nature | methods

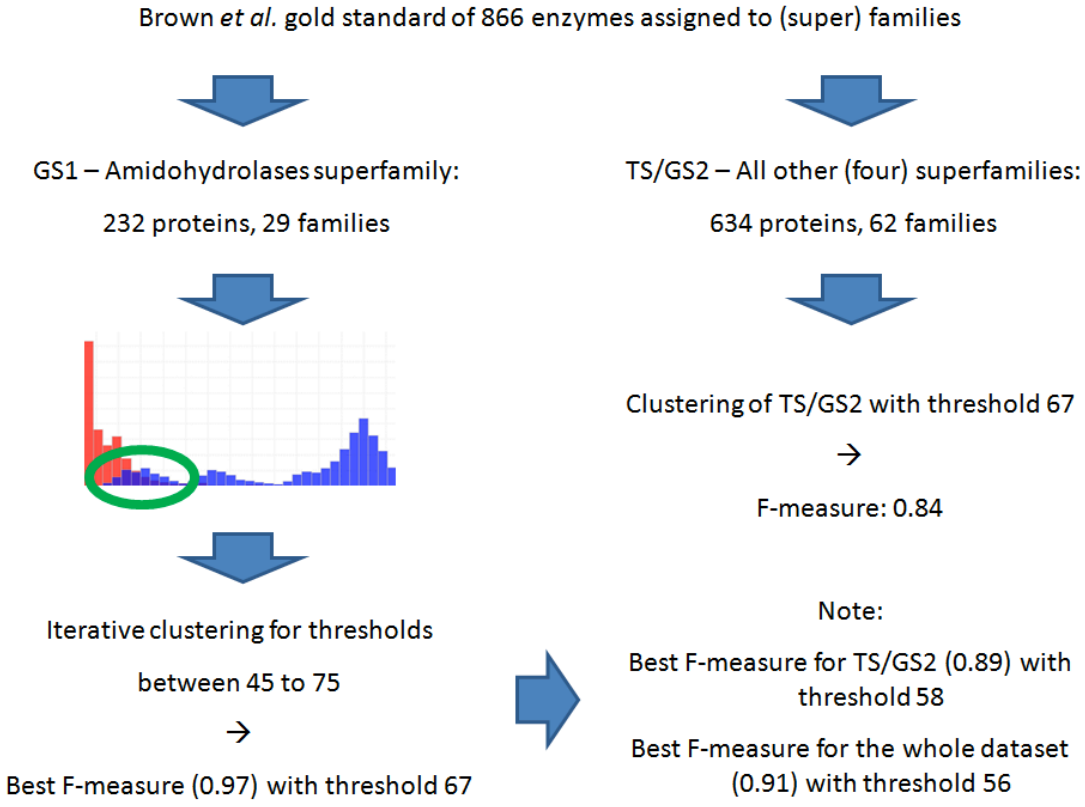
Partitioning biological data with transitivity clustering

Tobias Wittkop, Dorothea Emig, Sita Lange, Sven Rahmann, Mario Albrecht, John H Morris, Sebastian Böcker, Jens Stoye & Jan Baumbach

Supplementary figures and text:

Supplementary Figure 1	Transitivity Clustering application example workflow.
Supplementary Table 1	Main functionalities of the ClusterExplorer TransClust plug-in and its related biological questions.
Supplementary Methods	Detailed description of the used methods.

Supplementary figure 1 | Transitivity Clustering application example workflow. We utilize a manually curated dataset of protein families and superfamilies provided by Brown et al. First, the whole dataset is splitted into a gold standard dataset (GS1) consisting of all 232 protein sequences of the Amidohydrolases superfamily and a test set (TS/GS2), which consists of the remaining 634 amino acid sequences. The goal is to use the 29 known protein families of GS1 to identify a reasonable clustering threshold. This density parameter will be utilized to guess the 62 family assignments of the remaining 634 proteins from TS/GS2. Since we actually know the TS/GS2 family assignments, we may additionally use this application example to evaluate the proposed workflow. The Transitivity Clustering plug-in framework for Cytoscape aids at each step of the data analysis and evaluation. All datasets and tutorials guiding through the depicted workflow are publicly available at the Transitivity Clustering web site (http://transclust.cebitec.uni-bielefeld.de).



