Large scale clustering of protein sequences with FORCE - A layout based heuristic for weighted cluster editing

Tobias Wittkop¹,²,³, Jan Baumbach*¹,²,⁴, Francisco P Lobo¹,⁵ and Sven Rahmann⁶

Address: ¹Computational Methods for Emerging Technologies, Bielefeld University, Bielefeld, Germany, ²Genome informatics, Bielefeld University, Bielefeld, Germany, ³DFG Graduiertenkolleg Bioinformatik, Bielefeld University, Bielefeld, Germany, ⁴International Graduate School in Bioinformatics and Genome Research, Center for Biotechnology, Bielefeld, Germany, ⁵Laboratorio de Genetica Bioquimica, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil and ⁶Bioinformatics for High-Throughput Technologies, Technical University of Dortmund, Germany

Email: Tobias Wittkop - tobias.wittkop@cebitec.uni-bielefeld.de; Jan Baumbach* - jan.baumbach@cebitec.uni-bielefeld.de; Francisco P Lobo - francisco.pereira.logo@genetik.uni-bielefeld.de; Sven Rahmann - Sven.Rahmann@Uni-Dortmund.de

* Corresponding author

Abstract

Background: Detecting groups of functionally related proteins from their amino acid sequence alone has been a long-standing challenge in computational genome research. Several clustering approaches, following different strategies, have been published to attack this problem. Today, new sequencing technologies provide huge amounts of sequence data that has to be efficiently clustered with constant or increased accuracy, at increased speed.

Results: We advocate that the model of weighted cluster editing, also known as transitive graph projection is well-suited to protein clustering. We present the FORCE heuristic that is based on transitive graph projection and clusters arbitrary sets of objects, given pairwise similarity measures. In particular, we apply FORCE to the problem of protein clustering and show that it outperforms the most popular existing clustering tools (Spectral clustering, TribeMCL, GeneRAGE, Hierarchical clustering, and Affinity Propagation). Furthermore, we show that FORCE is able to handle huge datasets by calculating clusters for all 192,187 prokaryotic protein sequences (66 organisms) obtained from the COG database. Finally, FORCE is integrated into the corynebacterial reference database CoryneRegNet.

Conclusion: FORCE is an applicable alternative to existing clustering algorithms. Its theoretical foundation, weighted cluster editing, can outperform other clustering paradigms on protein homology clustering. FORCE is open source and implemented in Java. The software, including the source code, the clustering results for COG and CoryneRegNet, and all evaluation datasets are available at http://gi.cebitec.uni-bielefeld.de/comet/force/.

Background

The problem

High-throughput genome sequencing projects have generated massive amounts of DNA and protein sequence data, and will do so more rapidly in the near future. One major challenge continues to be determining protein
functions based solely on amino acid sequences. Large-scale pairwise sequence comparison directly results in pairwise similarity measures between protein sequences and is an efficient method to transfer biological knowledge from known proteins to newly sequenced ones. The most widely used method to search for sequence similarities is BLAST [1]. Three challenges arise:

1. Deriving a quantitative similarity measure from the sequence comparison that models homology as well as possible; frequently this is based on the negative logarithm of the BLAST E-value.

2. Inventing a clustering strategy that is sufficiently error-tolerant, since experience shows that sequence similarity alone does not lead to perfect clusterings. A common approach is to use a graph-based model, where proteins are represented as nodes and the similarities as weighted edges.

3. Implementing the chosen clustering strategy efficiently.

We note that many approaches do treat the three challenges separately. In this publication,

1. we use a family of different similarity functions, based on negative logarithms of BLAST E-values and sequence coverage.

2. we show that weighted graph cluster editing is an adequate model to identify protein clusters. Weighted graph cluster editing has been recently studied in [2] and is known to be NP-hard.

