Electronic Supplementary Material S1

Community landscapes: an integrative approach to determine overlapping network module hierarchy, identify key nodes and predict network dynamics

István A. Kovács, Robin Palotai, Máté S. Szalay, Péter Csermely*

Department of Medical Chemistry, Semmelweis University, Tűzoltó str. 37-47, H-1094 Budapest, Hungary

*E-mail: csermely@eok.sote.hu

Summary

In this Supporting Information (SI) we give a detailed description of the ModuLand network module determination method family. This integrative method is based on the construction of community landscapes from community heaps. (1) In the first part we describe three versions of the community heap construction methods, the NodeLand, LinkLand and PerturLand methods in detail. (2) In the second part the combination of community heaps to a community landscape is shown. We demonstrate the wide applicability of the ModuLand method to accommodate previous community detection methods in the examples of the BetweennessCentralityLand (BCLand) and CliqueLand community landscape determination methods resulting in distinct and overlapping network modules, respectively. (3) In the third part of the description we show the identification of module centers as hills of the community landscape using the PeakHill hill determination method. (4) Finally, the module membership of network elements and links is calculated using one of the developed module membership assignment methods, such as the GradientHill, ProportionalHill or TotalHill methods yielding modules of minimal, fair or detailed overlaps, respectively. (5) We also show that the ModuLand method-family enables a hierarchical analysis of network topology and the construction of a zoom-in network visualization method. Besides the detailed description of the ModuLand method the SI also contains 14 Supplementary Figures and their Supplementary Discussion, as well as a detailed summary of 18 module definitions, 129 different modularization methods, 13 module comparison methods as three Supplementary Tables and 370 references.

The Linux-based computer programs of the ModuLand-related methods or a Windows-based application can be downloaded from here: www.linkgroup.hu/modules.php.[1]

[1]A Linux/x86 compatible operating system with kernel version 2.6 is required for running the programs provided in the ModuLand program package. If you do not have access to such sytem, you may use a prebuilt VirtualBox image of a Linux system with all necessary programs as described in the package.

Table of content

Supplementary Figures **4**
Figure S1. Time-scale of the development of modularization methods. 4
Figure S2. Flowchart of the available ModuLand method algorithms. 5
Figure S3. Different approaches of community landscape construction. 6
Figure S4. Hill determination on community landscapes. 7
Figure S5. PeakHill methods for module membership assignment. 8
Figure S6. Hierarchical levels of the network science co-authorship network. 9
Figure S7. Modularization of the Zachary network. 10
Figure S8. Distribution of module sizes, module degrees, module overlap sizes and node membership numbers of the USF Word Association Network 11
Figure S9. Modules of two homonym words in the USF Word Association Network. 12
Figure S10. Comparison of the effect of four community heap construction methods on the modular structure of a school friendship social network. 13
Figure S11. Modular hierarchy of a school friendship network obtained by the LinkLand method. 14
Figure S12. Modular hierarchy of a school friendship network obtained by the PerturLand method. 15
Figure S13. Robustness of the ModuLand method in recovering natural modules of benchmark graphs. 16
Figure S14. Functional modules of the yeast protein-protein interaction network. 17

Supplementary Methods **18**
I. Networks used for the analysis 18
1. Network science collaboration network 18
2. Zachary karate club social network 18
3. Word association network 18
4. School-friendship network 18
5. Electrical power-grid of the USA 19
6. Yeast protein-protein interaction network 19
II. Overview of the ModuLand network module determination method family 20
III. Starting considerations and definitions 22
1. Terminology 22
2. Input data of the ModuLand method family 22
IV. Determination of the community landscape 24
1. The NodeLand and LinkLand community heap construction methods for weighted and undirected networks 24
a. The NodeLand community heap construction method 24
b. The LinkLand community heap construction method 26
2. The PerturLand community heap construction method for weighted and directed networks 27
3. Two extreme cases of community landscapes for weighted and directed networks 31
4. Transformation of widely used, former modularization methods to the community landscape framework of the ModuLand method 31
a, The CliqueLand community lanscape determination method for unweighted and undirected networks 31
b, The BetweennessCentralityLand (BCLand) community lanscape determination method for weighted and directed networks 32
c, Methods yielding partitions of the network 32
d, Stochastic module detection methods 32

5. Summary of the community landscape determination methods 32
V. Determining modules based on the community landscape 34
1. ThresholdHill: Horizontal cut of the community landscape as a detection threshold of previous community detection methods 34
2. PeakHill methods: hills are determined by local maxima 34
a. Finding the hill-tops and highlands of the community landscape 35
b. The ProportionalHill and the GradientHill module membership assignment methods (undirected and directed versions) 36
c. The TotalHill module membership assignment method for undirected networks 38
d. The TotalHill module membership assignment method for directed networks 40
e. The optimized version of the undirected TotalHill module membership assignment method 41
3. Other definitions of hills on a community landscape (the SameHill and NumberHill methods) 42
4. Finding the elements of the overlapping communities 43
5. Summary of the hill finder methods 43
6. Metrics based on the ModuLand modularization methods 43
a. Effective size of support 43
b. Effective number 44
c. Modular overlap 44
d. Bridgeness 44
e. Similarity of the elements 45
VI. Further opportunities and analysis of the ModuLand method 46
1. Merging the overlapping communities 46
2. Robustness of the ModuLand method for the measurement, sampling and computational errors 46
VII. Construction of the higher hierarchical level representations of the network 48
1. The strength of the link between two modules 48
2. The modularization of a higher hierarchical level 49
3. Projection of the modules of higher hierarchical levels 49
VIII. ModuLand-based network visualization methods 51

Supplementary Tables 52
Table S1. Definitions of network modules 52
Table S2. Comparison of network module determination methods 54
Table S3. Comparison of modularization methods 70

Supplementary Discussion 72

Supplementary References 78

Supplementary Figures

Figure S1. Time-scale of the development of modularization methods.
The figure shows the number of modularization methods listed in Table S2 as a function of publication years between 1956 and 2009 Books and reviews were omitted to help the direct assessment of the methodological enrichment of the field.

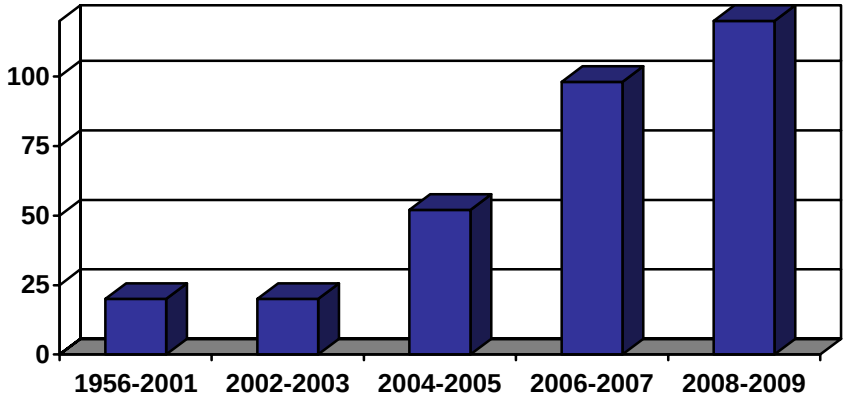


Figure S2. Flowchart of the available ModuLand method algorithms.
The figure shows the different phases and possible options of the ModuLand algorithm package. Cylinders represent data storage options, while boxes represent different operations. Box captions refer to the name of the operation, while the name in parentheses refers to the executable program name as found in the ModuLand program package downloadable from http://www.linkgroup.hu/modules.php. For a detailed explanation please see the User Guide included in the program package, Figure 1 and the main text as well as Sections IV., V. and VII.

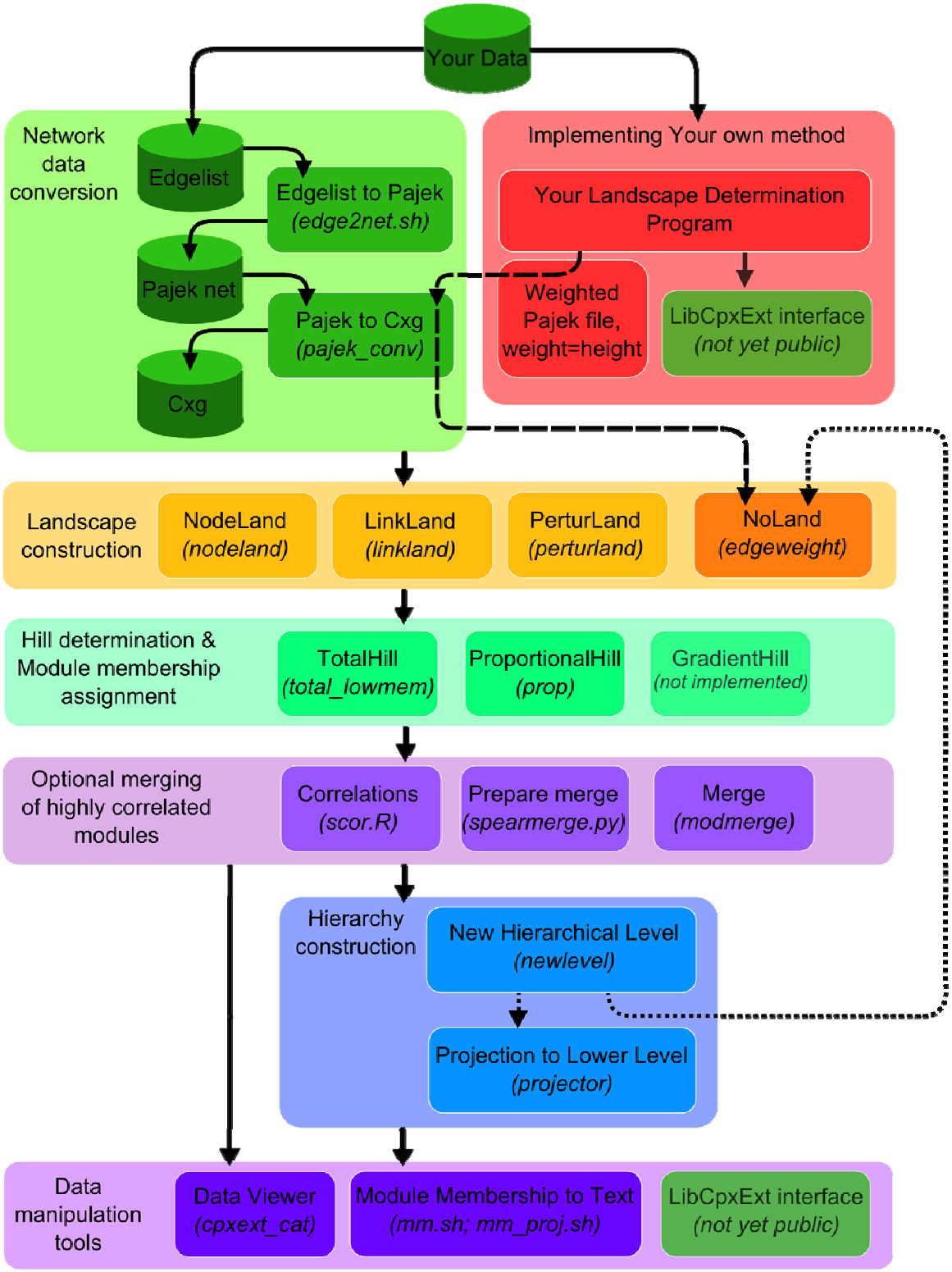


Figure S3. Different approaches of community landscape construction.

Panels **(a)** and **(b)**. Community heap-based community landscape construction. Panel **(a)** illustrates that a community heap (colored hill-like areas) may belong to each element or link of the network. Community heaps can be determined by any of the community heap construction methods described (see Figure 1. of the main text and Suppl. Methods, Section IV). Panel **(b)** shows that a community landscape may be constructed by summing up the individual community heaps belonging to the elements or links of the network (see Suppl. Methods, Section IV.1-3.). Panels **(c)** and **(d)**. Direct construction of a community landscape. It may be possible to construct the community landscape directly, omitting the community heap construction step. As an example, data acquired by former community detection algorithms can be directly transformed into a community landscape. On Panel **(c)** the betweenness centrality values are shown of a network, as heights of the links..The original algorithm by Girvan and Newman (2002) iteratively removes the links of highest betweenness, and defines modules as connected components (colored parts under the curve in the picture) of the remaining network. In this method the number of modules depends on the length of the iteration process. The horizontal line of Panel **(c)** represents the final round of the iterative process. As this algorithm removes links of high betweenness centrality from the network first, which are usually inter-modular links, the links removed last tend to be located at modular centers. Let the community landscape height c_{ij} of the link(*i,j)* be k *(k>0)*, if the given link was removed from the network in the *k*-th iteration. This way we get the 'BCLand community landscape' shown on Panel **(d)** (see Suppl. Methods, Section IV.4.b.), on which the original modularization can be gained by applying the ThresholdHill method. A similar conversion of an other partitioning technique to a community landscape yields overlapping modules as a rule. The horizontal lines below the panels represent links (or elements) of the network with the appropriate colors of their modules.

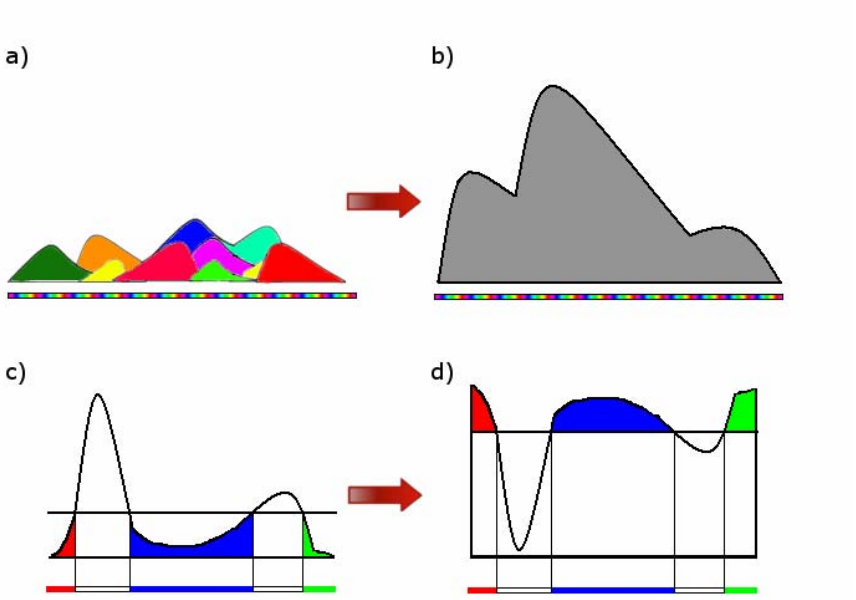


Figure S4. Hill determination on community landscapes.
Once a community landscape had been constructed, modules can be identified in different ways. On this illustrative figure the horizontal line below the Panels represents elements or links of the network with the color of their modules. The hill-like curve on the Panels represents the community landscape. The ThresholdHill hill determination method (see Suppl. Methods, Section V.1.) shown on Panel **(a)** identifies all connected components of the network above a given community landscape threshold as modules. The SameHill hill determination method (see Suppl. Methods, Section V.3.) shown on Panel **(b)** identifies connected components of height not differing more than a given percentage from the height of hill-top elements or edges as modules. We note that the ThresholdHill and SameHill methods in principle do not assign all elements or edges into modules. The PeakHill hill determination method (see Suppl. Methods, Section V.2.) shown on Panel **(c)** identifies the hill-tops of the community landscape as module-cores, and uses one of the PeakHill module membership assignment methods (e.g. the ProportionalHill or TotalHill module membership assignment methods, see Suppl. Methods, Section V.2.b.-e. and Figure S5) to assign each element and link of the network into overlapping modules.

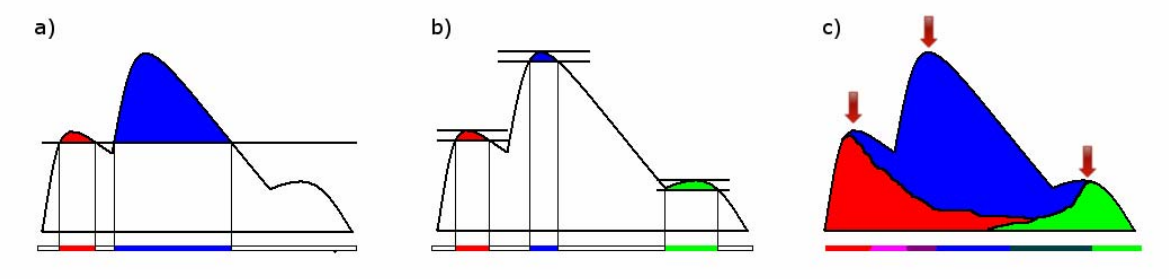


Figure S5. PeakHill methods for module membership assignment.
The figure illustrates three module-membership assignment methods resulting in increasing overlaps between the network modules. The horizontal colored bars illustrate links of the network, and the vertical lines connecting the links represent an element shared between the links. Link colors refer to the module membership assignment of the links. Vertical position of a given link indicates the ModuLand community landscape height of that link (see Figure 1 of the main text). Links A and E are selected as module-cores by the PeakHill hill detection algorithm, because they are local maxima (see Suppl. Methods, Section V.2.a.). Module-core links are always assigned to their own modules only. Panel **(a)**. The GradientHill module membership assignment method. In this module membership assignment method a non-core link is assigned only to the module of its highest neighboring link (or the module assignment is divided between modules of its highest neighbors, if more than one such neighbors exist). Based on this rule link C is assigned to the red module. As illustrated, this module membership assignment method results in a minimal overlap between the modules. Panel **(b)**. The ProportionalHill module membership assignment method. In this module membership assignment method a non-core link is assigned to the modules of its higher neighbors proportional to the height of the respective neighbors. Based on this rule link C is assigned more to the red module and only minimally to the blue module. This module membership assignment method results in a moderate overlap between the modules. Panel **(c)**. The TotalHill module membership assignment method. In this module membership assignment method a non-core link is assigned to the modules of its all neighbors proportional to the height of the respective neighbors. This type of assignment can be efficiently solved with a set of linear equations. Based on this rule not only link C is assigned both to the red and blue modules, but links B and D also become inter-modular links between the red module defined by the module-core, link A and the blue module defined by the module-core, link E. Additionally, the adjacent links of links A and E at the two edges of the figure – being also non-core links – will be assigned to more than one modules illustrated by their mixed color. Thus, the TotalHill module membership assignment method results in an extensively high overlap between the modules.

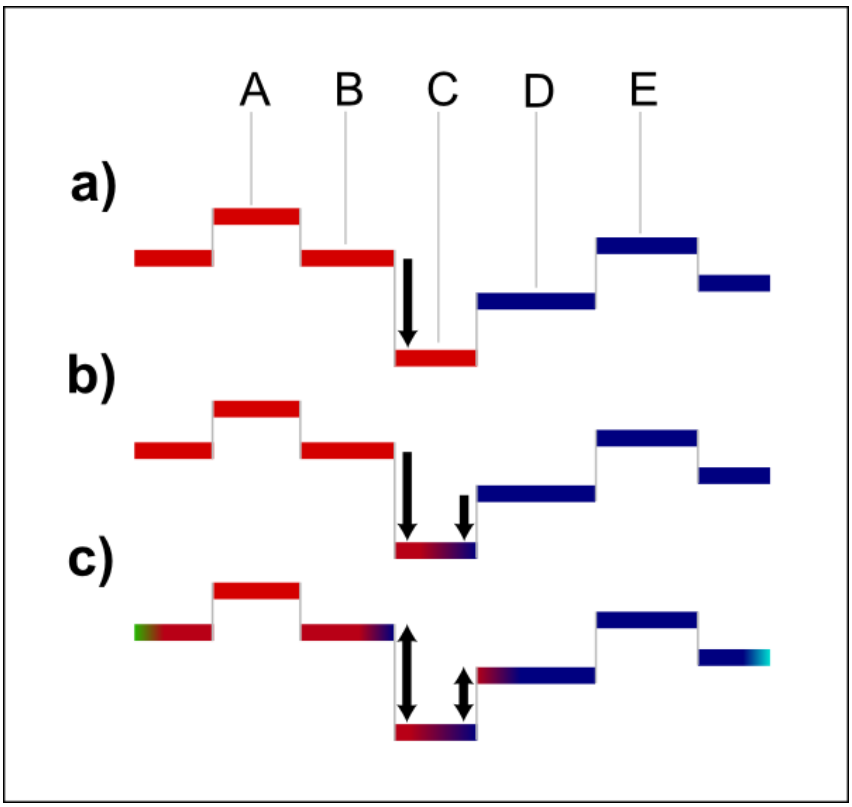


Figure S6. Hierarchical levels of the network science co-authorship network.
The first row of the figure shows the hierarchical levels of the network science collaboration network (Newman, 2006a) as uncovered by the LinkLand community heap construction method and the ProportionalHill module membership assignment method. Community heaps were determined only at the original network level. The network was visualized with the Kamada-Kawai algorithm. Modules of the original (zero-level) network shown on the left side of the figure became the elements of the first hierarchical level of the network labeled as 'level 1'. Modules of 'level 1' became the elements of 'level 2' until the network coalesced to a single element, which would constitute 'level 4' of the figure (not shown). It is also possible to project the module memberships of a higher level network back to the elements of any intermediary level network (see Suppl. Methods, Section VII.3.). The result of this projection procedure is shown in the second row of the figure, where the module memberships of the respective level are projected back to the elements of the original network and are represented by different colors.

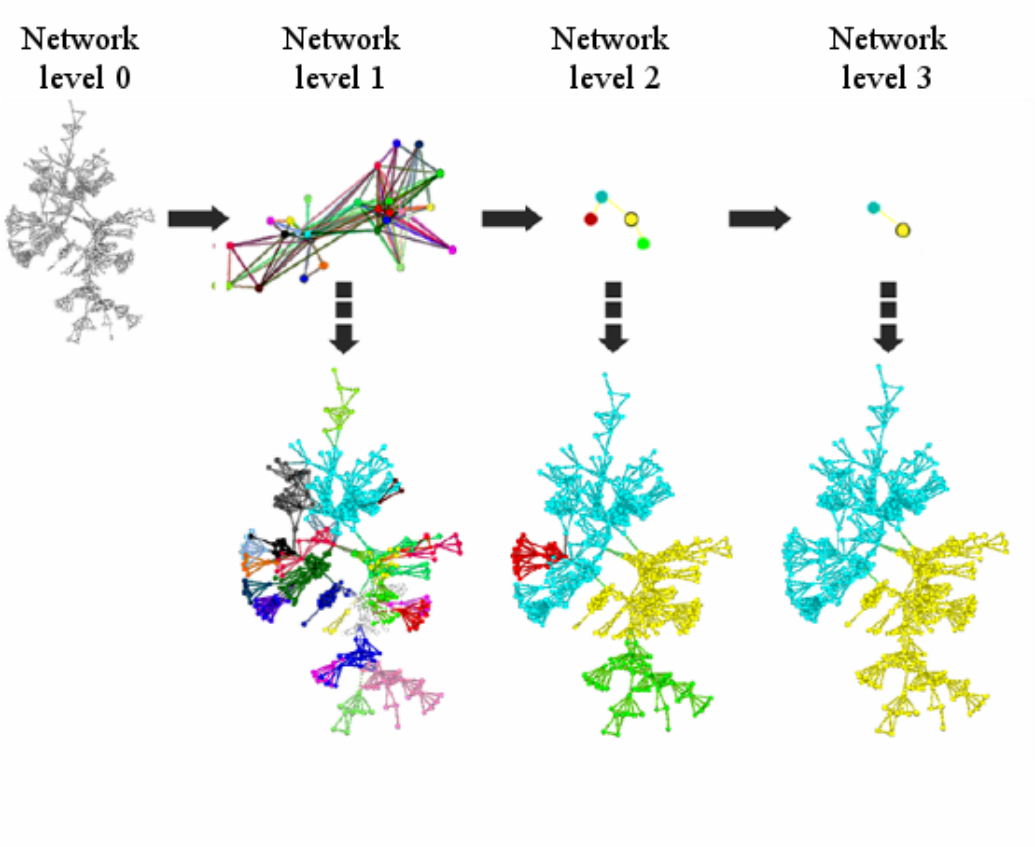


Figure S7. Modularization of the Zachary network.

This figure shows the modularization results of one of the gold-standard social networks for modular analysis, the Zachary karate club (Zachary, 1977). Panels **(a)** and **(b)**. Overlapping modules of the Zachary karate club network. The three overlapping modules of the Zachary network, as identified by the NodeLand community heap construction method together with either the ProportionalHill or the TotalHill module membership assignment methods, are shown on Panel **(a)** or Panel **(b)**, respectively. The network was visualized with the Kamada-Kawai algorithm. Green squares, red circles and blue triangles refer to the three modules identified by the methods. Element colors are mingled to the extent of their overlapping module membership. Element shapes refer to the module, which is the maximal-strength module of the given element. Numbers refer to the number of the respective karate club member in the original study (Zachary, 1977). The method correctly identifies the observed split of the original network, while uncovering several club-members in modular overlaps and a third module also identified in previous studies (Zhang et al., 2007a; Leskovec et al., 2008; Shen et al., 2009). Numbers on the insets at the bottom of Panels **(a)** and **(b)** show the effective number of modules (see Suppl. Methods, Section V.6.b.) where the respective element belongs to. Applying the ProportionalHill method Panel **(a)** results in a lower effective number of modules per element than the application of the TotalHill module membership assignment method Panel **(b)** indicating a smaller modular overlap, which is in agreement with the illustrative model of Figure S5.

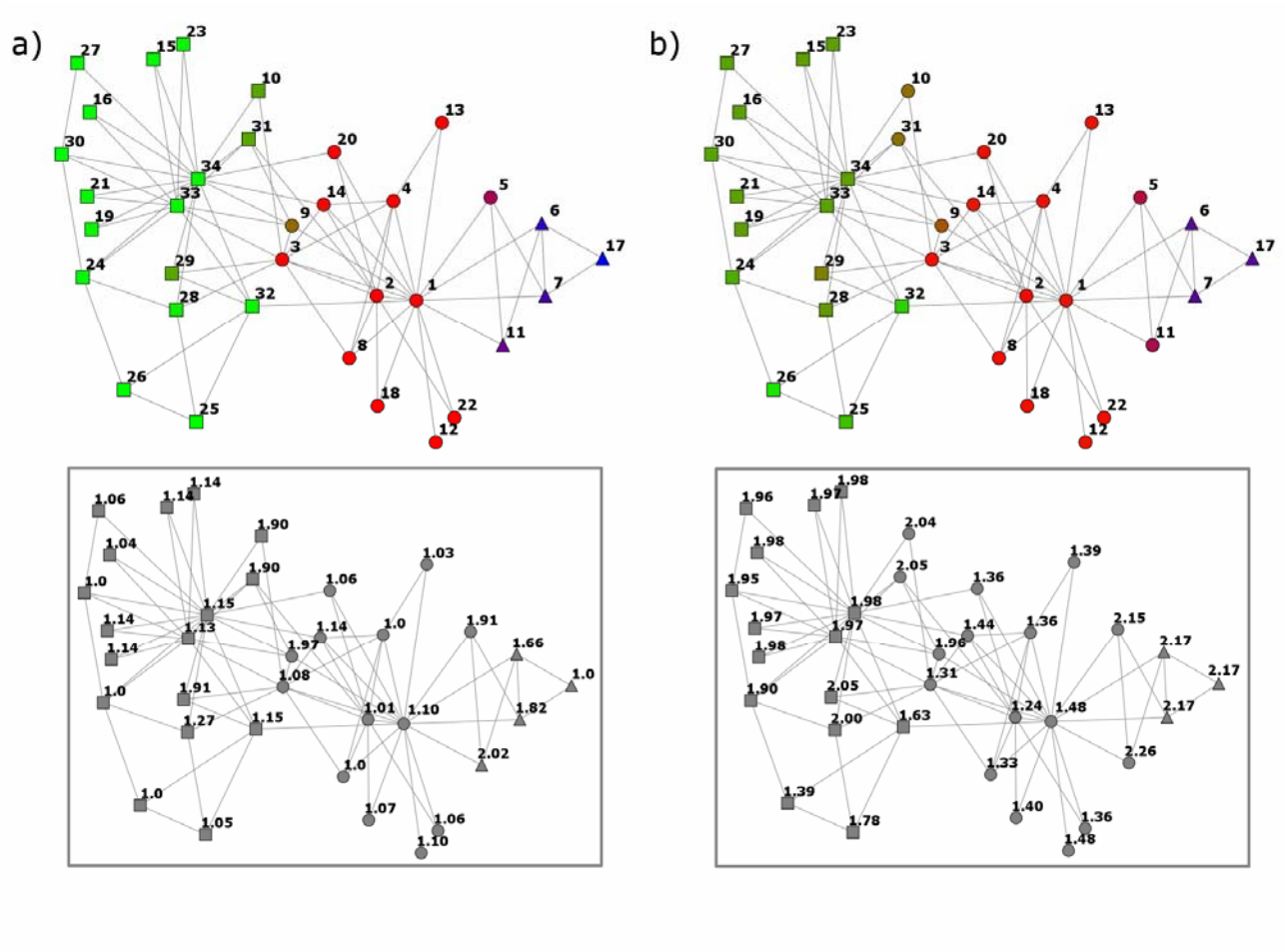
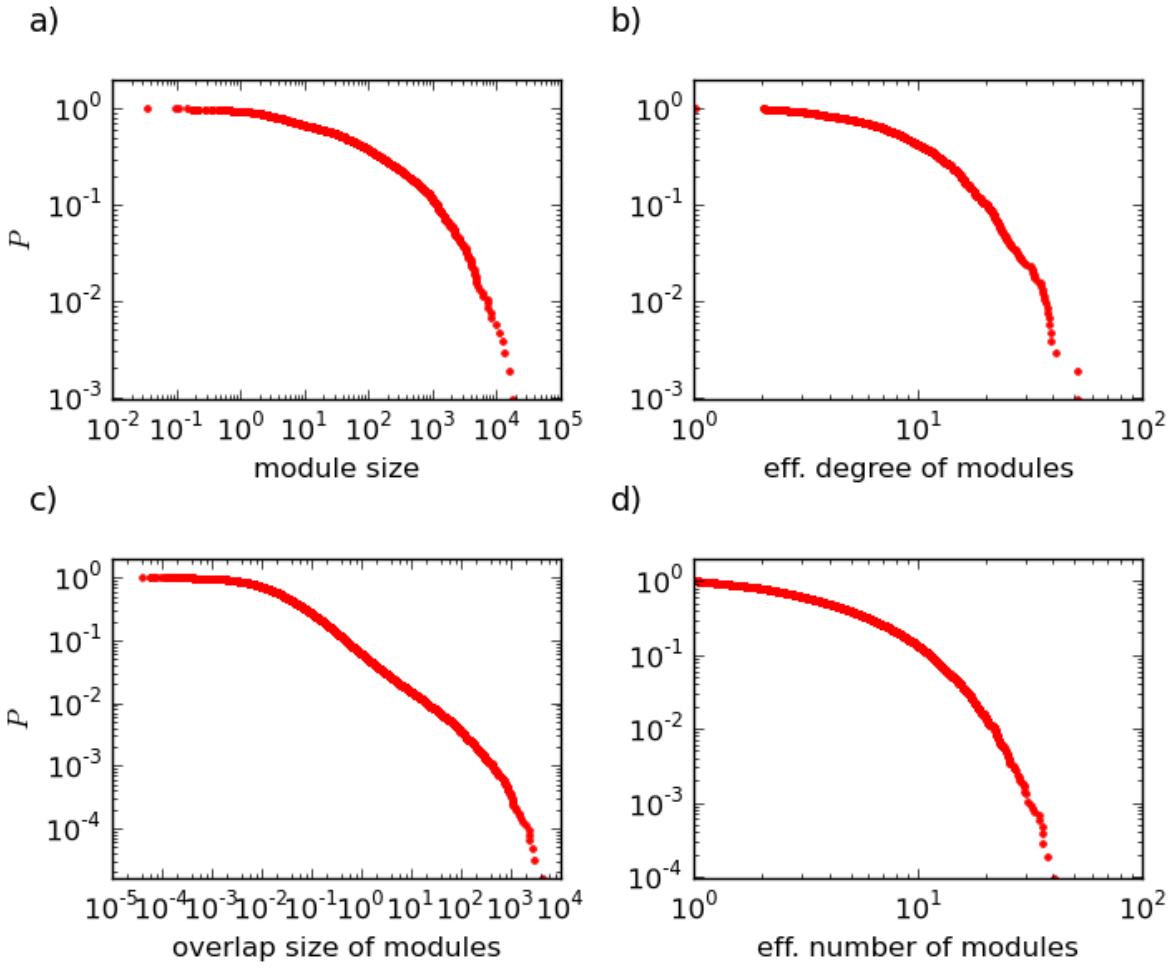


Figure S8. Distribution of module sizes, module degrees, module overlap sizes and node membership numbers of the USF Word Association Network.
Modules of the University of South Florida word association network (Nelson et al., 1998) were identified by applying the LinkLand community heap construction method and the TotalHill module membership assigment method. During the post-processing of the module assignment, we merged the modules with ProportionalHill module membership assignment-based correlation higher than 0.9. The modular structure was characterized by four metrics. Panel **(a)** shows the cumulative distribution of the size of modules, where the size of a given module is the summed membership assignment strength of each node to the given module. Panel **(b)** shows the cumulative distribution of the effective degree of modules, where weighted links are defined between modules based on their overlap as defined in Section VII.1., and the effective degree of a given module is the effective number (see Section V.6.b.) of such weighted links of the given module. Panel **(c)** shows the cumulative distribution of the overlaps between modules, where the overlap between two modules is the summed area-overlap between the respective modules (see Section V.6.d.) of each node of the network. Panel **(d)** shows the cumulative distribution of the effective number of modules per node (a similar measure is also known in the literature as 'node membership number'; González et al., 2007), where the effective number of modules of a node is given by the modular overlap measure of the given node (see Section V.6.c). All plots show the cumulative distributions on log-log scales. *P* on the vertical axes is defined as the fraction of modules (Panels **a** and **b**), the fraction of module pairs (Panel **c**) or the fraction of nodes (Panel **d**) for which the measured quantity equals or is greater than the value of the horizontal axes, for any given x.[2]



[2]For example, a point located at coordinates (x=10, y=0.42) on panel **(b)** means that 42% of the modules have an effective module degree of at least 10 (this corresponds to the situation that 42% of the modules are connected to at least 10 modules in the unweighted case).