Supplementary Table 1 | Main functionalities of the ClusterExplorer TransClust plug-in and its related biological questions. For the case of amino acid sequence clustering and depending on the specific question, the word "cluster" may correspond, for instance to "protein family" or "superfamily".

Functionality	Related (biological) question
Element analysis	
Similarity to all other elements	What are the most similar other proteins to the selected one. Which clusters (families, superfamilies) are they assigned to?
Similarity to all other clusters	If the cluster-assignment for the selected protein is questionable: Which other clusters are reasonable?
Cluster analysis	
Central element of cluster	Which proteins are the best representatives for the selected cluster?
Similarity to all other clusters	Having a selected a protein family (cluster) to be fused with others to form a superfamily, which families are most reasonable?
Similarity to all other elements	The selected cluster is incomplete. Which other proteins may be assigned to this cluster?
Histograms	
Intra- vs. inter-cluster edge weight distribution	What is a promising region for the optimal clustering threshold (Fig. 1 in the article, step 3 and Supplementary Fig. 1).
Cluster size	Plotting the cluster size distribution may help to find a reasonable threshold if no gold standard is available. For example, if the proteome of X organisms is clustered, for a "good" threshold we can expect peaks at cluster sizes X, 2X, 3X, etc.
Comparison	
Cluster comparison	How similar are two classifications? What is the best similarity threshold for clustering, given a gold standard subset (Fig. 1 in the article, step 4)?

Supplementary Methods

This supplementary material is structured as follows: We first describe the calculation of similarity networks for the purpose of amino acid sequence clustering based on all-vs.-all BLAST results. Afterwards, we illustrate the methodology of Transitivity Clustering and compare it to other popular methods. In the last part, we present an application example. Finally, we added supplementary items (one table and one figure).

At the Transitivity Clustering web site we provide access to the software, detailed information about input/output file formats etc. and tutorials guiding step-by-step through several application cases:

http://transclust.cebitec.uni-bielefeld.de

Computing similarity networks from BLAST results

Transitivity Clustering works on any kind of pairwise similarity values. For biological sequences, the standard tool for computing such values is BLAST[1]. BLAST can result in several hits between two proteins *a* and *b*. We denote these hits as $(a \rightarrow b)_i$, $i = 1,\ldots,n$ and $(b \rightarrow a)_j$, $j = 1,\ldots m$, where *n* and *m* are the number of hits for both directions of the comparison. Note that the number of hits as well as the E-values can differ for the two possible directions as BLAST is not symmetric. Since clustering requires a symmetric similarity function, i.e. the similarity of *a* to *b* is equal to the similarity of *b* to *a*, we use the higher E-value of both directions (the lower similarity value; conservative approach). If the BLAST search results in no match between two proteins (in at least one direction) the similarity value is set to zero. Otherwise, we offer two scoring strategies, i.e. two BLAST-based similarity functions:

1. Best Hit (BeH):
Only the lowest E-value of one direction is taken into account and the resulting similarity function for two proteins sequences *a* and *b* with at least one hit in both directions is defined as:

$$sim(a,b) = \min\left(\max_{i=1,\ldots n}(-\log_{10}(\text{E-value}((a \rightarrow b)_i)), \max_{j=1,\ldots m}(-\log_{10}(\text{E-value}((b \rightarrow a)_j))\right)$$

2. Sum of all hits (SoH):
This function takes all hits into account by merging them into a single similarity value. The resulting similarity function for two protein sequences *a* and *b* with at least one hit in both directions is defined as follows:

$$sim(a,b) = \min\left(\sum_{i=1}^{n}(-\log_{10}(\text{E-value}((a \rightarrow b)_i)), \sum_{j=1}^{m}(-\log_{10}(\text{E-value}((b \rightarrow a)_j))\right)$$

In addition to both functions, the coverage of the BLAST hits can be integrated with the similarity; refer to ref. [2] for more details.