3. we present a heuristic called FORCE to solve the problem. We show that it provides excellent quality results in practice when compared with an exponential-time exact algorithm, but has a running time that makes it applicable in practice when compared with an exponential-time exact algorithm. Here s: \(\left\lfloor \frac{V}{2} \right\rfloor \rightarrow \mathbb{R}\) denotes a similarity function and \(t\) a user-defined threshold. The resulting cost to add or delete an edge \(uv\) is set to \(\text{cost}(uv) = |s(uv) - t|\). The cost to transform a graph \(G = (V, E)\) into a graph \(G' = (V, E')\) is consequently defined as \(\text{cost}(G \rightarrow G') = \text{cost}(E \setminus E') + \text{cost}(E' \setminus E)\). As in the GCEP, the goal is to find a transitive graph \(G^* = (V, E^*)\), with \(\text{cost}(G \rightarrow G^*) = \min \{\text{cost}(G \rightarrow G'): G' \text{ transitive}\}\).

It can be easily seen that the WGCEP is NP-hard, since it is a straightforward generalization of the GCEP, where \(s: \left\lfloor \frac{V}{2} \right\rfloor \rightarrow \{-1, 1\}\) and \(t = 0\). The GCEP has been proved to be NP-hard several times, e.g., in [3,4].

Previous work and novel contributions

There are several approaches to cluster protein families. One of the earliest approaches that took the transitivity concept formally into account was ProClust [5]; however, the concept of editing the graph was not present in this work. The SYSTERS database [6], now at release 4, is based on a set-theoretic SYSTEmatic ReSearching approach and has existed for some time, but seems to have received little updates since early 2005. One of its main features is that it uses family-specific similarity thresholds to define clusters. It does not, however, employ a transitivity concept. In 2006, Paccanaro et al. [7] presented a comparison of the most popular cluster detection methods, like MCL [8], hierarchical clustering [9], GeneRAGE [10], and their own spectral clustering approach, which performs best when evaluated on a subset of the SCOP database. To evaluate our clustering model, we use the same datasets and performance figure. We furthermore include the recently published Affinity Propagation method in our comparison [11]. Additionally, we evaluate our approach against the COG database [12].
The weighted graph cluster editing problem was first considered for protein clustering in our extended abstract [2], where we also introduced the basic idea of the FORCE heuristic.

Here we present a detailed description of the method and an extended parameter estimation procedure using evolutionary training. Our main point in the paper is that the weighted cluster editing problem adequately models the biological homology detection problem if we use appropriate similarity functions and thresholds. The choice of the threshold and similarity function is, of course, critical, and we report the performance for a wide variety of them in the additional files accompanying this paper.

Methods

Clustering via graph layouting

We present an algorithm called FORCE that heuristically solves the WGCEP for a connected component and thus for a whole graph. FORCE is motivated by a physically inspired force-based graph layout algorithm developed by Fruchterman and Reingold [13]. The main idea of this approach is to find an arrangement of the vertices in a two-dimensional plane that reflects the edge density distribution of the graph, i.e., vertices from subgraphs with high intra-connecting edge weights should be arranged close to each other and far away from other nodes. This layout is then used to define the clusters by Euclidean single-linkage clustering of the vertices’ positions in the plane. To improve the solution, we implemented an additional postprocessing phase. All in all the algorithm proceeds in three main steps: (1) layouting the graph, (2) partitioning, and (3) postprocessing.

Layout phase

The goal in this phase is to arrange the vertices in a two-dimensional plane, such that the similarity values are respected. Subsets of nodes with high edge-density should be arranged next to each other, and far away from other nodes. To find a layout that satisfies this criterion, we use a model inspired by physical forces, i.e., nodes can attract and repulse each other. Starting with an initial layout (a circular layout with user defined radius and repulse each other. Starting with an initial layout (a model inspired by physical forces, i.e., nodes can attract and repulse each other otherwise. One can see that with increasing distance, attraction strength increases while repulsion strength decreases.

To improve convergence to a stable position with minimal interactions, we added a cooling parameter, also inspired by the algorithm of Fruchterman and Reingold. In our implementation, this means that if the displacement distance exceeds a maximal magnitude $M_i$ in iteration $i$, which starts at an initial value $M_0$ and decreases with every iteration $i$, the movement is limited to it.

The output of this phase is a two-dimensional array $pos$ containing the x-y-position of each node. Additional Files 1 and 2 illustrate the layout process and its convergence for two components with 41 and 10 nodes, respectively.