Figure S9. Modules of two homonym words in the USF Word Association Network.
Modules of the University of South Florida word association network (Nelson et al., 1998) were determined using the LinkLand community heap construction method together with the TotalHill module membership assignment method. During the post-processing of the module assignment, we merged the modules with ProportionalHill module membership assignment-based correlation higher than 0.9. The network was laid out using Graphviz (Gansner and North, 1999) and visualized using a custom program written in Python. Links were colored in proportion to the colors of their modules. In addition to the selected words "bright" (Panel **a**) or "focus" (Panel **b**), similar words above a similarity threshold of 13% or 20%, in case of "bright" or "focus", respectively, are also shown with a contrast corresponding to their degree of similarity. The calculation of similarity between any two words is described in section V.6.e. of Suppl. Methods.

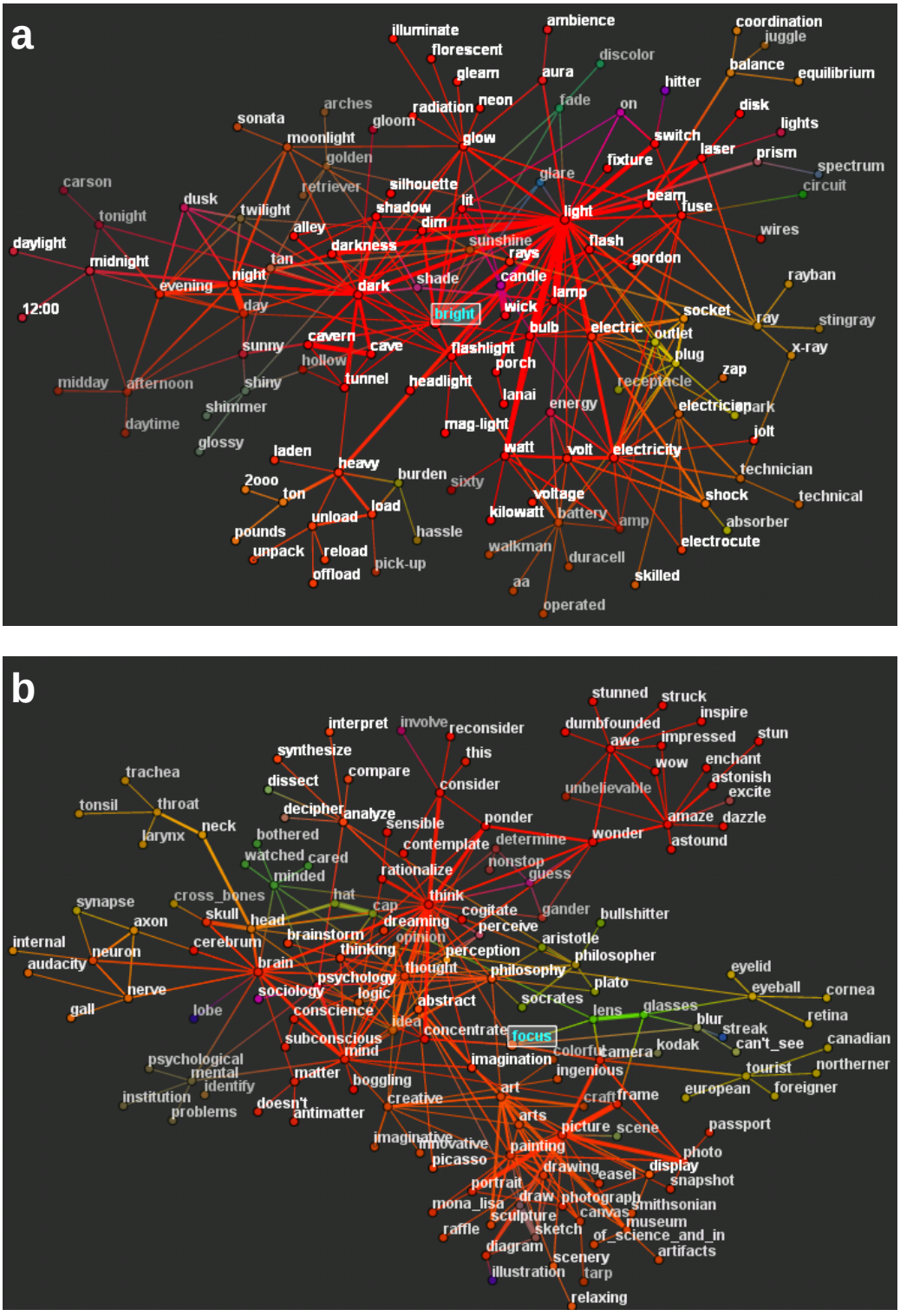
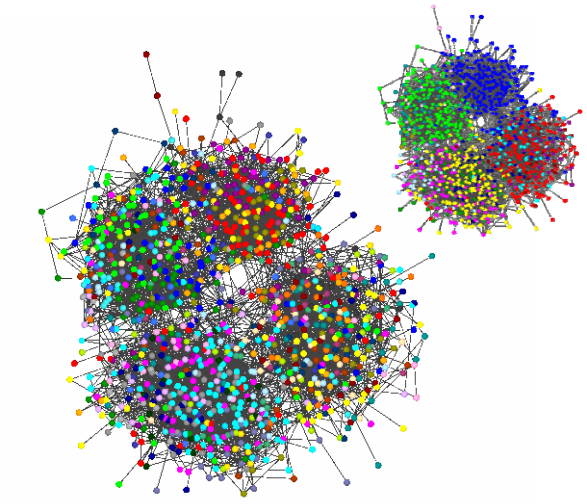


Figure S10. Comparison of the effect of four community heap construction methods on the modular structure of a school friendship social network.

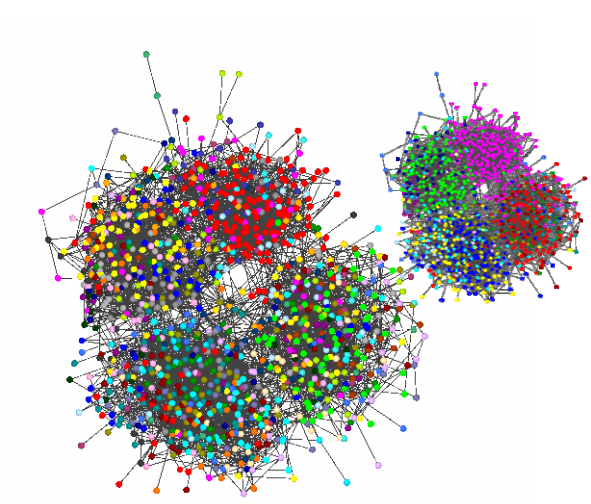
The modular structure of the school friendship social network of Community-44 of the Add Health dataset (see Suppl. Methods, Section I.4.; Moody, 2001; Newman, 2003) is shown at its first and second hierarchical levels (on the main images of the four panels and on their right-top insets, respectively), as uncovered by various community heap construction methods of the ModuLand method family using the ProportionalHill module membership assignment method. Community heaps were determined only at the original network level. During the post-processing of the module assignment, we merged the modules with ProportionalHill module membership assignment-based correlation higher than 0.9. The network was laid out with the Kamada-Kawai algorithm. Colors represent the color of the module, where the given element assigned most. The numbers in parentheses are the effective number of modules (see Suppl. Methods, Section V.6.b.). While the number of modules are rather consistent using the four community heap construction methods, the least accurate NoLand method (which simply takes link weights as community landscape heights) fails to distinguish the known four main sections (Newman, 2003) of the network

.



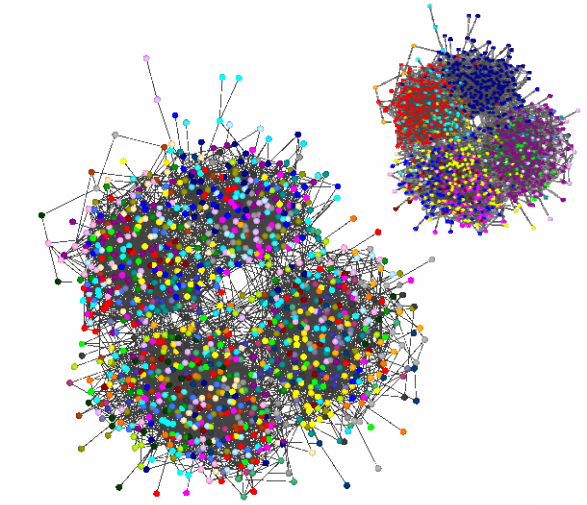
LinkLand

Level 0: 174 modules (27.55)
Level 1: 18 modules (7.18)



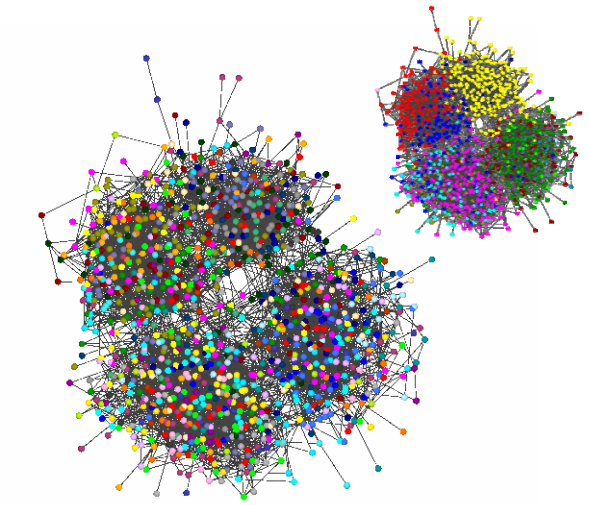
NodeLand

Level 0: 175 modules (31.48)
Level 1: 19 modules (8.22)



NoLand

Level 1: 229 modules (162.79)
Level 2: 24 modules (12.14)

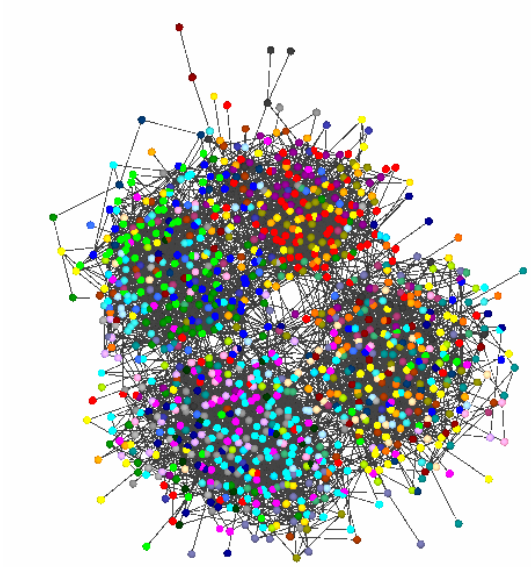


PerturLand (X=0.2)

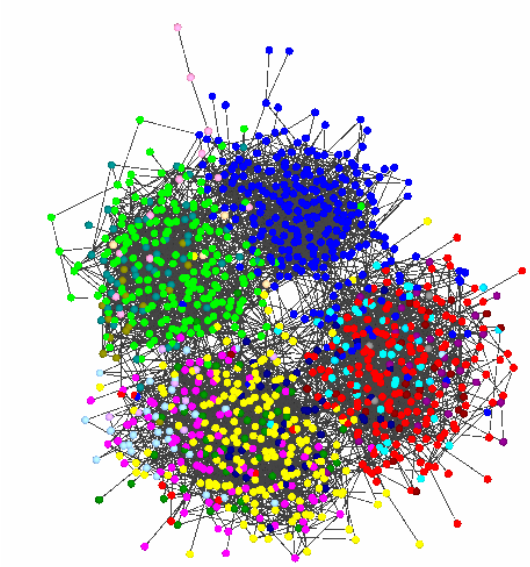
Level 1: 206 modules (123.17)
Level 2: 18 modules (9.34)

Figure S11. Modular hierarchy of a school friendship network obtained by the LinkLand method.

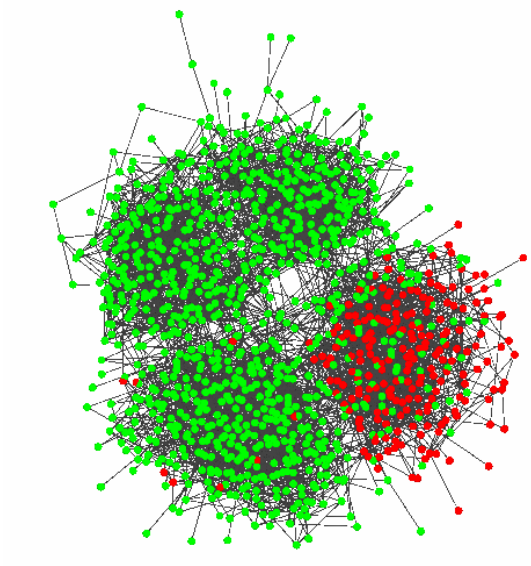
The modular structure of the school friendship social network of Community-44 of the Add Health dataset (see Suppl. Methods, Section I.4.; Moody, 2001; Newman, 2003) is shown at various hierarchical levels, as uncovered by the LinkLand community heap construction method using the ProportionalHill module membership assignment method. Community heaps were determined only at the original network level. During the post-processing of the module assignment, we merged the modules with ProportionalHill module membership assignment-based correlation higher than 0.9. The network was laid out with the Kamada-Kawai algorithm. Colors represent the color of the module, where the given element assigned most. The numbers in parentheses are the effective number of modules (see Suppl. Methods, Section V.6.b.). As shown, the higher hierarchical levels yield less, but more extended modules.



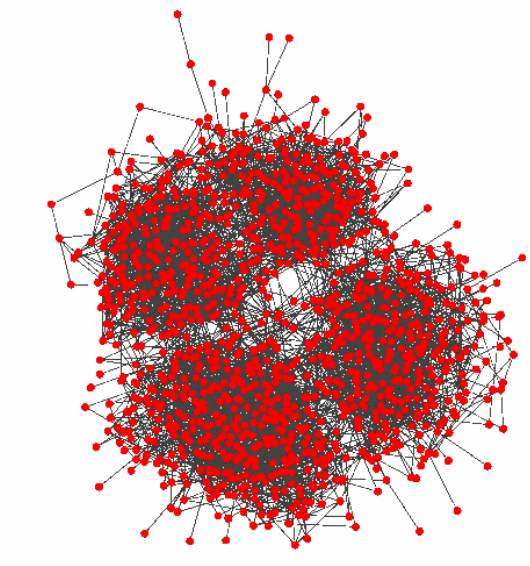
LinkLand – level 0
174 modules (27.55)



LinkLand – level 1
18 modules (7.18)



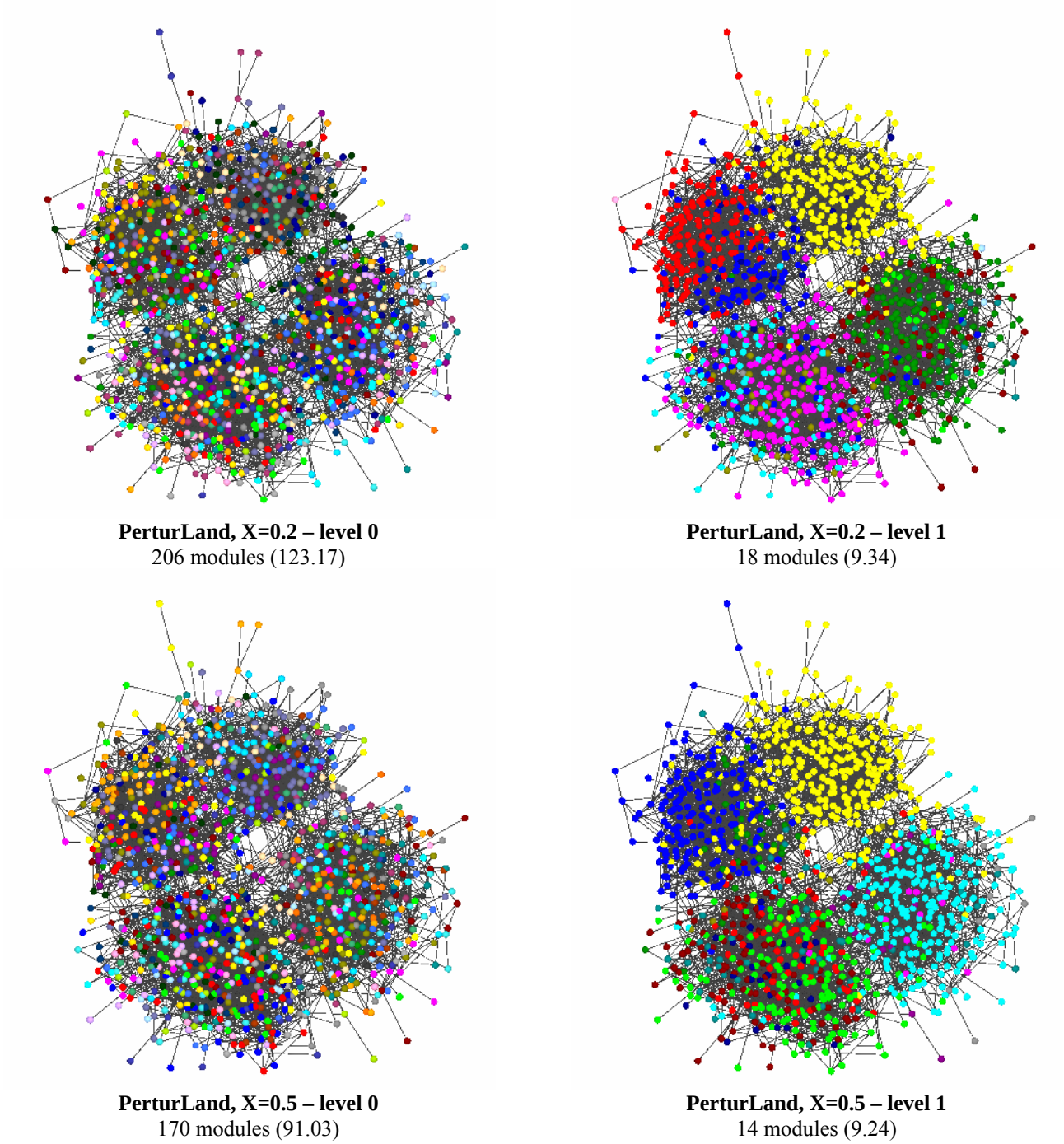
LinkLand – level 2
2 modules (1.62)



LinkLand – level 3
1 modules (1)

Figure S12. Modular hierarchy of a school friendship network obtained by the PerturLand method.

The modular structure of the school friendship social network of Community-44 of the Add Health dataset (see Suppl. Methods, Section I.4.; Moody, 2001; Newman, 2003) is shown at various hierarchical levels, as uncovered by the PerturLand community heap construction method with different *X* parameters, using the ProportionalHill module membership assignment method. Community heaps were determined only at the original network level. During the post-processing of the module assignment, we merged the modules with ProportionalHill module membership assignment-based correlation higher than 0.9. The network was laid out with the Kamada-Kawai algorithm. Colors represent the color of the module, where the given element assigned most. The numbers in parentheses are the effective number of modules (see Suppl. Methods, Section V.6.b.). The *X* parameter of the PerturLand community heap construction method controls the decay of the simulated perturbation. *X* values near 0 cause a rapid decay and thus result in small, nuclear community heaps, while *X* values near 1 cause a minimal decay and result in extended community heaps. As seen in the figure, the larger *X* parameter of 0.5 (inducing a slower perturbation-decay and yielding more extended community heaps) merged many of the smaller and even two of the four major modules (Newman, 2003) of Community-44.



PerturLand, X=0.2 – level 0
206 modules (123.17)

PerturLand, X=0.2 – level 1
18 modules (9.34)

PerturLand, X=0.5 – level 0
170 modules (91.03)

PerturLand, X=0.5 – level 1
14 modules (9.24)

Figure S13. Robustness of the ModuLand method in recovering natural modules of benchmark graphs.

Panels **(a)** and **(b)** show the correspondence of the identified modules to the modules of the benchmark graph of Lancichinetti et al. (2008). It can be seen that modules of the benchmark graph are consistently identified until the point, while modules can be defined in the strong sense ($\mu<0.5$). Representations of the benchmark graph were generated with degree and module size distribution exponents $\gamma=2$ and $\beta=1$ (panel **a**) or $\gamma=2$ and $\beta=2$ (panel **b**). The number of nodes was N=1000, the maximum degree was $K_{max}=50$, the average degree varied as K=15 (blue rectangles), K=20 (yellow triangles), K=25 (red diamonds), and the network fuzziness (μ of the x-axis of Panels **a** and **b**) was ranging from 0.1 to 0.6 (x-axis), where $\mu>0.5$ means that the modules are no longer defined in the strong sense. Higher normalized mutual information (shown on the y-axis) represents a better recovery of the original modules. Modules were identified using the NodeLand community heap construction method and the ProportionalHill module membership assignment method with merging highly correlated modules using an arbitrary chosen correlation threshold of 0.9 (see Section VI.1.). The figure shows the averaged results of 100 representations. Panels **(c)** and **(d)** show the effect of choosing different correlation thresholds for merging correlated modules. Representations of the benchmark graph were generated with degree and module size distribution exponents $\gamma=2$ and $\beta=1$, respectively, the number of nodes was N=1000, the maximum degree was $K_{max}=50$, the average degree was K=20, and the network fuzziness was set as $\mu=0.5$. Data show the average of 100 representations. Panel **(c)** shows the relative frequency (y-axis) of the given module-module correlation values (binned x-axis, bin size=0.1). It can be seen that the majority of module-pairs are weakly correlated while higher correlation is less probable. However, the increased probability of extremely high correlations indicates the existence of nearly identical artifact modules, which artifacts can result from using a PeakHill module determination method on a noisy community landscape. Panel **(d)** shows the effect of removing artifacts by merging correlated groups of modules into single modules above a given correlation threshold (x-axis) on the module identification accuracy measured via the normalized mutual information (y-axis). It can be seen that a threshold too low results in the erroneous merger of distinct modules, while a threshold too high misses to merge some artifact modules. Generally, the correlation threshold for merging modules may be chosen by inspecting the frequency curve of module-pair correlations and setting the threshold to merge the modules of extremely high correlation.

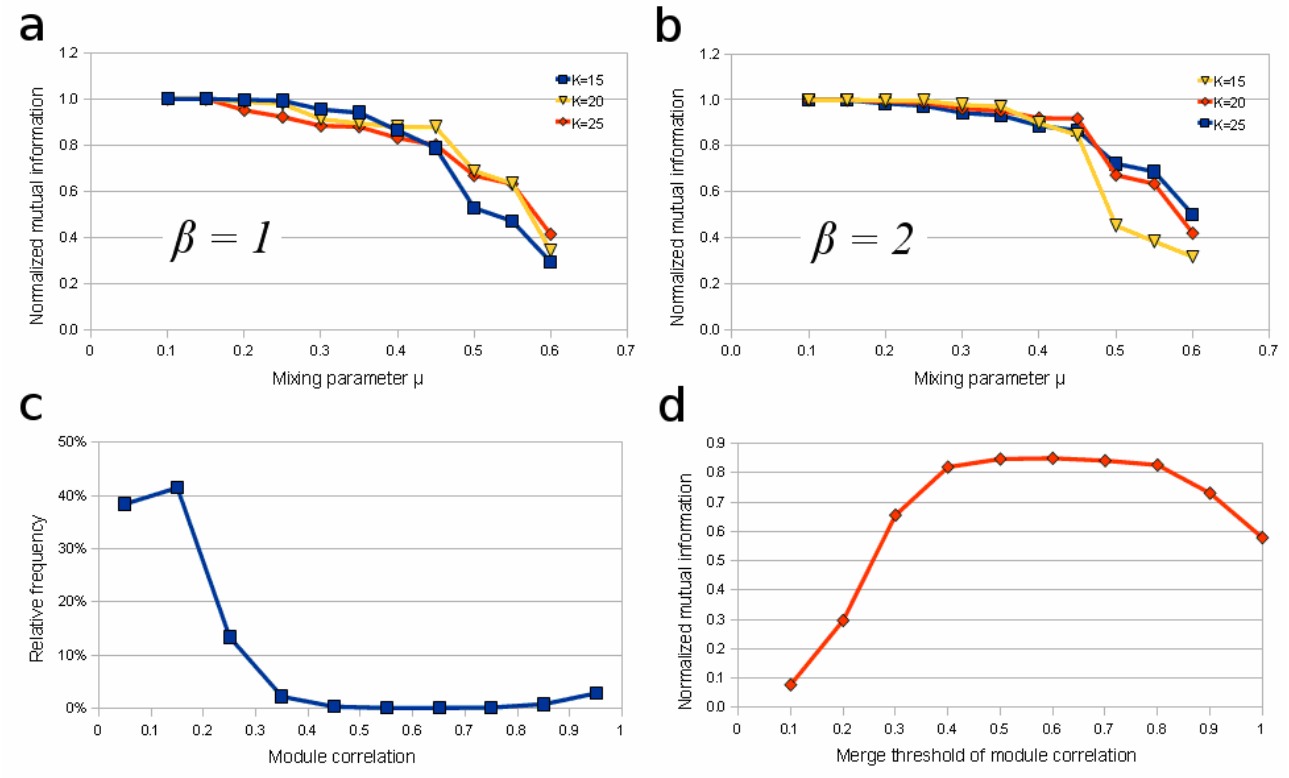
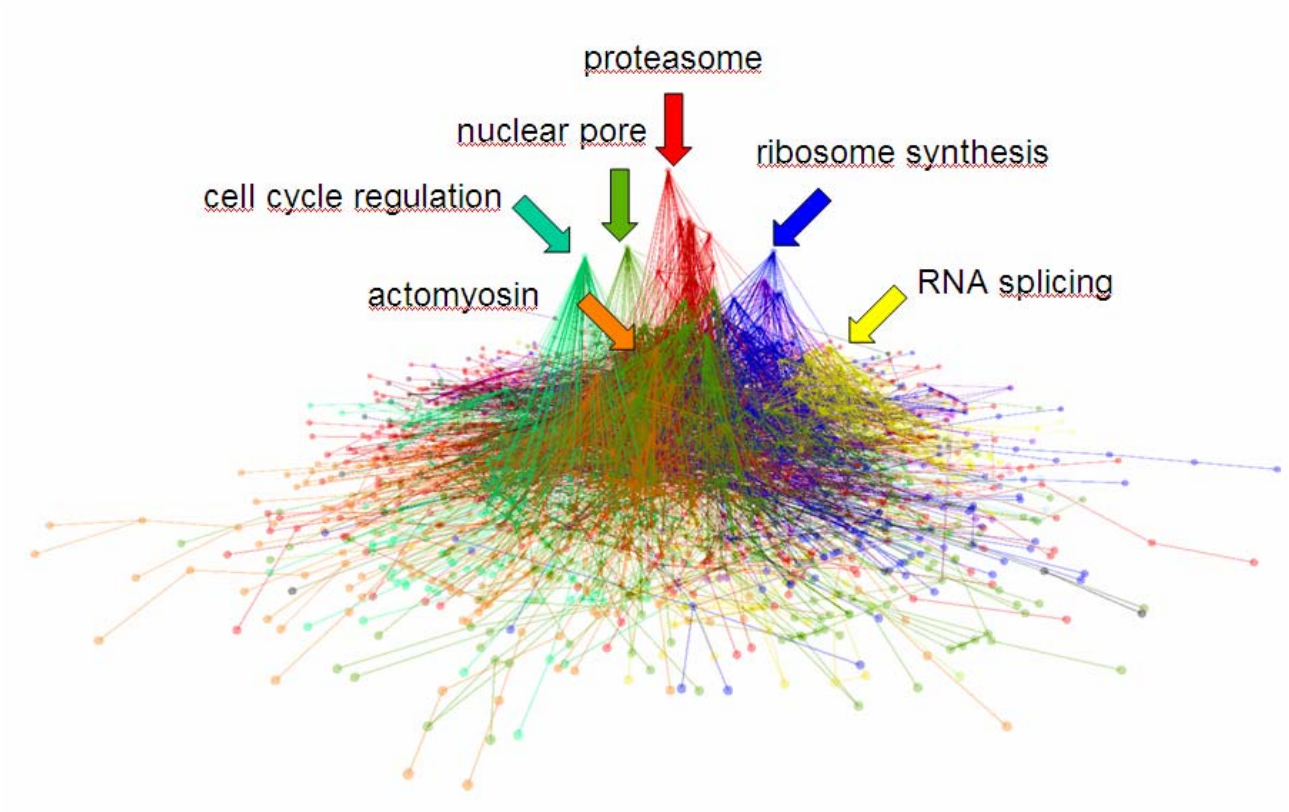


Figure S14. Functional modules of the yeast protein-protein interaction network.
Overlapping modules of the yeast protein-protein interaction network of Ekman et al. (2006) were identified using the LinkLand community heap construction method together with the TotalHill module membership assignment method. During the post-processing of the module assignment, we merged the modules with ProportionalHill module membership assignment-based correlation higher than 0.9. The modular structure of the lowest level of hierarchy is shown. The underlying 2D network layout was set by the the Kamada-Kawai algorithm. The vertical positions reflect the community landscape values of the elements on a linear scale. Elements were colored as the module of their maximum membership. The modular functions were assigned by the functions of the core modular proteins having at least 50% of the community landscape height of the local maximum of the module. In all cases these core-proteins showed a functional consensus. The functional labels and their arrows have the similar colors to their respective modules.



Supplementary Methods

I. Networks used for the analysis

1. Network science collaboration network

This network is the giant component of the undirected network science collaboration network as compiled by Mark J. Newman (2006a) containing 379 elements (network scientists) and weighted 914 links between them. Here a link represents a joint publication between two authors, and the weight is proportional with the number of these co-authorships. The original network data is available at the web-site: http://www-personal.umich.edu/~mejn/netdata/. The network data can also be downloaded from our web-site: <www.linkgroup.hu/modules.php>.

2. Zachary karate club social network

We analyze the weighted and undirected social network of a karate club as recorded by W. Zachary from 1970 to 1972 (Zachary, 1977). This network has 34 elements and 78 links, where elements were (initially) members of a karate club, and links are weighted with their interaction strength. The karate club network is a favored test-case for community detection, because in the course of recording the network data, members of the karate club have split into two factions. These factions provide insight about the natural communities of the karate club network. The original network data is available at the web-site: http://vlado.fmf.uni-lj.si/pub/networks/data/Ucinet/UciData.htm#zachary. The network data can also be downloaded from our web-site: <www.linkgroup.hu/modules.php>.

3. Word association network

We have used the University of South Florida word association network (http://www.usf.edu/FreeAssociation/), where 6,000 participants produced nearly three-quarters of a million responses to 5,019 stimulus words. This word association network gives a relative strength for each stimulus-response word pair, calculated by taking into consideration the count of associations to the response word given the count of stimuli by the stimulus word: The relative weight of an A → B link (called forward strength, FSG) is expressed as FSG = P/G, where G is the count of people who received the word A as the stimulus, and P is the count of people among them who responded with word B for that stimulus. Based on this data, a weighted and directed network can be built. While the direction of links provide insight to the complexity of human conceptual thinking, in the present study we considered the fact of association between words, and built an undirected network. Therefore, the parallel forward and backward links were collapsed into a single non-directed link, and weighted with the sum of the original weights. This process on the giant component of Appendix A of the University of South Florida word association network resulted in a weighted and undirected network. In this study we analyzed the largest connected component of this network consisting of 10,617 elements (English words) and 63,788 links (associations) between them. The network data can be downloaded from our web-site: <www.linkgroup.hu/modules.php>.

The antagonym, heteronym and homonym words chosen for our investigation were selected from lists found on the following websites:

- http://www-personal.umich.edu/~cellis/heteronym.html
- http://www-personal.umich.edu/~cellis/antagonym.html
- http://sb058.k12.sd.us/multiple%20meanings/multiple_meaning_words.htm

4. School-friendship network

We used the data of the high-scale Add-Health survey, which mapped social connections of high schools of the USA (Gonzaléz et al., 2007; Moody, 2001, Newman, 2003).[3] In the survey recorded between 1994 and 1995 social

[3]This research uses data from Add Health, a program project designed by J. Richard Udry, Peter S. Bearman, and Kathleen Mullan Harris, and funded by a grant P01-HD31921 from the National Institute of Child Health and Human Development, with cooperative funding from 17 other agencies. Special acknowledgment is due Ronald R. Rindfuss and Barbara Entwisle for assistance in the original design. Persons interested in obtaining data files from Add Health should contact Add Health, Carolina Population Center, 123 W. Franklin Street, Chapel Hill, NC 27516-2524 (addhealth@unc.edu).

connections of 90,118 students in 84 schools were recorded. For each friend named, the student was asked to check off, whether he/she participated in any of five activities with the friend. These activities were:

1. you went to (his/her) house in the last seven days;
2. you met (him/her) after school to hang out or go somewhere in the last seven days;
3. you spent time with (him/her) last weekend;
4. you talked with (him/her) about a problem in the last seven days;
5. you talked with (him/her) on the telephone in the last seven days.