Both ways to compute similarity networks, BeH and SoH, are implemented in the whole Transitivity Clustering framework: (1) the Cytoscape environment via the BLAST2SimGraph plug-in, (2) the stand-alone version of Transitivity Clustering, and (3) the web interface.

Note that we also allow the user to set an upper similarity threshold δ. All pairs of objects that exceed a similarity of δ are merged into one object and are automatically assigned to the same cluster. This decreases running time and may also result in increased accuracy.

An alternative tool for all-vs.-all protein sequence similarity computations may be the software FastBLAST[3]. For other application cases than biological sequence clustering, any other kind of similarity network may be utilized as input for Transitivity Clustering.

Transitivity Clustering

Transitivity Clustering is based on the so-called Weighted Transitive Graph Projection (WTGP) problem. Although not within the main focus of this paper, we briefly introduce the WTGP problem.

The transitive graph projection problem was originally defined on simple (unweighted) graphs, but can be extended for weighted graphs. We first need to define transitivity for graphs.

A graph $G = (V, E)$ with nodes V and edges $E \subseteq \binom{V}{2}$ is called transitive if each connected component is a clique, i.e. completely connected. This is equivalent to the intuitive condition for transitivity:

$$\forall u, v, w \in V;\ \{u, v\} \in E \wedge \{v, w\} \in E \Rightarrow \{u, w\} \in E$$

The unweighted transitive graph projection problem is now defined as follows:

Given a graph $G = (V, E)$, find that transitive graph $G' = (V, E')$ such that $|E \setminus E'| + |E' \setminus E|$ is minimal.

The weighted transitive graph projection problem is an extension of the unweighted problem. In this case, the edge changes are not treated equally but depend on edge weights. The objective function to be minimized here is the sum of the weights of all edge modifications.

Transitivity Clustering utilizes the similarities between objects to construct the similarity graph *G*. If an edge similarity exceeds a user-defined threshold (density threshold), the edge is treated as existent and as missing otherwise. Formally: given a set of objects V, a

density threshold $t \in \mathbf{R}$, and a pairwise similarity function $sim : \binom{V}{2} \rightarrow \mathbf{R}$, the graph G is defined as:

$$G = G(V, E) \; ; \; E = \left\{ \{u, v\} \in \binom{V}{2} \;\middle|\; sim(u, v) > t \right\}$$

The goal of the weighted transitive graph projection problem is to determine that transitive graph $G' = (V, E')$, such that there exists no other transitive graph $G'' = (V, E'')$, with $\text{cost}(G \rightarrow G'') < \text{cost}(G \rightarrow G')$. Hereby the modification costs are defined as:

$$\text{cost}(G \rightarrow G') = \underbrace{\sum_{\{u,v\} \in E, \notin E'} \left| sim(\{u, v\}) - t \right|}_{\text{deletion cost}} + \underbrace{\sum_{\{u,v\} \in E', \notin E} \left| sim(\{u, v\}) - t \right|}_{\text{addition cost}}$$

Evidently, it is very expensive to delete edges between highly similar objects or to add edges between very different objects.

Since the NP-hard unweighted problem is a special case of the weighted problem (with similarities of 1 and -1 respectively), the WTGP problem is NP-hard as well. Note: without loss of generality, we may solve the WTGP for all connected components of *G* separately. The advantage here is that TransClust, our Transitivity Clustering implementation, does not have to hold the whole similarity matrix in memory, which makes TransClust both memory- and runtime-efficient in practice. The obtained complete connected components of the resulting transitive graph will subsequently be reported as clusters.