Partitioning phase

Using the positions of the vertices from the layout phase, we define clusters by geometric single-linkage clustering, parameterized by a maximal node distance $\delta$. As described in Algorithm 2 (Appendix), we start with an arbitrary node $v_i \in V$ and define a new cluster $c_{v_i}. A node i belongs to $c_{v_i}$ if there exist nodes $v_i = i_0, \ldots, i_N = i \in V$ with $d(i_{j-1}, i_j) \leq \delta$ for all $j = 0, \ldots, N - 1$. Nodes are assigned to $c_{v_i}$ until no further nodes satisfy the distance cutoff. Then the next, not yet assigned, node $v_j \in V$ is chosen to start a new cluster until every node is assigned to some cluster. We denote with $G_\delta := \bigcup_{j=1}^m c_{v_j}$ the resulting graph obtained by adding all edges between two nodes of the same cluster and deleting all edges between two nodes of different clusters. To find a good clustering we calculate $cost(G \rightarrow G_\delta)$ for different $\delta$. Starting with $\delta \leftarrow \delta_{\text{init}} := 0$ we increase $\delta$ by a step size $\sigma$ up to a limit $\delta_{\text{max}} := 300$.

Experimentation shows that it is beneficial to also increase the step size, i.e. to start with $\sigma \leftarrow \sigma_{\text{init}} := 0.01$ and increase it by multiplying with a user-defined factor $f_{\sigma} := \cdot \cdot \cdot
1.1. The solution with lowest cost is returned as the resulting
clustering. Algorithm 2 returns the clustering in terms of an \( n \times n \) adjacency matrix \( E^* \in \{0,1\}^{n \times n} \), and the transformation cost \( c^* \).

**Postprocessing phase**
Although the best clustering is not guaranteed to be the optimal one, we often obtain a close to optimal solution in practice. To further improve the results we use a two-step postprocessing heuristic. We denote with \( \text{cost}(C) \) the cost to obtain the clustering \( C \).

1. To reduce the number of clusters and especially the number of singletons, the first step is to join two clusters if this reduces the overall cost:

Let \( C : = (c_1, \ldots, c_n) \) be the clustering obtained from the partitioning phase, ordered by size. For all cluster pairs \( 1 \leq i < j \leq n \) we calculate \( \text{cost}(c_i, ..., c_j \cup c_j, ..., c_n) \) until we find a clustering \( C' : = (c_1, \ldots, c_i \cup c_j, ..., c_n) \) with \( \text{cost}(C') < \text{cost}(C) \).

Let \( (c'_1, \ldots, c'_{n-1}) \) be the sorted vector \( C' \). Repeat to attempt joining more clusters until no more join is beneficial.

2. Similar to the Restricted Neighborhood Search Clustering (RNSC [14]), we move a vertex from one cluster to another if this move reduces the overall cost:

As above, let \( C : = (c_1, \ldots, c_n) \) be the clustering obtained from step 1, ordered by size. For \( i, j \in \{1, \ldots, n\} \), \( i \neq j \), and every \( k \in c_j \) we tentatively move \( k \) from \( c_i \) to \( c_j \) and calculate \( \text{cost}(c_1, ..., c_j \setminus \{k\}, ..., c_i \cup \{k\}, ..., c_n) \), until we find the first such modified clustering with lower cost than \( \text{cost}(C) \). We sort the resulting clusters again by size and use them as a new start configuration for the next iteration until no more re-assignments are beneficial.

**Analysis**
The worst-case running time of FORCE is given by the addition of those of the three main phases. Layouting runs in \( \Theta(R \cdot n^2) \), where \( R \) denotes the number of iterations and \( n \) is the number of nodes in the graph. Since \( R \) is determined by evolutionary training (see below), it might grow with \( n \), but we set an upper bound for \( R \) to \( R_{\text{max}} = 500 \) that in practice suffices even for very large datasets.

Partitioning runs in \( O(D \cdot n^2) \), where \( D \) is the number of different \( \delta \)-values used. This is seen as follows: Each \( \delta \)-value requires the construction of an auxiliary graph in \( O(n^2) \) time, the discovery of its connected components in \( O(|V| + |E_\delta|) = O(n^2) \) time, setting \( E^\prime \) to the transitive closure of \( E_\delta \) and computing its cost, which is also possible in \( O(n^2) \) after detecting connected components.

