Online Transitivity Clustering of Biological Data with Missing Values

Richard Röttger*[1,2], Christoph Kreutzer[1], Thuy Duong Vu[4], Tobias Wittkop[5], and Jan Baumbach[1,2,3]

1 **Max Planck Institute for Informatics, Saarbrücken, Germany,** {roettger,ckreutzer,jbaumbac}@mpi-inf.mpg.de
2 **Center for Bioinformatics, Saarbrücken, Germany**
3 **Cluster of Excellence on Multimodal Computing and Interaction, Center for Bioinformatics, Saarland University, Saarbrücken, Germany**
4 **Bioinformatics group, CBS-KNAW Fungal Biodiversity Centre, Utrecht, The Netherlands,** d.vu@cbs.knaw.nl
5 **Buck Institute for Age Research, Navato, NV, USA,** twittkop@buckinstitute.org

Abstract

Motivation: Equipped with sophisticated biochemical measurement techniques we generate a massive amount of biomedical data that needs to be analyzed computationally. One long-standing challenge in automatic knowledge extraction is clustering. We seek to partition a set of objects into groups such that the objects within the clusters share common traits. Usually, we have given a similarity matrix computed from a pairwise similarity function. While many approaches for biomedical data clustering exist, most methods neglect two important problems: (1) Computing the similarity matrix might not be trivial but resource-intense. (2) A clustering algorithm itself is not sufficient for the biologist, who needs an integrated online system capable of performing preparative and follow-up tasks as well.

Results: Here, we present a significantly extended version of Transitivity Clustering. Our first main contribution is its' capability of dealing with missing values in the similarity matrix such that we save time and memory. Hence, we reduce one main bottleneck of computing all pairwise similarity values. We integrated this functionality into the Weighted Graph Cluster Editing model underlying Transitivity Clustering. By means of identifying protein (super)families from incomplete all-vs-all BLAST results we demonstrate the robustness of our approach. While most tools concentrate on the partitioning process itself, we present a new, intuitive web interface that aids with all important steps of a cluster analysis: (1) computing and post-processing of a similarity matrix, (2) estimation of a meaningful density parameter, (3) clustering, (4) comparison with given gold standards, and (5) fine-tuning of the clustering by varying the parameters.

Availability: Transitivity Clustering, the new Cost Matrix Creator, all used data sets as well as an online documentation are online available at http://transclust.mmci.uni-saarland.de/.

Contact: roettger@mpi-inf.mpg.de

1998 ACM Subject Classification I.5.3 Clustering

Keywords and phrases Transitivity Clustering, Large Scale clustering, Missing Values, Web Interface

Digital Object Identifier 10.4230/OASIcs.GCB.2012.57

* to whom correspondence should be addressed

German Conference on Bioinformatics 2012 (GCB'12).
Editors: S. Böcker, F. Hufsky, K. Scheubert, J. Schleicher, S. Schuster; pp. 57–68

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Sophisticated biomedical measurement techniques generate a massive amount of data that needs to be analyzed efficiently. One example is the availability of next-generation sequencing (NGS) technology that provided us with almost two thousand whole-genome sequences [23]. This data is easily accessible for any researcher in the world-wide scientific community. However, the sequences are worthless without annotations of the biological meaningful elements, i.e. genes and non-coding functional elements, such as microRNAs. Analyzing millions of such DNA sequences individually to determine their function is an impossible task without the aid of specific bioinformatics tools [3, 25]. Just to give some numbers, the NCBI database hosts data about approximately 5.2 million bacterial and 3.2 million eukaryotic genes [22].

Although it appears that we have finally arrived in the post-genome era we still lack fundamental knowledge about crucial genetic programs, the interplay of genes and proteins as well as their biochemical regulation networks. We know very little about how cells, organs and tissues regulate survival, reproduction, movement, etc. in response to altering environmental conditions [20]. Therefore, we use so-called OMICS technology to generate even more data in order to unravel this crucial knowledge about interactions of all kinds of biological entities. The database content of the Gene Expression Omnibus (GEO) with data on more than 630,000 samples may exemplify this trend for the case of gene expression data [8].