Although not within the focus of this paper, we very briefly describe how we combine different heuristic and exact approaches to solve this computational NP-hard problem within reasonable running time and with good results in practice. First, we utilize an efficient, greedy heuristic called CAST[4] (Cluster Affinity Search Technique) to estimate an upper bound of the emerging costs to solve the WTGP problem for each connected component. With this upper bound, we decide if the respective instance may be used as input for an extended version of the exact fixed-parameter (FP) algorithm published in [5,6]. The running time of the FP approach essentially scales with the necessary costs. Hence, if the CAST-estimated costs are too high to successfully use the exact algorithm, we utilize FORCE[2] instead. It is a heuristic based on graph-layouting that usually provides exact results in practice; although not guaranteed. By using this combination of algorithms and depending on the compute architecture, the available main memory, and the similarity threshold (which is the density parameter), our implementation of Transitivity Clustering, TransClust, is generally able to cluster up to 500,000 amino acid sequences on a desktop PC with >4GB RAM. For large data sets, the running time of TransClust is generally bound by the time that the layout-based heuristic needs for handling the larger connected components. Hence, TransClust scales polynomial, $O(n^2)$, in practice with the number of

input sequences[2]. To give an impression: Clustering 192,187 prokaryotic protein sequences from the COG[7] database took 17 minutes with a weak BLAST-based similarity threshold of 10^{-10}, three minutes with a reliable threshold of 10^{-30}, and one minute by applying a restrictive similarity threshold of 10^{-50}. Here, we used the *BeH* similarity function and calculated the clusters on a SunFire 880 with 900 MHz UltraSPARC III+ processors and 32 GB of RAM. Note that the effective running time strongly depends on the chosen density parameter, i.e. the similarity threshold, since this influences the size of the connected components. These are processed separately and hence determine the real runtime. Clustering millions of sequences may be possible for restrictive thresholds *t* and by incorporating an upper bound threshold δ. The latter additionally reduces the problem complexity significantly.

The stand-alone TransClust implementation is able to utilize multiple CPUs by using Java threads, i.e. different connected components may be processed in parallel resulting in significantly improved running times (shared memory parallelism).

Cluster comparison

The ability to compare clustering results is crucial for evaluating the quality of a solution to a gold standard. We therefore provide several parameters and measures that facilitate the comparison of two clusterings. Precision and recall give an overview of the quality of a clustering with respect to the number of falsely assigned and correctly not assigned objects, respectively. The so-called F-measure integrates both precision and recall and can thus be used as a measure of correctness when comparing a clustering to a gold standard. The relative edit distance is an additional parameter that shows the number of edges differing between the gold standard and the given clustering. The lower the relative edit distance, the fewer edges differ between the two clusterings and thus the more similar they are. The measures are computed as follows, using the subsequent abbreviations: *TP* = number of true positive, *FP* = number of false positive, *FN* = number of false negative. Note that for a comparison between two clusterings it is necessary to decide which clusters are compared to which. For the F-measure I we compare each cluster from the gold standard to the cluster from the results with the maximal number of common elements (TPs). For the F-measure II we compare each cluster from the gold standard to the cluster from the results where we achieve the highest accuracy (see below). The F-measure II was introduced by Paccanaro *et al.* [8]. Formal descriptions follow:

$$\text{Precision } P = \frac{TP}{TP + FP}$$

$$\text{Recall } R = \frac{TP}{TP + FN}$$

$$\text{F-measure I} = \frac{2 \cdot P \cdot R}{P + R}$$

F-measure II = $F(K,C) = \frac{1}{n}\sum_{j=1}^{l} n^j \cdot \max_{1\le i\le m}\left(\frac{2n_i^j}{n_i + n^j}\right)$

with:

$$TP = \frac{1}{l}\sum_{j=1}^{l} \max_{1\le i\le m}\left(n_i^j\right)$$

$$FP = \frac{1}{l}\sum_{j=1}^{l}\left(n_{\arg\max_{1\le i\le m}\left(n_i^j\right)} - \max_{1\le i\le m}\left(n_i^j\right)\right)$$

$$FN = \frac{1}{l}\sum_{j=1}^{l} n^j - \max_{1\le i\le m}\left(n_i^j\right)$$

$K = (K_1,\ldots,K_m)$ = Clustering result obtained from algorithm
$C = (C_1,\ldots,C_l)$ = Reference (gold standard) clustering
n = number of nodes in graph
n_i, n^j = number of proteins in clusters K_i and C_j
n_i^j = number of proteins contained in $K_i \cap C_j$.