Based on these data, connections were assigned with weights from 1 to 6. A nomination as friend already resulted in a weight of one, and each checked category added one to that weight. In addition to the nomination data, these files include the gender, race, grade in school, school code, and total number of nominations made by each student.

In our study we analyzed one of the preferably examined school friendship network of the database, the Community-44 school network, because it contains a high number of students with a dense social network (Newman, 2003). This network has an approximately equal number of black and white students. The network contains 1,147 students with 6,189 directed links between them. In our current study directed parallel links were merged into a single undirected link with weight equal to the sum of the original weights and only the largest connected component of the network was used. This process resulted in a weighted undirected network consisting of 1,127 elements and 5,096 links with weight between 1 and 12. The network data can be downloaded from our web-site: <www.linkgroup.hu/modules.php>.

5. Electrical power-grid of the USA

In our studies we used the unweighted and undirected USA Western Power Grid network as an example from the field of engineered networks (Watts and Strogatz, 1998). The power grid network has 4,941 elements and 6,594 links, and is a favored network for studying error propagation and the effect of malicious attacks. The original network data were downloaded from the website of Prof. Duncan Watts (University of Columbia, http://cdg.columbia.edu/cdg/datasets). The network data can also be downloaded from our web-site: <www.linkgroup.hu/modules.php>.

6. Yeast protein-protein interaction network

We used the unweighted and undirected yeast protein-protein interaction network assembled by Ekman et al. (2006) consisting of 2,633 elements and 6,379 links covering approximately half the proteins of yeast genome. We analyzed the largest connected component of the network consisting of 2,444 elements and 6,271 links. Besides the high confidence of its data, we have chosen this network, because it has been involved in the identification of party and date hubs, an interesting dynamic feature of protein-protein interaction networks (Ekman et al., 2006). The network data can be downloaded from our web-site: <www.linkgroup.hu/modules.php>.

II. Overview of the ModuLand network module determination method family

The ModuLand network module determination method family is constructing a community landscape of networks, where the hills and highlands mark high community densities (corresponding to module-cores), and the valleys between them show the approximate positions of overlaps between network communities. In the closing step, the ModuLand method gives a network representation of the overlapping modules, and by a recursive process uncovers a hiearchical network structure in previously unprecedented detail enabling a fast, zoom-in analysis and visualization of large networks. The ModuLand method family is an integrative method consisting of the following four major steps (see main text Figure 1).

1) Determination of community heaps belonging to each element or link, and calculation of a value (called community heap value) for every links and/or elements showing how much the given link belongs to the community heap of the starting element or link.
2) Construction of community landscape, where the community landscape height of a link of the network is a value calculated by taking into consideration all community heap values, which were assigned to the given link in step 1.
3) Determination of overlapping network modules by finding the hills and highlands on the community landscape and assigning all links and elements to them.
4) Constructing the community hierarchy of the network using the original or higher level modules as elements of the second or higher layers of the community hierarchy, respectively.

All the four steps contain variable parameters. The appropriate way of constructing the community heaps may vary from network to network, depending on our information about the complex system and on the interaction, which we wish to analyze. In step 1) the community heap values may give a simple representation of the network communities setting the community heap value of those elements or links as a constant larger than zero, which belong to a community heap and zero to all others. However, more sophisticated community heap value structures can also be designed, and there is a wide variety of possible assignment criteria for the determination of the boundaries (or gradual decay) of community heaps. It is also possible that we construct the community heaps for an assembly of starting elements or links (e.g. cliques or motifs) and not for individual network elements or links what the detailed procedures of this paper will describe. From now on we assume that the community heap values are non-negative. (Numerous specific cases can be extended to other community heap values, including negative values, or even complex numbers, too.)

In a simple and straightforward version of step 2) the community heap values may be summed for each element or link. However, other community landscape construction methods may also give satisfying results. For instance, if the community heap value of the given element or link represents the local deviation of a measure, where the deviation is caused by the starting element of the community heap, the community landscape height may be calculated as the resulting local deviation of all community heaps based on the correlation matrix of their starting elements. In the case when the impacts of the community heaps are uncorrelated, the result is the square root of the sum of the community heap value squares. Thus, the summation we use in the implementations shown in this paper is only one of the many alternative community heap integration possibilities. From now on we assume that the community landscape is constructed by summing up the community heap values for each link.

In step 3) we suggest to use one of the PeakHill algorithms, which start the determination of community landscape hills by finding their hill-tops (or highlands). The rules of element or link assignment to the hill-tops/highlands may vary as we will describe in detail.

Importantly, several versions of the ModuLand methods give a continuous scale for the modular ‘distribution’ of all elements and links in the network marking a fraction of the given element or link belonging to various modules. This gives a much more detailed representation of the network structure than the usual yes/no answers, which unequivocally assign an element or link to one module (see Table S2). The ModuLand methods, which include one of the PeakHill hill determination method-based module membership assignment methods, do not require any previous knowledge on the possible number of network modules. Moreover, the ModuLand method family gives a series of hierarchical modular representations, where the hierarchical topology of the network is gradually uncovered from bottom to top. Top hierarchical levels give a bird’s eye approach of the network, and thus provide better overview, while lower levels contain more and more detailed views of the network. Several ModuLand methods – such as the NodeLand, LinkLand and PerturLand methods described in Sections IV.1. and IV.2., respectively – do not use strictly local or global information of the network for the determination of network communites, but explore many scales of network topology giving less and less weight of network segments being further and further away from the actually examined element or link of the community landscape. It is important to remark, that the main steps of the ModuLand method, which

perform well for weighted undirected networks, are also applicable for weighted directed networks (see e.g. the directed PerturLand Method in Section IV.2.).

In Section IV.3. we will show two extreme cases of community landscapes describing one of the most and the least stringent method of community landscape construction. We will outline three specific representations of the ModuLand method family, the NodeLand, LinkLand and PerturLand methods, as well as the ModuLand adaptations of the well known community detection methods of Girvan and Newman (2002; 2004) and Palla et al. (2005) named as BetweennesCentralityLand (BCLand for short) and CliqueLand methods, respectively. Before starting all these, in the following sub-sections of this Section we list a few basic definitions and considerations we followed during our work, and will describe the 4 major steps of the ModuLand method in detail.

III. Starting considerations and definitions

1. Terminology

- ***Using all elements/links.*** We have considered all (known) elements/links of the networks, and did not introduce any threshold values for excluding any segment of the network., except the elimination of self-loops. The reason for eliminating self-loops is that we are interested in the interactions between the elements.
- ***Non-existing links.*** Non-existing links (or non-identified, missing, hidden, 'secret' links) could be regarded as links having a zero weight.
- ***Community heap.*** Based on the direct interactions in the network we calculate the effective, indirect impact of the starting element to the rest of the network. In our work we represent this new indirect interaction as a property of the original links of the network, so we calculate a new weight, the so-called community heap value for every link from a given starting element. The community heap is a connected subgraph, surrounding the starting element in which the community heap values are all larger than zero. In general the community heap may start from a pair of elements, or from a link (like in the LinkLand method), or more generally from any selected motifs or subgraphs. Since the precise form of the effective interaction depends on the meaning of the links in the network, in principle each network may have its own, unique optimal community heap construction method. The optimal community heap construction method may differ from network to network, but for most networks general versions of both fast and accurate methods can be designed. Different versions of these generally applicable community heap construction methods of the ModuLand method family will be described later in detail, and adequate hints for their optimal use will also be given.
- ***Community heap value.*** The community heap value is a non-negative number rendered to every links of the original network showing how much the given link belongs to the community heap of the starting element or link. The community heap values depend on the community heap constructing method.
- ***Community landscape.*** Integrating all community heap values, which were assigned to a given link, we get a centrality-type value called as the community landscape height of that link. The community landscape height shows how much the given link is affected by the integrated indirect impact of all the starting elements of the network. From now on in this paper the community landscape is simply defined as the sum of the community heap values. We usually represent the community landscape as a 3 dimensional image of the original network, where the horizontal plane is a 2 dimensional, 'usual' representation of the network, while on the vertical axis the community landscape values of network links are plotted.

2. Input data of the ModuLand method family

The ModuLand method family requires a list of at least unweighted and undirected links between the elements of the network. The ModuLand method family can be applied for weighted and directed links as well without increased resource requirements. The present versions of the ModuLand method family accommodate only a single type of links with non-negative weights, where a weight is greater, if the connection is stronger between the endpoint elements. The available network data do not always fulfill these conditions, therefore a data conversion may be required. This is the case, if multiple types of links (colored graphs), links with negative weights, links with weights meaning distances between the elements or multi-element interactions (hypergraphs) are present. There is no universal recipe for converting these systems into a suitable weighted network, but basically weights representing endpoint similarity are advised.

For the present implementation of the ModuLand method family, linear link weights are necessary, as we will see at the introduction of the PerturLand method (Section IV.2.). In this case, parallel links of the same direction between the given elements can, and in practice should be merged, where the resulting weight will be the sum of the original weights. We note, that loop links are discarded, as we are only interested in links between elements.

In the usual case, the implementation of the ModuLand method family requires a detailed information on the links between the network elements. However, we note that in case we do not know the links between the elements, but do know certain features of each element, the ModuLand method family can also be applied by defining a (non-negative, linear) similarity metric between the network elements. Treating these similarity values as link weights, a network representation can be derived.

The principle of the ModuLand method family is the possibly most precise determination of the community heaps, therefore any further information determining the effective interactions is preferred. As an example of such an information, the *activity* of an element (p_i) can be taken into account by the ModuLand method family, if available. The activity of an element represents the relative strength of the community heap of the given element. If we know the

activity, the community heap values of the respective element are multiplied by the activity of that element. This option has high relevance, if we wish to investigate the behavior of a complex system under different circumstances, defining different activity of their elements. For example, news or an epidemic may not spread with equal probabilities from all elements, and this presumption can be modelled by assigning different activities to the elements.

When applying the ModuLand method family, elements in different connected components are always assigned into componentwise distinct sets of modules, therefore it is practical to analyze different connected components separately. This procedure is also useful by sparing computational resources.

IV. Determination of the community landscape

In a typical version of the ModuLand method family the community landscape is constructed by a specific integration of the community heaps of the network. However, there are situations, where the community landscape of the network is directly known instead of knowing the link-structure of the network. In these cases the community heap construction step can be omitted, because we may start directly from the community landscape to determine the modules.

We define the effective, indirect interactions as community heaps. In principle, an optimal method of community heap construction is a unique method for each given network, which may differ from network to network. However, different, generally applicable versions of community heap construction methods can be designed, and will be described here in detail. In all these methods the community heap value of an element for a given community heap means the indirect effect on that given element.

In the current applications it is not our aim to model the network dynamics, but to give static prediction of modules based on a static, or time interval-averaged description of the network. Therefore we only consider community heap construction methods, which result in time-independent, static indirect interactions, but we note that many elements of network dynamics may also be included to the framework we describe here.

1. The NodeLand and LinkLand community heap construction methods for weighted and undirected networks

The NodeLand and LinkLand methods are fast, but approximating methods for the determination of the community heaps in weighted, undirected networks.

In the generalized version of the NodeLand and LinkLand methods the community heap belonging to the starting element or link is determined by a network walk. During this walk the neighboring elements and links of the starting element or link are explored, and this procedure is continued and the community heap is extended until an appropriately selected criteria is fulfilled. Examples for the these criteria are given below, and will be listed in Section IV.1.a. for the NodeLand method, Section IV.1.b. for the LinkLand method, and Section IV.4. for the BetweennessCentralityLand (BCLand) and CliqueLand methods.

In the case of NodeLand and LinkLand methods, the construction of the community heap is governed by the so-called community heap-threshold, which is defined as ([the sum of the weights of the links belonging to the original community heap] / [number of elements in the community heap]).

As an example for the above criteria, the algorithms of the NodeLand and LinkLand methods follow the principle of ‘the community heap-threshold of the growing community heap is not allowed to decrease’. This principle is extended further in the NodeLand method by the conjuncture that ‘maximal growth is preferred’. These principles result in the following illustrative behavior of community heap growth.

- If the starting element of the community heap was at a connection-poor part of the original network, the community heap will extend towards more and more dense parts of the network. Having found a connection-rich region, the growth of the community heap stops.
- Conversely, if the starting element of the community heap was at a connection-rich part of the original network, the extension of the community heap will stop very soon, since there will be only such elements in the further neighborhood of the initial community heap, with which the community heap-threshold could only be decreased.

As a result of the above behavior illustrated on Figure 1A of the main text, elements and links belonging to a local connection rich region will be members of many community heaps, while elements and links of connection-poor segments will not be ‘attractors’ of community heap growth, and will only be members of very few community heaps.

In the end, after creating the community landscape by combining (in the methods detailed in the present paper simply summing up) community heaps, connection-poor segments of the network will have lower community landscape height, and thus will be valleys of the community landscape, while connection-rich segments will have high community landscape height, and thus will form hills of the community landscape.

a. The NodeLand community heap construction method

In the NodeLand community heap construction method the starting point of each community heap is an element of the original network. The starting element and later, its growing community heap are extended by only that neigboring element and its links linking it to the existing community heap, which will increase the community heap-threshold (as defined above: [the sum of the weights of the links belonging to the original community heap] / [number of elements in

the community heap]) of the existing community heap, and this increase will be maximal among all the possible increases supplied by any of the neigboring elements. If more than one elements exist, which fulfill the above criteria, all of them are added to the community heap, including the links connecting these elements both with each other and with the existing community heap. If additional elements are added to the community heap of the starting element, the community heap-threshold of the community heap generally increases, making the chances of element additions to the community heap from any further rounds of neighboring elements even more difficult. If there are no neighboring elements fulfilling the above, rather stringent requirements, the community heap of the starting element is considered to be ready, and the method continues with the determination of the community heap belonging to the next element of the original network. **Algorithm 1** below describes the community heap construction of the NodeLand method in detail.

Algorithm 1: Algorithm of the NodeLand community heap construction method

```
/*
Important variables used in the algorithm:

        startNode: the starting element.

        heapNodeList: elements of the community heap (initially empty).

        heapLinkList: links of the community heap (initially empty)..

        tempList: elements to be added to the community heap in the next round

        actualHeapThreshold: sum of the weight of all links in heapLinkList divided by the number of elements in
        heapNodeList.
*/
tempList := [ startNode ]
while tempList is not empty {
        add all elements of tempList to heapNodeList
        for each link e connected to any elements of tempList {
                if endpoints of e are already in heapNodeList { add e to heapLinkList }
        }
        clear tempList
        recalculate actualHeapThreshold
        maxNewHeapThreshold := actualHeapThreshold

        for each element n not in heapNodeList but having non-zero links lks with an endpoint in heapNodeList {
                newHeapThreshold := sum weight of links in heapLinkList + sum weight of link in lks
                newHeapThreshold := newHeapThreshold / (number of elements in heapNodeList + 1)
                if newHeapThreshold > maxNewHeapThreshold {
                        clear tempList
                        maxNewHeapThreshold := newHeapThreshold
                }
                if newHeapThreshold = maxNewHeapThreshold { add n to tempList }
        }
        if maxNewHeapThreshold = actualHeapThreshold {
                // cannot increase the threshold any more, stop algorithm
                clear tempList
        }
}
```

In the end of **Algorithm 1**, we find the elements and links of the community heap in the **heapNodeList** and **heapLinkList** lists of **Algorithm 1**, respectively. Identifying the community heap of one node in the NodeLand algorithm is structurally similar to a breadth-first search, therefore the runtime complexity of the algorithm is O(n(n+e)), where *n* is the number of nodes and *e* is the number of links in the network. However, in practice the algorithm is extremely fast as a community heap of any given node rarely covers the whole network.

For downloading the ModuLand program package including the NodeLand community heap construction method of **Algorithm 1** as the **nodeland** program see our homepage <http://www.linkgroup.hu/modules.php>.

In the NodeLand method the community heap value of a link in the community heap is set as the original weight of this link, while the community heap value of a link not belonging to the community heap is set as zero. Furthermore, if the activity of the elements is known, then all community heap values are multiplied by the activity of the starting element of the community heap, as described earlier in Section III.2.

The community landscape value of each elements or links are constructed by summing up the community heap values of that element or link for all community heaps. We note that in case of the NodeLand community heap construction method, the community landscape height value of non-zero weighted, but locally weak links may become zero, if that given link does not belong to any community heaps.

The NodeLand method resembles to the *l*-shell method of Bagrow and Bollt (2005) with the important difference that in the NodeLand method the community heaps resembling the *l*-shells are later summarized in order to create the community landscape. The NodeLand method can be easily applied for directed networks, too. In this case we take into consideration only the outgoing links from the existing communitiy heap by determining the next elements of the community heap.

b. The LinkLand community heap construction method

The LinkLand community heap construction method gives a less stringent method for the determination of network modules than the NodeLand method described in the previous section. In agreement with this property, the NodeLand method usually gives smaller modules and faster results than the LinkLand method.

In the LinkLand community heap construction method the same community heap-threshold is used as in the NodeLand method. Thus, the community heap-threshold is defined as ([the sum of the weights of the links belonging to the original community heap] / [number of elements in the community heap]). The starting point of each community heap is a link of the original network with its two end-elements. Since the starting link of the community heap connects its two end-elements, we define the starting community heap-threshold as half of the original weight of the starting link. The starting link and later its growing community heap are extended by those neigboring elements and their links linking them to the existing community heap and to each other, which will at least not decrease the community heap-threshold of the existing community heap. Please note, that the LinkLand method neither requires a direct increase in the community heap-threshold, nor asks for the maximal increase among all the possible increases, which were the characteristic features of the NodeLand method.

To see the meaning and scope of the LinkLand method from an other point of view, we can define the community heap connection strength of a given element in the network as the sum of the weights of those links, which are connecting the element to the existing community heap. With this definition, we can express the LinkLand method as follows. The growing community heap is extended by those neigboring elements, which have a community heap connection strength to the existing community heap *at least equal to* the community heap-threshold of the community heap.

After all eligible neighboring elements and their linking links have been already added to the community heap of the starting element, the community heap-threshold of the community heap is re-calculated. If there are no further neighboring elements having an equal or higher community heap connection strength than the community heap-threshold of the community heap, the community heap of the starting element is considered to be ready, and the method continues with the determination of the community heap belonging to the next link of the original network.

Algorithm 2 below describes the LinkLand community heap construction method in detail.

Algorithm 2: Algorithm of the LinkLand community heap construction method

*/**

Important variables used in the algorithm:

***startLink**: the starting link of the actual community heap.*

***heapNodeList**: elements of the community heap (initially empty).*

***heapLinkList**: links of the community heap (initially empty).*

***tempList**: elements to be added to the community heap in the next round.*

***actualHeapThreshold**: sum of the weight of all links in **heapLinkList** / number of elements in **heapNodeList**.*

```
*/
clear tempList
add the two end-elements of startLink to tempListwhile tempList is not empty {
        add all elements of tempList to heapNodeList.
        for each link e connected to any elements of tempList {
                if endpoints of e are already in heapNodeList { add e to heapLinkList }
        }
        clear tempList
        recalculate actualHeapThreshold
        maxNewHeapThreshold := actualHeapThreshold

        for each element n not in heapNodeList but having non-zero links lks with an endpoint in  heapNodeList {
                newHeapThreshold := sum of the weight of all links in heapLinkList + sum weight of link in lks
                newHeapThreshold := newHeapThreshold / (number of elements in heapNodeList + 1)
                if newHeapThreshold > maxNewHeapThreshold {
                        clear tempList
                        maxNewHeapThreshold := newHeapThreshold
                }
                if newHeapThreshold = maxNewHeapThreshold { add n to tempList }
        }
}
```

In the end of the **Algorithm 2**, we find the links and elements of the community heap in the **heapLinkList** and **heapNodeList** lists of **Algorithm 2**, respectively. Identifying the community heap of one link in the LinkLand algorithm is structurally similar to a breadth-first search, therefore the runtime complexity of the algorithm is O(e(n+e)), where *n* is the number of nodes and *e* is the number of links in the network. However in practice the algorithm is very fast as a community heap of any given link rarely covers the whole network.

For downloading the ModuLand program package including the LinkLand community heap construction method of **Algorithm 2** as the **linkland** program see our homepage <http://www.linkgroup.hu/modules.php>.

While the community heap value of a link outside the community heap is set to zero, the community heap value of the link in the community heap is calculated by multiplying the weight of the actual link by the weight of the starting link of the community heap. Thus community heaps with a starting link of significant weight gain higher influence on the community landscape, and communities of non-existing links (or links of zero weight) are automatically excluded from influencing the community landscape.The LinkLand community heap construction method could be regarded as taking pairs of elements instead of links as starting point of the community heaps. From this it follows that the LinkLand method will assign a zero community heap value to all links of the graph, which belong to the community heap of a pair of starting elements having no link between them.

In case the activity of the elements is known, then in the LinkLand method the community heap values are multiplied by the average activity of the endpoint elements of the starting link of the respective community heap.

The community landscape value of each elements or links are constructed by summing up the community heap values of that element or link for all community heaps. We note that, unlike the case of the NodeLand construction method (where the community landscape height value of non-zero weighted, but locally weak links may become zero, if that given link does not belong to any community heaps), in the LinkLand method the community landscape height value of a link with non-zero, positive weight will always be a non-zero, positive value.The LinkLand community heap construction method can be applied for directed networks as well by only considering the outgoing links for determining the neighboring elements of the growing community heap in the community heap construction process.

2. The PerturLand community heap construction method for weighted and directed networks

In the PerturLand community heap construction method the indirect effect of a given starting element is modelled as the amount of information reaching other elements and links of the network spreading out from the given starting element. The piece of information spreading from the starting element remains ‘identical’ throughout the spreading process, which can be envisioned as the very same piece of information reaching the other elements, but with decreasing reliability or confidence. Therefore, the maximum incoming and outgoing information quantity of a given element (and

not the sum information quantity) is the measure of how much information the given element receives from the network and sends to the network, respectively. The initial link weights (or more precisely, these values multiplied by the value of the *X* parameter, as seen later) give the ratio of the information, which is spreading from the starting element to the ending element. The initial information content of the elements is uniformly one. If we know and use the activity (defined in Section III. 2.) of the elements, it corresponds to the initial condition, where the initial information content of an element is the activity of that element. The PerturLand method assumes the link weights to be in the range of [0;1]. If the link weights are not in the range of 0 to 1, the link weights have to be normalized. In the followings s_{ij} will denote the link weight, if the link*(i,j)* exists, and $s_{ij}=0$, if no such link exists.

The community heaps of different starting elements are independent from each other. From their independence it follows that the community heaps can be determined in any order. The information-spread determining the community heap generally corresponds to a perturbation spreading process. In this process a perturbation (disturbance, impulse, information) spreads from the starting element through directed links, while its effect gradually weakens. If a unit of the perturbation is propagating from element *a* through the *(a,b)=e* link towards element *b*, then the quantity of perturbation arriving to element *b* from element a is $s_{ab}X$, where *X* is a parameter describing the attenuation of the perturbation during the propagation process. *X* can be in the range of [0;1]. The higher the value of *X*, the smaller the attenuation of the perturbation is spreading in the network. The value of *X* should be chosen by taking into account the properties of the analyzed network. *X* can be determined by estimating the information transfer efficiency between the network elements. The formal definitions and rules of the perturbation spreading process are given below. In the example detailed by these formalism the starting point is an element. In principle the starting point may also be a link, or a set of elements and/or links.

We define the vector *P[i]* to store the maximum quantity of perturbation that reached element *i*. Initially

$$P[i] = \begin{cases} 0, & i \neq a, \\ p_i, & i = a, \end{cases}$$

where p_i ($0<p_i\leq 1$) is the activity of element *a*, where *a* is the starting element of the perturbation. During the perturbation spreading process the values of the vector *P[i]* are iteratively modified. The alteration of the vector *P[i]* in one round of iteration can be described as

$$dP[b] := \begin{cases} \max_i(P[i]s_{ib}X), & P[b] = 0 \\ 0, & P[b] \neq 0 \end{cases}$$

where the *X* parameter is the attenuation coefficient of the perturbation as described before. Component *b* of the vector *dP* is the quantity of the perturbation that would reach element *b*, which was perturbation-free in the previous round of the iteration process, if the perturbation would flow towards element *b*. Knowing these values, perturbation only spreads towards the elements, which would be maximally effected by the perturbation. Matrix element *M[j][m]* denotes perturbation flowing from element *j* to element *m* through the link*(j,m)*, if there is such link, otherwise *M[j][m]* is equal to zero. Thus matrix *M* keeps an account of the perturbations flowing in the network. Initially, matrix *M* is filled with zeros.

$$M[j][m] := P[j]s_{jm}X, \ \forall m : dP[m] = \max_i dP[i], P[j] > 0$$

After determining the quantity of perturbation flowing from the community heap to the newly reached elements, the quantity of perturbation flowing into these newly reached elements can be calculated:

$$P[m] := dP[m], \ \forall m : dP[m] = \max_i dP[i]$$

After merging these new elements *m* to the community heap, the perturbation spreads on the *(m,n)* links sourcing from these new elements to other elements *n* in the community heap (including other newly merged elements), and the following update reflects this spreading:

$$M[m][n] := P[m]s_{mn}X, \ \forall m : dP[m] = \max_i dP[i], P[m] > 0$$

Please note that the normalization of link weights, the range of the *X* attenuation parameter and the max-greedy nature of the perturbation process described here together yield that the *P[i]* perturbation quantity of element *i* never increases once it was set to a non-zero value. This statement can be explained by the procedure first transforming the initial perturbation quantity and the ($s_{mn}X$) perturbation affinities of links by taking their negated logarithmic values, applying

Dijksta's shortest path algorithm (Dijkstra, 1959) and finally transforming back the resulting perturbation quantities, which yields the same result as the perturbation process described here.[4] It is known that Dijkstra's algorithm will never update the value of an element after updating it for the first time (Dijkstra, 1959). Similarly, the *M[m][n]* perturbation flow from element *i* to *j* will never increase once it has been set to a non-zero value.

The above steps describe one round of the iterative process for determining the perturbation quantity of each link and element of the network. The iteration stops once all components of the vector *dP* become zero, meaning that the non-zero (but maybe infinitesimally small) perturbation quantity reaching each link and element is already determined. At the end of the iteration process component *i* of vector *P* is the maximum perturbation quantity flowing into element *i*. The community heap value, h^a_{ij} of any link *(i,j)* to the community heap of element *a* (the starting element of the perturbation spreading process) is defined as the perturbation quantity flowing through link *(i,j)*, that is

$$h^a_{ij} = M[i][j] .$$

Let the community heap value of element *i* be $h^a_i = max_j\{M[i][j]\}$, i.e. the maximum outflowing perturbation quantity of element *i*. With this defintion, the determination of the community heap is completed. This process of determining community heaps of a directed network describes the PerturLand community heap construction method. Once all community heaps have been determined, the community landscape height of a given link is calculated by summing the community heap values of the given link for all community heaps. Note that the *X* parameter is not required for constructing the community heaps of higher hierarchical level networks, because these networks already have link weights proportional to the information flow rate on these links (see Section VII. for details).

While the PerturLand method is defined in the above desription for directed networks only, it can easily be applied in case of undirected networks by substituting undirected links with two directed links of opposing direction having the same weights as the original link. In this version of the PerturLand method the community landscape height of a given undirected link is defined as the maximum of the community landscape height of the substituted directed links.

Algorithm 3 below describes the PerturLand community heap construction method in detail.

Algorithm 3: Algorithm of PerturLand community-heap construction method

*/**
Input variables:

startElement: *the starting element of the actual heap.*

startPerturbation: *the starting perturbation (activation) of startElement*

X: *the intensity of the perturbation flow (free parameter, chosen with regards to the dynamical properties of the given network, $0<X\leq 1$)*

normWeight[i,j]: *the normalized weight of the directed $i\rightarrow j$ links*

Important variables used in the algorithm:

maxInPert[i]: *the maximal inflow perturbation of element i*

maxOutPert[i]: *the maximal outflow perturbation of element i*

tempMaxInPert[i]: *the maximal inflow perturbation reaching element i, if it would be added to the community-heap in the next round*

newElements: *the list of the elements to be added to the community-heap*

linkPert[i,j]: *the perturbation spreading trough the directed $i\rightarrow j$ link*

Output variables:

In the end of the algorithm, the community-heap assignment values of the elements will be stored in the ***maxOutPert*** *array, and the community-heap assignment values of the directed links in the*

[4]Although the behavior of the PerturLand method for determining the centrality of elements can be described by calculating shortest paths via the said transformations, the centrality measure provided by the PerturLand method greatly differs from the betweenness centrality, because PerturLand assigns centrality to elements based on properties of shortest paths, while the betweenness centrality characterizes a centrality based on the number rather than the properties of shortest paths.

```
        linkPert array
*/
clear maxInPert, maxOutPert, tempMaxInPert, linkPert variables (set all elements of the arrays to zero)
tempMaxInPert[startElement] := startPerturbation

do {
        clear list newElements
        newElements := the positions of the max values in tempMaxInPert

        for each element i of the newElements array {
                // setting the community-heap -> newElement links
                For each each j->i link where j not in newElements array but
                maxInPert[j]>0, so element j already belongs to the heap {
                        linkPert[j,i] := X * maxInPert[j] * normWeight[j,i]
                }
                // setting the newElements
                maxInPert[i] := tempMaxInPert[i]
        }

        for each each element i of the newElements array {
                // setting the newElement->heap links (including the cross-links between
                // the newElements)
                for each i->k link where i is the given newElement and
                maxInPert[k]>0, so element k already belongs to the heap; and
                linkPert[i,k] is zero so the given link does not belong to the heap {
                        linkPert[i,k] := X * maxInPert[i] * normWeight[i,k]
                }
        }
        clear array tempMaxInPert (set all elements to zero)
        // check the non-heap neighbors of the heap elements
        for each i->j link where maxInPert[i]>0 and maxInPert[j] is zero {
                tmp := X * maxInPert[i] * normWeight[i,j]
                if tmp > tempMaxInPert[j] then { tempMaxInPert[j] := tmp }
        }
} while there are any non-zero elements of the tempMaxInPert array

// setting the maxOutPert array
for each element i of the network {
        for each i->j link {
                if maxOutPert[i] < linkPert[i,j] then {
                        maxOutPert[i] := linkPert[i,j]
                }
        }
}
```

The runtime complexity of the PerturLand community lanscape determination method as given by **Algorithm 3** is O(ne) per community heap, where *n* is the number of nodes and *e* is the number of links in the network. However, there is space for more optimal algorithms, since the result of perturbation flow process can either be calculated with a modified Dijkstra algorithm (Dijkstra, 1959) effectively enabling a runtime complexity of O(e + n log n) per community heap, or by the Floyd–Warshall algorithm (Floyd, 1962) enabling a runtime complexity of $O(n^3)$ for the whole community landscape construction process.

For downloading the ModuLand program package including the PerturLand community heap construction method of **Algorithm 3** as the **perturland** program see our homepage <http://www.linkgroup.hu/modules.php>.

3. Two extreme cases of community landscapes for weighted and directed networks

We continue the explanation of the ModuLand method family by showing two extreme examples having one of the most, and one of the least stringent assignment criteria (see Figure S3 for an illustration).

One of the most stringent methods of the determination of the community heaps is, if only the links starting from the element *A* will form the community heap of element *A* and their community heap values will be equal to the weight of the given link. With this definition, in the directed case the summation of the community heap values will give the original weights of all links as their community landscape height (in the undirected case the community landscape height will be twice as much for each link).

One of the least stringent methods for the determination of community heaps, if we take all elements or links as parts of the community heap, which are in the same connected subgraph than our starting element or link, and we set their community heap values all equal, e.g. 1. Summing up the community heap values at the end of the assignment procedure we will get all connected subgraphs as plateaus of the community landscape with a community landscape height equal to the number of elements or links they contain.

These two extreme examples do not seem to be suitable for community detection, but at least show the general applicability of the ModuLand method for the analysis of network topology. However, in specific original networks (depending on the meaning of the given links in the network) even these extreme methods may be appropriate for the construction of the community landscape.

4. Transformation of widely used, former modularization methods to the community landscape framework of the ModuLand method

Numerous former module detection methods can be implemented in the ModuLand method family framework. In this process the former module detection gains the ability to detect modular overlaps even in the case, when this property was not among its original features. Furthermore, as a result of the implementation, the number of modules can be obtained automatically, instead of a previous choice or a pre-set parameter tuning. The implementation process may often result in the community landscape directly omitting the community heap construction step. The following CliqueLand and BetweennessCentralityLand (BCLand) community landscape construction methods are such examples.

a, The CliqueLand community lanscape determination method for unweighted and undirected networks

The CliqueLand method is an adaptation of the widely used, efficient clique percolation method (named as CFinder by the authors) of Tamás Vicsek and his collegaues (Adamcsek et al., 2006; Farkas et al., 2007; Palla et al., 2005; 2007a) for the determination of overlapping network modules using the terminology of the ModuLand method family. In the original version of the clique percolation method (Palla et al., 2005) links below a suitably selected threshold are left out from the analysis, and the residual links have a Boolean representation giving a weight of 1 to the link, if it exists, and zero otherwise.[5] The elements and links of the modules determined by this method form *k*-clique communities meaning a union of all *k*-cliques that can be reached from each other through a series of adjacent *k*-cliques (two *k*-cliques are adjacent, if they share *k*-1 elements). The clique percolation method determines the *k*-clique communities of the network for all *k* values, and selecting the optimal *k*-clique distribution treats these interconnected *k*-cliques as network communities. Two interconnected *k*-clique sets may contain *k*-2 common elements, which makes the clique percolation method very suitable for the detection of overlapping network communities.

