Supporting online material for Mapping change in large networks

M. Rosvall*

*Department of Biology, University of Washington, Seattle, WA 98195-1800**

C. T. Bergstrom†

*Department of Biology, University of Washington, Seattle, WA 98195-1800** and
Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501‡

(Dated: July 6, 2009)

Here we lay out the details of how we generate *significance clusters* and *alluvial diagrams* for mapping change in networks. Because this method assesses how much confidence we should have in the clustering of a network, we can detect, highlight, and simplify the significant structural changes that occur over time or between states in large networks, for example, citation networks, traffic networks, and monetary flow networks. The method consists of four steps, summarized here and described in detail below and in Fig. 1:

1. We partition or "cluster" the original real-world network, assigning each node to a single module, or community of closely associated nodes.

2. We generate a large number ($\sim$ 1000) of bootstrap replicate networks, constructed by parametric bootstrap resampling of the original network. We cluster each of those networks.

3. To identify the significant assignments of nodes to modules in the original network, we search for the largest subset of nodes in a module that co-occur in at least 95% of all bootstrap networks. To identify the significant modules, we search for all modules whose significant nodes are clustered with no other module's significant nodes in at least 95% of all bootstrap networks.

4. To map the changes in the network, we repeat the significance clustering for the different states of the network and generate an alluvial diagram, which highlights and simplifies changes in the significance clusterings.

Significance clustering and alluvial diagrams

This approach to mapping change in large networks works for any clustering algorithm. The choice of algorithm depends on the network type (undirected, directed, unweighted, weighted) and the scope of the study. Here we focus on the general case of weighted directed networks. We also assume that the weight of the links can be described by a Poisson-like process. That is, the weights represent, or can be modeled by, independent events in time. This can be generalized to other distributions of link weights; see section 2 below.

For simplicity of description, here we map the change between two states G^1 and G^2 of a network — but it is straightforward to extend the procedure to more states. We enumerate the N nodes by $\alpha = 1, 2, \ldots, N$. (The set of nodes in G^1 need not be identical to the set in G^2.) By $w_{\alpha\beta}$ we denote a directed link from node α to node β with weight w. Because the significance clustering procedure described below works exactly the same for each particular state of the network, we omit the superscript of G in what follows unless necessary to avoid confusion.

1. Cluster real-world network

We first partition the network G into the modular description M. In the modular description, each node is assigned to one and only one module. The number of modules depends on the network and the objective function of the clustering algorithm. To capture the dynamics across the links and nodes in directed weighted networks, we use the map equation as the objective function (1). In the section *Mapping directed weighted networks* below, we present a new efficient algorithm to search for a partition of the network that minimizes the expected description length of a random walk across the nodes and links of the network. This description length is quantified by the map equation, but the search algorithm can also be generalized for other objective functions.

2. Generate and cluster bootstrap-world networks

The bootstrap is a statistical method for assessing the accuracy of an estimate by resampling from the empirical distribution. This method is particularly powerful when the variance of the estimator cannot be derived analytically or when the underlying distribution is not accessible. Because the cluster assignments are a result of a

*Electronic address: rosvall@u.washington.edu
†Electronic address: cbergst@u.washington.edu
‡URL: http://octavia.zoology.washington.edu/

computational method and the network is idiosyncratic by nature, the bootstrap is indispensable for the process described here.

To generate a single bootstrap replicate network G_b^*, we resample every link weight $w_{\alpha\beta}$ of the original network G from a Poisson distribution with mean equal to the original link weight $w_{\alpha\beta}$. That is, $w_{\alpha\beta}^* \sim \mathrm{Pois}(w_{\alpha\beta})$ for each link in the bootstrap network. Because of the parametric resampling of the link weights, this method formally falls under parametric bootstrapping. If the link weights cannot be modeled by a Poisson process, or if the links are unweighted, the Poisson resampling should be replaced by an appropriate alternative resampling procedure (see for example refs. (2, 3)).