Relative edit distance = number of edges to be modified to transform the clustering result into the gold standard clustering; normalized by the number of all possible edge changes

Note: An F-measure near 1 indicates a "good" match with the gold standard, i.e. a "good" clustering result; values near 0 indicate "bad" results. In the remainder and the article, we refer to the F-measure II when an "F-measure" is mentioned.

Comparison to other clustering strategies

The focus of this publication is the whole clustering framework. With the Transitivity Clustering web site, the TransClust Cytoscape plug-ins, and the stand-alone software, we account for the need of user-friendly interfaces and provide the user with tools that help to address all of the typical problems that arise during data processing; features that are not available with any other clustering framework. Other bioinformatics clustering tools, such as jClust[9] and Power Graph Analysis[10], provide powerful visualization features and, in the case of jClust, implement various popular clustering strategies. However, the user is still left, at least, with identifying and specifying a problem-specific density parameter and with typical follow-up questions (see Supplementary Table 1).

In the last section of this material, we illustrate a typical, comprehensive clustering task that starts with a list of sequences, investigates similarity functions and parameter estimation problems, and finally deals with an integrated result interpretation; all of which can be done easily with TransClust but with no other clustering software. Additional practical advantages of Transitivity Clustering over other approaches are:

Direct leverage: Instead of solving the problem of hidden transitive substructures directly, many clustering approaches aim to find data objects that represent good centers for potential clusters, others perform flow simulations. Our approach directly attacks the actual problem.

Density parameter: For most clustering tools, finding and setting an appropriate density parameter is far from being intuitive for the user. For example Affinity Propagation[11] needs a preference and a damping factor; Markov Clustering[12] needs the user to enter an inflation parameter and a probability threshold. In contrast, Transitivity Clustering only asks for a single value: a similarity threshold, which is equally intuitive to the parameter *k* of k-means since directly based on the used similarity function. In addition, our TransClust implementation allows an easy identification of a "good" threshold for clustering, if a gold standard for a data subset is available.

Implementation and user-interface: TransClust is written in JAVA 6 and runs on any platform; no additional libraries or software are required. In contrast to most other tools TransClust can visualize the clustering results graphically within Cytoscape. Unlike all others it offers a one-click solution for simplified identification of the optimal, application-specific density parameter.

Although we see the main advantage of TransClust over other approaches in its integrative user-friendly software framework that aids the user from the beginning to the end of the data analysis, we also present a quality comparison of TransClust with other popular clustering tools in the following.

Since there is no need to replicate existing results, we evaluate TransClust with the same data sets and quality measures as in previous bioinformatics clustering publications. Paccanaro *et al.* used in ref. [8] a subset of the SCOP[13] database (ASTRAL95) as gold standard to compare their Spectral Clustering implementation against three other approaches. In ref. [2], we also used this data to compare our first, 2-dimensional layout-based clustering approach FORCE[2] with Spectral Clustering and other clustering software. SCOP is an expert, manually curated database that classifies protein sequences into clusters of four hierarchical levels: class, fold, superfamily, and family. Proteins in the same SCOP-superfamily are believed to be evolutionarily related. As in ref. [2,8], we use two subsets of the ASTRAL95 data set of SCOP: ASTRAL1 consists of 507 proteins assigned to six SCOP-superfamilies; ASTRAL2 consists of 511 proteins assigned to seven superfamilies. The aim is to reproduce the superfamily classification in both datasets based on the given amino acid sequences. By using the F-measure II as the quality score, several popular clustering tools have been compared: Centroid-based clustering with Affinity Propagation[11], Spectral Clustering as in ref. [8], connected component analysis with GeneRAGE[14], Markov Clustering with TribeMCL[12], Hierarchical Clustering as in ref. [8], and layout-based clustering with FORCE[2]. For ASTRAL1, the following F-measures have been achieved: 0.85 (FORCE), 0.81 (Spectral Clustering), 0.65 (Affinity Propagation), 0.47 (GeneRAGE), 0.32 (TribeMCL), and 0.26 (Hierarchical Clustering). For ASTRAL2, the following values have been achieved: 0.89