We may define a community landscape showing the modules of the clique percolation method in frame of the ModuLand method family. Let the community landscape height of a given link be equal to the maximal value of *k*, for which there exists a *k*-clique-community containing this link. If there is no *k*-clique containing the link, than the community landscape height is set to 1.

[5]Although we here only consider the original version of the clique percolation method for unweighted and undirected networks (Palla et al., 2005), it is important to note that the authors of the clique percolation method have later introduced important and efficient extensions of the original method, which are able to identify overlapping modules in directed (Palla et al., 2007a) and weighted (Farkas et al., 2007) networks as well.

b, The BetweennessCentralityLand (BCLand) community lanscape determination method for weighted and directed networks

The BetweennessCentralityLand (BCLand) method is an adaptation of the 'gold-standard' network module determination method of Girvan and Newman (2002; 2004) using the terminology of the ModuLand method family. The Girvan and Newman (2002; 2004) method first calculates the betweenness centrality of the network links (the number of shortest paths containing the given link), then iteratively deletes the link of highest betweenness centrality, and recalculates the betweenness centralities of the residual network. This way the original network is decomposed into non-overlapping components.

A community landscape over the links can be assigned to the Girvan-Newman method (Girvan and Newman, 2002; 2004), which is iteratively removing links in the order of their decreasing betweenness centrality. Let the community landscape height c_{ij} of the link*(i,j)* be k *(k>0)*, if the given link was removed from the network in the *k*-th iteration. The community landscape of this betweenness-based method is illustrated on Figure S3.

c, Methods yielding partitions of the network

In addition to the previously described community landscape implementation of the Girvan and Newman (2002; 2004) method, traditional agglomerative or divisive modularization methods, or any methods yielding partitions can be implemented as community landscape based methods, which determine modules by applying a threshold on the community landscape.

For example, in the case of agglomerative methods, the later an element or link is merged to the already existing clusters, the lower their community landscape height can be set. Figure S4a gives an illustration of the community landscape derived from an agglomerative method. In this interpretation of Figure S4a the positions of the horizontal lines correspond to cuts on the dendrogram of a hierarchical modularization method. The modules resulting from these horizontal cuts are the connected components of the original network corresponding to the community landscape segments above the horizontal lines (see the ThresholdHill method in Section V.1.). While the number of modules depends on the position of the horizontal lines (the cuts on the dendrogram), the same method implemented in the frame of the ModuLand method family with the PeakHill hill determination method (see Section V.2.) determines the number of modules automatically, and may yield overlapping modules, if needed.

d, Stochastic module detection methods

Methods yielding multiple possible partitions or yield partitions by heuristic optimalization of some measure (see Table S2) can easily result community landscapes by defining the community landscape height of the link*(i,j)* to be the number of cases, in which the endpoints *i* and *j* belong to the same module, taking into account the possible module structures or repeated stochastic runs[6]. Note that the different module structures do not need to be equally important, for example structures may be weighted by a fitness measure or probability function of the given structure.

5. Summary of the community landscape determination methods

In the examples above we showed that two of the widely applied previous methods, the method of Girvan and Newman (2002; 2004) finding distinct network communities as well as the method of Palla et al. (2005) finding overlapping network communities may be implemented as a wide sub-group of methods within the ModuLand network module determination method family. We also showed how several classes of other, previously described modularization methods of Table S2 may also be transcribed to the frame of the ModuLand method. Furthermore, we showed three illustrative examples, that the NodeLand, LinkLand and PerturLand methods use adaptively broader and broader scales of topological information to result in a community structure. We note, that our new methods (the NodeLand, LinkLand and PerturLand) experience no problem to detect the communities of *both* very low and very high density networks. Moreover, these methods can be efficiently applied to networks having any type of hierarchical topology including the simultaneous presence of very dense and very sparse segments.

Obviously, many other community heap construction methods of Table S2 such as those based on the *k*-clan, *k*-club, *k*-plex and other community definitions of Table S1 can be used in other variants of the ModuLand method. It is also possible that we construct the community heaps for an assembly of starting elements or links (e.g. cliques or motifs) and not for individual elements or links what we described here. Moreover, different community heap summation protocols

[6]This idea can also be found in the work of Sales-Pardo et al. (2007), altough the authors do not describe their method as a centrality landscape.

and community landscape exploration methods can be designed. The borders of the community heap must be defined by different algorithms for different types of networks depending on the meaning of the links. We believe that many more methods of the ModuLand family exist beyond the shown methods, which are optimal for one or another type of networks and/or community structures. We invite our colleagues to the exciting journey to explore and try them including – potentially – such methods, which are better for one or another type of networks than those we described in our current study.

V. Determining modules based on the community landscape

In this step we have to find the hills of the community landscape representing the overlapping communities of the original network. In the case of PeakHill methods described in this section, we first identify the module-cores, i.e. the links or connected plateau of links (hills or highlands), which have a local maximum on the community landscape (for illustration, see Figure S4). For the assignment of links (and elements) to module-cores we may use any module membership assignment methods described in Section V.2., such as the GradientHill, the ProportionalHill or the TotalHill methods.

As it can be seen on Figure S4, it is also possible to define modules by simple horizontal cuts on the community landscape. This method is described by the ThresholdHill method. First, we will describe this method in detail, then we will continue with the identification of local maxima, and finally we will introduce three module membership assignment methods (the GradientHill, the ProportionalHill and the TotalHill methods) yielding overlapping modules.

1. ThresholdHill: Horizontal cut of the community landscape as a detection threshold of previous community detection methods

To represent the modularization methods yielding partitions of the network in the ModuLand method family we discuss the ThresholdHill hill finder method. Most previous clustering or divisive methods of network community detection (see Table S2) can be represented as a horizontal cut of the community landscape. Cutting the community landscape at an appropriately selected height, these methods treat the connected subgraphs above (or occasionally: below) this horizontal plane as network modules (see Figure S4). If we treat the elements having no links above the horizontal cutting height as separate modules, we have assigned all elements to the community structure. These methods result in distinct network modules (having no overlaps) like in case of the widely used Girvan and Newman method (Girvan and Newman, 2002; 2004).

The horizontal cut model (making a visually conceivable meaning for the detection limit of network communities) helps us to explain why conventional methods are successful only seldom in finding big and small communities at the same time. These methods either raise the detection limit too high, and find only the largest communities, or set the detection limit too low, where most of the overlapping, large communities are already merged. This is called as the giant-component problem of the community detection methods, since by setting the detection limit too low, the modules collapse to a single giant-component (Berry et al., 2009; Fortunato, 2007; Fortunato and Barthélemy, 2007; Kumpula et al., 2007). The effect of the gradual shift in the detection limit on the development of more and more details of the modular network structure can be nicely followed in the hierarchical clustering methods. Another solution to find both small and large communities, if we simplify the network by leaving out the links below an appropriately selected arbitrary link weight threshold (Palla et al., 2005). This network simplification makes the communities more isolated, and enables to lower the detection limit to see smaller communities but leaving larger communities still more-less separated showing only a reasonably minor overlap. As we will describe in Section V.2., the PeakHill methods (the determination of hills and highlands of the community landscape based on the finding of local-maximas first) solve the giant-component problem, since they find the smaller and larger modules simultaneously.

If we modify the community landscape height values by a strictly increasing or decreasing transformation, the network modules determined by a horizontal cut of the community landscape will not change. However, in the case of PeakHill and similar methods the same strictly increasing or decreasing transformations can significantly modify the overlaps between the network modules on the community landscape. This property of the community landscape gives an another example, how a whole family of ModuLand methods can be designed around a previously applied, horizontal cut-type method, such as that of Girvan and Newman (2002; 2004) as well as Palla et al. (2005)[7].

2. PeakHill methods: hills are determined by local maxima

In this, closing step of the ModuLand method family we will first describe procedures (like the PeakHill method) to find the hill-tops/highlands of the networks as module-cores, and then we will assign all remaining links of the network

[7]In order to get the *k*-clique-communities as hills based on the CliqueLand method, we have to modify the hill definition of the ThresholdHill method. In the CliqueLand method the connected subgraph above the selected community landscape height threshold may contain more than one hills. Therefore, we define the *k*-connected subgraph of the community landscape above the threshold as the 'hill', i.e. as the module of the CliqueLand method. (In *k*-connected subgraphs all elements are reachable through a series of adjacent *k*-cliques; see Section IV.4.a.) Hills defined by this definition may have overlaps.

to these hill-tops/highlands constructing a complete procedure, where the community landscape height of the link will be fully distributed between the module-cores (now: modules) it belongs. We will describe three different methods for the assignment of links, the proportional, the gradient and the total distribution methods naming them ProportionalHill, GradientHill and TotalHill methods, respectively.

In the PeakHill methods, which finds the hill-tops/highlands of the network, hill-tops are determined by local maxima of the community landscape. Thus, the number of hill-tops/highlands will be equal to the number of network modules. Therefore a member of the PeakHill methods needs no previous assumption on the number of network modules to give a final result.

Let us naively illustrate the ProportionalHill, GradientHill, TotalHill and other module membership assignment methods as a paint-flow process on the community landscape. Let us allow paints flow down from hill-tops pouring differently colored paints on each different hill-top. With this process the whole community landscape will be painted at the end. With appropriate flow rules certain links may be painted by more than one color. These links will represent the overlapping links between the modules.

a. Finding the hill-tops and highlands of the community landscape

We give the definiton of the hill-tops (or highlands) of the community landscape as follows.

Directed networks: The centers (or cores) of modules are defined as the hill-tops of the community landscape. A hill-top is either a link, whose outbound links have a smaller community landscape height [by the outbound links of a link*(i,j)* we mean the outbound links of element *j*, the end-point of link*(i,j)*], or a strongly connected component (this means that every element is reachable from each other element by a directed walk on this component) consisting of links with equal community landscape height, whose outbound links have a smaller community landscape height.

Undirected networks: The hill-top of the community landscape contains all connected links, which have the same community landscape heights, and whose neighboring links all have lower community landscape heights than theirs.

Algorithm 4 describes the PeakHill method for finding of community landscape hill-tops or highlands in detail.

Algorithm 4: Algorithm of the PeakHill method to find the hill-tops or highlands of community landscapes

/
As a result of the algorithm, network links will be assigned to the four different categories below:

> ***'c':*** *This link is a module-core link (a hill-top of the community landscape).*
> *Please note that in the current implementation a module core consisting of multiple links still has just one link marked as the module-core link. This link the representative link for that core.*
>
> ***'h':*** *This link is part of a highland (a module-core consisting of several links).*
>
> ***'s'****: This link is a small link having at least one neighboring link, which has a higher community landscape height.*
>
> ***'e':*** *This link is "equal", having no neighbors with higher community landscape height, but at least one neighboring link with an equal community landscape height. Such a link can be a part of a highland forming a module-core, or part of a plateau. This is a temporal type, since the OldBoyWalk procedure will determine, whether such links are of type 's' or 'h'.*

Important variables used in the algorithm:

> ***maxNeighbor:*** *the community landscape height of the highest neighboring link (zero, if there is no neighboring link).*
>
> ***height(e):*** *the community landscape height of link e.*
>
> ***type(e):*** *the type of link e as described above.*

```
*/

for each link l in the network {
        if l has no neighbor { maxNeighbor := 0 }
        else { maxNeighbor := maximum of height(l') for l' in neighbors of link l }

        if height(l) = maxNeighbor{ type(l) := 'e' }
        else if height(l) < maxNeighbor{ type(l) := 's' }
        else if height(l) > maxNeighbor{ type(l) := 'c' }
}

// let OldBoyWalk resolve the type of "equal" links
for each link l in the network {
        if type(l) = 'e' { Call the OldBoyWalk procedure on link l }
}
```

Description of the OldBoyWalk procedure: the OldBoyWalk procedure behaves like a fitt, quick old man with a cradle. OldBoyWalk can walk everywhere on a field containing links with equal community landscape heights, but cannot step higher or lower than the field. After exploring the whole plateu, OldBoyWalk checks every neighbor of the plateau. If OldBoyWalk finds at least one neighboring link having a higher community landscape height than that of the plateau, sets every link of the plateau to type **'s'**, otherwise sets every link of the plateau to type **'h'**.

The PeakHill hill detection method is incorporated into the downloadable implementation of the ProportionalHill and TotalHill membership assignment methods. For downloads, examples and usage instructions please see our homepage <http://www.linkgroup.hu/modules.php>.

The directed version of the PeakHill method differs from the undirected version in two points. First, for a given directed link*(i,j)* only the links outbound from element *j* are regarded in the assignment process, because these links can be regarded as neighbors in the directed case. Second, the module cores identified by the OldBoyWalk procedure are strongly connected, that is each link of a module core can be reached from any other link of the same module core by a directed walk.

b. The ProportionalHill and the GradientHill module membership assignment methods (undirected and directed versions)

In the ProportionalHill module membership assignment method network links are assigned to modules of their non-lower neighboring links in the proportion of the absolute community landscape height of the respective neighboring links. The already mentioned module-core sets of links are assigned with full weight to the respective modules defined by themselves. In the GradientHill module membership assignment method a link is assigned to modules of the highest neighboring links of the given link.

In the start of the ProportionalHill module membership assignment method all links are marked as unassigned. After this, multiple rounds of link-assignments are performed: in all rounds, links are assigned to modules based on the assignment of previously assigned links. In each round, we descend to next slice of links, starting from the top community landscape slice. In this procedure a community landscape slice is formed by all links having the same community landscape height.

Here we describe the steps of a single round of the ProprotionalHill module membership assignment method:

- The first step: the hill-tops/highlands of the community landscape are marked, with the understanding that each of them becomes a new module-core. Each link of all these connected components are assigned to their own modules with an assignment-strength of their full community landscape height.
- In consecutive steps, unassigned links of the next descending community landscape slice, having at least one neighboring link already assigned to the growing modules, are assigned to modules proportional to the assignment-strength of their neighbors already assigned to existing modules. In such a step, links assigned in the current step are not considered as 'assigned neighbors' during the whole length of the respective step. The step described here is repeated until there are any unassigned links remained at the actual community landscape slice. Once all links of

the actual community landscape slice have been already assigned to modules, the round is over, and the next round begins, unless there are no more (lower) community landscape slices left, where the whole assignment procedure ends.

As an outcome of the ProportionalHill module membership assignment process, for each link the sum of the assignment-strength values of the given link is equal to the community landscape height of the link.[8]

The GradientHill module membership assignment method is structurewise very similar to the ProportionalHill module membership assignment method described above.[9] However, in the GradientHill module membership assignment method we do not assign the links of the given round proportionally to the community landscape heights of all their neighbors assigned previously, but only to those neighbors, which have the maximal community landscape height among all neighbors. Obviously, this assignment procedure may also give an assignment of a new link to multiple network modules (if the new link has more than one neighbors having an equally maximal community landscape height), but produces much smaller modular overlaps than the ProportionalHill module membership assignment method. We suggest to apply the GradientHill module membership assignment method, if large overlaps between various modules are not interesting, or are not feasible.

Algorithm 5 below describes the ProportionalHill and GradientHill module membership assignment methods for link assignment in detail.

Algorithm 5: Algorithm of the ProportionalHill and GradientHill module membership assignment methods

```
/*
Important variables used in the algorithm:
        tmpEqualList: Contains all links of the actual slice. (A slice of the community landscape is a set of
        links, where all links have the same community landscape height.)
        actualLinkList: Contains links set to be assigned to a module.
        markedList: Links from which the module membership of the current link is calculated.
        height:(e): Community landscape height of the link e.
        neighLinks(e): List of neighboring links of link e.
        moduleMembership[e][m]=x: link e belongs to module m with a strength of x.
        In the beginning all moduleMembership[… ][… ] values are set = 0, except module core links,
        whose module membership for the respective module is set to the height of the link.
*/

cycle on the slices in the descending order of the community landscape height of slices {

        put every link of the given slice to tmpEqualList
        while length of tmpEqualList > 0 {
                clear actualLinkList
                for each link e of tmpEqualList {
                        if e is not yet assigned to any modules and e has at least one assigned neighbor {
                                add e to actualLinkList
                        }
                }
```

[8]We may set the summation of assignment-strengths equal the weight of the starting, module-core link, by multiplying the assignment-strengths by the weight of the starting link and divide by the community landscape height. Henceforth we use only the assignment-strengths fulfilling the requirement that the sum of these assignment-strengths equals the community landscape height of a link. (Please note that in the current implementation a module core consisting of multiple links still has just one link marked as the module-core link – this link the representative link for that core.)
[9]Therefore, the GradientHill method is not found as a separate method in the algorithm package.

```
                // module membership assignment
                for each link e of actualLinkList {
                        // module membership assignment
                        clear markedList
                        if using the Proportional method {
                                for e' in neighLinks(e) {
                                        if height(e') >= height(e) { add e' to markedList }
                                }
                        }
                        else if using the Gradient method {
                                maxh := maximum of height(e') for e' in neighLinks(e)
                                for e' in neighLinks(e) {
                                        if height(e') = maxh { add e' to markedList }
                                }
                        }

                        sum_nHeight := sum of height(e') for e' in markedList
                        for m :=1..number of modules {
                                for e' in markedList {
                                        delta := moduleMembership[e'][m] / sum_nHeight * height(e)
                                        increment(moduleMembership[e][m], delta)
                                }
                        }
                }
                remove all links of the actualLinkList from tmpEqualList
        }
}
```

The runtime complexity of the ProportionalHill and GradientHill module membership assignment methods is O(edm), where e is the number of links of the network, d is the average degree of nodes and m is the number of modules. Assuming practically that the average degree is bounded by a constant and that the number of modules is not more than the number of nodes, the runtime complexity is $O(n^3)$.

For downloading the ModuLand program package including the ProportionalHill module membership assignment method see our homepage: <http://www.linkgroup.hu/modules.php>.

The ProportionalHill and GradientHill module membership assignment methods for directed networks. The basic idea of the the ProportionalHill and GradientHill module membership assignment methods was to assign a link to the modules of the neighboring (and in terms of community landscape height, non-lower) links of the given link. The directed and undirected cases have the single difference that in case of a given directed link*(i,j)* only the links outbound from element *j* are regarded in the assignment process, because these links can be regarded as neighbors in the directed case.

c. The TotalHill module membership assignment method for undirected networks

In the TotalHill module membership assignment method the assignment of module-cores is performed as described previously for the ProportionalHill module membership assignment method, but when assigning a non-core link to modules of the neighboring links in proportion of the community landscape height of the neighboring links, the neighboring links of both non-lower and lower community landscape height are regarded. This module membership assignment method is especially important, since it yields the most detailed information of the network module structure.

From now on we use the following notations:

s_{ij}: the original weight of the link*(i,j)*
m_{ij}: community landscape height of the given link*(i,j)*
$b_{ij}[k]$: is defined only for not module-core links. $b_{ij}[k]$ is the module assignment strength of link*(i,j)* to the *k*-th module giving us the value, that how strongly link*(i,j)* belongs to the *k*-th module.
$a_{ij}[k]$: is the module assignment strength of link*(i,j)* to the *k*-th module, if the link*(i,j)* is the core of the *k*-th module. $a_{ij}[k]$ is the community landscape height of the given link*(i,j)*, if the link*(i,j)* is the core of the *k*-th module, otherwise $a_{ij}[k]$ is the zero vector. This vector does not change during the procedure. For module-core links b_{ij} does not exist.
$d_{ij}[k]$: the module assignment strength of link*(i,j)* to the *k*-th module: for a module-core link*(i,j)*, d_{ij} equals a_{ij}, and for non-core links, d_{ij} equals b_{ij}.

In the following steps we will get an equation system with the number of variables equal to the non-core links of the network. Note that the b_{ij} variables are not scalar, but vector values with the vector length equal to the number of modules. However, the components are independent of each other, so the equation system can be decomposed into equation systems, which have scalar-variables, and can be solved independently. Therefore for the sake of simplicity we write simply b_{ij} instead of $b_{ij}[k]$ in the equation.

Unlike the directed case, in the undirected case each quantity is symmetric for index swapping, so $b_{ij}=b_{ji}$. For non-core links:

$$b_{ij} = m_{ij}\frac{\sum_{k,k\neq j} d_{ki} + \sum_{l,l\neq i} d_{lj}}{\sum_{k,k\neq j} m_{ki} + \sum_{l,l\neq i} m_{lj}}$$

Module-core links are exceptions, these links are assigned to the respective modules with their full height. If the link*(i,j)* is the core of module *k*, then $a_{ij}[k] = m_{ij}$, while other components of a_{ij} are zero. For non-core links a_{ij} is the zero vector. In the current implementation, the community landscape height of a link is distributed between modules, and an element is assigned to modules with a quantity equal to the sum of the assignment quantity (now community landscape height) of the links of the given element. Let *A* be the set of module-core links, *B* the set of non-core links of non-zero community landscape height. The main point here is, that altough the equation system has the same form for all modules, the *A* and *B* sets depend on the choosen module.

Now we define the module assignment strength vectors of elements based on the module assignment strength of links. Let we use the

$$a_i \equiv \sum_{k,(i,k)\in A} a_{ik}$$
$$b_i \equiv \sum_{k,(i,k)\in B} b_{ik}$$
$$d_i \equiv a_i + b_i = \sum_k d_{ik}.$$

notations.

Now we give an alternative form of the described equation system, where variables are assigned to elements instead of links. After solving the equation system, the links can be assigned to modules based on the assignment of the elements.

$$b_{ij} = m_{ij}\frac{\sum_{k,k\neq j} d_{ki} + \sum_{l,l\neq i} d_{lj}}{\sum_{k,k\neq j} m_{ki} + \sum_{l,l\neq i} m_{lj}} = \\ = \frac{m_{ij}}{m_i + m_j - 2m_{ij}}(d_i + d_j - 2b_{ij})$$

The module membership assignment equations can only be written for non-core links. We can rewrite the equation so that the module membership assignment of links can be easily calculated, given that the module membership assignment of elements is known:

$$b_{ij} = m_{ij}\frac{d_i + d_j}{m_i + m_j} = r_{ij}\,(d_i + d_j)$$

where we introduced $r_{ij} = \frac{m_{ij}}{m_i+m_j}$. With

$$b_i = \sum_{j,(i,j)\in B} b_{ij} = \sum_{j,(i,j)\in B} r_{ij}\,(d_i + d_j)$$

we give a form of the equation system, where variables are assigned to elements. Now b_i can be written as

$$b_i = \eta_i + \sum_{j,(i,j)\in B} \rho_{ij} b_j,$$

$$b = (I - \rho)^{-1}\eta,$$

$$\eta_i = \frac{r_i a_i}{1 - r_i} + \frac{1}{1 - r_i} \sum_{j,(i,j)\in B} \frac{m_{ij} a_j}{m_i + m_j}$$

$$\rho_{ij} = \frac{m_{ij}}{(m_i + m_j)(1 - r_i)}$$

.

d. The TotalHill module membership assignment method for directed networks

In the directed case, besides assigning an element *i* to modules, that is, determining how strongly element *i* is connected to the modules (measured by the module assignment strength of element *i* noted by the symbol, d_{i*}) another quantity (the 'module influence strength' of element *i* noted by the symbol, d_{*i}) can also be determined, namely that how strongly the modules connect to element *i*.

If a link*(i,j)* is a module-core link of *k*-th module, then the module assignment strength of link*(i,j)* to the *k*-th module is $a_{ij}[k] = m_{ij}$ (i.e. the community landscape height), while other components of a_{ij} are zero. For non-core links, a_{ij} is the zero vector. For a non-core link*(i,j)* the module assignment strength of the link, b*(i,j)* is

$$b_{ij} = m_{ij} \frac{\sum_k d_{jk}}{\sum_k m_{jk}}$$

where d_{jk} is the module assignment strength of link*(i,j)* to the modules: for a module-core link*(i,j)*, d_{ij} equals a_{ij}, and for non-core links, d_{ij} equals b_{ij}. That is, the given link is assigned to modules based on the module membership assignment of the neighboring outbound links of the given link, in proportion of the community landscape height of the respective links. The module membership assignment strength of element *i*, d_{i*} is calculated by summing the module membership assignment-strengths of the outbound links of element *i*: $d_{i*} \equiv b_{i*} + a_{i*} \equiv \sum_j d_{ij}$. Respectively, the module influence strength of element *i*, d_{*i} is calculated by summing the module membership assignment-strengths of the inbound links of element *i*: $d_{*i} \equiv b_{*i} + a_{*i} \equiv \sum_j d_{ji}$.

This way we get an equation system with the number of variables equal to the non-core links of the network. Note that the variables are not scalar, but vector values with the vector length of the number of modules. However, the components are independent of each other, so the equation system can be decomposed into equation systems, which have scalar-variables, and can be solved independently.

Based on the module membership assignment rule of elements and the equation system assigning links to modules, we give an alternative form of the equation system, where variables are assigned to elements instead of links. Once the module membership assignment of elements is calculated, links can also be assigned to modules. If *(i,j)* $\in B$, then

$$b_{ij} = \frac{m_{ij}}{m_{j*}} (b_{j*} + a_{j*})$$

Given the module membership assignment of elements, the module membership assignment of links is easily calculated. Introducing

$$m_{i*} = \sum_{j,(i,j)\in B} m_{ij}$$

$$m_{*i} = \sum_{j,(j,i)\in B} m_{ji}$$

we can write, that

$$b_{i*} = \sum_{j,(i,j)\in B} b_{ij} = \sum_j \frac{m_{ij}}{m_{j*}} (b_{j*} + a_{j*})$$

$$b_{*i} = \sum_{j,(j,i)\in B} b_{ji} = \frac{m_{*i}}{m_{i*}} (b_{i*} + a_{i*}) .$$

The sought module membership assignment-strengths are

$$d_{i*} = a_{i*} + \sum_{j,(j,i)\in B} \frac{m_{ij}}{m_{j*}} (b_{j*} + a_{j*})$$

$$d_{*i} = a_{*i} + \frac{m_{*i}}{m_{i*}} (b_{i*} + a_{i*})$$

This way we rewrote the equation system in an alternative form assigning variables to elements.

e. The optimized version of the undirected TotalHill module membership assignment method

Algorithm 6 of the optimized version consists of two procedures: an equation system solving function and the TotalHill membership assignment method itself. We have chosen the Gauss-Seidel iterative equation solver algorithm, which does not have a run-time guarantee but in practice finds a solution with a low error extremely fast.

Algorithm 6: Optimized algorithm for the undirected TotalHill module membership assignment method

```
Procedure GaussSeidel ( A, b, x, maxError ) {
        x:=b
        error := maxError
        while error >= maxError {
                error := 0
                for i:=1..N {
                        sum := b[i]
                        for j:=1..N {
                                if i != j { sum :=sum-A[i,j] * x[j] }
                        }
                        sum:=sum /A[i,j]
                        error := error + |x[i] - sum|
                        x[i] := sum
                }
        }
}

Procedure TotalHill ( G(V,E), maxError ) {
        // get the sum height of all elements of the network
        SumHeight : Array[N]
        for i:=1..N { SumHeight[i] := 0 }
        for every e_ij link { SumHeight[i] := SumHeight[i]+ height(e) }

        // make the NxN matrix, representing the linear equation system
        // (in practice it could be more optimal to use a sparse matrix representation)
        A : Matrix [NxN]
        for i:=1..N {
                lineSum := 0
                for j:=1..N { A[i,j] := 0 }
                for every e_ij link {
                        If notModuleCenter(e) {
                                prop := (SumHeight(i) + SumHeight(j)) / height(e)
                                lineSum := lineSum + prop
                                A[i,j] := prop
                        }
                }
                A[i,i] := lineSum-1
        }

        LinkBelong : Matrix[NumberOfModules, NumberOfLinks]

        // For every module-core link we set and solve the equation system (Ax=b)
        // Please note that in the current implementation a module core consisting of multiple links still
```

```
        // has just one link marked as the module-core link making this link is the representative link for that
        // core. Thereby there are as many module-core links as many modules.

        for every e_ij where e_ij is a module-core link {
                m:=the index of module, which has e_ij as module-core

                // set the right column vector of of the equation system
                b : Array[N]
                for k:=1..N { b[k]:=0 }
                b[i] := - height(e)

                // solve the system
                GaussSeidel( A, b, x, maxError)

                // by the result (x vector) – the module membership of nodes – we calculate the
                // module membership of links

                for every l_ab, where l_ab is a non-module-core link {
                        sumHeights := SumHeight[a] + SumHeight[b]
                        if sumHeights > 0 {
                                LinkBelong[m,l] := (x[a] + x[b]) * height(l) / sumHeights
                        } else {
                                LinkBelong[m,l] := 0
                        }
                }

                // only the e module-core link belongs to the m module,
                // the other module-core links not
                for every l_ab, where l_ab is a module-core link { LinkBelong[m,l]:=0 }
                LinkBelong[m,e] := height(e)
        }
        return LinkBelong
}
```

The runtime complexity of the TotalHill module membership assignment methods is $O(n^2 + mg)$, where n is the number of nodes, m is the number of modules and g is the runtime complexity of solving a linear equation system of n variables and n equations. Using a trivial Gauss-elimination, $O(g)$ equals $O(n^3)$, and assuming practically that the number of modules is not more than the number of nodes, the theoretical runtime complexity of the TotalHill method is $O(n^4)$. However, we would like to note that the Gauss-Seidel linear equation system solver is very fast in practice, and that the computations can be easily parallelized regarding the number of modules. The linear equations could also be solved by a sparse LU decomposition-based methods (Duff et al., 1986).

For downloading the ModuLand program package including the TotalHill module membership assignment method see our homepage: <http://www.linkgroup.hu/modules.php>.

3. Other definitions of hills on a community landscape (the SameHill and NumberHill methods)

In this section we introduce two hill definitions, which use a different perspective than those introduced previously.

SameHill: Inspired by the Potts model and spectral modularization methods (see Table S2), we introduce the SameHill hill definition. In this method values are assigned to elements or links, and modules are defined as the parts of connected components with 'similar' values. For example, in the simplest spectral case elements with negative values constitute one module, and positive elements form the other. Hills of the community landscape of these methods are the links or elements of connected components in the neighborhood of a hill-top, whose community landscape height is not

lower than *p*% of the community landscape height of the respective hill-top. If hills of the community landscape are flat enough, with this method we get a fast, non-overlapping modularization for most of the elements.

NumberHill: Let us chose a hill-top not-yet assigned (on the community landscape over the elements or links), and assign it to a new module totally. Then let us continue adding the neighboring elements or links of the elements or links already in this new module with the highest community landscape height, until the module reaches a predetermined size (for example a given percent of the *N* number of elements). This method can result overlapping modules by adding an element to more than one hill-tops during the process. In general there will be unassigned elements too, because of the fixed size of the modules.

Both of the SameHill and NumberHill methods are fast, and may yield overlapping links or elements. Naturally, several other hill definitions are possible, and their applicability is determined by the properties of the community landscape.

4. Finding the elements of the overlapping communities

After assigning all links of the network to various modules by any of the procedures described in Sections V.1-3. the assignment of the elements may use a relatively simple method, called detailed module membership assignment. An element is assigned to a given module to the extent its links belong to that module. Elements having no links are obviously not cores of any conventional modules. However, for other purposes, such as the visualization of the network, it may be useful to assign them as one-element modules of the network. From the element module membership assignment procedures we tried, the above, detailed module membership assignment procedure explores and utilizes the richest information on the network community structure. However, for some applications the detailed element module membership assignment might be too complex. In these cases we may assign the given element to that module, where its links belong the most.

The PeakHill network module determination methods described in Section V.2. greatly differ from the horizontal cuts resembling to most of the previously designed network module determination methods of Table S2 described in Section V.1., since

1.) it assigns all links and elements of the network to network modules;
2.) automatically determines the total number of modules;
3.) avoids the giant-component problem giving a detail-rich picture on the community structure of the network to the required extent.

5. Summary of the hill finder methods

Obviously, besides the module membership assignment methods we described above, a large number of assignment procedures can also be designed. Such a module membership assignment method can be a variant of the ProportionalHill method, where the module membership assignment is performed proportionally to the community landscape height-difference between the actual link and its already assigned neighbors, not to the absolute height of the neighbors described above. The ProportionalHill or TotalHill module membership assignment methods can use non-linear (e.g. exponential, or other) functions for the assignment of links to the modules of their previously assigned neighbors. We are inviting our colleagues to explore this rich field of possibilities.

6. Metrics based on the ModuLand modularization methods

a. Effective size of support

The effective size of support (ESS) measure of a random variable was defined by Grendar (2006). For a better understanding of what the effective size of support means, let us see first the usual definiton of the size of the support of a random variable. Let *X* be a discrete random variable with probability mass function *p* consisting of *m* elements. The definiton for the support of *X* is the set of those $x \in X$ whose weight (probability) is non-zero, that is

$$A(p(X)) = \{x \in X : p_x > 0\}$$

and $|A(p(X))|$, the size of the support equals the number of the members of this set.

While the size of support for random variables of the probability mass function $p = [0.5, 0.5]$ and $q = [0.99, 0.01]$ both equal two, the properties of the two random variables differ significantly. While the former random variable takes both possible values with equal probability, the latter takes only one of the two possible values most of the time. To distinguish between such differences, it would be useful to devise a measure $S \in [1;m]$, where the value of this measure describes the random variable from the introduced point of view, i.e. that the number of the members is not the discrete

number of members occurring with a non-zero probability, but it is expressed as a continuous measure taking into account their probability. This new measure is the effective size of support *S(p(X))* or *S(X)*.