Analyzing this impressive amount of data manually is infeasible. In order to draw conclusions from such huge data sets successfully, we often start with compressing the raw data such that the contained information becomes visible. One typical computational tool is clustering [2]. Clustering is a computer science method that partitions data objects into groups such that the objects share common traits, i.e. the elements within the groups are more similar to each other than to objects from other groups. In bioinformatics, for many tasks we have given a pair-wise similarity function $s(u, v)$ that assigns each pair of objects (u and v) a similarity value [12]. The corresponding similarity matrix serves as input for many computational biology clustering problems: protein complex detection from protein-protein interaction networks, cancer sub-typing from gene expression data and protein homology detection from all-vs.-all BLAST [1] results. In our paper we exemplarily focus on the later data type: protein sequence similarities utilized for finding clusters of potentially homologous proteins. Various tools have been developed for this purpose: k-means, Affinity Propagation, Markov Clustering, and FORCE as well as Transitivity Clustering, to name just a few [11, 10, 9, 16, 27].

Here, we concentrate on the problem of finding clusters of homologous proteins with Transitivity Clustering. While this task was extensively studied and published before (refer to [26, 28, 29]), we will focus on two sideline challenges arousing in this context: (1) running time aspects for computing the similarity matrix and (2) online access to a web interface that allows for performing all necessary steps of a typical cluster analysis efficiently without the need for downloading, installing and running the software tool on the local desktop PC. While there exist clustering tools, which allow for clustering with incomplete information, they mainly focus on repairing noise and incompleteness due to the underlying assays (e.g. microarray studies or patient data collections) [13, 7, 18, 21]. Here, we present a systematic approach of saving runtime and memory by omitting the calculation of a share of the similarity values for homology detection while still using the well accepted standard NCBI BLAST. This approach allows the user to save time and enables large-scale studies with almost no drawbacks in the quality of the clustering results. Furthermore, the web front-end offers

easy-to-use interfaces to typical data pre-processing, clustering as well as post-processing tasks. In the following, we briefly describe our previous work on Transitivity Clustering followed by a summary of our new contributions. Afterwards, we describe our approach in detail. Finally, we will evaluate our improved method and discuss its' robustness for protein homology detection.

1.1 Our previous work

We recently developed Transitivity Clustering, an integrated software package dedicated to partitioning biomedical data sets. In previous publications, we studied and evaluated its' performance for remote homology detection and protein (super)family reconstruction [27, 28] as well as protein complex identification [29]. Based on gold standard data from Paccanaro *et al.* [16] and Brown *et al.* [5] we demonstrated that Transitivity Clustering performs equally well or outperforms other popular bioinformatics clustering tools for these tasks. It is based on exact and heuristic algorithms for solving the Weighted Transitive Graph Projection (WTGP) problem [19]. Given a similarity matrix and a user-given threshold (density parameter), we compute a cost graph by removing all edges below the threshold. Afterwards, we make the graph transitive by adding/deleting edges with respect to a cost function that we minimize. Finally, we report the emerging cliques as clusters (see *Definitions* in the Methods section). As demonstrated in [29], the average similarity within the clusters is above the threshold while the average similarity between elements from different clusters is below the density cutoff. So far, Transitivity Clustering comes as stand-alone tool, Java library and as Cytoscape [24] plugin.

1.2 Our new contributions

However, Transitivity Clustering as well as most other approaches neglect two important problems: First, computing the similarity matrix might not be trivial but resource-intense (main memory, running time, costs for allocating appropriate data sets, etc.). Second, a clustering algorithm itself is not sufficient for the biomedical researcher, who needs an integrated online system capable of performing preparative and follow-up tasks as well. The goal of Transitivity Clustering is to assist the user through the entire clustering pipeline. We present a significantly extended version of Transitivity Clustering:

1. Missing similarity values: We added the capability of dealing with missing values in the similarity matrix to the Transitivity Clustering framework such that we may save time and memory. Hence, we reduce one main bottleneck of computing all pairwise similarity values. We will describe how we integrated this functionality into the underlying Weighted Graph Cluster Editing model such that we still achieve accurate results.
2. Evaluation of the robustness: We use the gold standard data from Brown *et al.* [5], as well as a larger data set containing the protein sequences of 27 different *corynebacteria*, to study the robustness of our extended approach. We utilize all-vs.-all BLAST results and remove a varying amount of similarity values based on a block-wise scheme. We study how this affects the accuracy of the clustering performance as well as the running time improvement.
3. Web interface: We present a new, intuitive web interface that aids with all important steps of a typical cluster analysis. We give examples and describe a workflow through the interface.

