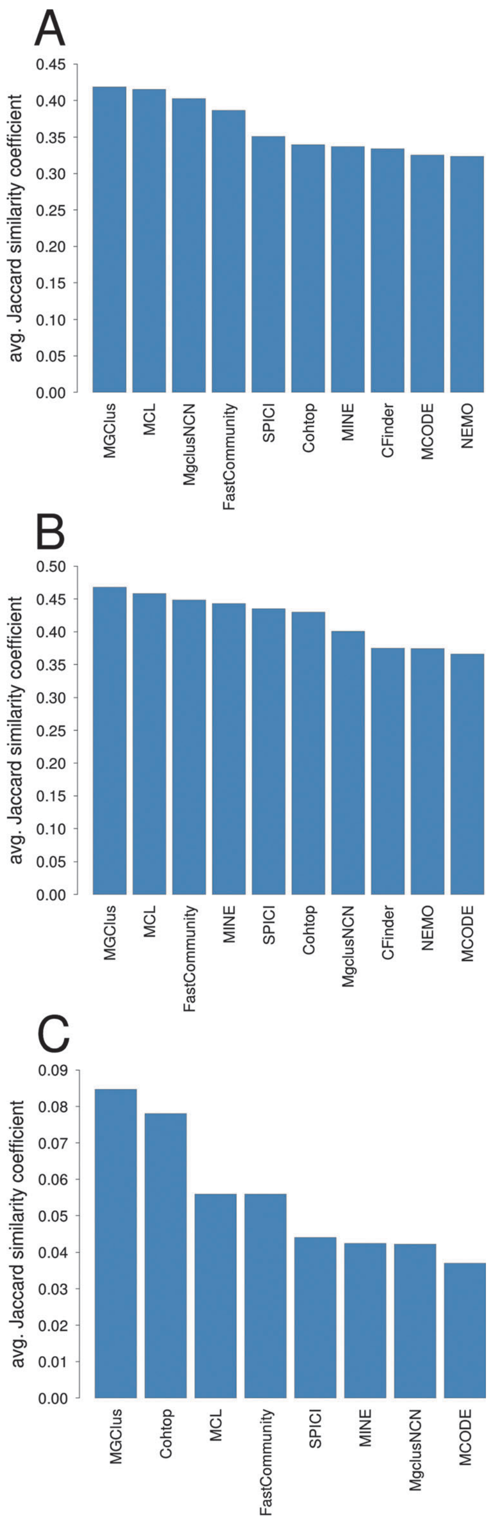

Fig. 2 Accuracy benchmark using biological networks. Shown is the average Jaccard similarity coefficient between predicted and reference clusters for (a) the small *S. cerevisiae* network that was limited to data from yeast two-hybrid experiments, (b) the large *S. cerevisiae* network that contained all interactions, and (c) the human FunCoup network. Only genes present in both the network and the reference sets were included in the Jaccard score calculation. For the *S. cerevisiae* networks, complexes from the CYC2008 database were used as reference clusters, while KEGG pathways were used for the FunCoup network. Two of the methods are not included in (c) because, in the case of CFinder, it took more than 24 hours to complete and, in the case of *NEMO*, it crashed.

Table 1 Runtime benchmark. Shown are ''user runtimes'' on (a) the small *S. cerevisiae* network that was limited to data from yeast two-hybrid experiments, (b) the large *S. cerevisiae* network that contained all interactions, and (c) the human FunCoup network. All benchmarks were conducted on an Intel Core i7 with 16 GB of memory. Runtimes longer than 24 hours were not measured. *NEMO* is only available as a Cytoscape plugin, and crashed on the human network

	Time (s)		
	S. cerevisiae (small)	*S. cerevisiae* (large)	Human
SPICi	0.0	0.0	1.3
FastCommunity	0.3	1.0	66.4
MCL	0.4	1.7	89.4
MGclusNCN	1.4	2.0	110.0
Cohtop	2.1	7.0	701.2
MGClus	1.8	3.2	861.1
CFinder	0.2	7.4	>24 h
MINE	118.9	2114.9	36666.1
MCode	2.0	11.0	27788.4
NEMO	12.0	34.0	Plugin crashed

We then applied all methods to a number of real biological networks, where *MGclus* also performed at the top. A problem that we encountered is that many methods only predict clusters for a small part of the networks, which can hinder their practical usefulness. It also makes it hard to compare the results between methods. To make the comparison as fair as possible, we counted unclustered nodes as singletons.