ESS should have certain properties, dictated by common sense (Grendar, 2006):

P1) *S(p)* should be continuous, symmetric function (i.e., invariant under exchange of p_i and p_j, $i,j = 1... m$).

P2) $S(\delta_m) = 1 \leq S(p_m) \leq S(u_m) = m$, where u_m denotes the uniform probability mass function on m-element support, δ_m denotes an m-element probability mass function with probability concentrated at one point, p_m denotes a probability mass function with $|A(p)| = m$.

P3) $S([p_m, 0]) = S(p_m)$.

P4*) $S(p(X, Y)) \leq S(p(X))S(p(Y))$, and equal if and only if X and Y are independent random variables. This means that the effective number of states of the joint system (X,Y) cannot be greater than the number resulting from multiplying the effective number of states for the single systems X and Y.

Grendar concludes that an $S(p(X))$ conforming to these properties can be written as $exp(H(p(X)))$, where $H(p(X)) = -\sum_{i=1...m} (p_i \ln p_i)$ is the Shannon-entropy.

So we can define the effective size of support as $S(p(X)) = exp(H(p(X)))$. Earlier the term 'perplexity' has also been defined (Jelinek et al., 1977; Bahl et al., 1983) for this quantity.

b. Effective number

The effective size of support measure described above can be used for the measurement of the effective number of members belonging to any group (e.g. network community or module). The 'effective number of members' from now on will be called shortly as *effective number*. Here 'effective' means that the number of the members is not counted in a discrete way, but it is expressed as a continuous measure taking into account the weight of the members, by giving larger weight to the important elements. With other words, the effective number shows us the number of important or frequent elements in the set. As an example a member with a relatively small weight does not count as one full member, just as a small fraction of a full member. We introduce the effective number as follows.

$$n_e\{V[i]\} = \exp\left(-\sum_i p_i \log p_i\right)$$

The equation above defines the effective number of the members of set V, where $V[i]$ is the weight of member i, and

$$p_i = \frac{V[i]}{\sum_j V[j]}.$$

While counting the number of significant members in the traditional, discrete way introduces a threshold, and discards members of relatively insignificant weight, counting the effective number of members does not require any threshold, and summarizes the total information available. Moreover, the effective number is a continuous function of the weights of the members. See Lee et al. (2009a) for an example of the application of the effective number for network degrees.

c. Modular overlap

Let us assume that the module membership assignment-strength of element *i* to module *j* ($j = 1..m$) is $d_i[j]$. The modular overlap of element *i* is the effective number of modules that element *i* is assigned to:

$$O[i] := n_e\{d_i[a]\}.$$

d. Bridgeness

Graph theory traditionally refers to a link as a *bridge*, whose removal would increase the number of components of the graph. In sociology, a *bridge node* is an element connecting two otherwise distinct groups (Burt, 1995; Nepusz et al., 2008; see Suppl. Discussion). We introduce the *bridgeness* measure of an element or link as the overlap of the given element or link between two or more modules relative to the overlap of the other elements or links.

$$B[a][b][i] := \frac{T[a][b][i]}{\sum_j T[a][b][j]},$$

is the bridgeness of element *i* between modules a and b, where

$$T[a][b][i] := \min\{d_i[a], d_i[b]\}$$

is the area-overlap, or common area of element between modules a and b.

The total bridgeness of element *i* describes the bridgeness of that element between all modules:

$$B[i] := \sum_{a=1}^{m} \sum_{b \neq a, b=1}^{m} B[a][b][i].$$

e. Similarity of the elements

The similarity of the elements *i* and *j* is based on their module membership vectors, d_i and d_j:

$$Sim(d_i, d_j) = \sum_{k} \min \left(\frac{d_i[k]}{\sum_t d_i[t]}; \frac{d_j[k]}{\sum_t d_j[t]} \right)$$

The value of $Sim(d_i, d_j)$ is in the $[0; 1]$ interval, and it is maximal, if the d_i and d_j module membership vectors are exactly the same.

VI. Further opportunities and analysis of the ModuLand method

1. Merging the overlapping communities

The PeakHill method (as opposed to many other network module determination methods) does not require a pre-set value for the maximal number of modules expected. This may result in a rather large number of modules in networks with a complex community structure yielding a 'rough' community landscape with many local maxima. However, modules of such networks with rough community landscapes can be merged. This also allows the tailoring of the final module-structure to the needs of the investigator. Large overlaps, small differences in both the distance and community landscape height of modular hill-tops/highlands are all appropriate reasons for merging of two neighboring modules. In many practical applications merging the adjacent modules can be very useful (such as the reorganization of overlapping firm departments, or computer program subroutines, to name but a few). While in Section VII. we will describe a hierarchical representation of the community structure, which merges the appropriate modules based on the detailed topological information, here we introduce a simple yet effective post-processing step for merging artificial modules:

Our post-processing step relies on the fact that the nodes of the network can be ordered based on their module membership assignment strength to any given module. Therefore we define the correlation of two modules as the Spearman's rank-order correlation of the corresponding orderings of the nodes of the network. After calculating the correlation of all module pairs, groups of highly correlated modules, where the correlation is above a chosen threshold, are merged into a single module. The module membership strength of the nodes of the network to a given resulting module equals the sum module membership strength to the corresponding merged modules. In Figures S13c and S13d we show the effectiveness of this post-processing step and also investigate the effect of choice of threshold. Generally the correlation threshold for merging modules may be chosen by inspecting the histogram of module-pair correlations and setting the threshold to merge the modules of extremely high correlation, but throughout this paper we used an arbitrary chosen correlation threshold of 0.9 for the sake of simplicity.

We note that applying the TotalHill module membership assignment method yields massively overlapping modules unsuitable for merging modules based on this threshold, therefore in this case we merge the modules based on the ProportionalHill module membership assigment method-based module information.

2. Robustness of the ModuLand method for the measurement, sampling and computational errors

Measurement and sampling errors obviously compromise any community detection method (Massen and Doye, 2006; Karrer et al., 2008; Yip and Horvath, 2007; Lusseau et al., 2008). If these errors are beyond the tolerance level set by the purpose of the study and/or the habits of the investigator, specially designed versions of the ModuLand method family may take this problem into consideration. If we have an estimate of the typical measurement error we may check the effect of this error by discretizing the link-weight values rounding them to the next available value of a scale in the range of the expected error and measure the differences in the outcome of the modular structure. Alternatively, both the measurement errors and the community heap-finding errors can be minimized by discretization of the community landscape heights to a scale comparable with the expected errors and examine the differences it caused in the modular structure. If none of these solutions works properly, it may happen that due to the sensitivity of certain heap construction methods the cumulative errors will occasionally show little, artificial hill-tops on a side of a hill of the community landscape. This yields to a number of unreal modules in the case of the PeakHill module assignment method. If experiencing this trouble, the merging procedures of Section VI.1. or specific ways of error reduction can be very helpful. However, in most cases we can neglect these artificial hill-tops, since at a higher level of the hierarchical community structure, these new, artificial communities will be often merged to real communities, and will not disturb the community structure any longer.

In our work we applied the benchmark graph generation method published by Lancichinetti et al. (2008), which produces non-overlapping modules, in order to check the correspondence of our highly overlapping modules with the surely known partitions of the benchmarks. The comparison of our results with the recently published updated benchmark graph generation method by Lancichinetti et al. (2009a), which also supports overlapping modules, is an interesting and challenging problem of its own, and would require a study which is beyond the scope of our current paper.

In Figures S13a and S13b we show that the identified modules correspond consistently to the modules of the benchmark graph of Lancichinetti et al. (2008) over a range of parameter settings, where modules can be defined in the strong

sense. Strong sense means here, at least the half of the neighboring nodes are assigned to the same module as the given node, see. Lancichinetti et al. (2008).

VII. Construction of the higher hierarchical level representations of the network

The ModuLand method family offers the way to create higher levels of hierarchical representation of the original network. Here the elements of the higher level correspond to the modules of the original network, and the links of the higher level correspond to the overlaps between the respective modules (Figure 1D and Figure S6). This hierarchical representation can be established recursively in several steps, until the whole original network becomes represented by one or more single elements without interaction between them. The higher levels of hierarchical representation are very useful, since several versions of the ModuLand method family, like the PerturLand method may yield a rather detailed community structure at the first hierarchical level of modules, which gives an extremely rich information on the communities causing an 'overflow' of normal human cognitive capacities. In such cases (which often occurs at larger networks having more than 10,000 elements) the complexity of the community structure can be easily reduced to the comprehension limit of a few modules examining a higher hierarchical level. Moreover, the occasional artificial hill-tops and modules, mentioned in Section VI.1., will also be eliminated at higher hierarchical levels. In the following Sections we describe a detailed assignment procedure of these hierarchical representation layers allowing a fast, zoom-in type analysis of large networks.

1. The strength of the link between two modules

Let $s^{(2)}_{ab}$ denote the strength of the directed or undirected link between module *a* and module *b* (thus the strength of a link at the second hierarchical level of the modular representation noted by the upper index), which will be defined for both cases in this section. We will see, that $s^{(2)}_{ab}$ has usually a nonzero value even in case if *a*=*b*, meaning that loop links may occur on higher network levels. These loop links will be discarded, since they do not give information about inter-element connections on the higher level.

In the followings $d_i[j]$ ($d_{i*}[j]$ in directed networks) will denote the strength of element *i* belonging to module *j*, and m_i will denote the centrality of element *i*.

In directed networks the activity (see Section III.2.) of a module (where the module is regarded as an element of the second – or any higher – hierarchical level), $p_a{}^{(2)}$ is given by the sum activity of the elements belonging to the given module, weighted by the module assignment-strength of those elements. This can be expressed by the formula $p_a^{(2)} = \sum_i p_i d_{i*}[a]$. If we set the activity of the elements all equal to 1 (as it is in case of the test-networks used in this paper) having the assumption of $p_i \equiv 1 \quad \forall i$, the formula can be rewritten as $p_a^{(2)} = \sum_i d_{i*}[a]$. In directed networks let the $s^{(2)}_{ab}$ strength of link*(a,b)* be the maximum of the overlap of the modules *a* and *b* over the elements, divided by the activity of module *a*, that is:

$$s_{ab}^{(2)} := \frac{\max_i\{T[a][b][i]\}}{p_a},$$

where *T[a][b][i]* is the overlap ('common area') of element *i* between module *a* and *b*:

$$T[a][b][i] := \min\{d_i[a], d_i[b]\}.$$

The overlap of an element between two modules increases linearly proportional to its assignment-strength to each of the two given modules. Now, it is clear, that $s^{(2)}_{ab}$ scales linearly as the function of the community landscape height of the *i* element.

In undirected networks, we may use other definitions according to the model of the interactions used in the community heap construction procedure. We now present a simple definition, in which the overlap between two modules is summarized over all elements. For the node *i* let the overlap be propotional to its assignment-strength to both modules. The expression $d_i[a]d_i[b]$ scales as quadratic instead of linear in the function of the community landscape height of the element, therefore, the desired strength between modules *a* and *b* can be defined as

$$s_{ab}^{(2)} := \sum_i \frac{2d_i[a]d_i[b]}{m_i}.$$

The constant multiplier of 2 in the formula is a consequence of our convention, regarding an undirected link to two directed links with opposite directions.

In the undirected case, the activity of a module (where the module is regarded as an element of the second – or any higher – hierarchical level) is also given by the sum of the activities of the elements belonging to that module, weighted by the module assignment-strength to that module, so $p_a^{(2)} = \sum_i p_i d_i[a]$. While $p_i \equiv 1 \quad \forall i$ for the networks considered

in our current study, the above formula can be rewritten as $p_a^{(2)} = \sum_i d_i[a$. We note that in the undirected case, $s^{(2)}{}_{ab}$ naturally equals $s^{(2)}{}_{ba}$.

It is important to note, that the strength of the link between two modules already comprises the effect of direct and indirect interactions between the elements of the original network, too. In other words this means that the higher level link strengths contain already the indirect impact between the elements of the lower level network.

2. The modularization of a higher hierarchical level

The structure of the next hierarchical level of the original network reflects the essential structure of the original network, making it easier to visualize, overview, describe and understand the original network. If the network of the next hierarchical level is still too complex, the construction of even higher hierarchical levels may be required. This is done by finding the modules of the representation of the network at the next hierarchical level. This process can be continued until the resulting network has no links and colaesces to individual single elements, which is – obviously – the top level network. The number of top level elements depends on the applied community heap construction method, among others. For example applying the PerturLand or LinkLand method yields a number of top level elements equal to the connected components in the original network, while in the case of the NodeLand method more top level elements than the number of original connected components may appear as a final result.

Technically it is important to remark, that each element without links of the network is defined as a separate module. According to this definition such an element makes a module containing one single element on every higher hierarchical levels of the network.

The link strengths between the modules can be directly used as community landscape heights, without any heap contruction method, due to the fact, that the higher level link strength already takes into account the indirect interactions between the original elements. But generally, one may be interested in other type of interections between the modules, than between the original elements. For example the interaction between groups of people (e.g. cities or countries) may be of quite different nature than the interactions between the constituent individuals. In this case one should apply a heap construction method selected and/or adjusted according to the type of the interaction and re-run a modularization protocol at the second (or higher) level of the hierarchical structure in order to get the third (or, generally 'higher+1') level community landscape.

3. Projection of the modules of higher hierarchical levels

Not only the elements of the (*n-1*)th hierarchical level can be assigned to the elements of the (*n*)th level as shown in the previous sections, but also elements of any lower, (*n-2*)th, etc. hierarchical levels can be directly assigned to the elements of the (*n*)th level. For example elements of the (*n-2)*th level can be assigned to the elements of the (*n*)th level (distributing the assignment-strengths of the (*n-2)*th level elements to the (*n-1)*th level elements between the (*n*)th level elements). Since in practice the importance of this direct assignment procedure is used 'in reverse', that is to show how many elements of e.g. the original network are beloning to very high hiearchical representation of this network having only a few modules already, we call this process as the 'projection' of the (*n)*th hierarchical level to the (*n-2)*th (or any lower) hierarchical level.

The following steps have to be taken to calculate the assignment-strength of element *i* of the (*n-2)*th hierarchical level to element *j* of the (*n*)th hierarchical level: For any element *k* of the (*n-1)*th hierarchical level, let us multiply the assignment-strength of element *i* to element *k* (note that element *k* is not on the same hierarchical level as element *i*, but at the hierarchical level of element *i* element *k* is actually one of the modules, where element *i* belongs) with the proportion with which element *k* is assigned to element *j*. The sum of the results of these multiplications gives the module assignment-strength of element *i* to element *j*.

To achieve an easy generalization of this procedure let us introduce the $M^{(n+1)}{}_{(n)}$ matrix, where $M^{(n+1)}{}_{(n)}[i,j]$ is the proportion with which the element *i* of the (*n*)th hierarchical level is assigned to module *j* of the same hierarchical level (or, to the element *j* of the (*n+1*)th hierarchical level):

$$M_{(n)}^{(n+1)}[i][j] \equiv \frac{d_i^{(n)}[j]}{m_i^{(n)}},$$
$$m_i^{(n)} \equiv \sum_j d_i^{(n)}[j]$$

Now any $M^{(a+k)}_{(a)}$ *(k>1)* matrix can be derived, which project the module structure of a higher hierarchical level to the elements of a lower hierarchical level, using

$$M^{(a+k)}_{(a)} := M^{(a+1)}_{(a)} M^{(a+2)}_{(a+1)} (...) M^{(a+k)}_{(a+k-1)}$$

that is, multiplying the module assignment matrices of consecutive levels. Naturally, the number of columns of any left-side matrix (the number of modules) by definition equals the number of rows of the right-side matrix (elements of the higher hierarchical level). The assignment of element *i* of the (*a*)th level to the higher (*a+k)*th hierarchical level, element *j* is given by the equation $d_i[j] = M^{(a+k)}_{(a)}[i][j] m^{(a)}_i$.

Applying the projection described here, we can not only analyze elements of the previous hierarchical level from the viewpoint of a given hierarchical level, but also elements of any previous hierarchical levels, including the original network. This gives rise to an exquisite possibility of an easy analysis, visualization and comprehension of extremely large original systems, such as telecommunication networks, neuronal cells, the human proteome, to name only a few.

A simple case illustrating this scenario can be seen on Figure S6 showing the hierarchical levels of the network science collaboration network (Newman, 2006a). The upper part of the figure shows the different hierarchical levels of the network. The bottom part of the figure shows the community structure of the higher hierarchical levels projected to the original network. The top hierarchical level would consist of separate elements. It can be seen that the projection of a higher and higher hierarchical levels generally yields larger and larger modules, as expected.

VIII. ModuLand-based network visualization methods

The hierarchical network representation can be used for a fast visualization of extremely large networks, where conventional network visualization methods became too slow to apply. A visualization method based on the ModuLand method starts with the modularization of the network as described in this paper and treats the elements of the top-network not only as a representation of the respective community of the bottom-network as described in Section VII., but also as the center of the community of the bottom-network in the 2D space. The ModuLand method family gives numerical values to characterize the contribution of network elements to each network community, and assigns community landscape heights to each link (please note that from the community landscape heights of the links the community landscape heights of the elements can be derived). This data can be used to calculate the relative position of the elements of the bottom-network from the community-centers. The ModuLand-based visualization protocol can be combined with other, existing network visualization methods. It is important to note that the extra time to run the ModuLand program before the visualization is not a disadvantage in case of extremely large networks, since many visualization tools run slower than the expected polynomial complexity, $O(n^3)$ of the ModuLand programs (Walshaw, 2003), and often give overpacked final results, which are difficult to comprehend. As a further advantage of the hierarchical visualization based on the ModuLand method the user may zoom-in to deeper and deeper layers of the hierarchical complexity of the modular structure described in Section VII. just in the section of interest in a large network.

Supplementary Tables

Table S1. Definitions of network modules

Name	Definition	References
Definitions based on 'local' topology[a-d]		
Clique	a complete subgraph of size *k*, where complete means that any two of the *k* elements are connected with each other	Luce and Perry, 1949; Wasserman and Faust, 1994
***k*-clan**	a maximal connected subgraph having a subgraph-diameter ≤ *k*, where the subgraph-diameter is the maximal number of links amongst the shortest paths *inside* the subgraph connecting any two elements of the subgraph	Alba, 1973; Mokken, 1979; Wasserman and Faust, 1994
***k*-club**	a connected subgraph, where the distance between elements of the subgraph ≤ *k*, and where no further elements can be added that have a distance ≤ *k* from all the existing elements of the subgraph	Alba, 1973; Mokken, 1979; Wasserman and Faust, 1994
***k*-clique**	a maximal connected subgraph having a diameter ≤ *k*, where the diameter is the maximal number of links amongst the shortest paths (including those *outside* the subgraph), which connect any two elements of the subgraph	Luce, 1950; Alba, 1973; Mokken, 1979; Wasserman and Faust, 1994
***k*-clique community**	a union of all cliques with *k* elements that can be reached from each other through a series of adjacent cliques with *k* elements, where two adjacent cliques with *k* elements share *k*-1 elements *(please note that in this definition the term k-clique is also often used, which means a clique with k elements, and not the k-clique as defined in this set of definitions; the definition may be extended to include variable overlap between cliques)*	Alba and Moore, 1978; Palla et al., 2005; Zubcsek et al., 2008
***k*-component**	a maximal connected subgraph, where all possible partitions of the subgraph must cut at least *k* edges	Matula, 1972
***k*-plex**	a maximal connected subgraph, where each of the *n* elements of the subgraph is linked to at least *n*-*k* other elements in the same subgraph	Seidman and Foster, 1978; Wasserman and Faust, 1994
strong LS-set	a maximal connected subgraph, where each subset of elements of the subgraph (including the individual elements themselves) have more connections with other elements of the subgraph than with elements outside the subgraph	Wasserman and Faust, 1994; Radicchi et al., 2004
LS-set	a maximal connected subgraph, where each element of the subgraph has more connections with other elements of the subgraph than with elements outside of the subgraph	Wasserman and Faust, 1994; Flake et al., 2002; Radicchi et al., 2004
lambda-set	a maximal connected subgraph, where each element of the subgraph has a larger element-connectivity with other elements of the subgraph than with elements outside of the subgraph (where element-connectivity means the minimum number of elements that must be removed from the network in order to leave no path between the two elements)	Borgatti et al., 1990; Wasserman and Faust, 1994
modified (weak) LS-set	a maximal connected subgraph, where the sum of the inter-modular links of the subgraph is smaller than the sum of the intra-modular edges	Wasserman and Faust, 1994; Radicchi et al., 2004

Table S1. Definitions of network modules (continuation)

Name	Definition	References
Definitions based on 'local' topology (continuation)[a-d]		
***k*-core**	a maximal connected subgraph, where the elements of the subgraph are connected to at least *k* other elements of the same subgraph; alternatively: the union of all *k*-shells with indices greater or equal *k*, where the *k*-shell is defined as the set of consecutively removed nodes and belonging links having a degree ≤ *k*	Seidman, 1983; Wasserman and Faust, 1994; Radicchi et al., 2004; Carmi et al., 2007; Kitsak et al., 2008
Definitions based on 'global' topology		
Modularity(Q)-based definitions	a set of subgraphs, which displays a larger modularity (Q) measure, than the same set of subgraphs in an appropriate null-model (for the definition of edge- or motif-based modularity measures and – usually randomized – null-models see the references and Table S3)	Girvan and Newman, 2002; Newman and Girvan, 2004; Reichardt and Bornholdt, 2006a; Arenas et al., 2008a; Fortunato and Castellano, 2007; Fortunato, 2010
Definitions based on element similarity		
Element similarity definitions	a subgraph containing element-pairs, which are similar to each other based on their distance, eigenvector components, etc.; similarity may also be functional similarity coming from the emergent network properties – this latter similarity measure, however, may add an element of redundancy to the definition, since the emergent function often emerges from the modular structure	Leicht et al., 2006; Fortunato and Castellano, 2007; Fortunato, 2010
Definitions based on information content		
Information content-based definitions	a set of subgraphs, which allow the greatest compression of network structure with a minimal loss of information (the Network Information Bottleneck)	Ziv et al., 2005; Rosvall and Bergstrom, 2007; 2008a
Definitions based on information propagation		
Information propagation-based definitions	a set of elements, displaying a larger communication among them than to the rest of the network	Flake et al., 2002; and all network walk-based methods of Table S2
Global influence	a set of elements, displaying a larger influence (sum of weighted paths) to each other than to the rest of the network	Ghosh and Lerman, 2008
Definitions based on network dynamics[e]		
Co-Set	a subgraph, where all elements occur or change simultaneously	Papin et al., 2004

[a]The definitions based on 'local' topology are listed in the order of their approximate stringency starting with the most stringent. A summary of additional (less exact) local community definitions can be found in Hinne (2007).
[b]The most widely used module definition is the LS-set or sometimes the modified LS-set. However, all the other definitions (and possibly even more) are satisfying our common sense of community formation based on the higher relative frequency/strength of links between community members compared to non-members ('static definition') as well as on the larger closeness or reachability of community members compared to non-members ('dynamic definition').
[c]In all the 'local' topology-based definitions no loops or multiple links were allowed.
[d]Most of the 'local' topology-based definitions do not include link weights or directedness.
[e]While there are plenty of mathematically explicit 'static' definitions, there is a remarkable paucity of explicit definitions, which include a dynamic network property.

Table S2. Comparison of network module determination methods

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Agglomerative methods										
Hierarchical agglomeration (clustering)	+	+	–	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Johnson, 1967; Aldenfelder and Blashfield, 1984; Wasserman and Faust, 1994; Slater 2008
Clustering with previous overlap 'distribution'	+	–	–	parameter dependent	refined	+	~O(n^3), local version: O(n log$n$)	N.A.	N.A.	Gregory 2009
Shortest path-similarity and hierarchical agglomeration	+	–	–	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Rives and Galitsky 2003; Arnau et al., 2005
Neighborhood similarity and hierarchical agglomeration	+	–	–	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Andreopoulos et al., 2007
Topological overlap methods	+	+	–	parameter dependent	yes/no	+	≥O(n^3)	N.A.	N.A.	Ravasz et al., 2002; Zhang and Horvath, 2005; Li and Horvath, 2007; Yip and Horvath, 2007
Restricted neighborhood search/flow Markov clustering	+	+	–	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Enright et al., 2002; Spirin and Mirny, 2003; Dorow et al., 2004; King et al., 2004
Clustering and a random walk process	+	+	+	parameter dependent	yes/no	–	~O(n^2)	N.A.	1	Delvenne et al., 2008; E et al., 2008; Li et al., 2008b
Clustering and a flocking process	+	+	+	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Li et al., 2008c
Markov clustering with noise	+	–	–	parameter dependent	yes/no	+	N.A.	N.A.	(1)	Gfeller et al., 2005; 2007

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Agglomerative methods (continuation)										
Clustering similarity- and distance similarity-based method	+	–	–	parameter dependent	yes/no	+	N.A.	N.A.	N.A.	Poyatos and Hurst, 2004
Dissimilarity matrix reordering based visual clustering	+	–	–	parameter dependent	yes/no	–	$O(n^2)$	N.A.	N.A.	Yang et al., 2006
Multilevel clustering (coarsening/de-coarsening)	+, –	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Oliviera and Seok, 2006
Network Information Bottleneck (NIB) clustering	+	–	–	automatic	with 'soft clustering': refined	With 'soft clustering: +	N.A.	0.4, 0.45	N.A.	Ziv et al., 2005
Bayesian clustering	+	–	–	automatic	yes/no	–	$O(n^2)$	N.A.	0	Hofman and Wiggins, 2008
Expectation-maximization, kernel-derived method	+	+	–	automatic	refined	+	N.A.	N.A.	3 overlaps	Ren et al., 2009
Potts-model	+	+	+	parameter dependent	yes/no	–	~O(nlogn)	0.88	N.A.	Blatt et al., 1996; Spirin and Mirny, 2003; Guimera et al., 2004; Ronhovde and Nussinov, 2008, 2009
Fuzzy Potts-model	+	+	–	parameter dependent	yes/no	+	parameter dependent	0.7	N.A.	Reichardt and Bornholdt, 2004; 2006b; Ispolatov et al., 2006; Heimo et al., 2008b
Informational coherence and fuzzy Potts-model	+	–	–	parameter dependent	yes/no	+	$\geq O(n^3)$	N.A.	N.A.	Shalizi et al., 2007
Laplacian clustering	+	+	–	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Kim et al., 2008

Supplementary Table 2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Agglomerative methods (continuation)										
Enforced frustration method	+	+	–	automatic	yes/no	–	$O(n^{3.2})$	N.A.	3	Son et al., 2006
Dynamical clustering (sync-dependent hierarchy detection)	+	–	–	parameter dependent	yes/no	–	$O(n^2)$	0.45	0 (3 modules)	Arenas et al., 2006; Boccaletti et al., 2007; Pluchino et al., 2008
Dynamical simplex evolution method	+	–	–	automatic	yes/no	–	$O(n^2)$	0.83	N.A.	Gudkov et al., 2008
Symmetric connectivity and noise/continuous dynamics	+	+	–	automatic	yes/no	–	$O(n^2)$	N.A.	N.A.	Krawczyk and Kulakowski, 2008
k-means (*k*-median, *p*-median) of cluster distance minimization	+	–	–	parameter dependent	yes/no	–	$\sim O(n^4)$	0.26, 0.9	0	MacQueen, 1967; Gustafsson et al., 2006; Angelini et al., 2007a; Brusco and Köhn, 2008
fuzzy *k*-means clustering (fuzzy c-means clustering)	+	+	–	parameter dependent	refined	+	$<O(n^3)$	N.A.	0	Bezdek, 1981; Dunn, 1974; Liu (2010)
Network vectorization clustering	+	+	+	parameter dependent	yes/no	–	N.A.	0.87	0	Ren et al., 2008
Co-occurrence methods	+	–	–	automatic	refined	+	N.A.	N.A.	N.A.	Papin et al., 2004; Jin et al., 2008
Highly connected subgraph spectral analysis method	–, +	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Kleinberg, 1997; Gibson et al., 1998; Hartuv and Shamir, 2000; Bu et al., 2003
Distance-based *k*-clique clustering methods	+	–	–	automatic	yes/no	–	$O(n^3)$	N.A.	N.A.	Edachery et al., 1999
Game-based clustering	+	+	+	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Li et al., 2008d

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Network walk-based agglomerative methods[i]										
Clique percolation methods	–, +	+	+	automatic	Refined	+	O(*exp(n)*) – in real applications it runs faster, O(n^2)	N.A.	N.A.	Alba and Moore, 1978; Palla et al., 2005; Adamcsek et al., 2006; Zotenko et al., 2006; Farkas et al., 2007; Palla et al., 2007a; Du et al., 2008; Kumpula et al., 2008; Lehmann et al., 2008; Shen et al., 2008; Zubcsek et al., 2008
Bipartite cliques	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Tanay et al., 2004
Local community-based methods (basins of attraction)	–, +	+	–	automatic	yes/no	+	O($n \log n$), O(n^3)	0.18	1	Altaf-Ul-Amin et al., 2006; Luo et al., 2006; Bagrow, 2008; Carmi et al., 2008; Hu et al., 2008b
Local fitness optimization	+	–	–	automatic	refined	+	>O(n^2), fast	0.6	5 overlaps	Baumes et al., 2005a; 2005b; Lancichinetti et al., 2009a
Local community with fuzzy clustering	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Hu et al., 2007
k-core-based methods	–	–	–	automatic	yes/no	+	~O(n^3)	N.A.	N.A.	Bader et al., 2003; Wuchty and Almaas, 2005; Alvarez-Hamelin et al., 2006; Dorogovtsev et al., 2006; Baskerville et al., 2007; Carmi et al., 2007
Local communities with *l*-shell label propagation	–, +	–	–	automatic	yes/no	+	>O(n^3)	N.A.	3	Bagrow and Bolt, 2005; Porter et al., 2007
Local communities with *t*-shell (initial triangles) label propagation	–, +	–	–	automatic	yes/no	+	N.A.	N.A.	4 overlaps	Eckmann and Moses, 2002; Kelsic, 2005
Local communities with bridge-bounding	+	–	–	automatic	yes/no	–	~O(n)	N.A.	N.A.	Papadopoulos et al., 2009

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Network walk-based agglomerative methods[i] (continuation)										
Local communities with (hub-based) label propagation/hub duplication	–, +	+	+	automatic	yes/no	–	~O(n)	N.A.	1	Gibson et al., 1998; da Fontoura Costa, 2004; Ucar et al., 2006; Zhang et al., 2008
Similarity-based network evolution methods	+	+	+ (signed network)	automatic	yes/no	–	≤O(n^3)	0.82	1	Yang, 2006; Xiang et al., 2009
Maximum flow, minimum cut methods (electric circuit models w/o or with spectral analysis)	+	+	–	parameter dependent	yes/no	–	O(n), >O(n^3)	0.57	0,1	Elias et al., 1956; Ahuja et al., 1993; Flake et al., 2002; Eriksen et al., 2003; Simonsen et al., 2004; Wu and Huberman, 2004; Alves, 2007; Slater 2008
Community profile plot assessment method	+	+	–	parameter dependent	yes/no	–	N.A.	N.A.	three modules	Leskovec et al., 2008
Communication-related Green's function with spectral analysis	+	–	–	parameter dependent	yes/no	+	>O(n^3)	N.A.	overlaps	Estrada and Hatano, 2008
Communicability graph and clique identification	+	–	–	parameter dependent	yes/no	+	exp(n) but in reality faster	N.A.	0 or overlaps	Estrada and Hatano, 2009
Random walk-based similarity distance and hierarchical agglomeration	+	+	+	automatic	yes/no	–	~O(n^3)	N.A.	N.A.	Pons and Latapy, 2005
Diffusion-based hierarchical communities	+	+	–	automatic	yes/no	–	O(n^3)	~0.75	0	Zhou 2003a; 2003b; Zhou and Lipowski, 2004
Diffusion kernel-based similarity method	+	–	–	automatic	yes/no	–	N.A.	0.68	1	Zhang et al., 2007b

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Network walk-based agglomerative methods[i] (continuation)										
Agent propagation, voting, mergers	+	–	–	automatic	yes/no	–	N.A.	N.A.	0	Young et al., 2004
Opinion propagation with decaying confidence	+	–	–	automatic	yes/no	–	N.A.	N.A.	1	Morarescu and Girard, 2009
Autonomous agent-based method for dynamic networks	+	–	–	automatic	yes/no	–	$>O(n^2)$	N.A.	1	Yang et al., 2010
Belief propagation Bayesian method to solve the Potts-model	+	–	+	automatic	yes/no	–	$O(nlog^{\alpha}n)$	0.95	1	Hastings et al., 2006; Sulc and Zdeborova, 2009
Label propagation with random link removal	+	+	–	automatic	yes/no	+	$\sim O(n)$	N.A.	3 solutions (1 correct)	Raghavan et al., 2007; Tibély and Kertész, 2008; Xiaodong et al., 2008; Barber and Clark, 2009; Leung et al., 2009; Gregory, 2009; Liu and Murata, 2010
Affinity propagation methods	+	+	+	automatic	yes/no	–	$>O(n^2)$	N.A.	N.A.	Frey and Dueck, 2007; 2008; Leone et al., 2008; Wang et al., 2007; Brusco and Köhn, 2008
Information flow-based methods	+	+	–	automatic	refined	+	$O(n^2logn)$	N.A.	N.A.	Cho et al., 2006; 2007; Hwang et al., 2006; 2008
Signal propagation with F-statistics and fuzzy C-means clustering	+	+	–	automatic	yes/no	–	$>O(n^2)$	0.8	0	Hu et al., 2008a
Neuronal activation propagation times with principal component analysis	+	–	+	automatic	yes/no	–	N.A.	N.A.	N.A.	da Fontoura Costa, 2008a; 2008b