2 Methods

2.1 Extension of the Weighted Transitive Graph Projection model

Given a set of objects V, a threshold $t \in \mathbb{R}$, and a pairwise similarity function $\mathrm{sim}\colon \binom{V}{2} \to \mathbb{R}$, we define a graph G as

$$G = (V, E);\ E = \left\{ uv \in \binom{V}{2}; \mathrm{sim}(uv) > t \right\}.$$

The WTGP problems is now defined as the determination of a transitive graph $G' = (V, E')$, such that there exists no other transitive graph $G'' = (V, E'')$, with $\mathrm{cost}\,(G \to G'') < \mathrm{cost}\,(G \to G')$. The costs for edge additions/deletions are defined as

$$\mathrm{cost}\,(G \to G') := \underbrace{\sum_{uv \in E \setminus E'} |\mathrm{sim}\,(uv) - t|}_{\text{deletion cost}} + \underbrace{\sum_{uv \in E' \setminus E} |\mathrm{sim}\,(uv) - t|}_{\text{addition cost}}.$$

This problem is NP-hard [14] and APX-hard [6] and we tackle it with a combination of exact and heuristic algorithms (refer to [19, 27]). This problem is also known as "Cluster Editing" and several other exact approaches are mentioned in literature. For an overview of exact methods refer to [4].

Now we need to integrate a possibility to deal with missing edges in the graph G. Therefore, we slightly adjust the underlying similarity function.

$$\mathrm{sim}\,(uv) := \begin{cases} \in \mathbb{R} & \text{if similarity available} \\ t & \text{if missing value} \end{cases}$$

The similarity of the missing values is set to the user-given threshold t. As a result, the costs for adding/remove an edge with a missing value is

$$\underbrace{|\mathrm{sim}\,(uv) - t|}_{\text{deletion cost}} = \underbrace{|\mathrm{sim}\,(uv) - t|}_{\text{addition cost}} = |t - t| = 0$$

In consequence, the overall costs for transforming a graph G into the transitive G' is not affected by the missing values:

$$\mathrm{cost}\,(\text{missing values}) = \sum_{uv \in E \setminus E'} |\mathrm{sim}\,(uv) - t| = \sum_{uv \in E \setminus E'} |t - t| = 0$$

In summary, since the missing values have no information content we adapted our approach such that they do not impact the clustering process. The remaining edges with existing similarity values have to account for the clustering result. We are confident that this does not affect the clustering quality much, as for most transitive most of the similarity information is redundant anyways. Thus, whenever we find a cluster with high intra-cluster similarity it is likely that missing similarities within this cluster are comparably high as well. Our transitivity model is perfectly suited for this key feature, as the missing values do not influence the clustering process itself. Thus, the remaining high similarities are capable of forming the final cluster.

2.2 Costmatrix creation with missing values

With the presented method, the user is able to include missing values in the similarity input files for Transitivity Clustering. In comparison to available methods [13, 7, 18, 21], we not

only allow for missing values, which are repaired either *a priori* or during the clustering process (in our case by exploiting the transitivity properties of the clusters), but we also provide the user with the possibility to systematically benefit from missing values in order to save calculation time of the similarity file and resulting cost matrices.

In this paper we concentrate on protein homology detection based on a BLAST all-vs-all run. In our approach, we do not use missing values on randomly chosen positions but omit the calculation of entire blocks of the similarity file. Therefore, we have developed the new CostMatrixCreator (CMC) assisting the user in creating a similarity file with missing values. First, a BLAST database for all protein sequences is created. Afterwards, only a certain percentage of the proteins are BLASTed against the database. Figure 1 depicts this process. CMC can either choose these proteins randomly (as in our evaluation) or the user can provide a manually created list if the user wants to incorporate prior knowledge. Consequently, we compute similarity values for only those pairs of sequences where at least one of them is in the list of proteins compared against the database. The similarities for the other sequence pairs are missing. That keeps the similarity file small, as the missing values are not required to be explicitly marked as missing. The only additional information needed are the IDs of the proteins left out. Note that this procedure distinguishes between missing similarity values due the BLAST cut-off and missing values due to an omitted comparison. The missing values resulting from the BLAST cut-off indeed carry information: They are very dissimilar and are set to a minimum value. The missing values resulting from omitted comparisons do not carry any information and are set to the density threshold.

This approach has four major advantages: (1) The user is still able to utilize the well accepted standard BLAST. (2) The user saves the calculation time for the missing similarities. (3) As the missing values are not stored explicitly, which reduces the memory consumption of the result file drastically. (4) The created cost matrices can afterwards be analyzed with the standard Transitivity Clustering workflow as before. Also note that our approach would also work with any other similarity function than BLAST as well.