Subsequently we partition the bootstrap replicate network with the same clustering method we used on the original network; this yields the bootstrap modular description M_b^*. This procedure — generating a bootstrap replicate network and clustering it into modules — is repeated to generate a large number $B \sim 1000$ of bootstrap modular descriptions $\mathsf{M}^* = \{\mathsf{M}_1^*, \mathsf{M}_2^*, \ldots, \mathsf{M}_B^*\}$. Each *Bootstrap world* panel in Fig. 1 illustrates four of these modular descriptions for four different bootstrap replicate networks, each created by the Poisson resampling procedure described above. Because approximately 1000 networks must be clustered in this step, we have developed a new fast stochastic and recursive search algorithm for finding an accurate modular description of a given network (see section *Mapping directed weighted networks*).

3. Identify significant assignments

The basic idea behind significance clustering is that we can look at the bootstrap replicates to see which aspects of the modular description of the original network are best supported by the data. Features of the original network that occur in all or nearly all of the bootstrap replicates are well-supported by the data; features that occur in only some of the bootstrap replicates are less well-supported.

What features do we consider? First, we consider the assignment of each node to a module. By looking at the set of bootstrap modular descriptions, we can assess which of these assignments are strongly supported by the data, and which node assignments are less certain. To identify the nodes that are significantly assigned to a module, we search for the largest subset of nodes in each module of the original modular description M that are also clustered together in at least 95% of all bootstrap modular descriptions M^*. To pick the largest subset, of course we need some measure of size. The size of a subset could simply correspond to the number of nodes in the subset, but in line with our general clustering philosophy, we use the volume of flow through the subset (4).

To efficiently search the large space of possible subsets in each cluster, we use simulated annealing (5). Initially the nodes are randomly assigned to be members or non-members of the candidate largest subset. The score S of the configuration is the size of the subset minus a penalty to account for the constraint that only nodes that are clustered together in at least 95% of all bootstrap modular descriptions should be included. To implement the penalty, we first, and for each bootstrap modular description, count the number of nodes in the subset that do not belong to the largest group of nodes assigned to the same cluster. These are the mismatch nodes that break the constraint. To allow for a 5% error, we add together the number of mismatch nodes for all bootstrap modular descriptions, excepting the 5% with the highest number of mismatches. Finally we multiply this sum by ten times the cluster size, to make sure that the subset size and the penalty are of comparable size. This is necessary for an efficient search and a zero penalty at the end of the procedure (this *ad hoc* scaling factor of 10 was found by optimizing the convergence to a configuration with zero penalty and maximal subset size). After initiating with random assignments, we follow the standard simulated annealing scheme (5). At successively lower temperatures T, a node's subset assignment (member or non-member) is flipped and the score S' for the new state is calculated. As in the Metropolis-Hastings algorithm (6, 7), the new state is always accepted if the new score is higher ($\Delta S = S' - S > 0$) or, conversely, if the new score is lower, the new state is accepted with probability equal to the Boltzmann factor of the score difference $\exp(\Delta S/T)$. Starting at $T = 1$, we iterate this step as many times as there are nodes in the cluster, and then reduce the temperature according to $T' = 0.99T$. We repeat this procedure for as long as at least one new state is accepted for a given temperature. The nodes assigned to the subset at the final state serve as our approximation of the largest significant subset.

In addition to telling us about the assignment of individual nodes to specific modules, the set of bootstrap replicates also contains information about which modules stand alone and which are possibly subsets of other modules. To reveal this information, we need to identify the modules that are always, or almost always, separate from any other module. We consider a module to be significant if its significant subset is clustered with no other significant subset in at least 95% of all bootstrap modular descriptions. Conversely, two clusters are mutually nonsignificant if their significant subsets are clustered together in more than 5% of all bootstrap modular descriptions. In this way, each module can be mutually nonsignificant with more than one other module. In the alluvial diagram described in section 4, we want to associate each nonsignificant module with the module together with which it most likely forms a subset. The search for these pairs of modules is straightforward: For each pair of modules, we count in how many bootstrap modular descriptions all nodes in the two significant subsets are clustered together and record this number if it exceeds 5% of all bootstrap modular descriptions (the criterion for nonsignificant modules). Then, starting at the smallest module, we associate the module with the other larger module with which it is most often clustered, and proceed to the next smallest module, and so on.