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Network walk-based agglomerative methods[i] (continuation)										
All shortest paths with principal component analysis	+	–	–	automatic	yes/no	–	N.A.	N.A.	0	da Fontoura Costa and Rodrigues, 2008a; Zhang et al., 2008
Message percolation method	+	+	–	automatic	refined	+	N.A.	0.15	N.A.	Meng Muntz and Rezaei, 2006
Structure-connected clusters (common neighbors)	+	–	–	automatic	yes/no	yes/no	$O(n^2)$	N.A.	N.A.	Mete et al., 2008
ModuLand method family	+	+	+	automatic	refined	+	implementation-dependent, $\leq O(n3)$ possible	N.A.	3 modules with overlaps	Kovács et al., 2006; current paper
Methods (both agglomerative and divisive) based on modularity (Q) optimization[j]										
Hierarchical agglomeration with (greedy) optimization	+	–	–	parameter dependent	yes/no	–	$\sim O(n^2)$, $O(nlog^2n)$	0.33	0,1	Newman, 2004a; Clauset et al., 2004; Gustafsson et al., 2006
Hierarchical agglomeration with a random walk	+	–	–	parameter dependent	yes/no	–	$\sim O(n)$	N.A.	1	Pujol et al., 2006
Multi-step greedy algorithm with vertex mover refinement	+	+	–	parameter dependent	yes/no	–	$\geq O(nlogn)$	N.A.	N.A.	Schuetz and Caflish, 2008; Noack and Rotta, 2008; Sun et al., 2009
Extremal division optimization	+	+	+	parameter dependent	yes/no	–	$O(n^2logn)$	0.82	5 modules	Duch and Arenas, 2005
Simulated annealing	+	+	+	automatic	yes/no	–	$\geq O(n^3)$	0.9	N.A.	Guimera and Amaral, 2005; Massen and Doye, 2006
Mean field annealing	+	–	–	automatic	yes/no	–	$\geq O(n^2)$	N.A.	N.A.	Lehmann and Hansen, 2007
Basin hopping	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Massen and Doye, 2005

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Methods (both agglomerative and divisive) based on modularity (Q) optimization[j] (continuation)										
Convex optimization	+	–	–	automatic	yes/no	–	N.A.	0.29	1	Hildebrand, 2008
Linear or vector programming relaxation methods	+	–	–	automatic	yes/no	–	$O(n^2)$, $O(n^3)$	N.A.	1 (4 modules)	Agarwal and Kempe, 2008
Hierarchical community detection based on affinity matrices	+	–	–	automatic	refined	+	$>O(n^3)$	N.A.	N.A.	Sales-Pardo et al., 2007
Genetic algorithm method	+	–	–	automatic	yes/no	–	$O(n)$	N.A.	0,1	Tasgin et al., 2007
Mixed integer mathematical programming	+	–	–	automatic	yes/no	–	N.A. (high)	N.A.	N.A.	Xu et al., 2007
Modularity preserving pre-modularization size-reduction	–	+	–	automatic	yes/no	–	speed increase up to a factor of 4.27	N.A.	0	Arenas et al., 2007
Local modularity algorithm	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Hinne, 2007
Multiscale Q-optimization Using self-loops	+	+	+	automatic	yes/no	–	N.A.	N.A.	0	Arenas et al., 2008b
Methods using spectral properties of the network	+	+	+	automatic	yes/no	–	$\geq O(n^2)$	0.57, 0.72	0	Donetti and Munoz, 2004; 2005; White and Smyth, 2005; Newman, 2006a; 2006b; Barber, 2007; Leicht and Newman, 2008; Richardson et al., 2008; Ruan and Zhang; 2008

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Methods (both agglomerative and divisive) based on modularity (Q) optimization[j] (continuation)										
Multilevel partitioning using minimum weight cut of a derived complete graph	+	–	–	automatic	yes/no	–	$\geq O(nlog^2n)$	0.7	1	Djidjev, 2008
Optimization methods using alternative modularity definitions										
Hierarchical agglomeration for heterogeneous modules	+	–	–	parameter dependent	yes/no	–	$O(nlog^2n)$	>0.33	0	Danon et al., 2006
Cluster density-based optimization	+	–	–	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Spirin and Mirny, 2003
Maximal clique-based Q optimization	+	–	–	parameter dependent	refined	+	N.A.	N.A.	4 modules + overlaps	Shen et al., 2009
Modularity for directed graph and overlaps	+	–	+	automatic	yes/no	+	N.A.	N.A.	2 overlaps	Nicosia et al., 2009
Modularity for positive and negative links	+	+	+ (signed network)	automatic	yes/no	–	N.A.	N.A.	N.A.	Bansal et al., 2004; Gómez et al., 2009; Kaplan and Forrest, 2008; Traag and Bruggeman, 2009
Local modularity optimization	–, +	–	+	automatic	yes/no	+	$\sim O(n^2)$	~0.5	N.A.	Clauset, 2005; Muff et al., 2005; Rodrigues et al., 2007
Local modularity optimization and hierarchical agglomeration	+	+	–	automatic	yes/no	–	N.A. (fast)	0.67	4 modules	Blondel et al., 2008; Wallace and Gingras, 2008
Multiscale (local → global) modularity refinement	+	+	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Pons, 2006

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Optimization methods using alternative modularity definitions (continuation)										
Community strength modularity optimization	+	–	–	automatic	yes/no	–	N.A.	N.A.	3 modules	Medus and Dorso, 2009
Local influence-based modularity optimization	+	+	+	automatic	yes/no	–	N.A.	N.A.	0	Ghosh and Lerman, 2008
Motif-based modularity maximizing methods	+	+	+	parameter dependent	yes/no	–	N.A.	N.A.	0	Arenas et al., 2008a
Centrality-based modularity optimization	+	–	–	parameter dependent	yes/no	–	N.A.	N.A.	4 modules	Ghosh and Lerman, 2009; Lerman and Ghosh, 2009
Statistical distribution-based modularity	+	–	–	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Pei and Zhang, 2007
Blockmodeling-based modularity	+	–	+	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Reichardt and White, 2007
Partition-coverage starting modularity generalization	+	–	–	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Gaertler et al., 2007
Mutual information-based modularity maximizing methods (link to hierarchical clustering)	+	+	+	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Angelini et al., 2007b; Bickel and Chen, 2009
Facility location theory-based method using strongly local modularity	+	–	–	automatic	yes/no	–	$O(nlog^2n)$	N.A.	0	Berry et al., 2007
Fuzzy modularity optimization using *c*-means clustering	+	+	–	automatic	refined	+	$\sim O(n)$	N.A.	4 modules	Zhang et al., 2007a

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Optimization methods using alternative modularity definitions (continuation)										
Fuzzy c-means modularity optimization based on improved Shared Nearest Neighbor method	+	+	–	Automatic	Refined	+	N.A.	N.A.	2 modules	Xie et al., 2009
Markov random walk and oscillator-synchronization-based modularity	+	+	+	automatic	yes/no	–	N.A.	N.A.	N.A.	Lambiotte et al., 2008; Li et al., 2008a
K-means clusters of node synchronization correlation matrix	+	+	+	automatic	yes/no	–	N.A.	N.A.	2 modules	Li et al., 2010
Synchronization based evolutionary subnetworks	+	–	–	automatic	yes/no	+	$O(n^2)$	N.A.	4 modules	Li et al., 2009
Random walk link partition on weighted line graph (edge-to-vertex dual)	+	+	+	automatic	yes/no	+	N.A.	N.A.	4 modules	Evans and Lambiotte 2009a
Time-dependent Q optimization (multislice networks)	+	–	–	parameter dependent	yes/no	–	N.A.	N.A.	2–4 modules	Mucha et al., 2009

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Optimization methods using alternative modularity definitions (continuation)										
Q optimization on weighted line graph (edge-to-vertex dual)	+	+	–	automatic	yes/no	+	N.A.	N.A.	N.A.	Evans and Lambiotte 2009b
Self-organization-based modularity	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Ahnert et al., 2009b
Other divisive methods										
Centrality-based methods (betweenness random walk, current-based and information centralities)	+	–	–	parameter dependent	yes/no	–	$O(n^3)$, $O(n^4)$	0.3, 0.16	0, 1,3	Girvan and Newman, 2002; Fortunato et al., 2004; Newman, 2004c; Newman and Girvan, 2004; Andrade et al., 2009
Fuzzy betweenness centrality methods	+	–	–	parameter dependent	refined	+	$\leq O(n^3)$	N.A.	N.A.	Wilkinson and Huberman, 2004; Pinney and Westhead, 2006; Gregory, 2007
Conductance optimization	+	–	–	parameter dependent	yes/no	–	(NP-hard)	N.A.	N.A.	Bollobas, 1998; Cheng and Shen, 2009
Cut-size optimization	+	–	–	parameter dependent	yes/no	–	(NP-hard)	N.A.	N.A.	Wei and Cheng, 1989
Division optimization with branch and bound method	+	+	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Hager et al., 2009
Division optimization based on edge-clustering (loops)	+	+	+	parameter dependent	yes/no	–	$\geq O(n2)$	0.82	5 modules	Radicchi et al., 2004; Vragović and Louis, 2006

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Other divisive methods (continuation)										
Mutual information-based methods	+	–	–	automatic	yes/no	–	>O(n3)	N.A.	0	Strehl and Ghosh, 2002; Rosvall and Bergstrom, 2007; 2008a; Sun et al., 2007; Zhang et al., 2008; Kraskov and Grassberger, 2009
Bootstrap resampling and significance clustering	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Rosvall and Bergstrom, 2008b
Maximum likelihood method	+	+	+	automatic	refined	+	fast	N.A.	0,1	Čopič et al., 2005; Newman and Leicht, 2007; Mitrovic and Tadic, 2008; Mungan and Ramasco, 2008; Ramasco and Mungan, 2008; Vazquez, 2008a; 2008b; Wang and Lai, 2008; Zanghi et al., 2008
Dendrogram maximum likelihood method	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Clauset et al., 2008
Tree mapping	+	+	–	automatic	yes/no	–	N.A.	N.A.	N.A.	da Fontoura Costa and Rodrigues, 2008b
Similarity optimization with simulated annealing	+	+	–	automatic	refined	+	N.A.	N.A.	N.A.	Nepusz et al., 2008

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Other divisive methods (continuation)										
Hypergraph mutual information	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Strehl and Ghosh, 2002
Random walk-based heat kernel pagerank and Cheeger inequality local cut	–	–	–	parameter dependent	yes/no	–	~O($n^{1.5}$)	N.A.	N.A.	Chung, 2007
Random walk-based LinkRank	+	+	+	parameter dependent	yes/no	–	N.A.	N.A.	N.A.	Kim et al., 2010
Clustering (eigenvector) centrality and repetitive matrix bipartition	+	+	+ (signed network)	automatic	yes/no	–	~O(n)	N.A.	N.A.	Yang and Liu, 2007
Methods using spectral properties of the network or its local communities	–, +	+	+	automatic	yes/no	–	≥O(n)	N.A.	N.A.	Barnes, 1982; Donath and Hoffman, 1972; Kleinberg, 1997; Gibson et al., 1998; Hartuv and Shamir, 2000; Capocci et al., 2005; Tibély et al., 2006; Heimo et al., 2008a; Sahai et al., 2009
Division optimization based on community sub-matrix eigenvalue maximization	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Chauhan et al., 2009
Truncated singular value decomposition of the modular contribution matrix	+	–	–	automatic	yes/no	–	N.A.	N.A.	N.A.	Arenas et al., 2009

Table S2. Comparison of network module determination methods (continuation)

Name of method	Complete Data-set[a]	Weighted graph[b]	Directed graph[b]	Number of modules[c]	Assignment to modules[d]	Overlaps[e]	Polynomial complexity (speed)[f]	Test-modules[g]	Zachary-network[h]	References
Other divisive methods (continuation)										
Matrix factorization (+ semi supervised clustering)	+	–	–	automatic	yes/no	–	N.A. (slow)	0.53	3	Zhang et al., 2007c; Wang et al., 2008b; Ma et al., 2010
Spectral properties of the complement graph	+	+	+	automatic	yes/no	–	N.A.	N.A.	N.A.	Zarei and Samani, 2009; Zarei et al., 2009
Symmetric community measurement (on graph and its complement)	+	–	–	automatic	yes/no	–	N.A.	N.A.	3 modules	Wang and Lai, 2009
Methods for finding overlaps after any given non-overlapping clustering method										
Find overlappings by paralel genetic algorithms	+	+	–	automatic	yes/no	+	N.A.	N.A.	2 modules with overlaps	Carchiolo et al., 2009
Overlapping modularity measurement	+	–	–	automatic	yes/no	+	N.A.	N.A.	N.A.	Lázár et al., 2009

Data-assembly of the Table was closed on February 12th 2010. While we have taken a considerable effort to detect and read a large number of modularization methods, obviously the above list is extremely far from being complete. We would like to deeply apologize to all respectful colleagues, whose methods and significant efforts have been inadvertedly omitted from this list in this voluminous and extremely fast-growing field. Comparison of the methods was helped by the reviews of Newman (2004b), Danon et al. (2005), Fortunato and Castellano (2007) and Fortunato (2010). 35 clustering algorithms have been nicely reviewed and clustered to a network by Jain et al. (2004). Several algorithms were compared by Lancichinetti and Fortunato (2009b) and the modularity maximization methods were critically assessed by Good et al. (2009). Where more than one references are given, notes and numbers in the columns may refer to only one or a few of them. In case of multiple values separated with commas, each of them is referring to different reference(s). N.A. = data not available.

[a]At the column "Complete data-set" a "+" sign means that the method used all data from the original data set. The "–" sign denotes that some of the original data were deleted or comprised (like at coarse-graining methods), or the analysis was a fully local method using only a sub-segment of the original network. If both signs are present, some of the methods used all original data, while others not. Note, that many currently available network data are, in fact, only samples of a larger data-set, and in this sense, even the "+" methods use only a partial information. A detailed elucidation of the effects of sampling biases on modularization awaits further analysis (for an initial study see Lusseau et al., 2008).

[b]"Weighted graph" and "Directed graph" notes, if the method uses the additional information of weights or directedness for determining the modules. Some of the methods marked as "–" in these columns were applied to weighted and directed networks, but did not use these properties for the refinement of the modular structure.

[c]"Number of modules" is "parameter dependent", if the maximal number of modules does not derived "automatic"-ally from the method.
[d]"Assignment of modules" is "refined", if the method gives a continuous scale for all links and elements as their assignment to various modules (fuzzy clustering/partition), and "yes/no", if only a decisive "yes" or "no" answer (hard/crisp community clustering/partition) is given.
[e]"Overlaps" notes, if the method calculates overlaps between modules. If the method had a "yes/no" assignment, the existence of overlaps means that certain elements/edges were assigned to multiple modules simultaneously with an equal weight.
[f]"Speed" refers to the computational speed (computational complexity) of the method, where symbol *n* denotes the number of elements. To simplify the formulas and help comparison, many times it was assumed that *n* is roughly equal to the number of edges, i.e. the method is applied to sparse networks.
[g]Numbers in the "Test modules" column refer to the fraction of correctly identified elements in the model network proposed by Girvan and Newman (2002) having four communities with 32 links each having 16 number of neighbors in average, half of them being intra-modular and the other half of the elements pointing towards other modules.
[h]The "Zachary network" column gives the number of misplaced members of Zachary's karate club (Zachary, 1977), where the hidden modular division was later exposed by a real split, which made this network a gold-standard for the assessment of module determination methods (Girvan and Newman, 2002). Values in parentheses refer to the number of overlapping values, which are not real misplacements.
[i]Network walk-based methods also contain a large variety of methods, where the mutual information of the local network environment has been assessed and used for module determination.
[j]The modularity function (Q) has been suggested by Newman (2004a) as a measure of the "statistically surprising fraction of the links in a network fall within the chosen communities" (Leicht and Newman, 2008) meaning that the edge-density larger than that of an appropriate model system. The exhaustive optimization of this function is an NP-complete problem (Brandes et al., 2007), which makes this straightforward method computationally untractable with larger size real networks. Therefore, several heuristic, optimum-search strategies have been applied to find the global optimum and to circumvent the traps of local optima using an algorithm with a reasonably low computational complexity.

Table S3. Comparison of modularization methods

Name of parameter, tool or method	Description of the parameter, tool or method helping the comparision of modularization methods	References
Modularity (Q)	Difference between the fraction of links within modules and the expected number of such links under an appropriate null model (like the one, which preserves the degree sequence of the original network, but otherwise randomizes link positions producing an ensemble of networks) – shown to be a necessary but not sufficient condition, since large random graphs may also have a division resulting in a high modularity and smaller modules may escape detection.	Newman and Girvan, 2004; Guimera et al., 2004; Reichardt and Bornholdt, 2006a; Fortunato and Barthélemy, 2007; Pei and Zhang, 2007; Good et al., 2009
Weighted modularity (Q)	The same as above for weighted networks, where in the null model the element strength and not only the element degree is preserved.	Newman, 2004c
Maximal clique-based modularity-like function (Q_c)	Uses a continuous scale for module assignment and the assumption that a highly coeherent maximal clique is usually found only in one module.	Shen et al., 2009
Benchmark graphs	Here a test-graph is examined and the recovery of the pre-set modular structure is tested for divisive methods or – more recently – for overlapping modules.	Zachary, 1977; Newman and Girvan, 2004; Lancichinetti et al., 2008, 2009a; Reichardt and Leone, 2008; Lancichinetti and Fortunato, 2009a, 2009b; Sawardecker et al., 2009
Network Information Bottleneck (M)	Perceives modularization as a coarse-graining process, which finds an optimum between minimizing the number of relevant modules and preserving the emergent information of the whole network and defines M as the area under the 'information curve', which is the function of the output information (after modularization) as the function of the input information (before modularization) of the network.	Ziv et al., 2005
Local modularity scores	Utilize a network walk and determine a local community around the starting element resulting in a large number of 'stopping criteria', where the local community is said to be complete; defines 'outwardness' of an element as the number of its neighbors outside the community minus the number of neighbors inside normalized by the degree and follows the quality of community-growth as the change in the number of extra-modular links.	Bagrow, 2008
Pairwise membership probability based consistency	The module memberships of network nodes are compared after different modularization methods and a consistency measure of all comparisions for the whole network is defined.	Kwak et al., 2009
Clustering stability during a Markov chain random walk	The autocovariance of the network partition during a Markov chain random walk is characterizing the stability of the clusters. If the clustering shows a high stability during the Markov process it is good representation of the real network communities.	Delvenne et al., 2008
Statistical significance	The statistical significance of communities is calculated using Extreme Statistics, and artificial communities arising as structural fluctuations of random graphs are detected.	Lancichinetti et al., 2009b

Table S3. Comparison of modularization methods (continuation)

Parameter	Description	References
Eigenvector stability	The statistical significance of communities is calculated using the stability of the eigenvectors of the Laplacian matrix.	Hu et al., 2010
Perturbation resilience	Altering the position of a few links will not change the modular structure (random graphs have multiple competing modularity maxima, while real networks typically have a single global maximum).	Massen and Doye, 2006; Karrer et al., 2008; Yip and Horvath, 2007; Hu et al., 2009
Resolution limit	Optimization of global modularity measures may not detect small modules, if the size distribution of modules is large in the network (this problem is also known ad the "giant-component problem" referring to the fact, that the detection of small modules is often possible by the concurrent coalescence of larger modules to a giant component), these problems also warn that many quality functions of modularization work best only, if results with the same number of modules are compared.	Fortunato, 2007; Fortunato and Barthélemy, 2007; Kumpula et al., 2007; Berry et al., 2009
Computational time	This parameter is usually given as the polynomial complexity of the modularization algorithm, where the exponent of the number of elements and (especially) links of the starting network is critical.	Newman, 2004b; Dannon et al., 2005; Gustaffson et al., 2006; Fortunato and Castellano, 2007; Fortunato, 2010

The Supplementary Table lists the parameters, which have been designed so far to help the comparison of modularization methods judging their efficiency, reproducibility and usefulness. Most of the measures however have not been accommodating link weights (positive and negative) and directedness (+ link colors), which are highly significant parameters of module membership assignment in real-world networks.

Supplementary Discussion

The network approach, i.e. the description of complex systems as an assembly of their connected elements, became an increasingly useful method to describe, understand and visualize the enormous data sets of current science and everyday life from cells to society (Boccaletti et al., 2006; Barabasi and Oltvai, 2004; Csermely, 2006; Newman, 2003; Strogatz, 2001; Watts, 1999). The importance of network modules has been recognized rather early (Homans, 1950; Rice, 1927; Simon, 1962; Weiss and Jacobson, 1955), and their determination drew more and more interest, since modules are key players to understand the functional organization and evolution of networks (Danon et al., 2005; Fortunato, 2010; Fortunato and Castellano, 2007; Hartwell et al., 1999; Newman, 2004b; Porter et al., 2009). Modular overlaps and their key function in social organization have been described by Georg Simmel a long ago (Simmel, 1922) and gained recently a growing attention and interest (Palla et al., 2005).

The ideal network modularization method delivers both meaningful partitions and community overlaps. Additionally, an ideal method describes both the connected hierarchy and the disjoint embeddedness of the autonomous sub-networks. Finally, an ideal method is both fast and accurate at the same time. The simultaneous optimization of the above requirement-pairs is impossible, therefore a novel method can not be 'better' than the previous methods, it may only be different in the ratio how it fulfills one or another ascpect of the above, diverse requirements. There are several modularization methods, which explore the local topology of hubs (da Fontoura Costa, 2004), extend the communities by label propagation (Raghavan et al., 2007; Leung et al., 2009), employ a highly efficient optimization of the Potts-model (Ronhovde and Nussinov, 2009), or a gradual hierarchical agglomeration process (Pujol et al., 2006), and thus achieve a computational time up to nearly linear with the number of network elements. These methods are particularly suitable to assess the modular structure of extremely large networks. However, they use only a subset of the available structural information and, therefore, in many cases have only a limited accuracy. A large number of methods exist, which provide a rather accurate representation of network communities (Table S2). However, most of these accurate methods are computationally expensive. Similar statements apply for handling the modular overlaps and network hierarchy. The ModuLand method family includes both accurate (but somewhat slower) and fast (but relatively inaccurate) methods, as well as methods handling variable extent of modular overlaps at multiple hierarchical levels. Thus the ModuLand method family is not 'better' than the previous methods, but – as a significant and important novelty – provides the experimenter a general framework to respond to the otherwise intractably extreme requirements. This general framework allows an easy shift of emphasis from one optimization parameter to another thus finding an optimal method for the analysis of the given network or data structure. Moreover, the general framework of the ModuLand family also gives a unified background and tool for the easy comparision and evaluation of various modularization attempts, techniques and ideas.

Comparison of the ModuLand method family with existing methods

As we have summarized in the main text, the ModuLand method family is novel, since (a) it includes an unparalleled variety of integrated community heap-determination methods; and (b) uses the hills of community landscapes as a basis of module determination.

(a) The various community heap construction methods of the ModuLand family resemble to many local methods described in Table S2. One of the closest methods is that of Bagrow and Bollt (2005), who not only define local communities by the spreading of *l*-shells from the elements of the network, but also include a process aiming to achieve a community-assignment 'consensus' of the *l*-shells including the given element.

(b) Previous network landscape methods used clustering coefficients (Eckmann and Moses, 2002), edge number per visualized network unit area (Ramani et al., 2005), loop-coefficients (Vragovic and Louis, 2006) or degrees (Axelsen et al., 2006) to define the landscape-height. These definitions utilize mainly local elements of topology, while the ModuLand method assesses a wide range of structural information. Moreover, none of the previous authors used their landscapes for module determination. Recently a number of publications showed a 'hidden metric space' behind network topologies, which links network structure to a landscape-type representation (Krioukov et al., 2008; Narayan et al., 2009). The 'consensus'-building approach of Bagrow and Bollt (2005) resembles to the construction of the community landscape of our method. The recent work of Roswall and Bergstrom (2008) published during the course of the current study (Kovacs et al., 2006) uses the probability flow of random walks to construct a map of scientific communication. This method is similar to our PerturLand community heap construction method, but its application

yields non-overlapping modules. Moreover, none of the methods mentioned above and listed in Table S2 use the hills of a community landscape-type network representation to determine the modular structure. The hill-finding approach, which is the second phase of the ModuLand methods, gives an additional layer of flexibility where the relatively inaccurate results of simpler 'consensus'-building algorithms and the large computational costs of accurate optimization processes can be tailored to the network and to the experimenter's needs and possibilities.

Scale-free distribution of module parameters

The community size-distribution, the community degree distribution, the community overlap-size and the node membership number distribution of the University of South Florida word association network (Nelson et al., 1998) are shown on Figure S8. We have used cumulative distributions, since they were shown more accurate, than frequency-distributions (Tanaka et al., 2005). The distributions recover the formerly observed, highly heterogenous patterns (Arenas et al., 2004; Clauset et al., 2004; González et al., 2006; 2007; Guimera et al., 2003; Lancichinetti et al., 2009a; Palla et al., 2005; Pollner et al., 2006; Radicchi et al., 2004) but significantly deviate from the expected linearity on the log-log scale at both low and high values. This apparent discrepancy has two major reasons.

1.) While many of the previous distribution patterns of module parameters were similar to a scale-free distribution, quite often rather significant deviations from linearity on the log-log scale could also be observed. For example a non-linear log-log distribution pattern was quite obvious at the community degree ditribution of the South Florida work association network (Palla et al., 2005), at the community size distribution of the amazon.com network (Clauset et al., 2004) and the Add-Health school firendship network (González et al., 2006) as well as at the node membership number distribution of the Add-Health school friendship network (González et al., 2007).
2.) The properties of the ModuLand method may also influence a scale-free distribution of modular parameters to the fashion observed on Figure S8.
    - Those ModuLand method versions, which apply a module membership assignment method based on the PeakHill hill determination method do not require any previous knowledge about the possible number of network modules. However, due to the fact that the PeakHill methods only investigate the community landscape without further information about the underlying community heap structure, these methods are almost totally blind regarding the size of the underlying community heaps. In other words, the PeakHill methods can not discriminate between a community landscape built up from many small community heaps (contaning only a few elements) and a community landscape built up from a few large community heaps. Even if the original community heaps are small, the modules can still be of arbitrary size almost independently from the size of the heaps, thus significantly lowering the number of small modules.
    - Noise-like errors (coming either from the inaccuracy of the data or from the approximative nature of the community heap construction methods) may cause the appearance of small local maxima on the community landscape. Thus, by the introduction of 'artificial' local modules, even a very low level of noise may significantly fragment large modules. The effect of additional small modules will become rather noticeable in the case of the TotalHill module membership assignment method, where the size of modules assigned to the noise-induced small local maxima may become larger than that of the ProportionalHill or GradientHill module membership assignment methods. The module merging method described in Section VI.1. addresses this problem, but still residual discrepancies may remain.

    The above two properties shift both the small and large modules towards intermediate size entities causing the curvilinear pattern of Figure S8a.
    - The deviations of the effective module degree distribution on Figure S8b from linearity may be explained partly by the reasons listed before. We would like to add here that the preferential attachment model of modules (Pollner et al., 2006) takes into account a direct interaction of modules and constituent network elements. However, the community heap construction step of the ModuLand method is based on the effective, indirect impact of the starting element to the rest of the network (see Section III.1.), which again diminishes the number of both low-degree and extremely high-degree entities contributing to the curvilinear pattern of Figure S8b.
    - The above considerations also influence the overlap size and membership number distributions (Figures S8c and S8d).

    At higher hierarchical levels deviations from the ideal scale-free distribution of modular properties largely come from the definition of link weights at these levels reflecting the overlaps of primary modules (see Section VII.1.).

Robustness of the results obtained by the ModuLand method
In our work we applied the benchmark graph generation method published by Lancichinetti et al. (2008), which produces non-overlapping modules, in order to check the correspondence of our highly overlapping modules with the surely known partitions of the benchmarks. The comparison of our results with the recently published updated benchmark graph generation method by Lancichinetti et al. (2009a), which also supports overlapping modules, is an interesting and challenging problem of its own, and would require a study which is beyond the scope of our current paper.

In Figures S13a and S13b we show that the identified modules correspond consistently to the modules of the benchmark graph of Lancichinetti et al. (2008) over a range of parameter settings, where modules can be defined in the strong sense. Strong sense means here, at least the half of the neighboring nodes are assigned to the same module as the given node, see. Lancichinetti et al. (2008).

Overlapping word-association modules
The inter-modular, central position of multiple meaning words (heteronyms, antagonyms and homophones) shown by the ModuLand analysis of the University of South Florida word association network (Nelson et al., 1998) gives a further support of the earlier findings of Sigman and Cecchi (2002) showing that word ambiguity greatly improves the small-world character of the Word-net. The multiple meaning words of 'bright' and 'focus' shown on Figure S9 are homophones, where the multiple meanings are neither extremely disjoint (as would be the case for heteronym words with a different pronunciation) nor opposing (as would be the case for antagonym words). As expected, the analyzed homophone words are located in densely interconnected modules. Interestingly, our representation of the associative neighborhood of 'bright' does not show words referring to the intellect, which has been successfully picked up by the k-clique method of Palla et al. (2005). This shows the importance of the similarity threshold, which we set to 13% making the image clearer, by cutting all words having less than 13% modular similarity than the modular content of the examined word, 'bright'. Words related to the intellect were less than this threshold in this case.

Interestingly, words with the highest community landscape height representing centrally important elements of American thinking based on word-associations (as determined by the NodeLand method yielding almost a thousand modules total) are the following in decreasing order of their importance: money > cold > car > water > food > tree > book > church > dinner. The list represents well the major human needs and circumstances (water, food, dinner, cold, tree), values of the American society (book, church) and major icons of the American lifestyle (money, car).

Comparison of the hierarchical network representation of the ModuLand method family with that of other methods
The complexity of most networks exceeds the cognitive limits of the 'logical', left hemisphere of the brain, which is able to handle a handful of separate pieces of information only. This necessity of data-reduction together with the early recognition of the embedded structure of networks, where elements of complex networks are networks themselves (such as the cells of our brain can be represented as networks of proteins, etc.) made the hierarchical representation of networks a centerpiece of network studies. A number of hierarchy-representations came from the tree-structures of clustering schemes, or utilized the development of various renormalization-type processes, where the information content of network topology was gradually reduced utilizing the 'fractal-like' behavior of many real networks (Guimerá et al., 2003; Hartwell et al., 1999; Ravasz et al., 2002; Song et al., 2005; Wiuf et al., 2006). Recently several powerful methods have been published to extract network hierarchy (Clauset et al., 2008; Pan and Sinha, 2009; Sales-Pardo et al., 2007) and/or to detect the overlapping and hierarchical structure of network modules (Ahn et al., 2009; Lancichinetti et al., 2009a; Shen et al., 2008). The exposed hierarchy can be used for the simplified visualization of large networks (Walshaw, 2003) demonstrated by the example of GenPro (Vlasblom et al., 2006), a visualization subprogram of Cytoscape (Shannon et al., 2003).

The ModuLand method treats primary network modules as elements of the next hierarchical layer of the network, thus gives a hierarchical representation of the network community structure (Section VII.). The hierarchy of the network science collaboration network (Newman, 2006b) is shown on Figure S6, while that of Community-44 of the Add-Health school friendship network (Moody, 2001; González et al., 2007) is presented on Figure S10 and (partially) on Figure S12. The hierarchical network representation can be used for a fast visualization of extremely large networks, where

conventional network visualization methods became too slow to apply (see Section VIII.). The hierarchical representation of modules can also be a highly efficient method for efficient packet delivery (Danon et al., 2008; Palotai, 2008; Palotai and Csermely, 2009).

Modular hierarchy of a school friendship network

The hierarchical module structure of Community-44 of the Add-Health database (Moody, 2001; González et al., 2007) obtained by various methods of the ModuLand method family is shown on Figs. S10-S12. Most modularization methods obtain several or all of the 4 major modules at higher hierarchical levels. However, the methods also show a highly refined modular structure at lower levels up to 232 modules. Interestingly, if we decreased the *X* value of the Perturland method, thus decreased the putative exchange of information between the schoolchildren of the network, their modules have been better separated (cf. panels of Figure S12) – which is in agreement with our expectations.

Comparison of the community landscape height with other centrality measures

The vertical scale of the community landscape represents the combination (in its simplest form: sum) of the individual community heaps for the given link or element. Since the community heaps correspond to the indirect impact of their start-sites to all links of the network, their sum gives a centrality-type measure, which is high, if the overall impact of all network segments (elements, links or groups) is large. However, combining the community heaps is not the only option for constructing the community landscape. The concept of centrality has a long-standing tradition in network science, and any centrality measure defined in the past or to be defined in the future can serve as a basis of a community landscape. The seminal work of Linton C. Freeman (1978/79) clarified three structural measures of topological centrality: degree (the number of direct neighbors), betweenness centrality (the number of shortest paths traversing through the element) and closeness (the sum of shortest paths leading to all other elements). Later a number of other centrality measures have been defined, which take into account more, or different details of the overall topology of the network. These measures include

- the lobby index, which is the largest integer k of a node x fulfilling the criterion that node x has at least k neighbors with a degree of at least k (Korn et al., 2009)
- the eigenvector centrality, which is the principal eigenvector of the adjacency matrix related to the combined degree of the element and its neighbors (Bonacich, 1972),
- PageRank, which is the damped random-walk based prestige-measure of Google related to the principal eigenvector of the transition matrix describing the damped random walk (Brin and Page, 1998; Perra et al., 2008),
- approximation of PageRank-type measure by local node properties (in- and out-degrees) and by the centrality of the module the nodes maximally belongs (Masuda et al., 2009),
- Katz centrality, which is the total number of paths linking the given node to other nodes in the network exponentially weighted by the length of the path (Katz, 1953), or Bonacich centrality, where the total number of paths is attenuated by two factors α or β for indirect and direct links, respectively (Bonacich, 1987),
- subgraph centrality, which is related to the closed walks starting and ending at the given element (Estrada and Rodríguez-Velázquez, 2005) or centrality measures based on biased random walks (Estrada et al., 2008; Lee et al., 2009b),
- information centrality, which is the drop of graph performance removing the given element or link (Latora and Marchiori, 2007),
- information-flow score, which is a complex measure of information centrality modelling the network as an electrical circuit (Missiuro et al., 2009) and
- a large variety of centrality-type measures used in the network descriptiptions of ecosytems (for their summary see Jordán and Scheuring, 2004).