2.3 Data sets

We utilize the gold standard data set from Brown *et al.* for studying the effect of missing values to the clustering performance, i.e. accuracy and running time. The data set comprises of five enzyme superfamilies (amidohydrolase, crotonase, enolase, haloacid dehalogenase, and vicinal oxygen chelate) with different levels of sequence diversity. On the one hand, the enolase and crotonase superfamilies contain a very discrete set of sequences, i.e. high sequence similarities. The other extreme are the haloacid dehalogenase and parts of the amidohydrolase, which include a very divers set of sequences with a comparably high number of outliers. Therefore, in congruence with Brown *et al.*, this set of protein sequences serves as an evaluation data set for clustering tools.

The five superfamilies consist of 4,887 proteins that are further divided into 91 families. Each of the amino acid sequences is either annotated to a gold or a silver standard family. Gold standard families only contain sequences with experimentally determined functions, while silver standard families are less restrictive. As done in previous studies, when we compared Transitivity Clustering to other approaches [27], we only used the 866 sequences that are assigned to a gold standard family.

As the data set from Brown *et al.* resembles a rather small problem instance, we can not expect huge improvements in terms of reduced computational time. Thus, we also apply our methods to a larger remote homology detection data set consisting of 66,000 proteins of 27 different *corynebacteria*. In the remainder of this manuscript, we refer to this data set as the "Coryne-Data".

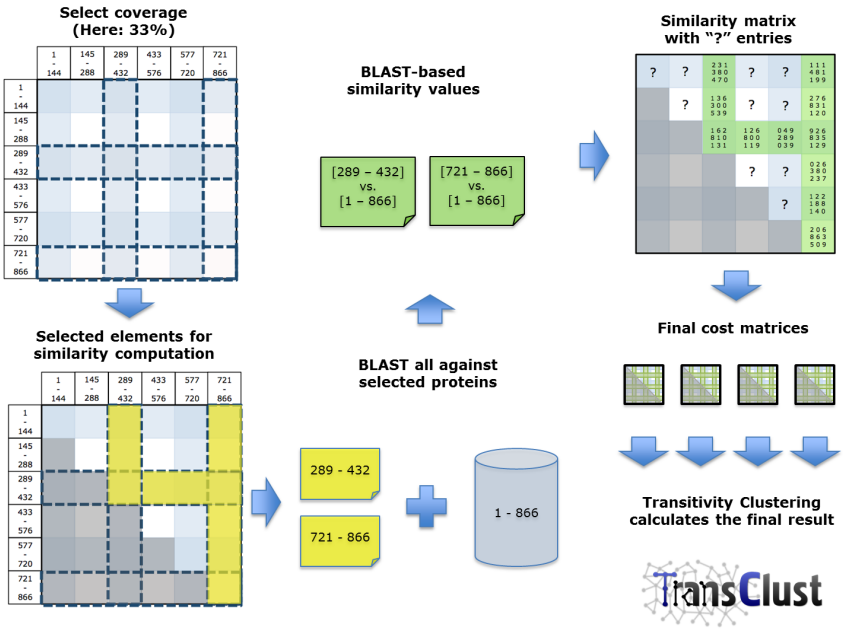


Figure 1 Illustration of the block-based scheme for saving similarity value computation time. For illustration reasons, we arbitrarily ordered the data to form six blocks. Now, only the data from block three and six are selected to be BLASTed against the database. The remaining values in the similarity matrix are displayed as "?". These are the steps used by the CostMatrixCreator to create cost matrices for Transitivity Clustering.

2.4 Evaluation

We use the F-measure for comparing the results of Transitivity Clustering to the above described gold standard. Essentially, the F-measure is an equal combination of precision and recall. It gives a value between 0 and 1, where values near 0 mean "bad" results and a value of 1 means a perfect overlap between clustering result and gold standard, i.e. protein families in our case.

Let $C = \{C_1, \ldots, C_n\}$ be the clusters obtained by Transitivity Clustering and $K = \{K_1, \ldots, K_m\}$ be the gold standard. Furthermore let $T = (t_{i,j}) \in \mathbb{N}^{m \times n}$ denote the matrix where each entry is the number of common objects between K_i and C_j,

$$t_{i,j} := |\{K_i \cap C_j\}|,\ 1 \leq i \leq m,\ 1 \leq j \leq n.$$

We follow the scheme of Paccanaro *et al.* [16] for computing the F-measure for a clustering C against a reference clustering K in the following way:

$$\text{F-measure}(K_i) = \max_{1 \leq j \leq n} \frac{2 \cdot t_{i,j}}{|C_j| + |K_i|}.$$

The overall F-measure is then defined as

$$\begin{aligned} \text{F-measure}(C, K) &= \frac{1}{\sum_{i=1}^{m} |K_i|} \sum_{i=1}^{m} (|K_i| \cdot \text{F-measure}(K_i)) \\ &= \frac{1}{\sum_{i=1}^{m} |K_i|} \sum_{i=1}^{m} \left(|K_i| \cdot \max_{1 \leq j \leq n} \frac{2 \cdot t_{i,j}}{|C_j| + |K_i|} \right). \end{aligned}$$

Note that we refer to the F-measure when we use the term “accuray”.