4. Construct alluvial diagram

To reveal change over time or between states of real-world networks, we summarize the results of the significance clusterings of the different states $G^1, G^2, \ldots$ in an alluvial diagram. The diagram is constructed to highlight the significant changes, fusions, and fissions that the modules undergo between each pair of successive states G^i and G^{i+1}. Each significance clustering for a state G^i occupies a column in the diagram and is horizontally connected to preceding and succeeding significance clusterings by stream fields. Each block in a row of the alluvial diagram represents a cluster, and the height of the block reflects the size of the cluster (here in units of flow through the cluster, though other size measures, such as number of nodes, could be used instead). The clusters are ordered from bottom to top by size, with mutually nonsignificant clusters placed together and separated by a third of the standard spacing. We use a darker color to indicate the significant subset of each cluster. Different colors can be used for clusters or groups of clusters to highlight particular stories in the data.

We use the stream fields to reveal the changes in cluster assignments and in level of significance between two adjacent significance clusterings. The height of a stream field at each end, going from the significant or nonsignificant subset of a cluster in one column to the significant or nonsignificant subset of a cluster in the adjacent column, represents the total size of the nodes that make this particular transition. By following all stream fields from a cluster to an adjacent column, it is therefore possible to study in detail the mergers with other clusters and the significance transitions. To reduce the number of crossing stream fields, the stream fields are ordered by the position of the clusters to which they connect. For smooth transitions, we draw the stream fields with splines and use gradient shading for the component colors. Finally, to reduce the amount of ink and improve clarity, we apply a threshold and do not show the thinnest stream fields.

Mapping directed weighted networks

Here we briefly review our information theoretic approach to revealing community structure in weighted and directed networks (1) and present a new fast stochastic and recursive search algorithm to minimize the map equation — the objective function of our method. We have developed this algorithm to be able to accurately partition the large number of bootstrap networks (8). The search algorithm can also be generalized for other objective functions.

The map equation

The objective of our flow-based and information-theoretic method known as the map equation is to find the structures within a network that are significant with respect to how information or resources flow through that network. For a detailed description of the map equation, see ref. (9). For a dynamic visualization of the mechanics of the map equation, see http://www.tp.umu.se/~rosvall/livemod/mapequation/. The following is a short review.

There is a duality between the problem of compressing a data set, and the problem of detecting and extracting significant patterns or structures within those data (10, 11, 12). We have developed the map equation approach to make use of this duality to detect community structure within directed and weighted networks that inherently are characterized by flow. For a given network partition M, the map equation specifies the theoretical limit $L(\mathsf{M})$ of how concisely we can describe the trajectory of a random walker on the network. The underlying code structure of the map equation is designed such that the description can be compressed if the network has regions in which the random walker tends to stay for a long time. Therefore, with a random walk as a proxy for real flow, minimizing the map equation over all possible network partitions reveals important aspects of network structure with respect to the dynamics on the network.

To take advantage of the regional structure of the network, one index codebook and m module codebooks, one for each module in the network, are used to describe the random walker's movements. The module codebooks have codewords for nodes within each module (and exit codes to leave the module), which are derived from the node visit/exit frequencies of the random walker. The index codebook has codewords for the modules, which are derived from the module switch rates of the random walker. Therefore, the average length of the code describing a step of the random walker is the average length of codewords from the index codebook and the module codebooks weighted by their rates of use. This is the map equation:

$$L(\mathsf{M}) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(\mathcal{P}^i). \tag{1}$$

The first term of this equation gives the average number of bits necessary to describe movement between modules, and the second term gives the average number of bits necessary to describe movement within modules. In the first term, $q_{\curvearrowright}$ is the probability that the random walker switches modules on any given step, and $H(\mathcal{Q})$ is the entropy of the module names, i.e. the frequency-weighted average length of codewords in the index codebook. In the second term, $H(\mathcal{P}^i)$ is the entropy of the within-module movements — including an "exit code" to signify departure from module i, i.e. the frequency-weighted average length of codewords in module codebook i — and