A common issue for non-expert users is that some methods have a number of user-adjustable parameters that can greatly influence their performance. However, without knowing the optimal clustering beforehand, it is impossible to know how parameters should be set. It is therefore important that methods have reasonable and robust standard parameters. We generally found *MGclus* to perform well with the default merge-gain cut-off (C_{MG}) and only recommend to change C_{MG} if one wants to enforce a smaller or a larger cluster. As a rule of thumb, a higher C_{MG} will result in a larger number of clusters of smaller size and higher density.

In conclusion, *MGclus* has proven to be suitable for detecting modules at both protein complex- and pathway-scales in small as well as large-scale biological networks. It performed notably better than other methods when applied to random clusters, and better or equally well when applied to high throughput PPI networks.

References

1 S. Brohée and J. van Helden, *BMC Bioinf.*, 2006, **7**, 488.
2 J. Wang, M. Li, Y. Deng and Y. Pan, *BMC Genomics*, 2010, **11**, S10.
3 M. Girvan and M. E. J. Newman, *Proc. Natl. Acad. Sci. U. S. A.*, 2002, **99**, 7821–7826.
4 J. Song and M. Singh, *Bioinformatics*, 2009, **25**, 3143–3150.
5 C. G. Rivera, R. Vakil and J. S. Bader, *BMC Bioinf.*, 2010, **11**, S61.
6 L. H. Hartwell, J. J. Hopfield, S. Leibler and A. W. Murray, *Nature*, 1999, **402**, C47–C52.

7 G. D. Bader and C. W. V. Hogue, *BMC Bioinf.*, 2003, **4**, 2.

8 J. B. Pereira-Leal, A. J. Enright and C. A. Ouzounis, *Proteins*, 2004, **54**, 49–57.

9 A. W. Rives and T. Galitski, *Proc. Natl. Acad. Sci. U. S. A.*, 2003, **100**, 1128–1133.

10 T. Ideker, O. Ozier, B. Schwikowski and A. F. Siegel, *Bioinformatics*, 2002, **18**, S233–S240.

11 R. R. Nayak, M. Kearns, R. S. Spielman and V. G. Cheung, *Genome Res.*, 2009, **19**, 1953–1962.

12 C.-C. Lin, J.-T. Hsiang, C.-Y. Wu, Y.-J. Oyang, H.-F. Juan and H.-C. Huang, *BMC Syst. Biol.*, 2010, **4**, 138.

13 E. Cerami, E. Demir, N. Schultz, B. S. Taylor and C. Sander, *PLoS One*, 2010, **5**, e8918.

14 V. Spirin and L. A. Mirny, *Proc. Natl. Acad. Sci. U. S. A.*, 2003, **100**, 12123–12128.

15 A. Alexeyenko and E. L. L. Sonnhammer, *Genome Res.*, 2009, **19**, 1107–1116.

16 A. Alexeyenko, T. Schmitt, A. Tjärnberg, D. Guala, O. Frings and E. L. L. Sonnhammer, *Nucleic Acids Res.*, 2012, **40**, D821–D828.

17 C. Huttenhower, E. M. Haley, M. A. Hibbs, V. Dumeaux, D. R. Barrett, H. A. Coller and O. G. Troyanskaya, *Genome Res.*, 2009, **19**, 1093–1106.

18 B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi and T. Vicsek, *Bioinformatics*, 2006, **22**, 1021–1023.

19 A. Clauset, M. E. J. Newman and C. Moore, *Phys. Rev. E: Stat. Nonlinear Soft Matter Phys.*, 2004, **70**, 066111.

20 A. J. Enright, S. Van Dongen and C. A. Ouzounis, *Nucleic Acids Res.*, 2002, **30**, 1575–1584.

21 K. Rhrissorrakrai and K. C. Gunsalus, *BMC Bioinf.*, 2011, **12**, 192.

22 P. Jiang and M. Singh, *Bioinformatics*, 2010, **26**, 1105–1111.

23 A. Alexeyenko, D. M. Wassenberg, E. K. Lobenhofer, J. Yen, E. Linney, E. L. L. Sonnhammer and J. N. Meyer, *PLoS One*, 2010, **5**, e10465.

24 C. Stark, B.-J. Breitkreutz, A. Chatr-Aryamontri, L. Boucher, R. Oughtred, M. S. Livstone, J. Nixon, K. Van Auken, X. Wang, X. Shi, T. Reguly, J. M. Rust, A. Winter, K. Dolinski and M. Tyers, *Nucleic Acids Res.*, 2011, **39**, D698–D704.

25 S. Pu, J. Wong, B. Turner, E. Cho and S. J. Wodak, *Nucleic Acids Res.*, 2009, **37**, 825–831.

26 M. Kanehisa and S. Goto, *Nucleic Acids Res.*, 2000, **28**, 27–30.