In order to use the above described Brown *et al.* gold standard data set we first downloaded all amino acid sequences for all proteins and BLASTed them all-vs.-all (E-value threshold 0.01). We furthermore downloaded the corresponding protein family annotations. For the Coryne-Data, we obtained all protein sequences of all 27 fully-sequenced *corynebacteria* from the NCBI website. As there is no gold standard for this data set, we performed an all-vs.-all BLAST and performed a TC clustering with a threshold of 20 in order to produce our own "gold standard". Anyway, we may use this data set for studying the effect of missing values compared a full-coverage similarity matrix and for evaluating run time savings.

For creating a similarity matrix with missing values we utilize the new CostMatrixCreator to construct a BLAST database containing all proteins of a dataset. Here, only a certain percentage of the available proteins are compared to the database (again with E-value threshold 0.01). Figure 1 depicts this process. For systematic evaluation, we vary the coverage from 1% to 90% and create the cost matrices accordingly. We call this method the “block-based scheme”, as we exclude no single entries from the similarity matrix calculation but entire blocks or rather stripes (Figure 1 illustrates that as well).

The emerging cost matrices are then used to perform the clustering with TC. For each clustering we compute the accuracy (F-Measure) and measure the runtime. The running time for the entire clustering is the time for CMC plus the successive clustering process. In order to assess the variability (robustness) of the best threshold, we clustered both data sets with different thresholds and always picked (and reported) the threshold leading to the best F-measure.

2.5 Web interface

TransClust, our Transitivity Clustering implementation, and the new CostMatrixCreator can be downloaded as stand-alone Java programs. Nevertheless, in order to increase the accessibility of the presented tools, a web interface helps in attracting more users. Especially non computer experts benefit from the now obsolete step of an installation process. Furthermore, the results can be accessed from any computer with Internet connection, which allows for an optimized workflow and eases collaborations. For the web interface, we mainly used the following libraries and programming languages:

LAP: We use Linux, Apache, and PHP to generate the HTML code, for data handling and for executing the Transitivity Clustering software.

JavaScript and jQuery: jQuery is a JavaScript library under GPL or MIT license that can be used to automate common tasks in web design. We make use of this for the toggle switches that show/hide information such as advanced options for different steps in the clustering process. We also used JavaScript to check input forms for correctness and for changing contents of input forms, when using sliders, for instance.

jQueryUI: jQueryUI is an extension of jQuery specifically designed for providing extended functionality such as widgets and animations. We use jQueryUI for the tabs that can be used to toggle the different outputs and for the sliders that can be used to change different input parameters.

Highcharts: Highcharts is a charting library from Highsoft Solutions under the Creative Commons Attribution-NonCommercial 3.0 License (free for non-academics). We use to create the graphical representations of the results. We use jQuery/JavaScript to process the output files and feed them to Highcharts.

The new web interface is publicly available online at http://transclust.mmci.uni-saarland.de/.