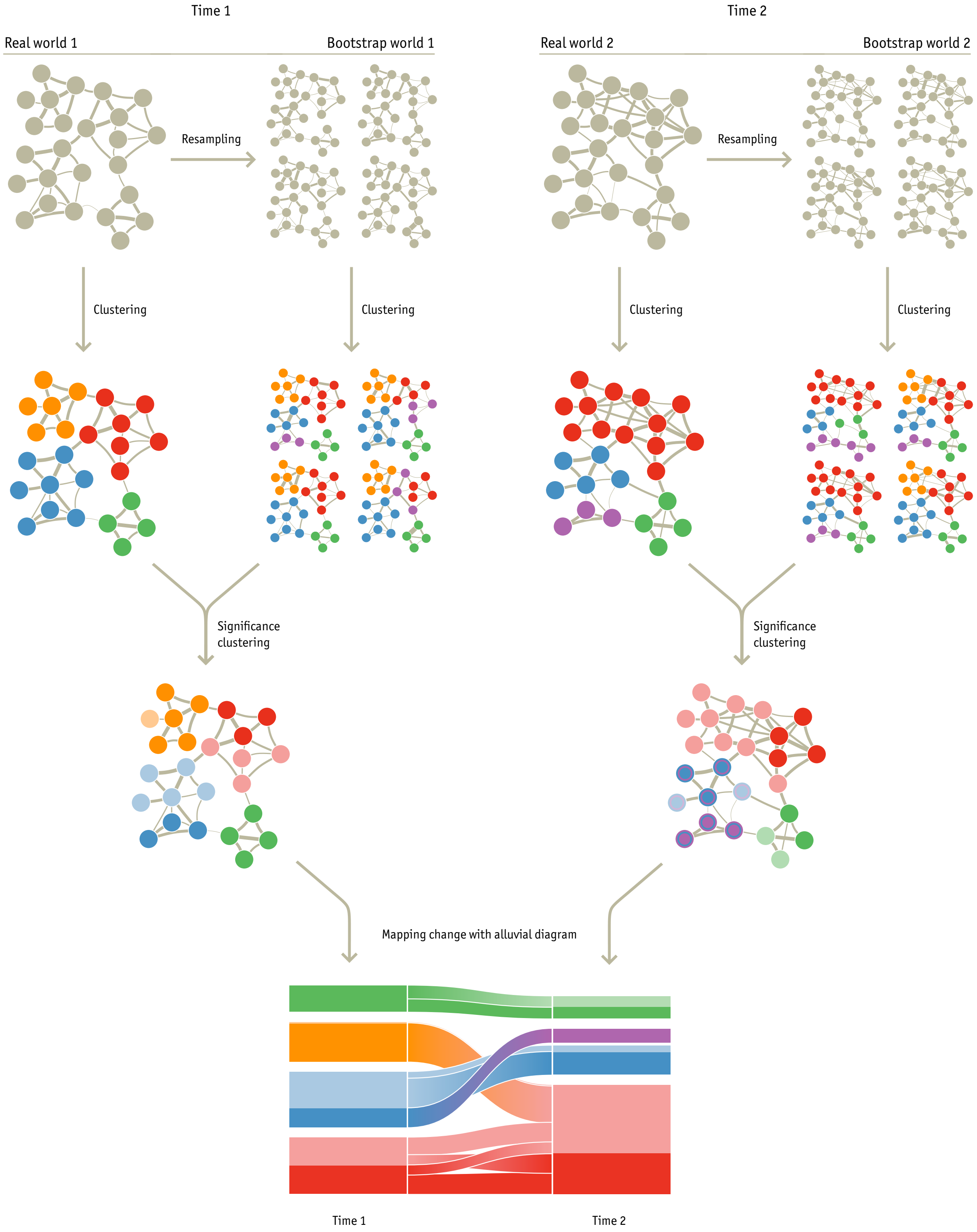


FIG. 1 Significance clustering and alluvial diagram for mapping change in large networks. By repeatedly resampling of the weighted links from the original networks, we create "bootstrap worlds" of 1000 resampled networks. By clustering these bootstrap networks, and comparing to the clustering of the original networks, we can estimate the degree of support that the data provide in assigning each node to a cluster. In the bottom networks, the darker colors represent nodes that are clustered together in at least 95% of the 1000 bootstrap networks. The alluvial diagram highlights and summarizes the structural changes between the time 1 and time 2 significance clusters. The height of each block represents the volume of flow through the cluster (4). The clusters are ordered from bottom to top by their size, with mutually nonsignificant clusters placed together and separated by a third of the standard spacing. The orange module merges with the red module, but the nodes are not clustered together in 95% of the bootstrap networks. The blue module splits, but the significant nodes in the blue and purple modules are clustered together in more than 5% of the bootstrap networks. Neither change is significant.

the weight $p_{\circlearrowright}^{i}$ is the fraction of within-module movements that occur in module i, plus the probability of exiting module i such that $\sum_{i=1}^{m} p_{\circlearrowright}^{i} = 1 + q_{\curvearrowright}$.