During postprocessing, each iteration takes \( O(n^2) \) time, since the number of clusters is bonded by \( n \). The total time is thus \( O(P \cdot n^2) \), where \( P \) is the number of postprocessing iterations. While theoretically \( P \) can grow with \( n \), in practice we observe only a small number of iterations until no more improvement occurs.

Thus for all practical purposes, the overall runtime of FORCE is quadratic in the number of nodes.

**Evolutionary parameter training**
There are several user-defined parameters to assign, such as the number of iterations \( R \), the attraction and repulsion scaling factors \( f_{\text{att}}, f_{\text{rep}} \), the magnitude \( M_\theta \) and the initial circular layout radius \( \rho \). A practical method to find good values is evolutionary training. FORCE implements such a strategy in two different ways.

First, a good parameter combination is determined that can be applied to most of the graphs. This is done during a pre-computation on a training data set. Since, however, the optimal parameter constellation depends on the specific graph, we additionally apply such a training algorithm to each graph. FORCE allows to specify the number of generations to train, and thus to adjust runtime and the quality of the result.

Training works as follows: First we start with a set of 25 randomly generated parameter sets and the initial parameters mentioned above. The parameter sets are sorted by the cost to solve the WGCEP on the given graph. For each generation, we use the best 10 parameter constellations as parents, to generate 15 new combinations. In order to obtain fast convergence to a good constellation, as well as a wide spectrum of different solutions without running into local minima, FORCE splits these 15 new combinations into 3 groups, with 5 members each. The first group consists of parameters obtained only by random combinations of the 10 best already known parameter constellations. The next group is generated with random parameters, while the third group is obtained by a combination of the previous methods. To reduce the runtime for small or very easy to compute solutions, we added a second terminating condition: If at most two different cost appear while calculating the 25 start parameters, the best one is chosen. No more generations are computed.

**Datasets, similarity functions, and parameters**
Here we describe the datasets used for the subsequent evaluation. First the ASTRAL dataset from SCOP, as used in [7], is introduced. We also describe a considerably larger dataset obtained from the COG database. BLAST is used for all-against-all similarity searches in all datasets. The similarity measure is a function of the BLAST E-values; we describe three reasonable functions to convert E-values
into similarities. The results are used as input for FORCE. All datasets can be downloaded from the FORCE website.

**SCOP and Astral95**

SCOP is an expert, manually curated database that groups proteins based on their 3D structures. It has a hierarchical structure with four main levels (class, fold, superfamily, family). Proteins in the same class have the same type(s) of secondary structures. Proteins share a common fold if they have the same secondary structures in the same arrangement. Proteins in the same superfamily are believed to be evolutionarily related, whereas proteins in the same family exhibit a clear evolutionary relationship [15]. We take the SCOP superfamily classification as ground truth against which we evaluate the quality of a clustering generated by a given algorithm, using reasonable quality measures, such as the F-measure (see below).

Since the complete SCOP dataset contains many redundant domains that share a very high degree of similarity, most researchers choose to work with the ASTRAL compendium for sequence and structure analysis in order to generate non-redundant data [16]. ASTRAL allows to select SCOP entries that share no more sequence similarity than a given cutoff, removing redundant sequences.

We extracted two subsets of the ASTRAL dataset of SCOP v1.61 with a cutoff of 95 percent, which means that no two protein sequences share more than 95% of sequence identity. We consider ASTRAL95 as the best possible available reference for remote homology detection on a structural basis.

The two subsets are exactly those used in Paccanaro et al.’s work [7]. The first comprises 507 proteins from 6 different SCOP superfamilies, namely Globin-like, EF-hand, Cupredoxins, (Trans)glycosidases, Thioredoxin-like, and Membrane all-alpha. We refer to this dataset as ASTRAL95_1_161.