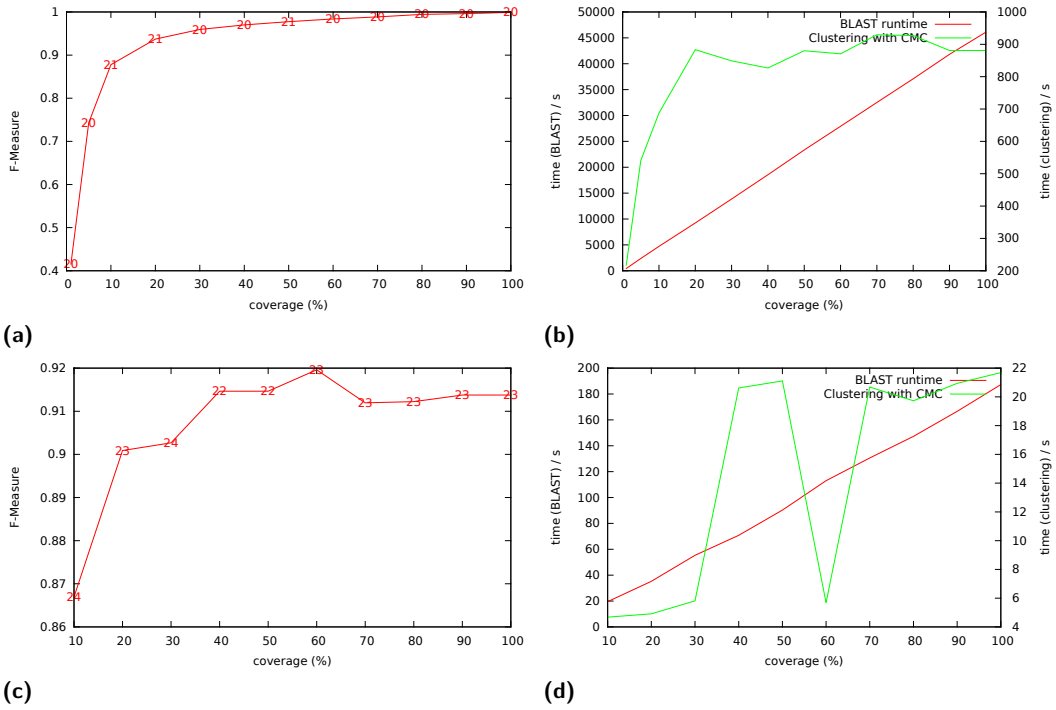


Figure 2 The plots depict the runtime and accuracy of both datasets, Coryne-Data is on the top ((**a**) and (**b**)), the data set of Brown *et al.* on the bottom ((**c**) and (**d**)). Figures (**a**) and (**c**) display the F-measure as a function of the coverage. The numbers on the line give the threshold, which yielded to the corresponding F-meassure. Figures (**b**) and (**d**) give two runtime measures, again as a function of the coverage: **red** displays the time consumed for calculating the BLAST results, **green** depicts the runtime for the clustering plus CMC. Note that both runtime plots have different timescales displayed on the left for BLAST and on the right for clustering respectively.

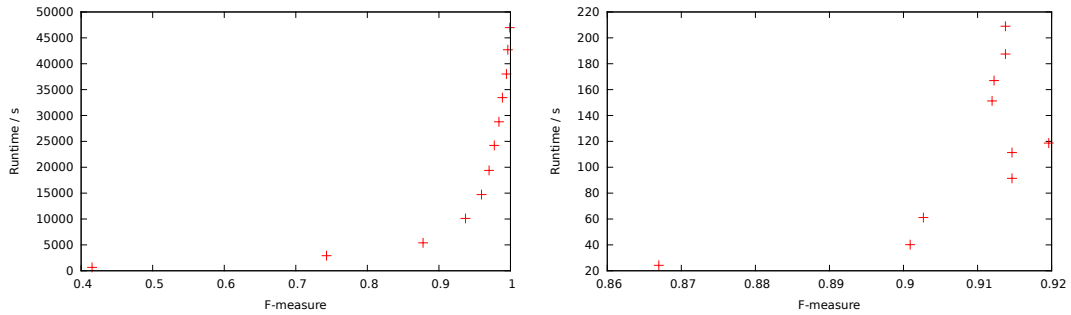
3 Results and discussion

3.1 Clustering with missing values

We first investigate the robustness of our approach by comparing the F-Measure for different percentages of missing values (inverse similarity coverage). The results are depicted in Figure 2. We can see that even for a low coverage (high amount of missing values) the F-Measure only drops by a few percent when comparing against the protein families from Brown *et al.*, as well as for the Coryne-Data. Furthermore, it is important to notice, that the threshold delivering the best F-measure is very stable for all coverages. That means in conclusion, that all methods for finding a good threshold (e.g. using a smaller gold standard dataset for parameter training or utilizing the same threshold from a comparable study) can be applied for clustering with missing values as well.

Running time scales as expected: BLAST computing times grow linearly with the number of sequence comparisons while the runtime for the clustering process is essentially constant with little variation.

Figure 3 plots the average runtime of the cost matrix creation CMC with missing values against the F-Measures. It shows that with our approach the runtime can be drastically reduced, while the drop of the F-Measure is comparably moderate, i.e. less than 10% F-Measure drop for about 70% runtime saving. All runtimes are based on a single thread



(a) **(b)**

Figure 3 Runtime of CMC as a function of the F-Measure for **(a)** *Coryne-Data* and **(b)** Brown *et al.* dataset. Due to the small size of the Brown *et al.* dataset and the resulting short BLAST runs, we observe a certain variation of the clustering runtime. For the larger dataset, this is not observed anymore.

execution of BLAST and CMC. Note, that CMC also supports parallel execution, which may further reduce the runtime.