(FORCE), 0.82 (Spectral Clustering), 0.69 (Affinity Propagation), 0.54 (GeneRAGE), 0.52 (TribeMCL), and 0.42 (Hierarchical Clustering).

By using the TransClust user-interface, the application of Transitivity Clustering to the same data sets and its comparison with the gold standard is an easy task: with just a few mouse clicks, we achieve similar F-measures as with the command-line tool FORCE: 0.83 for ASTRAL1, and 0.91 for ASTRAL2 respectively. This demonstrates that the TransClust software is a reasonable alternative to other methods at least for the task of protein sequence clustering. In the following, we evaluate TransClust with another gold standard data set and illuminate how the integrated non-clustering functionalities support efficient data analyses.

A typical clustering workflow by using a gold standard data set of enzyme families

We used a dataset published by Brown *et al.* in ref. [15] consisting of 866 enzymes that were manually assigned to protein families and superfamilies. Since this dataset is hand-curated it is well-suited as a gold standard. To exemplify the power of our TransClust implementation and to evaluate Transitivity Clustering, we separated this gold standard into two sets; one was used to optimize the density threshold given the optimal gold standard outcome, while the second was used for evaluating the clustering method by applying the obtained threshold; see Supplementary Fig. 1 for an overview of the proposed workflow. We first all-vs.-all BLASTed the protein sequences given in the two datasets using an E-value cutoff of 100. The similarity networks were generated by the BLAST2SimGraph plug-in. We used the *SoH* similarity function for both datasets. Note that this visualization may already provide us with valuable information regarding the relationships across diverse protein families and superfamilies[16].

For the small dataset consisting of Amidohydrolases only, we obtained the initial density threshold range as follows: We assigned all proteins contained in the similarity network with the appropriate family affiliation. Next, we employed the ClusterExplorer plug-in for plotting the intra-cluster vs. inter-cluster edge weight distributions according to this gold standard (see Supplementary Table 1 and Fig. 1, step 3 in the article). From the histogram, we could see that the best separation in the gold standard could probably be achieved within a threshold range of 50 to 70. We next ran the gold standard comparison method of TransClust. To play it safe we add a little more tolerance and set a minimal threshold of $\alpha = 45$, a maximal threshold of $\beta = 75$, and a stepsize of 2. TransClust computed the clustering of the similarity network for each threshold between α and β and compared the respective clustering results to the given gold standard. The comparison results included the above mentioned measures (see Cluster Comparison) and showed the highest F-measure II for a density threshold of 67 (refer to step 4 of Fig. 1 in the article for a simplified illustration).

This threshold was then applied to the second dataset (steps 5-7 in Fig. 1 of the article) that contained all but the Amidohydrolase sequences from the dataset by Brown *et al.* We compared the clustering result for this dataset to the hand-curated annotations using ClusterExplorer and found an F-measure II of approx. 0.84. This shows that the chosen

density threshold provides reasonable results for partitioning of unknown sets of protein sequences into the appropriate protein families. Clustering the complete gold standard dataset with a threshold of 67 finally results in a precision of 0.87, recall of 0.85, and F-measure 0.88. The best F-measure (0.91) for the whole dataset can be achieved at a threshold of 56, which proves our workflow to be expedient. Note that this also demonstrates the robustness of Transitivity Clustering for sequence-based identification of functionally related proteins regarding small fluctuations in the similarity threshold: while the threshold varies between 67 and 56, the F-measure remains stable at around 0.9 (between 0.88 and 0.91). Furthermore, we compared TransClust with two standard bioinformatics clustering approaches: Markov Clustering[17] and Affinity Propagation[11]. We used the same dataset, optimized the input parameters, and reconstructed the protein families with best F-measures of 0.89 (MCL) and 0.67 (AP), respectively.