To efficiently describe a random walk using a two-level code of this sort, the choice of partition M must reflect the patterns of flow within the network, with each module corresponding to a cluster of nodes in which a random walker spends a long period of time before departing for another module. To find the best such partition, we therefore seek to minimize the map equation over all possible partitions M.

Fast stochastic and recursive search algorithm

Any greedy (fast but inaccurate) or Monte Carlo-based (accurate but slow) approach can be used to minimize the map equation. To provide a good balance between the two extremes, we have developed a fast stochastic and recursive search algorithm, implemented it in C++, and made it available online both for directed and undirected weighted networks (8). As a reference, the new algorithm is as fast as the previous high-speed algorithms (the greedy search presented in the supporting appendix of ref. (1)), which were based on the method introduced in ref. (13) and refined in ref. (14). At the same time, it is also more accurate than our previous high-accuracy algorithm (a simulated annealing approach) presented in the same supporting appendix.

The core of the algorithm follows closely the method presented in ref. (15): neighboring nodes are joined into modules, which subsequently are joined into supermodules and so on. First, each node is assigned to its own module. Then, in random sequential order, each node is moved to the neighboring module that results in the largest decrease of the map equation. If no move results in a decrease of the map equation, the node stays in its original module. This procedure is repeated, each time in a new random sequential order, until no move generates a decrease of the map equation. Now the network is rebuilt, with the modules of the last level forming the nodes at this level. And exactly as at the previous level, the nodes are joined into modules. This hierarchical rebuilding of the network is repeated until the map equation cannot be reduced further. Except for the random sequence order, this is the algorithm described in ref. (15).

With this algorithm, a fairly good clustering of the network can be found in a very short time. Let us call this the core algorithm and see how it can be improved. The nodes assigned to the same module are forced to move jointly when the network is rebuilt. As a result, what was an optimal move early in the algorithm might have the opposite effect later in the algorithm. Because two or more modules that merge together and form one single module when the network is rebuilt can never be separated again in this algorithm, the accuracy can be improved by breaking the modules of the final state of the core algorithm in either of the two following ways:

> *Submodule movements.* First, each cluster is treated as a network on its own and the main algorithm is applied to this network. This procedure generates one or more submodules for each module. Then all submodules are moved back to their respective modules of the previous step. At this stage, with the same partition as in the previous step but with each submodule being freely movable between the modules, the main algorithm is re-applied.
>
> *Single-node movements.* First, each node is reassigned to be the sole member of its own module, in order to allow for single-node movements. Then all nodes are moved back to their respective modules of the previous step. At this stage, with the same partition as in the previous step but with each single node being freely movable between the modules, the main algorithm is re-applied.