3.2 Web interface

The clustering process with the web interface is divided into three steps. In the first step, the user provides the input (cost-matrix file, similarity file, or BLAST and FASTA files). In the second step, a review of the given similarities is presented as a similarity distribution plot. The user may then specify further clustering options, such as the clustering threshold(s) or a gold standard file to compare against, if available/desired. In the third step the results are presented as intra/inter-cluster similarity distribution graphs. The best way, however, to evaluate our new web interface is navigating with your browser to the new Transitivity Clustering web site. It is easy-to-use and provides start-to-end solutions for each important step of a typical cluster analysis. Briefly, the new interface now assists with the following typical data processing tasks: (1) computing and post-processing of a similarity matrix, (2) estimation of a meaningful density parameter, (3) running the clustering process itself, (4) automatic comparison with given gold standards, and (5) fine-tuning of the clustering by varying the threshold and by visualizing inter-cluster vs. intra-cluster similarity distributions.

4 Conclusion and outlook

To sum up, we directly integrated the concept of missing similarity values with the weighted transitive graph projection model of Transitivity Clustering in a straight forward fashion. We demonstrated the power of the approach for protein homology detection based on all-vs.-all BLAST results. The accuracy only drops slightly while run time for computing the similarity matrix can be reduced linearly, with the presented tool. The new web interface saves time and effort with download, configuration and installation.

The new CostMatrixCreator is a JAVA implementation and supports the user with the creation of cost matrices with missing values step by step. Note that we do not use an own BLAST implementation but we execute the standard NCBI Blast binaries from within CMC. Thus our CMC implementation could easily be adapted for performing the necessary steps with any kind of similarity function. The resulting cost matrices can finally be included

easily into the normal clustering workflow by using our new Transitivity Clustering web interface, for instance.

In the future, we aim to apply our method to more run time intense similarity computation problems (3D structures of proteins, for instance). We will also integrate the new version of Transitivity Clustering into clusterMaker [15]. The most important future work, however, is to evaluate our method on more data sets. We are working closely together with the SFLD (Structure-Function Linkage Database) team [17] to make this possible in the near future. Another example for future applications is huge gene expression data sets.

Acknowledgement

JB and RR are grateful for financial support from the Cluster of Excellence on Multimodal Computing and Interaction of the German Research Foundation (DFG). RR's work was also supported by the International Max Planck Research School in Computer Science. JB, RR and CK wish to thank the Center for Bioinformatics Saar (ZBI).

References

1 S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

2 Bill Andreopoulos, Aijun An, Xiaogang Wang, and Michael Schroeder. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefing in Bioinformatics*, Feb 2009.

3 Enrique Blanco and Josep F Abril. Computational gene annotation in new genome assemblies using geneid. *Methods Mol Biol*, 537:243–61, 2009.

4 S. Böcker, S. Briesemeister, and G.W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2011.

5 Shoshana D Brown, John A Gerlt, Jennifer L Seffernick, and Patricia C Babbitt. A gold standard set of mechanistically diverse enzyme superfamilies. *Genome Biology*, 7(1):R8, 2006.

6 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71:360–383, 2003.

7 S. Dubnov, R. El-Yaniv, Y. Gdalyahu, E. Schneidman, N. Tishby, and G. Yona. A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles. *Machine Learning*, 47(1):35–61, 2002.

8 Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res*, 30(1):207–10, Jan 2002.

9 A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, Apr 2002.

10 A. J. Enright and C. A. Ouzounis. GeneRAGE: a robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16(5):451–457, May 2000.

11 B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

12 John A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

13 Dae-Won Kim, Ki-Young Lee, Kwang H Lee, and Doheon Lee. Towards clustering of incomplete microarray data without the use of imputation. *Bioinformatics*, 23(1):107–113, Jan 2007.

14 Mirko Křivánek and Jaroslav Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986.

15 John H Morris, Leonard Apeltsin, Aaron M Newman, Jan Baumbach, Tobias Wittkop, Gang Su, Gary D Bader, and Thomas E Ferrin. clustermaker: a multi-algorithm clustering plugin for cytoscape. *BMC Bioinformatics*, 12(1):436, Nov 2011.