All the above described data analysis steps except for BLAST have been performed with the Transitivity Clustering plug-in framework for Cytoscape. No extra software was necessary. The whole workflow may be performed in less than 30 minutes with just a few mouse clicks.

Future directions

In the future, we will develop, implement, and test a parallelized TransClust based on message passing to allow distributed memory computing.

We would like to emphasize that our software may be a powerful tool for large-scale meta-genomics datasets. Here, one of the key challenges is the discovery of novel protein families[18,19]. Transitivity Clustering may be used to find a reasonable similarity threshold for known protein families. If we can reconstruct this gold standard, we may subsequently apply our approach to the unknown sequences. Resulting large clusters are likely to be novel protein families.

Furthermore, Transitivity Clustering will find application, for instance in Word Sense Disambiguation (Text Mining), in Public Health surveys, in gene expression data processing, in the identification of protein complexes in protein-protein interaction networks, and in spectrometry data analysis in proteomics and metabolomics.

References

1. S. F. Altschul, T. L. Madden, A. A. Schaffer et al., *Nucleic Acids Res* **25** (17), 3389 (1997).
2. T. Wittkop, J. Baumbach, F. P. Lobo et al., *BMC Bioinformatics* **8** (1), 396 (2007).
3. M. N. Price, P. S. Dehal, and A. P. Arkin, *PLoS One* **3** (10), e3589 (2008).
4. A. Ben-Dor, R. Shamir, and Z. Yakhini, *J Comput Biol* **6** (3-4), 281 (1999).
5. S. Rahmann, T. Wittkop, J. Baumbach et al., *Comput Syst Bioinformatics Conf* **6**, 391 (2007).

6. S. Böcker, S. Briesemeister, and G. W. Klau, *Algorithmica* **in press** (2009).
7. R. L. Tatusov, N. D. Fedorova, J. D. Jackson et al., *BMC Bioinformatics* **4**, 41 (2003).
8. A. Paccanaro, J. A. Casbon, and M. A. Saqi, *Nucleic Acids Res* **34** (5), 1571 (2006).
9. Georgios A. Pavlopoulos, Charalampos N. Moschopoulos, Sean D. Hooper et al., *Bioinformatics* **25** (15), 1994 (2009).
10. Loic Royer, Matthias Reimann, Bill Andreopoulos et al., *PLoS computational biology* **4** (7), e1000108 (2008).
11. B. J. Frey and D. Dueck, *Science* **315** (5814), 972 (2007).
12. A. J. Enright, V. Kunin, and C. A. Ouzounis, *Nucleic Acids Res* **31** (15), 4632 (2003).
13. A. Andreeva, D. Howorth, J. M. Chandonia et al., *Nucleic Acids Res* **36** (Database issue), D419 (2008).
14. A. J. Enright and C. A. Ouzounis, *Bioinformatics* **16** (5), 451 (2000).
15. S. D. Brown, J. A. Gerlt, J. L. Seffernick et al., *Genome Biol* **7** (1), R8 (2006).
16. H. J. Atkinson, J. H. Morris, T. E. Ferrin et al., *PLoS One* **4** (2), e4345 (2009).
17. A. J. Enright, S. Van Dongen, and C. A. Ouzounis, *Nucleic Acids Res* **30** (7), 1575 (2002).
18. D. Wu, P. Hugenholtz, K. Mavromatis et al., *Nature* **462** (7276), 1056 (2009).
19. S. Yooseph, G. Sutton, D. B. Rusch et al., *PLoS Biol* **5** (3), e16 (2007).