Another set of centrality measures can be derived from the community structure of the network. Measures, like the bridgeness, defined differently by Nepusz et al. (2008) and this paper, take into account the inter-modularity of an element. On the contrary, the height of the community landscape may merge the local (degree-, or eigenvector centrality-based), mesoscopic (community-based) and global centrality measures in different ratios, depending on the exact method of the ModuLand method family. The complexity of centrality measures is further substantiated by the findings of Pollner et al. (2008), who demonstrated that the inter-modular position is resulting in high centralities by a number of conventional centrality measures. Similarly, the grossly inter-modular 'creative elements' defined by Csermely (2008) display an extremely high centrality in a large variety of networks. The community landscape height could combine both the 'traditional', local centrality increments (mostly characteristic

to the cores of well-defined network modules) and the 'bridge-related' centrality increments typical to key information channels of modular overlaps. Different realizations of the ModuLand method family give different weight to these two types of centrality-increments, and thus regulate the 'roughness' of the community landscape, i.e. the overall extent of modular overlaps. (A rough community landscape has rather well defined hills, where overlaps are minor. On the contrary, a flat community landscape provides large overlaps and less well-defined individual modules.)

Central elements of power-grid a network

We have tested the centrality measures of the ModuLand method by removing the most central elements of the Western Power Grid network of the USA (Watts and Strogatz, 1998, Figure 4 of the main text) and calculating the efficiency of the network as defined by Latora and Marchiori (2001) after the removal. When removing elements in the order of their decreasing centrality from the network, we did not re-calculate the centrality values after each removal. While this may sound as a large simplification at the first time, it is worth to consider that in a real case, a rather voluminous dynamics can be observed in the real network after each removal, which rearranges the links of the network rather significantly. Thus, lacking simulations reflecting the possible network dynamics, the starting model itself is already simple enough to accommodate this further simplifying assumption for comparative purposes.

The bridgeness centrality measure – for its definition, see Section V.6.d. – of the ModuLand method defined more important elements of the power-grid network than either the degree or the betweenness centrality measures (Figure 4).

Discrimination between date- and party-hubs

We have analyzed the modular structure of the yeast proteome using the high-confidence protein-protein interaction data of Ekman et al. (2006). Our analysis using either the LinkLand or the NodeLand methods (Figure 5A of the main text, Figure S14) uncovered a number of major modules with a well-known biochemical function such as the proteasome cell cycle regulation, ribosome biogenesis, nuclear pore complex, actomyosin, RNA splicing, etc. Importantly, those modules, which were functionally related, such as the proteasome and cell cycle regulation as well as the ribosome biogenesis, nuclear pore complex and RNA splicing showed a large overlap. The functional analysis of the modules showed several, novel interesting features. As an example of this, rather surprisingly, the superoxide dismutase has been recovered as a part of the mitochondrial ribosome (Kovacs et al., 2006; Mihalik et al., 2008). It turned out from a literature search that, in fact, a functional relationship has been uncovered between this enzyme and the mitochondrial ribosome before by Zielinski et al. (2002), which gives an experimental evidence for the functional meaning of the ModuLand-based module membership assignment in a biological network. We have also identified characteristic changes in the yeast proteome after stress (Mihalik et al., 2008; Palotai et al., 2008), which were in agreement with the expectations (Szalay et al., 2007).

The dissection of date- and party-hubs of protein interaction networks, i.e. proteins sequentially or simultaneously interacting with a large number of neighbors, has been proved notoriously difficult (Ekman et al., 2006; Han et al., 2004; Kim et al., 2006; Komurov and White, 2006) and has been intensively debated (Batada et al., 2006; 2007; Bertin et al., 2007; Agarwal et al., 2009). We were curious, if our modularization method may help to discriminate between these different hub classes. In agreement with earlier findings (Han et al., 2004, Yu et al., 2007; Yu et al., 2008) we assumed that date-hubs should have a more inter-modular position than party-hubs. Date-hubs had a larger bridgeness than party-hubs having a similar centrality (Figure 5 of main text). Similarly, date-hubs had a larger modular overlap than party-hubs of the same degree. We have obtained similar results, if we made the modularization using the NodeLand, LinkLand or Perturland methods of the ModuLand method family (Kovács et al., 2006). As we have described in the main text the bridgeness-based dissection of date- and party hubs misclassified only a single date-hub (201 total) and 28 party-hubs (318 total) of the consensus-based identification of these proteins. The large variability of date- and party-hub identification may come in part by the differences of the initial datasets. As noted by Yu et al. (2008) the two-hybrid methods pick up more inter-modular contacts (date hubs), while the immunprecipitation-based methods enrich intra-modular interactions (party hubs). Our result almost exclusively identifying date-hubs as inter-modular proteins poses this important class as multifunctional, 'moonlighting' proteins (Gianchandani et al., 2006), mediators of cross-modular effects (Andreopoulos et al., 2007), non-hub bottlenecks (Yu et al., 2007) and creative proteins (Csermely, 2008) helping to survive unprecedented, novel challenges, and playing a key role in the

development and evolvability of complex systems. The modular analysis is an important tool to identify these latter, creative elements of complex systems (Csermely, 2008; Kovacs et al., 2006).

Conclusions and perspectives

As a summary, the ModuLand module determination method is a powerful and novel modularization tool, which recovers many results of earlier determinations, uncovers a rich hierarchy of complex networks, provides a set of novel centrality and other measures to characterize network elements and is able to predict the dynamic behavior of network elements from their topology. The method has an important potential to assess network dynamics and evolution (Palla et al., 2007b), to predict missing elements or to identify hidden or mislabeled elements or links (Clauset et al., 2008), to design efficient packet routing algorithms (Danon et al., 2008; Palotai, 2008; Palotai and Csermely, 2009), or to assess the roughness of various landscapes including fracture surfaces (Haavig Bakke and Hansen, 2007) or energy landscapes. The ModuLand method family may also help us to identify the inter-modular contacts and increasing overlaps of the developing brain (Fair et al., 2008) as well as to predict bankruptcies of financial networks (Fujiwara and Aoyama, 2008) as a few of the myriads of exciting applications.

We invite our colleagues to explore these possibilities and offer the method as a freely downloadable software at our web-site, <www.linkgroup.hu/modules.php>. We are happy to list, link and accommodate the upgrades and versions of this method developed by other groups at this platform. Moreover, we highly welcome any results of direct comparison of the more than hundred different modularization techniques described in Table S2 (and obviously those, which we may have left out for which we deeply apologize) using the unifying platform of the ModuLand method for their conversion and direct comparison as described in Section IV.4.

Supplementary References

Adamcsek, B., Palla, G., Farkas, I. J., Derényi, I. and Vicsek, T. CFinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics* **22**, 1021–1023 (2006).

Agarwal, G. and Kempe, D. Modularity-maximizing network communities via mathematical programming. *Eur. Phys. J. B* **66**, 409–418 (2008).

Agarwal, S., Deane, C. M., Porter, M. A. and Jones, N. S. Revisiting date and party hubs: novel approaches to role assignment in protein interaction networks. http://arxiv.org/abs/0911.0408 (2009).

Ahn, Y.-Y., Bagrow, J. and Lehmann, S. Communities and hierarchical organization of links in complex networks. http://arxiv.org/abs/0903.3178 (2009).

Ahnert, S. E., Johnston, I. G., Fink, T. M. A., Doye, J. P. K. and Louis, A. A. Self-assembly, modularity and physical complexity. http://arxiv.org/abs/0912.3464 (2009).

Ahuja, R. K., Magnati, T. L. and Orlin, J. B. *Network Flows: Theory, Algorithms and Application.* (Prentice Hall USA, 1993).

Alba, R. D. A graph theoretic definition of a sociometric clique. *J. Math. Sociol.* **3**, 113–126 (1973).

Alba, R. D. and Moore, G. Elite social circles. *Sociol. Meth. Res.* **7**, 167–188 (1978).

Aldenfelder, M. S. and Blashfield, R. K. *Cluster Analysis*. (Sage, Newbury Park CA, 1984).

Altaf-Ul-Amin, M., Shinbo, Y., Mihara, K., Kurokawa, K. and Kanaya, S. Development and implementation of an algorithm of protein complexes in large interaction networks. *BMC Bioinformatics* **7,** 207 (2006).

Alvarez-Hamelin, I., Dall'Asta, L., Barrat, A. and Vespignani, A. *k*-core decomposition: a tool for the analysis of large-scale Internet graphs. *Adv. Neur. Info. Processing Syst.* **18**, 41–50 (2006).

Alves, N. A. Unveiling community structures in weighted networks. *Phys. Rev. E* **76,** 036101 (2007).

Andrade, R. F. S., Pinho, S. T. R. and Lobao, T. P. Identification of community structure in networks using higher order neighborhood concepts. *Intl. J. Bifurc. Chaos* **19,** 2677–2685 (2009).

Andreopoulos, B., An, A., Wang, X., Faloutsos, M. and Schroeder, M. Clustering by common friends finds locally significant proteins mediating modules. *Bioinformatics* **23**, 1124–1131 (2007).

Angelini, L., Boccaletti, S., Marinazzo, D., Pellicoro, M. and Stramaglia, S. Identification of network modules by optimization of ratio association. *Chaos* **17,** 023114 (2007a).

Angelini, L., Marinazzo, D., Pellicoro, M. and Stramaglia, S. Natural clustering: the modularity approach. *J. Stat. Mech.* L08001 (2007b).

Arenas, A., Danon, L., Díaz-Guilera, A., Gleiser, P. M. and Guimerà, R. Community analysis in social networks. *Eur. Phys. J. B* **38**, 373–380 (2004).

Arenas, A., Díaz-Guilera, A. and Pérez-Vincente, C. J. Synchronization reveals topological scales in complex networks. *Phys. Rev.Lett.* **96**, 114102 (2006).

Arenas, A., Duch, J., Fernández, A. and Gómez, S. Size reduction of complex networks preserving modularity. *New J. Phys.* **9**, 176 (2007).

Arenas, A., Fernández, A., Fortunato, S. and Gómez, S. Motif-based communities in complex networks. *J. Phys. A: Math. Theor.* **41**, 224001 (2008a).

Arenas, A., Fernández, A. and Gómez, S. Multiple resolution of the modular structure of complex networks. *New J. Phys.* **10**, 053039 (2008b).

Arenas, A., Borge-Holthoefer, J. Gómez, S. and Zamora, G. Optimal map of the modular structure of complex networks. http://arxiv.org/abs/0911.2651 (2009).

Arnau, V., Mars, S. and Marín, I. Iterative cluster analysis of protein interaction data. *Bioinformatics* **21,** 364–378 (2005).

Axelsen, J. B., Bernhardsson, S., Rosvall, M., Sneppen, K. and Trusina, A. Degree landscapes in scale-free networks. *Phys. Rev. E.* **74,** 036119 (2006).

Bader, G. D. and Hogue, C. W. V. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics* **4,** 2 (2003).

Bagrow, J. P. Evaluating local community methods in networks. *J. Stat. Mech.* P05001 (2008).

Bagrow, J. P. and Bolt, E. M. A local method for detecting communities. *Phys. Rev. E* **72,** 046108 (2005).

Bahl, L.R., Jelinek, F. and R.L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Analysis Machine Intelligence* **5**, 179–190 (1983).

Bansal, N., Blum, A. and Chawla, S. Correlation clustering. *Mach. Learn.* **56**, 89–113 (2004).

Barabasi, A. L. and Oltvai, Z. N. Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.* **5**, 101–113 (2004).

Barber, M. J. Modularity and community detection in bipartite networks. *Phys. Rev. E* **76,** 066102 (2007).

Barber, M. J. and Clark, J. W. Detecting network communities by propagating labels under constraints. *Phys. Rev. E* **80**, 026129 (2009).
Barnes, E. R. An algorithm for partitioning of nodes of a graph. *SIAM J. Alg. Discr. Meth.* **3**, 541–550 (1982).
Baskerville, K., Grassberger, P. and Paczuski, M. Graph animals, subgraph sampling and motif search in large networks. *Phys. Rev. E* **76,** 036107 (2007).
Batada, N. N., Reguly, T., Breitkreutz, A., Boucher, L., Breitkreutz, B. J., Hurst, L. D. and Tyers, M. Stratus not altocumulus: a new view of the yeast protein interaction network..*PLoS Biol.* **4**, e317 (2006).
Batada, N. N., Reguly, T., Breitkreutz, A., Boucher, L., Breitkreutz, B. J., Hurst, L. D. and Tyers, M. Still stratus not altocumulus: further evidence against the date/party hub distinction. *PLoS Biol.* **5**, e154 (2007).
Baumes, J., Goldberg, M., and Magdon-Ismail. M., Efficient identification of overlapping communities, In: *IEEE Intl. Conf. Intelligence Security Inform. (ISI)* pp. 27–36 (2005a).
Baumes, J., Goldberg, M., Krishnamoorthy, M., Magdon-Ismail. M. and Preston, N. Discovering hidden groups in communication networks. In: *IADIS* (Eds.: Guimaraes, N. and Isaias, P. T.) pp. 97–104 (2005b).
Berry, J. W., Hendrickson, B., La Violette, R. A., Leung, V. J. and Phillips, C. A. Community detection via facility location. http://arxiv.org/abs/0710.3800 (2007).
Berry, J. W., Hendrickson, B., Randall, A., La Violette, R. A. and Phillips, C. A. Tolerating the community detection limit with edge weighting. http://arxiv.org/abs/0903.1072 (2009).
Bertin, N., Simonis, N., Dupuy, D., Cusick, M. E., Han, J. D. J., Fraser, H. B., Roth, F. P. and Vidal, M. Confirmation of organized modularity in the yeast interactome. *PLoS Biol.* **5**, e154 (2007).
Bezdek, J. C. *Pattern recognition with fuzzy objective function algorithms.* Kluwer Academic Publishers (Norwell, USA) (1981).
Bickel, P. J. and Chen, A. A nonparametric view of network models and Newman-Girvan and other modularities. *Proc. Natl. Acad. Sci. USA* **106**, 21068–21073 (2009).
Blatt, M., Wiseman, S. and Domany, E. Superparamagnetic clustering of data. *Phys. Rev.Lett.* **76**, 3251–3254 (1996).
Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech.* P10008 (2008).
Boccaletti, S., Latora, V., Moreno, Y., Chavez, M. and Hwang, D.-U. Complex networks: Structure and dynamics. *Physics Rep.* **424**, 175–308 (2006).
Boccaletti, S., Ivanchenko, M., Latora, V., Pluchino, A. and Rapisarda, A. Detecting complex network modularity with dynamical clustering. *Phys. Rev. E* **75,** 045102 (2007).
Bollobas, B. *Modern graph theory.* Springer Verlag (New York, USA) (1998).
Bonacich, P. Factoring and weighing approaches to clique identification. *J. Math. Sociol.* **2**, 113–120 (1972).
Bonacich, P. Power and centrality: a family of measures. *J. Am. Sociol.* **92**, 1170–1182 (1987).
Borgatti, S. P., Everett, M. G. and Shirey, P. R. LS sets, lambda sets and other cohesive subsets. *Social Networks* **12**, 337–358 (1990).
Brandes, U., Brandes, D., Delling, M., Gaertler, R. Görke, M. Hoefer, Z. Nikoloski and D. Wagner. On finding graph clusterings with maximum modularity. *LNCS* **4769**, 121–132 (2007).
Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine. *Comp. Networks ISDN Systems* **30,** 107–117 (1998).
Brusco, M. J. and Köhn, H.-F. Comment on "Clustering by passing messages between data points". *Science* **319**, 726c (2008).
Bu, D., Zhao, Y., Cai, L., Xue, H., Zhu, X., Lu, H., Zhang, J., Sun, S., Ling, L., Zhang, N., Li, G. and Chen, R. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Res* **31**, 2443–2450 (2003).
Burt, R.S. *Structural Holes: The Social Structure of Competition*. Harvard University Press, Cambridge (1995).
Capocci, A., Servedio, V. D. P., Caldarelli, G. and Colaiori, F. Detecting communities in large networks. *Physica A* **352,** 669–676 (2005).
Carchiolo, V., Malgeri, M., Longheu, A., Mangioni, G. Search for overlapped communities by parallel genetic algorithms. *(IJCSIS) Intl. J. Comp.Sci.Inform.Secur.* **6**, 2 (2009).
Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y. and Shir, E. A model of Internet topology using *k*-shell decomposition. *Proc. Natl. Acad. Sci. USA* **104**, 11150–11154 (2007).
Carmi, S., Krapivsky, P. L. and ben-Avraham, D. Partition of networks into basins of attraction. *Phys. Rev. E* **78**, 066111 (2008).

Chauhan, S., Girvan, M. and Ott, E. Using network function to define and identify community structure. http://arxiv.org/abs/0911.2735 (2009).
Cheng, X-Q. and Shen, H.-W. Uncovering the community structure associated with the diffusion dynamics of networks. http://arxiv.org/abs/0911.2308 (2009).
Chin, C. S. and Samanta, M. P. Global snapshot of a protein interaction network – a percolation based approach. *Bioinformatics* **19**, 2413–2419 (2003).
Cho, Y. R., Hwang, W. and Zhang, A. Identification of overlapping functional modules in protein interaction networks: information flow-based approach. *Proc. 6th IEEE Intl. Conf. Data Mining* 147–152 (2006).
Cho, Y. R., Hwang, W., Ramanathan, M.and Zhang, A. Semantic integration to identify overlapping functional modules in protein interaction networks. *BMC Bioinformatics* **8**, 265 (2007).
Chung, F. The heat kernel as the pagerank of a graph. *Proc. Natl. Acad. Sci. USA* **104**, 19735–19740 (2007).
Clauset, A. Finding local community structure in networks. *Phys. Rev. E* **72**, 026132 (2005).
Clauset, A., Newman, M. E. J. and Moore, C. Finding community structure in very large networks. *Phys. Rev. E* **70**, 066111 (2004).
Clauset, A., Moore, C. and Newman, M. E. J. Hierarchical structure and prediction of missing links in networks. *Nature* **453**, 98–101 (2008).
Čopič, J., Jackson, M. O. and Kirman A. Identifying community structures from network data via maximum likelihood methods. http://www.stanford.edu/~jacksonm/netcommunity.pdf (2005).
Csermely, P. *Weak links: Stabilizers of Complex Systems from Proteins to Social Networks.* Springer Verlag, Heidelberg (2006).
Csermely, P. Creative elements: network-based predictions of active centres in proteins, cellular and social networks. *Trends Biochem. Sci.* **33,** 569–576 (2008).
da Fontoura Costa, L. Hub-based community finding. http://arxiv.org/abs/cond-mat/0405022 (2004).
da Fontoura Costa, L. Detecting neuronal communities from beginning of activation patterns. http://arxiv.org/abs/0801.4269 (2008a).
da Fontoura Costa, L. Communities in neuronal complex networks revealed by activation patterns. http://arxiv.org/abs/0801.4684 (2008b).
da Fontoura Costa, L. and Rodrigues, F. A. What is there between two nodes of a complex network? http://arxiv.org/abs/0801.4068 (2008a).
da Fontoura Costa, L. and Rodrigues, F. A. Trees = networks ?!? http://arxiv.org/abs/0808.0730 (2008b).
Danon, L., Duch, J., Díaz-Guilera, A. and Arenas, A. Comparing community structure identification. *J. Stat. Mech.* P09008 (2005).
Danon, L., Díaz-Guilera, A. and Arenas, A. The effect of size heterogeneity on community identification in complex networks. *J. Stat. Mech.* P11010 (2006).
Danon, L., Arenas, A. and Díaz-Guilera, A. Impact of community structure on information transfer. *Phys. Rev. E.* **77**, 036103 (2008).
Delvenne, J.-C., Yaliraki, S. N. and Barahona, M. Stability of graph communities across time scales. http://arxiv.org/abs/0812.1811 (2008).
Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1S, 269–271 (1959).
Djidjev, H. N. A scalable multilevel algorithm for graph clustering and community structure detection. *LNCS* **4936**, 117–128 (2008).
Donath, W. E. and Hoffman, A.J. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connections matrices. *IBM Techn. Disclosure Bull.* **15**, 938–944 (1972).
Donetti, L. and Munoz M. A. Detecting network communities: a new systematic and efficient algorithm. *J. Stat. Mech.* P10012 (2004).
Donetti, L. and Munoz M. A. Improved spectral algorithm for the detection of network communities. *AIP Conf. Proc.* **779,** 104–107 (2005).
Dorogovtsev, S. N., Goltsev, A. V. and Mendes, J. F. F. *k*-core percolation and *k*-core organization on complex networks. *Phys Rev Lett* **96,** 040604 (2006).
Dorow, B., Widdows, D., Ling, K., Eckmann, J-P., Sergi, D. and Moses, E. Using curvature and Markov clustering in graphs for lexical and word-sense discrimination. http://arxiv.org/abs/cond-mat/0403693 (2004).
Du, N., Wu, B., Wang, B. and Wang, Y. Overlapping community detection in bipartite networks. *IEEE/WIC/ACM 2008*, 176–179 (2008).
Duch, J. and Arenas, A. Community detection in complex networks using extremal optimization. *Phys. Rev. E* **72**, 027104 (2005).

Duff, I.S., Erisman, A.M. and Reid, J. K. *Direct Methods for Sparse Matrices*, Oxford: Clarendon Press (1986).
Dunn, J. C. A fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters. *J. Cybernetics* **3**, 32–57. (1974).
E, W., Li, T. and Vanden-Eijnden, E. Optimal partition and effective dynamics of complex networks. *Proc. Natl. Acad. Sci. USA* **105**, 7907–7912 (2008).
Eckmann, J-P. and Moses, E. Curvature of co-links uncovers hidden thematic layers in the World Wide Web. *Proc. Natl. Acad. Sci. USA* **99,** 5825–5829 (2002).
Edachery, J., Sen, A. and Brandenburg, F. J. Graph clustering using distance-k cliques. *LNCS* **1731**, 98–106 (1999).
Ekman, D., Light, S., Björklund, A. K. and Elofsson, A. What properties characterize the hub proteins of the protein-protein interaction network of *Saccharomyces cerevisiae*? *Genome Biol.* **7**, R45 (2006).
Elias, P., Feinstein, A. and Shanon, C. E. A note on the maximum flow through a network. *IEEE Trans. Info. Theor.* **2,** 117–119 (1956).
Enright, A. J. van Dongen, S. and Ouzounis, C. A. An efficient algorithm for large-scale detection of protein families. *Nucl. Ac. Res* **30**, 1575–1584 (2002).
Eriksen, K. A., Simonsen, I., Maslov, S. and Sneppen, K. Modularity and extreme links of the Internet. *Phys. Rev. Lett.* **90**, 148701 (2003).
Estrada, E. and Hatano, N. Communicability in complex networks. *Phys. Rev. E* **77**, 036111 (2008).
Estrada, E. and Hatano, N. Communicability graph and community structures in complex networks. *Appl. Math. Comput.* **214**, 500-511 (2009).
Estrada, E. and Rodríguez-Velázquez, J.A. Subgraph centrality in complex networks. *Phys. Rev. E* **71**, 056103 (2005).
Estrada, E., Higham, D. J. and Hatano, N. Communicability betweenness in complex networks. *Phys. A* **388**, 764–774 (2008).
Evans, T. S. and Lambiotte, R. Line graphs, link partitions, and overlapping communities. *Phys.Rev.E,* **80**, 016105 (2009a).
Evans, T. S. and Lambiotte, R. Edge partitions and overlapping communities in complex networks. http://arxiv.org/abs/0912.4389 (2009b).
Fair, D. A., Cohen, A. L., Dosenbach, N. U., Church, J. A., Miezin, F. M., Barch, D. M., Raichle, M. E., Petersen, S. E. and Schlaggar, B. L. The maturing architecture of the brain's default network. *Proc. Natl. Acad. Sci. USA* **105**, 4028–4032 (2008).
Farkas, I., Ábel, D., Palla, G. and Vicsek, T. Weighted network modules. *New J. Phys.* **9,** 180 (2007).
Flake, G. W., Lawrence, S., Giles, L. and Coetze, F. M. Self-organization and identification of web-communities. *IEEE Computer* **35**, 66–71 (2002).
Floyd, R. W. Algorithm 97. *Comm. ACM* **5-6**, 345 (1962).
Fortunato, S. Quality functions in community detection. *Proc. SPIE* **6601**, 660108 (2007).
Fortunato, S. Community detection in graphs. *Phys. Rep.* **486**, 75-174 (2010).
Fortunato, S. and Barthélemy, M. Resolution limit in community detection. *Proc. Natl. Acad. Sci. USA* **104,** 36–41 (2007).
Fortunato, S. and Castellano, C. Community structure in graphs. http://arxiv.org/abs/0712.2716 (2007).
Fortunato, S., Latora, V. and Marchiori, M. Method to find community structures based on information centrality. *Phys. Rev. E* **70**, 056104 (2004).
Freeman, L. C. Centrality in social networks conceptual clarification. *Social Networks* **1**, 215–239 (1978/79).
Frey, B. J. and Dueck, D. Clustering by passing messages between data points. *Science* **315**, 972–976 (2007).
Frey, B. J. and Dueck, D. Response to comment on "Clustering by passing messages between data points". *Science* **319**, 726d (2008).
Fujiwara, Y. and Aoyama, H. Large-scale structure of a nation-wide production network. http://arxiv.org/abs/0806.4280 (2008).
Gaertler, M., R. Gorke and Wagner, D. Significance-driven graph clustering. *LNCS* **4508,** 11–26 (2007).
Gansner, E. R. and North, S. C. An open graph visualization system and its applications to software engineering. *Softw. Pract. Exp.* **30**, 1203–1233 (1999).
Gfeller, D., Chappelier, J. C. and De Los Rios, P. Finding instabilities in the community structure of complex networks. *Phys. Rev. E* **72,** 056135 (2005).
Gfeller, D., De Los Rios, P., Caflish, A. and Rao, F. Complex network analysis of free energy landscapes. *Proc. Natl. Acad. Sci. USA* **104,** 1817–1822 (2007).

Ghosh, R. and Lerman, K. Community detection using a measure of global influence. http://arxiv.org/abs/0805.4606 (2008).
Gianchandani, E. P., Brautigan, D. L. and Papin, J. A. Systems analyses characterize integrated functions of biochemical networks. *Trends Biochem Sci* **31**, 284–291 (2006).
Gibson, D., Kleinberg, J. and Raghavan, P. Inferring web-communities from link topology. In: Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space – structure in hypermedia systems: links, objects, time and space---structure in hypermedia systems (Pittsburgh PA) 225–234 (1998).
Girvan, M. and Newman, M. E. J. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **99,** 7821–7826 (2002).
Gómez, S., Jensen, P. and Arenas, A. Analysis of community structure in networks of correlated data. *Physical Review E,* **80** 016114 (2009).
González, M. C., Lind, P. G. and Herrmann, H. J. System of mobile agents to model social networks. *Phys. Rev. Lett.* **96**, 088702 (2006).
González, M. C., Herrmann, H. J., Kertész, J. and Vicsek, T. Community structure and ethnic preferences in school friendship networks. *Physica A* **379**, 307–316 (2007).
Good, B. H., de Montjoye, Y.-A. and Clauset, A. The performance of modularity maximization in practical contexts. http://arxiv.org/abs/0910.0165 (2009).
Gosh, R. and Lerman, K. Structure of heterogeneous networks. *Proc. Intl. Conf. Comput. Sci. Eng,* **4**, 98–105 (2009).
Gregory, S. An algorithm to find overlapping community structure in networks. In: *Proc. 11th Eur. Conf. Principles Practice Knowledge Discov. Databases (PKDD)* Springer Verlag, Berlin, pp. 91–102 (2007).
Gregory, S. Finding overlapping communities in networks by label propagation. http://arxiv.org/abs/0910.5516 (2009).
Gregory, S. Finding overlapping communities using disjoint community detection algorithms. In: *Complex Networks.* (Eds.: Fortunato, S., Menezes, R., Mangioni, G. and Nicosia, V.) Springer Verlag, Berlin, *Studies Comput. Intelligence* vol. **207** pp. 47–62 (2009).
Grendar, M. Entropy and effective support size. *Entropy* **8**, 169–174 (2006).
Gudkov, V., Montealegre, V., Nussinov, S. and Nussinov, Z. Community detection in complex networks by dynamical simplex evolution. *Phys. Rev. E* **78**, 016113 (2008).
Guimerà R. and Amaral, L. A. N. Functional cartography of complex metabolic networks. *Nature* **433,** 895–900 (2005).
Guimera, R., Danon, L., Díaz-Guilera, A., Giralt, F. and Arenas, A. Self-similar community structure in a network of human interactions. *Phys. Rev. E* **68**, 065103 (2003).
Guimera, R., Sales-Pardo, M. and Amaral, L. A. N. Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E* **70**, 025101 (2004).
Gustafsson, M., Hörnquist, M. and Lombardi, A. Comparison and validation of community structures in complex networks. *Physica A* **367,** 559–576 (2006).
Haavig Bakke, J. O. and Hansen, A. The accuracy of roughness exponent measurement methods. *Phys. Rev. E* **76,** 031136 (2007).
Habib, M. and Paul, C. A Survey on algorithmic aspects of modular decomposition. http://arxiv.org/abs/0912.1457 (2009).
Hager, W. W., Phan, D. T., and Zhang, H. An exact algorithm for graph partitioning. http://arxiv.org/abs/0912.1664 (2009).
Han, J.-D. J., Bertin, N., Hao, T., Goldberg, D. S., Berriz, G. F., Zhang, L. V., Dupuy, D., Walhout, A. J. M., Cusick, M. E., Roth, F. P. and Vidal, M. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature* **430,** 88–93 (2004).
Hartuv, E. and Shamir, R. A clustering algorithm based on graph connectivity. *Info. Proc. Lett.* **76,** 175–181 (2000).
Hartwell, L. H., Hopfield, J. J., Leibler, S. and Murray, A. W. From molecular to modular cell biology. *Nature* **402,** C47–C52 (1999).
Hastings, M. B. Community detection as an interference problem. *Phys. Rev. E* **74**, 035102 (2006).
Heimo, T., Tibély, G., Saramäki, J., Kaski, K. and Kertész, J. Spectral methods and cluster structure in correlation-based networks. *Physica A* **387,** 5930–5945 (2008a).
Heimo, T., Kumpula, J. M., Kaski, K. and Saramäki, J. Detecting modules in dense weighted networks with the Potts method. *J. Stat. Mech.* P08007 (2008b).
Hildebrand, R. Identification of community structure in networks with convex optimization. http://arxiv.org/abs/0806.1896 (2008).
Hinne, M. Local identification of web-graph communities. *ICTIR* 261–278 (2007).