16 A. Paccanaro, J. A. Casbon, and M. A. Saqi. Spectral clustering of protein sequences. *Nucleic Acids Research*, 34(5):1571–1580, 2006.

17 Scott C-H Pegg, Shoshana D Brown, Sunil Ojha, Jennifer Seffernick, Elaine C Meng, John H Morris, Patricia J Chang, Conrad C Huang, Thomas E Ferrin, and Patricia C Babbitt. Leveraging enzyme structure-function relationships for functional inference and experimental design: the structure-function linkage database. *Biochemistry*, 45(8):2545–55, Feb 2006.

18 J. Poland and T. Zeugmann. Clustering pairwise distances with missing data: Maximum cuts versus normalized cuts. pages 197–208, 2006.

19 Sven Rahmann, Tobias Wittkop, Jan Baumbach, Marcel Martin, Anke Truss, and Sebastian Böcker. Exact and heuristic algorithms for weighted cluster editing. *Comput Syst Bioinformatics Conf*, 6:391–401, 2007.

20 Richard Röttger, Ulrich Rückert, Jan Taubert, and Jan Baumbach. Towards the size of gene regulatory networks – how little do we actually know? *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, in press, 2012.

21 M. Sarkar and T. Y. Leong. Fuzzy k-means clustering with missing values. *Proc AMIA Symp*, pages 588–592, 2001.

22 Eric W Sayers, Tanya Barrett, Dennis A Benson, Evan Bolton, Stephen H Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M Church, Michael DiCuccio, Scott Federhen, Michael Feolo, Ian M Fingerman, Lewis Y Geer, Wolfgang Helmberg, Yuri Kapustin, David Landsman, David J Lipman, Zhiyong Lu, Thomas L Madden, Tom Madej, Donna R Maglott, Aron Marchler-Bauer, Vadim Miller, Ilene Mizrachi, James Ostell, Anna Panchenko, Lon Phan, Kim D Pruitt, Gregory D Schuler, Edwin Sequeira, Stephen T Sherry, Martin Shumway, Karl Sirotkin, Douglas Slotta, Alexandre Souvorov, Grigory Starchenko, Tatiana A Tatusova, Lukas Wagner, Yanli Wang, W John Wilbur, Eugene Yaschenko, and Jian Ye. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39(Database issue):D38–51, Jan 2011.

23 Eric W Sayers, Tanya Barrett, Dennis A Benson, Evan Bolton, Stephen H Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M Church, Michael Dicuccio, Scott Federhen, Michael Feolo, Lewis Y Geer, Wolfgang Helmberg, Yuri Kapustin, David Landsman, David J Lipman, Zhiyong Lu, Thomas L Madden, Tom Madej, Donna R Maglott, Aron Marchler-Bauer, Vadim Miller, Ilene Mizrachi, James Ostell, Anna Panchenko, Kim D Pruitt, Gregory D Schuler, Edwin Sequeira, Stephen T Sherry, Martin Shumway, Karl Sirotkin, Douglas Slotta, Alexandre Souvorov, Grigory Starchenko, Tatiana A Tatusova, Lukas Wagner, Yanli Wang, W. John Wilbur, Eugene Yaschenko, and Jian Ye. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, Nov 2009.

24 Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, Nov 2003.

25 Vasily Tcherepanov, Angelika Ehlers, and Chris Upton. Genome annotation transfer utility (gatu): rapid annotation of viral genomes using a closely related reference genome. *BMC Genomics*, 7:150, 2006.

26 Tobias Wittkop, Jan Baumbach, Francisco P Lobo, and Sven Rahmann. Large scale clustering of protein sequences with force -a layout based heuristic for weighted cluster editing. *BMC Bioinformatics*, 8:396, 2007.

27 Tobias Wittkop, Dorothea Emig, Sita Lange, Sven Rahmann, Mario Albrecht, John H Morris, Sebastian Böcker, Jens Stoye, and Jan Baumbach. Partitioning biological data with transitivity clustering. *Nat Methods*, 7(6):419–20, Jun 2010.

28 Tobias Wittkop, Dorothea Emig, Anke Truss, Mario Albrecht, Sebastian Böcker, and Jan Baumbach. Comprehensive cluster analysis with transitivity clustering. *Nat Protoc*, 6(3):285–95, Mar 2011.

29 Tobias Wittkop, Sven Rahmann, Richard Röttger, Sebastian Böcker, and Jan Baumbach. Extension and robustness of transitivity clustering for protein-protein interaction network analysis. *Internet Mathematics*, 7(4):255–273, 2011.