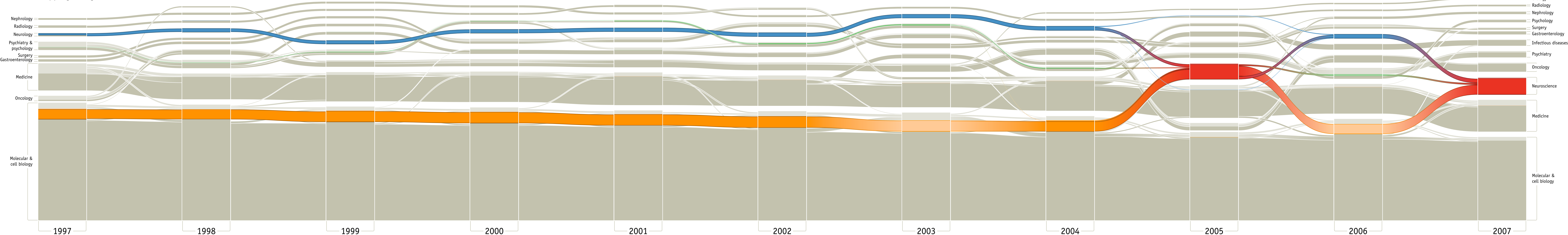
In practice, we repeat the two extensions to the core algorithm in sequence and as long as the clustering is improved. Moreover, we apply the submodule movements recursively. That is, to find the submodules to be moved, the algorithm first splits the submodules into subsubmodules, subsubsubmodules, and so on until no further splits are possible. Finally, because the algorithm is stochastic and fast, we can restart the algorithm from scratch every time the clustering cannot be improved further and the algorithm stops. The implementation is straightforward and, by repeating the search 100 times, the final partition is less likely to correspond to a bad clustering of a local minimum. For each iteration, we record the clustering if the description length is shorter than the previously shortest description length. In practice, for the citation networks presented in this paper, which have on the order of 10,000 nodes and 1,000,000 directed and weighted links, each iteration takes about 5 seconds on a modern PC. We generate the significance clusterings by repeating the algorithm 100 times for each network and bootstrap network.

References

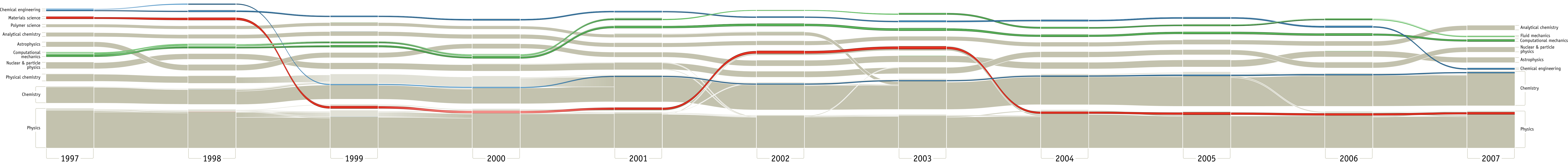
1. M. Rosvall, C. T. Bergstrom, *PNAS* **105**, 1118 (2008).
2. B. Karrer, E. Levina, M. E. J. Newman, *Phys Rev E* **77**, 046119 (2008).
3. D. Gfeller, J.-C. Chappelier, P. D. L. Rios, *Phys Rev E* **72**, 056135 (2005).
4. This is the total *PageRank* of the cluster, which corresponds to the steady-state flow of random walkers that we use in the information-theoretic clustering algorithm.
5. S. Kirkpatrick, J. C. D. Gelatt, M. P. Vecchi, *Science* **220**, 671 (1983).
6. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, *J. Chem. Phys* **21**, 1087 (1953).
7. W. K. Hastings, *Biometrika* **57**, 97 (1970).
8. M. Rosvall (2008). The code can be downloaded from http://www.tp.umu.se/˜rosvall/code.html.

9. M. Rosvall, D. Axelsson, C. Bergstrom, arXiv:0906.1405 (2009).
10. J. Rissanen, *Automatica* **14**, 465 (1978).
11. P. Grünwald, I. J. Myung, M. Pitt, eds., *Advances in minimum description length: theory and applications* (MIT Press, London, England, 2005).
12. C. E. Shannon, W. Weaver, *The mathematical theory of communication* (Univ of Illinois Press, 1949).
13. A. Clauset, M. E. J. Newman, C. Moore, *Phys Rev E* **70**, 066111 (2004).
14. K. Wakita, T. Tsurumi, arXiv:cs/07020480 (2007).
15. V. Blondel, J. Guillaume, R. Lambiotte, E. Mech, *J Stat Mech: Theory Exp* **2008**, P10008 (2008).

Mapping change in medicine 1997–2007



Mapping change in physics & chemistry 1997–2007


Chemical engineering
Materials science
Polymer science
Analytical chemistry
Astrophysics
Computational mechanics
Nuclear & particle physics
Physical chemistry
Chemistry
Physics
Analytical chemistry
Fluid mechanics
Computational mechanics
Nuclear & particle physics
Astrophysics
Chemical engineering
Chemistry
Physics