Hofman, J. M. and Wiggins, C. H. A Bayesian approach to network modularity. *Phys. Rev. Lett.* **100**, 258701 (2008).
Homans, G. C. *The human groups.* Harcourt, Brace & Co. New York NY USA (1950).
Hu, X., Sokhansanj, B., Wu, D. And Tang, Y. A novel approach for mining and fuzzy simulation of subnetworks from large biomolecular networks. *IEEE Trans. Fuzzy Syst.* **15**, 1219–1229 (2007).
Hu, Y., Li, M., Zhang, P., Fan, Y. and Di, Z. Community detection by signalling on complex networks. *Phys. Rev. E* **78**, 016115 (2008a).
Hu, Y., Chen, H., Zhang, P., Li, M., Di, Z. and Fan, Y. Comparative definition of community and corresponding identifying algorithm. *Phys. Rev. E* **78**, 026121 (2008b).
Hu, Y., Nie, Y., Yang, H., Cheng, J., Fan, Y. and Di, Z. Measuring structure in complex networks. http://arxiv.org/abs/0902.3331 (2009).
Hu, Y., Ding, Y., Fan, Y. and Di, Z. How to measure singificance of community structure in complex networks. http://arxiv.org/abs/1002.2007 (2010).
Hwang, W., Cho, Y. R., Zhang, A. and Ramanathan, M. A novel functional module detection algorithm for protein-protein interaction networks. *Algorithms Mol. Biol.* **1**, 24 (2006).
Hwang, W., Cho, Y. R., Zhang, A. and Ramanathan, M. CASCADE: a novel quasi all paths-based network analysis algorithm for clustering biological interactions. *BMC Bioinformatics* **9**, 64 (2008).
Ispolatov, I., Mazo, I. and Yuryev, A. Finding mesoscopic communities in sparse networks. *J. Stat. Mech.* P09014 (2006).
Jain, A. K., Topchy, A. K., Law, M. H. C. and Buchmann, J. M. Landscape of clustering algorithms. In: *Proc. 17th ICPR, Cambridge UK* (Kittler, J., Petrou, M. and Nixon, M. S., eds.) **1**, 260–263 (2004).
Jelinek, F., Mercer, R.L., Bahl, R.L. and Baker, J.K. Perplexity – *A* measure *of* difficulty of speech recognition tasks. *J. Acoust. Soc. Am.* **62**, S63 (1973).
Jin, R., McCallen, S., Liu, C.-C., Xiang, Y., Almaas, E. and Zhou, X. J. Identifying dynamic network modules with temporal and spatial constraints. *Pac. Symp. Biocomput.* 203–214. (2009)
Johnson, S. Hierarchical clustering schemes. *Psychometrika* **32**, 241–254 (1967).
Jordán, F. and Scheuring, I. Network ecology: topological constraints on ecosystems dynamics. *Physics Life Rev.* **1**, 139–172 (2004).
Kaplan, T.D. and Forrest, S. A dual assortative measure of community structure. http://arxiv.org/abs/0801.3290 (2008).
Karrer, B., Levina, E. and Newman, M. E. J. Robustness of community structure in networks. *Phys. Rev. E* **77**, 046119 (2008).
Katz, L. A new status index derived from sociometric analysis. *Psychometrika* **18**, 39–40 (1953).
Kelsic, E. D. Understanding complex networks with community-finding algorithms. *SURF 2005 Final Report*, Caltech, www.its.caltech.edu/~mason/research/kelsic.pdf (2005).
Kim, P. M., Lu, L. J., Xia, Y. and Gerstein, M. B. Relating three-dimensional structures to protein networks provides evolutionary insights. *Science* **314**, 1938–1941 (2006).
Kim, C., Cheon, M., Kang, M. and Chang, I. A simple and exact Laplacian clustering of complex networking phenomena: application to gene expression profiles. *Proc. Natl. Acad. Sci. USA* **105**, 4083–4087 (2008).
Kim, Y., Son, S.-W. and Jeong, H. Link Rank: finding communities in directed networks. *Phys. Rev. E* **81**, 016103 (2010).
King, A. D., Pržulj, N. and Jurisica, I. Protein complex prediction via cost-based clustering. *Bioinformatics* **20,** 3013–3020 (2004).
Kitsak, M., Riccaboni, M., Havlin, S., Pammolli, F. and Stanley, H. E. Structure of business firm networks and scale-free models. http://arxiv.org/abs/0810.5514 (2008).
Kleinberg, J. Authoritative sources in a hyperlinked environment. *IBM Research Report RJ* 10076(91892) (1997).
Komurov, K. and White, M. Revealing static and dynamic modular architecture of the eukaryotic protein interaction network. *Mol. Syst. Biol.* **3**, 110 (2007).
Korn, A., Schubert, A. and Telcs, A. Lobby index in networks. *Physica A* **388,** 2221–2226 (2009).
Kovacs, I., Csermely, P., Korcsmaros, T. and Szalay, M. Method for analyzing the fine structure of networks. *Patent application* WO 2007/093960 (2006).
Kraskow, A. and Grassberger, P. MIC: mutual information based hierarchical clustering. In: Information Theory and Statistical Learning. Frank Emmert-Streib and Matthias Dehmer (Eds.) Springer Verlag, New York, pp. 101–123 (2009).
Krawczyk, M. J. and Kulakowski, K. Communities in networks – a continuous approach. http://arxiv.org/abs/0709.0923 (2008).

Krioukov, D., Papadopoulos, F., Boguná, M. and Vahdat, A. Efficient navigation in scale-free networks embedded in hyperbolic metric spaces. http://arxiv.org/abs/0805.1266 (2008).
Kumpula, J. M., Saramäki, J., Kaski, K. and Kertész, J. Limited resolution in complex network community detection with Potts model approach. *Eur. Phys. J. B* **56,** 41–45 (2007).
Kumpula, J. M., Kivelä, M., Kaski, K. and Saramäki, J. A sequential algorithm for fast clique percolation. *Phys. Rev. E* **78**, 026109 (2008).
Kwak, H., Eom, Y.-H., Choi, Y., Jeong, H. and Moon, S. Consistent community identification in complex networks. http://arxiv.org/abs/0910.1508 (2009).
Lambiotte, R., Delvenne, J.-C. and Barahona, M. Laplacian dynamics and multiscale modular structure in networks. http://arxiv.org/abs/0812.1770 (2008).
Lancichinetti, A. and Fortunato, S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E* **80**, 016118 (2009a).
Lancichinetti, A. and Fortunato, S. Community detection algorithms: a comparative analysis. *Phys. Rev. E* **80**, 056117 (2009b).
Lancichinetti, A., Fortunato, S. and Radicchi, F. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**, 046110 (2008).
Lancichinetti, A., Fortunato, S. and Kertész, J. Detecting the overlapping and hierarchical community structure of complex networks. *New J. Phys.* **11**, 033015 (2009a).
Lancichinetti, A., Radicchi, F. and Ramasco, J.J. Statistical significance of communities in networks. http://arxiv.org/abs/0907.3708 (2009b).
Latora, V. and Marchiori, M. Efficient behavior of small-world networks. *Phys. Rev. Lett.* **87**, 198701 (2001).
Latora, V. and Marchiori, M. A measure of centrality based on network efficiency. *New J. Phys.* **9,** 188 (2007).
Lázár, A., Ábel, D. and Vicsek, T. Modularity measure of networks with overlapping communities. http://arxiv.org/abs/0910.5072 (2009).
Lee, S. H., Kim, P.-J., Ahn, Y.-Y. and Jeong, H. Googling hidden interactions: Web search engine based weighted network construction. http://arxiv.org/abs/0710.3268 (2009a).
Lee, S., Yook, S.-H. and Kim, Y. Centrality measure of complex networks using biased random walks. *Eur. Phys. J. B* **68,** 277–281 (2009b).
Lehmann, S. and Hansen, L. K. Deterministic modularity optimization. *Eur. Phys. J. B* **60**, 83–88 (2007).
Lehmann, S., Schwartz, M. and Hansen, L. K. Bi-clique communities. *Phys. Rev. E* **78**, 016108 (2008).
Leicht, E. A. and Newman, M. E. J. Community structure in directed networks. *Phys. Rev. Lett.* **100**, 118703 (2008).
Leicht, E. A., Holme, P. and Newman, M. E. J. Vertex similarity in networks. *Phys. Rev. E* **73**, 026120 (2006).
Leone, M., Sumedha and Weigt, M. Unsupervised and semi-supervised clustering by message passing: soft-constraint affinity propagation. *Eur. Phys. J. B* **66**, 125–135 (2008).
Lerman, K. and Ghosh, R. Parameterized centrality for network analysis. http://arxiv.org/abs/0911.1095 (2009).
Leskovec, J., Lang, K. J., Dasgupta, A. and Mahoney, M. W. Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. http://arxiv.org/abs/0810.1355 (2008).
Leung, J. X. Y., Hui, P., Lio, P. and Crowcroft, J. Towards real-time community detection in large networks. *Phys. Rev. E* **79**, 066107 (2009).
Li, A. and Horvath, S. Network neighborhood analysis with the multi-node topological overlap measure. *Bioinformatics* **23,** 222–231 (2007).
Li, D., Leyva, I., Almendral, J.A., Sendina-Nadal, I., Buldú, J. M., Havlin, S. and Boccaletti, S. Synchronization interfaces and overlapping communities in complex networks. *Phys. Rev. Lett.* **101**, 168701 (2008a).
Li, Q., He, Y. and Jiang, J.-P. A novel clustering algorithm based on a modified model of random walk. http://arxiv.org/abs/0810.5484 (2008b).
Li, Q., He, Y. and Jiang, J.-P. A new clustering algorithm based upon flocking on complex network. http://arxiv.org/abs/0812.5032 (2008c).
Li, Q., Chen, Z., He, Y. and Jiang, J.-P. A novel clustering algorithm based upon games on evolving network. http://arxiv.org/abs/0812.5064 (2008d).
Li, M., Wang, X. and Lai, C.-H. Evolutionary subnetworks in complex systems. http://arxiv.org/abs/0908.2659 (2009).

Li, X., Li, M., Hu, Y, Di, Z., Fan, Y. Detecting community structure from coherent oscillation of excitable systems. *Physica A* **389**, 164–170 (2010).
Liu, J. Detecting the fuzzy clusters of complex networks. *Pattern Recognition* **43**, 1334–1345 (2010).
Liu, X. and Murata, T. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A* **389**, 1493–1500 (2010).
Luce, R. D. Connectivity and generalized cliques in sociometric group structure. *Psychometrika* **15**, 169–190 (1950).
Luce, R. D. and Perry, A. A method of matrix analysis of group structure. *Psychometrika* **14**, 94–116 (1949).
Luo, F., Wang, J. Z. and Promislow, E. Exploring local community structures in large networks. *Web Intelligence* 233–239 (2006).
Lusseau, D., Whitehead, H. and Gero, S. 2008. Incorporating uncertainty into the study of animal social networks. *Animal Behaviour* **75**, 1809–1815 (2008).
Ma, X., Gao, L., Yong, X. and Fu, L. Semi-supervised clustering algorithm for community structure detection in complex networks. *Physica A* **389**, 187–197 (2010)
MacQueen, J. B. Some methods for classification and analysis of multivariate observations. In: *Proc. 5th Berkeley Symp. Math. Statistics Probability* (Eds.: Cam, L. M. L. and Neyman, J.) Univ. California Press, Berkeley USA, vol. **1** pp. 281–297 (1967).
Massen, C. P. and Doye, J. P. K. Identifying "communities" within energy landscapes. *Phys. Rev. E* **71**, 046101 (2005).
Massen, C. P. and Doye, J. P. K. Thermodynamics of community structure. http://arxiv.org/abs/cond-mat/0610077 (2006).
Masuda, N., Kawamura, Y. and Kori, H. Impact of hierarchical modular structure on ranking of individual nodes in directed networks. *New J.Physics,* **11,** 113002 (2009).
Matula, D. W. *k*-components, clusters and slicings in graphs. *SIAM J. Appl. Math.* **22**, 459–480 (1972).
Medus, A. D. and Dorso, C. O. Alternative approach to community detection in networks. *Phys. Rev. E* **79**, 066111 (2009).
Meng Muntz, A. H-Y. and Rezaei, B. A. Method and apparatus for distributed community finding. US patent application US2006/0271564 (2006).
Mete, M., Tang, F., Xu, X. and Yuruk, N. A structural approach for finding modules for biological networks. *BMC Bioinformatics* **9**, S19 (2008).
Mihalik, Á., Palotai, R. and Csermely, P. The effect of stress on the modular structure of yeast protein-protein interaction network. *Biochemistry (Hung.)* **32**, 67 (2008).
Missiuro, P. V., Liu, K., Zou, L., Ross, B. C., Zhao, G., Liu, J. S. and Ge, H. Information flow analysis of interactome networks. *PLoS Comput. Biol.* **5,** e1000350 (2009).
Mitrovic, M. and Tadic, B. Search of weighted subgraphs on complex networks with maximum likelihood methods. *LNCS* **5102**, 551–558 (2008).
Mokken, R. J. Cliques, clubs and clans. *Quality Quantity* **13**, 161–173 (1979).
Moody, J. Race, school integration and friendship segregation in America. *Am. J. Sociol.* **107** 679–716 (2001).
Morarescu, I. C. and Girard, A. Opinion dynamics with decaying confidence: application to community detection in graphs. http://arxiv.org/abs/0911.5239 (2009).
Mucha, P. J., Richardson, T., Macon, K., Porter, M. A. and Onnela, J.-P. Community structure in time-dependent, multiscale, and multiplex networks. http://arxiv.org/abs/0911.1824 (2009).
Muff, S., Rao, F. and Caflisch, A. Local modularity measure for network clusterizations. *Phys. Rev. E* **72**, 056107 (2005).
Mungan, M. and Ramasco, J. J. Who is keeping you in that community? http://arxiv.org/abs/0809.1398 (2008).
Narayan, O. and Sanice, I. The large scale curvature of networks. http://arxiv.org/abs/0907.1478 (2009).
Nelson, D. L., McEvoy, C. L. and Schreiber, T. A. The University of South Florida word association, rhyme and word fragment norms. http://www.usf.edu/FreeAssociation/ (1998).
Nepusz, T., Petróczi, A., Négyessy L. and Bazsó, F. Fuzzy communities and the concept of bridgeness in complex networks. *Phys. Rev. E* **77**, 016107 (2008).
Newman, M. E. J. (2003) The structure and function of complex networks. *SIAM Rev.* **45**, 167–256.
Newman, M. E. J. Fast algorithm for detecting community structure in networks. *Phys. Rev. E* **69**, 066133 (2004a).
Newman, M. E. J. Detecting community structure in networks. *Eur. Phys. J. B* **38,** 321–330 (2004b).
Newman, M. E. J. Analysis of weighted networks. *Phys. Rev. E* **70**, 056131 (2004c).

Newman, M. E. J. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **103**, 8577–8582 (2006a).
Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **74**, 036104 (2006b).
Newman, M. E. J. and Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113 (2004).
Newman, M. E. J. and Leicht, E. Mixture models and exploratory analysis in networks. *Proc. Natl. Acad. Sci. USA* **104**, 9564–9569 (2007).
Nicosia, V., Mangioni, G., Carchiolo, V. and Malgeri, M. Extending the definition of modularity to directed graphs with overlapping communities. *J. Stat. Mech.* P03024 (2009).
Noack, A. and Rotta, R. Multi-level algorithms for modularity clustering. http://arxiv.org/abs/0812.4073 (2008).
Oliviera, S. and Seok, S. C. A multilevel approach to identify functional modules in yeast protein-protein interaction network. *LNCS* **3992**, 726–733 (2006).
Palla, G., Derenyi, I., Farkas, I. and Vicsek, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435,** 814–818 (2005).
Palla, G., Farkas, I. J., Pollner, P., Derenyi, I. and Vicsek, T. Directed network modules. *New J. Phys.* **9,** 186 (2007a).
Palla, G., Barabasi, A.-L. and Vicsek, T. Quantifying social group evolution. *Nature* **446,** 664–667 (2007b).
Palotai, R. Increased network throughput using the module flow routing algorithm. MSc thesis. Technical University of Budapest (2008).
Palotai, R., Szalay, M. S. and Csermely, P. Chaperones as integrators of cellular networks: changes of cellular integrity in stress and diseases. *IUBMB Life* **60**, 10–18 (2008).
Palotai, R. and Csermely, P. (2009) Network modules help the identification of key transport routes, signaling pathways in cellular and other networks. *Ann. Physik* **18**, 822–829 (2009).
Pan, R. K. and Sinha, S. Modular networks with hierarchical organization: the dynamical implications of complex structure. *PRAMANA J. Phys.* **71**, 331–340 (2009).
Papadopoulos, S., Skusa, A., Vakali, A., Kompatsiaris, Y. And Wagner, N. Bridge bounding: a local approach for efficient community dicovery in complex networks. http://arxiv.org/abs/0902.0871 (2009).
Papin, J. A., Reed, J. L. and Palsson, B. O. Hierarchical thinking in network biology: the unbiased modularization of biochemical networks. *Trends Biochem. Sci.* **29**, 641–647 (2004).
Pei, P. and Zhang, A. A "seed-refine" algorithm for detecting protein complexes from protein interaction data. *IEEE Transactions Nanosci.* **6,** 43–50 (2007).
Perra, N., Zlatic, V., Chessa, A., Conti, C., Donato, D. and Caldarelli, G. Schröedinger-like PageRank equation and localization in the WWW. http://arxiv.org/abs/0807.4325 (2008).
Pinney, J. W. and Westhead, D. R. Betweenness-based decomposition methods for social and biological networks. In: *Interdisc. Stat. Bioinf.* (Eds.: Barber, S. Baxter, P.D., Mardia, K. V. and Walls, R. E.) Leeds Univ. Press, Leeds UK, pp. 87–90 (2006).
Pluchino, A., Rapisarda, A. and Latora, V. Communities recognition in the Chesapeake Bay ecosystem by dynamical clustering algorithms based on different oscillators. *Eur. J. Phys. B* **65,** 395–402 (2008).
Pollner, P., Palla, G. and Vicsek, T. Preferential attachment of communities: the same principle, but a higher level. *Europhys. Lett.* **73**, 478 (2006).
Pollner, P., Palla, G., Ábel, D., Vicsek, A., Farkas, I. J., Derényi, I. and Vicsek, T. Centrality properties of directed module members in social networks. *Physica A* **387**, 4959–4966 (2008).
Pons, P. Post-processing hierarchical community structures: quality improvements and multi-scale view. http://arxiv.org/abs/cs.DS/0608050 (2006).
Pons, P. and Latapy, M. Computing communities in large networks using random walks. *LNCS* **3733**, 284–293 (2005).
Porter, M. A., Mucha, P., Newman, M. E. J. and Friend, A. J. Community structure in the United States House of Representatives. *Physica A* **386**, 414–438 (2007).
Porter, M. A., Onnela, J.-P. and Mucha, P. Communities in networks. *Notices Am.Math.Soc.* **56**, 1082–1097, 1164–1166 (2009).
Poyatos, J. F. and Hurst, L. D. How biologically relevant are interaction-based modules in protein networks? *Genome Biol.* **5**, R93 (2004).
Pujol, J. M., Béjar, J. and Delgado, J. Clustering algorithm for determining community structure in large networks. *Phys. Rev. E* **74**, 016107 (2006).

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V. and Parisi, D. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA* **101,** 2658–2663 (2004).
Raghavan, U. N, Albert, R. and Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **76**, 036106 (2007).
Ramani, A. K., Bunescu, R. C., Mooney, R. J. and Marcotte, E. M. Consolidating the set of known human protein-protein interactions in preparation for large-scale mapping of the human interactome. *Genome Biol.* **6,** R40 (2005).
Ramasco, J. J. and Mungan, M. Inversion method for content-based networks. *Phys. Rev. E* **77**, 036122 (2008).
Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. and Barabási, A.-L. Hierarchical organization of modularity in metabolic networks. *Science* **297**, 1551–1555 (2002).
Reichardt, J. and Bornholdt, S. Detecting fuzzy community structures in complex networks with a Potts model. *Phys. Rev. Lett.* **93,** 218701 (2004).
Reichardt, J. and Bornholdt, S. When are networks truly modular? *Physica D* **224,** 20–26 (2006a).
Reichardt, J. and Bornholdt, S. Statistical mechanics of community detection. *Phys. Rev. E* **74,** 016110 (2006b).
Reichardt, J. and Leone, M. (Un)detectable cluster structure in sparse networks. *Phys. Rev. Lett.* **101**, 078701 (2008).
Reichardt, J. and White, D. R. Role models for complex networks. *Eur. Phys. J. B* **60,** 217–224 (2007).
Ren, W., Yan, G., Lin, G., Du, C. and Han, X. Detecting community structure by network vectorization. *LNCS* **5092**, 245–254 (2008).
Ren, W., Yan, G., Liao, X. and Xiao, L. Simple probabilistic algorithm for detecting community structure. *Phys. Rev. E* **79**, 036111 (2009).
Rice, S. A. The identification of blocs in small political bodies *Am. Polit. Sci. Rev.* **21**, 619–627 (1927).
Richardson, T., Mucha, P. J. and Porter, M. A. Beyond bisection: spectral partitioning of networks into multiple communities. *Phys. Rev. E* **80** 036111 (2008).
Rives, A. W. and Galitski, T. Modular organization of cellular networks. *Proc. Natl. Acad. Sci. USA* **100**, 1128–1133 (2003).
Rodrigues, F. A., Travieso, G. and da Fontoura Costa, L. Fast community identification by hierarchical growth. *Int. J. Mod. Phys. C* **18**, 937–947 (2007).
Ronhovde, P. and Nussinov, Z. An improved Potts-model applied to community detection. http://arxiv.org/abs/0803.2548 (2008).
Ronhovde, P. and Nussinov, Z. Multiresolution community detection for mega-scale networks by information-based replica correlations. *Phys. Rev. E* 80, 016109 (2009).
Rosvall, M. and Bergstom, C. T. An information-theoretic framework for resolving community structure in complex networks. *Proc. Natl. Acad. Sci. USA* **104**, 7327–7331 (2007).
Rosvall, M. and Bergstom, C. T. Maps of random walks reveal community structure in complex networks. *Proc. Natl. Acad. Sci. USA* **105**, 1118–1123 (2008a).
Rosvall, M. and Bergstom, C. T. Mapping change in large networks. http://arxiv.org/abs/0812.1242 (2008b).
Ruan, J. and Zhang, W. Identifying network communities with a high resolution. *Phys. Rev. E* **77**, 016104 (2008).
Sahai, T., Speranzon, A. and Banaszuk, A. Hearing the clusters in a graph: a distributed algorithm. http://arxiv.org/abs/0911.4729 (2009).
Sales-Pardo, M., Guimerá, R., Moreira, A. A. and Amaral, L. A. N. Extracting the hierarchical organization of complex systems. *Proc. Natl. Acad. Sci. USA* **104**, 15224–15229 (2007).
Sawardecker, E. N., Sales-Pardo, M. and Amaral, L. A. N. Detection of node group membership in networks with group overlap. *Eur. Phys. J. B* **67**, 277 (2009).
Schuetz, P. and Caflish, A. Efficient modularity optimization: multi-step greedy algorithm and vertex mover refinement. *Phys. Rev. E* **77**, 046112 (2008).
Seidman, S. B. Network structure and minimum degree. *Social Networks* **5**, 269–287 (1983).
Seidman, S. B. and Foster, B. L. A graph-theoretic generalization of the clique concept. *J. Math. Sociol.* **6**, 139–154 (1978).
Shalizi, C. R., Camperi, M. F. and Klinker, K. L. Discovering functional communities in dynamical networks. *LNCS* **4503,** 140–157 (2007).
Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B. and Ideker, T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**, 2498–2504 (2003).
Shen, H., Cheng, X., Cai, K. and Hu, M.-B. Detect overlapping and hierarchical community structure in networks. *Physica A* **388**, 1706–1712 (2008).

Sigman, M. and Cecchi, G. A. Global organization of the Wordnet lexicon. *Proc. Natl. Acad. Sci. USA* **99**, 1742–1747 (2002).
Simmel, G. *Conflict and the web of group affiliations.* Glencoe, Ill. USA, Free Press (1922; 1955).
Simon, H. The architecture of compexity. *Proc. Am. Philos. Soc.* **106**, 467–482 (1962).
Simonsen, I., Eriksen, K. A., Maslov, S. and Sneppen, K. Diffusion on complex networks: a way to probe their large-scale topological structure. *Physica A* **336**, 163–173 (2004).
Slater, P. B. Discernment of hubs and clusters in socioeconomic networks. http://arxiv.org/abs/0807.1550 (2008).
Son, S.-W., Jeong, H. and Noh, J. D. Random field Ising model and community structure in complex networks. *Eur. Phys. J. B* **50,** 431–437 (2006).
Song, C., Havlin, S. and Makse, H.A. Self-assembly of complex networks. *Nature* **433,** 392–395 (2005).
Spirin,V. and Mirny, L. A. Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. USA* **100,** 12123–12128 (2003).
Strehl, A. and Ghosh, J. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *J. Machine Learning Res.* **3,** 583–617 (2002).
Strogatz, S. H. Exploring complex networks. *Nature* **410,** 268–276 (2001).
Šulc, P. and Zdeborova, L. Belief propagation for graph partitioning. http://arxiv.org/abs/0912.3563 (2009).
Sun, J., Faloutsos, C., Papadimitriou, S. and Yu, P. S. GraphScope: parameter-free mining of large time-evolving graphs. In: *Proc. 13th ACM SIGKDD Intl. Conf. Konwledge Discov. Data Mining*, ACM, New York USA, pp. 687–696 (2007).
Sun, Y., Danila, B., Josic, K. and Bassler, K. E. Improved community structure detection using a modified fine tuning strategy. *EPL* **86**, 28004 (2009).
Szalay, M. S., Kovács, I. A., Korcsmáros, T., Böde, C. and Csermely, P. Stress-induced rearrangements of cellular networks: consequences for protection and drug design. *FEBS Lett.* **581**, 3675–3680 (2007).
Tanaka, R., Yi, T.-M. and Doyle, J. Some protein interaction data do not exhibit power law statistics. *FEBS Lett.* **579**, 5140–5144 (2005).
Tanay, A., Sharan, R., Kupiec, M. and Shamir, R. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proc. Natl. Acad. Sci. USA* **101,** 2981–2986 (2004).
Tasgin, M., Herdagdelen, A. and Bingol, H. Community detection in complex networks using genetic algorithms. http://arxiv.org/abs/0711.0491 (2007).
Tibély, G., Onnela, J.-P., Saramäki, J., Kaski, K. and Kertész, J. Spectrum, intensity and coherence in weighted networks of a financial market. *Physica A* **370**, 145–150 (2006).
Tibély, G. and Kertész, J. On the equivalence of the label propagation method on community detection and a Potts model approach. *Physica A* **387**, 4982–4984 (2008).
Traag, V. A. and Bruggeman, J. Community detection in networks with positive and negative links. *Phys. Rev. E* **80**, 036115 (2009).
Ucar, D., Asur, S., Catalyurek, U. and Parthasarathy S. Improving functional modularity in protein-protein interactions graphs using hub-based subgraphs. *LNCS* **4213,** 371–382 (2006).
Vazquez, A. Population stratification using a statistical model on hypergraphs. *Phys. Rev. E* **77**, 066106 (2008a).
Vazquez, A. Bayesian approach to clustering real value, categorical and network data: solution via variational methods. http://arxiv.org/abs/0805.2689 (2008b).
Vlasblom, J., Wu, S., Pu, S., Superina, M., Liu, G., Orsi, C. and Wodak, S. J. GenePro: a Cytoscape plug-in for advanced visualization and analysis of interaction networks. *Bioinformatics* **22**, 2178–2179 (2006).
Vragović, I. and Louis, E. Network community structure and loop coefficient method. *Phys. Rev. E* **74**, 016105 (2006).
Wallace, M. L., Gingras, Y. and Duhon, R. A new approach for detecting scientific specialties from raw cocitation networks. *J. Am. Soc. Info. Sci.***60**, 240–246 (2008).
Walshaw, C. A multi-level algorithm for force-directed graph-drawing. *J. Graph Algorithms Appl.* **7,** 253–285 (2003).
Wang, J. and Lai, C.-H. Detecting groups of similar components in complex networks. *New J. Phys.* **10,** 123023 (2008).
Wang, J. and Lai, C.-H. Symmetry consideration in identifying network structures. http://arxiv.org/abs/0908.0870 (2009).

Wang, K., Zhang, J., Li, D., Zhang, X. And Guo, T. Adaptive affinity propagation clustering. *Acta Automatica Sinica* **33**, 1242–1246 (2007a).
Wang, R. S., Zhang, S. H., Wang, Y., Zhang, X. S. and Chen, L. Clustering complex networks and biological networks by nonnegative matrix factorization with various similarity measures. *Neurocomputing* **72**, 134–141 (2007b).
Wasserman, S. and Faust, K. *Social Network Analysis.* Cambridge Univ. Press, Cambridge (1994).
Watts, D. J. *Small worlds. The dynamics of networks between order and randomness*. Princetown University Press (1999).
Watts, D. J. and Strogatz, S. H. Collective dynamics of ' small-world' networks. *Nature* **393,** 440–442 (1998).
Wei, Y.-C. and Cheng, C.-K. Toward efficient hierarchical designs by ratio cut partitioning. In: *Proc. IEEE Intl. Conf. Computer Aided Design,* IEEE New York, USA, pp. 298–301 (1989).
Weiss, R. S. and Jacobson, E. A method for the analysis of the structure of complex organizations. *Am. Sociol. Rev.* **20,** 661–668 (1955).
White, S. and Smyth, P. A spectral clustering approach to finding communities in graphs. In: Kargupta, H., Srivastava, J., Kamath, C. and Goodman, A. (eds.), *Proceedings of the 5th SIAM International Conference on Data Mining* (Society for Industrial and Applied Mathematics, Philadelphia, 2005).
Wilkinson, D. M. and Huberman, B. A. A method for finding communities of related genes. *Proc. Natl. Acad. Sci. USA* **101,** 5241–5248 (2004).
Wiuf, C., Brameier, M., Hagberg, O. and Stumpf, M. P. H. A likelikood approach to analysis of network data. *Proc. Natl. Acad. Sci. USA* **103**, 7566–7570 (2006).
Wu, F. and Huberman, B. A. Finding communities in time: a physics approach. *Eur. Phys. J. B* **38**, 331–338 (2004).
Wuchty, S. and Almaas, E. Peeling the yeast protein network. *Proteomics* **5**, 444–449 (2005).
Xiang, B., Chen, E.-H. and Zhou, T. Finding community structure based on subgraph similarity. *Studies in Computational Intelligence* 207 73-81 (2009).
Xiaodong, D., Cunrui, W., Xiandong, L. and Yanping, L. Web community detection model using particle swarm optimization. *CEC2008* 1074–1079 (2008).
Xie, F., Ji, M., Zhang, Y. and Huang, D. The detection of community structure in network via an improved spectral method. *Physica A* **388,** 3268–3272 (2009).
Xu, G., Tsoka, S. and Papageorgiou, L. G. Finding community structures in complex networks using mixed integer optimisation. *Eur. Phys. J. B* **60**, 231–239 (2007).
Yang, B. Self-organizing network evolving model for mining network community structure. *LNAI* **4093**, 404–415 (2006).
Yang, B. and Liu, D-Y. A heuristic clustering algorithm for mining communities in signed networks. *J. Comput. Sci. Technol.* **22**, 320–328 (2007).
Yang, S., Luo, S. and Li, J. A novel visual clustering algorithm for finding community in complex network. *LNCS* **4093**, 396–403 (2006).
Yang, B., Liu, J. and Liu, D. An autonomy-oriented computing approach to community mining in dsitributed and dynamic networks. *Auton. Agent Multi-Agent Syst.* **20**, 123–157 (2010).
Yip, A. M. and Horvath, S. Gene network interconnectedness and the generalized topology overlap measure. *BMC Bioinformatics* **8,** 22 (2007).
Young, M., Sager, J., Csárdi, G. and Hága, P. An agent-based algorithm for detecting community structure in networks. http://arxiv.org/abs/cond-mat/0408263 (2004).
Yu, H., Kim, P. M., Sprecher, E., Trifonov, V. and Gerstein M. The importance of bottlenecks in protein networks: correlation with gene essentiality and expression dynamics. *PLoS Comput. Biol.* **3**, e59 (2007).
Yu, H., Barun, P., Yildirim, M. A., Lemmens, I., Venkatesan, K., Sahalie, J., Hirozane-Kishikawa, T., Gebreab, F., Li, N., Simonis, N., Hao, T., Rual, J. F., Dricot, A., Vazquez, A., Murray, R. R., Simon, C., Tardivo, L., Tam, S., Svrzikapa, N., Fan, C., de Smet, A. S., Motyl, A., Hudson, M. E., Park, J., Xin, X., Cusick, M. E., Moore, T., Boone, C., Snyder, M., Roth, F. P., Barabási, A. L., Tavernier, J., Hill, D. E. and Vidal, M. High-quality binary protein interaction map of the yeast interactome network. *Science* **322**, 104–110 (2008).
Zachary, W. W. An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* **33**, 452–473 (1977).
Zanghi, H., Ambroise, C. and Miele, V. Fast online graph clustering via Erdös Renyi mixture. *Pattern Recogn.* **41**, 3592–3599 (2008).
Zarei, M. and Samani, K. A. Eigenvectors of network complement reveal community structure more accurately. *Physica A* **388,** 1721–1730 (2009).

Zarei, M., Samani, K. A. and Omidi, G. R. Complex eigenvectors of network matrices give better insight into the community structure. *J. Stat. Mech. Theor. Exp.* P10018 (2009).
Zhang, B. and Horvath, S. A general framework for weighted gene co-expression network analysis. *Stat. Appl. Gen. Mol. Biol.* **4**, 17 (2005).
Zhang, S. H., Wang, R. S. and Zhang, X-S. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A* **374**, 483–490 (2007a).
Zhang, S., Ning, X-M. and Zhang, X-S. Graph kernels, hierarchical clustering and network community structure: experiments and comparative analysis. *Eur. Phys. J. B* **57**, 67–74 (2007b).
Zhang, S. H., Wang, R. S. and Zhang, X-S. Uncovering fuzzy community structure in complex networks. *Phys. Rev. E* **76**, 046103 (2007c).
Zhang, J., Zhang, S. and Zhang, X-S. Detecting community structure in complex networks based on a measure of information discrepancy. *Physica A* **387**, 1675–1682 (2008).
Zhou H. Network landscape from a Brownian's particle perspective. *Phys. Rev. E* **67**, 041908 (2003).
Zhou H. Distance, dissimilarity index and network community structure. *Phys. Rev. E* **67**, 061901 (2003).
Zhou, H. and Lipowsky, R. Network Brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. *LNCS* **3038**, 1062–1069 (2004).
Zielinski, R., Pilecki, M., Kubinski, K., Zien, P., Hellman, U. and Szyszka R. Inhibition of yeast ribosomal stalk phosphorylation by Cu-Zn superoxide dismutase. *Biochem Biophys Res Commun* **296**, 1310–1316 (2002).
Ziv, E., Middendorf, M. and Wiggins, C. An information-theoretic approach to network modularity. *Phys. Rev. E* **71**, 046117 (2005).
Zotenko, E., Guimarães, K. S., Jothi, R. and Przytycka, T. M. Decomposition of overlapping protein complexes: a graph theoretical method for analyzing static and dynamic protein associations. *Algorithms Mol. Biol.***1**, 7 (2006).
Zubcsek, P. P., Chowdhury, I. and Katona, Z. Information communities: the network structure of communication. http://www.cs.bme.hu/~zskatona/pdf/comm.pdf (2008).