Due to the fact that SCOP is continuously updated, we decided to evaluate both the original data from [7] (SCOP v1.61) and more recent data from the current SCOP version (SCOP v1.71). The novel version is slightly different. For example, the superfamily Membrane all-alpha has been removed in the meantime, and most of its proteins are assigned to different superfamilies. Also, several other proteins have been reassigned to one of the five other superfamilies. This provides another dataset of 589 sequences from the remaining 5 superfamilies, which we refer to as ASTRAL95_1_171.

The second subset consists of 511 sequences from 7 superfamilies, namely Globin-like, Cupredoxins, Viral coat and capsid proteins, Trypsin-like serine proteases, FAD/NAD(P)-binding domain, MHC antigen-recognition domain, and Scorpion toxin-like. We refer to this as ASTRAL95_2_161 and ASTRAL95_2_171 respectively. SCOP can be found at [17], while the protein sequences are available at [18].

**Protein sequences from the COG database**

The Cluster of Orthologous Groups (COG) of proteins database is a repository whose main goal is a phylogenetic classification of proteins encoded by complete genomes. It currently consists of 192,187 prokaryotic protein sequences from 66 complete genomes distributed across the three domains of life [12]. COG contains clusters in which at least three individual proteins (or groups of paralogs), originating from three different species, are each other’s best BLAST hit in both directions. This strategy is believed to generate clusters of groups of orthologous genes.

We consider COG as the best possible representation of orthology detection, based on sequence data alone. We refer to this dataset as the COG dataset. COG can be found [19], while the protein sequences are available at [20].

**Similarity functions and thresholds**

Any attempt to (optimally) solve the WGCEP would be in vain if the target function did not model our goal appropriately. As mentioned earlier, the main challenge is to identify appropriate similarity functions and thresholds. We have used a variety of similarity functions that we describe below.

Assume we are given a set of proteins $V$ and a BLAST output file containing multiple high-scoring pairs (HSPs) in both directions. For two proteins $u$ and $v$ we denote with $(u \leftarrow v)$ and $(u \rightarrow v)$, where $i = 1, \ldots, k$ and $j = 1, \ldots, l$, the corresponding $k$ HSPs in one and $l$ HSPs in the other direction, respectively.

We consider the following three similarity functions.

**Best hit (BeH)**

This widely used method concentrates on the E-value of a single HSP: For both directions, one looks for the best hit, i.e., the HSP with lowest E-value. To obtain a symmetric similarity function $s: \binom{V}{2} \rightarrow \mathbb{R}$, the negative logarithm of the worst (largest) of the two E-values is taken as similarity measure between $u$ and $v$. The resulting symmetric similarity function is then defined as

$$s(uv) := -\log(\min_{i \in \{u \leftarrow v\}} \max_{j \in \{u \rightarrow v\}} \min_{u \leftarrow v} \text{E-value}(u \leftarrow v) \text{E-value}(u \rightarrow v)).$$

**Sum of hits (SoH)**

This approach is similar to BeH, but additionally includes every HSP with an E-value smaller than a threshold $m = \ldots$
We use this threshold as penalty for every additional HSP. This leads to the similarity function

\[ s(uw) = -\log_{10}\left( \max \left\{ m^{-A_{ij}} \prod_{i=1}^{k} E\text{-value} \left( \{ v \leftarrow v \} \right), m^{-A_{ij}} \prod_{j=1}^{l} E\text{-value} \left( \{ u \leftarrow v \} \right) \right\} \right) \]

Coverage (Cov)

The third approach integrates the lengths of a HSP into the similarity function. To determine the coverage, we need the following indicator function:

\[ \Gamma_{uv}(i) = \begin{cases} 1 & \text{if in } u \text{ the position } i \text{ is covered by any HSP } \{ u \leftarrow v \}_{i=1}^{\mu} \text{ or } \{ v \leftarrow v \}_{i=1}^{\nu}, \\ 0 & \text{otherwise.} \end{cases} \]

The coverage can now be defined as

\[ \text{Coverage}(uw) := \min \left\{ \frac{1}{|u|} \sum_{i=1}^{k} \Gamma_{uv}(i), \frac{1}{|v|} \sum_{j=1}^{l} \Gamma_{uv}(i) \right\}. \]

In order to obtain a good similarity function, we control the influence of the coverage on the overall similarity function by a user-defined factor \( f \), and set

\[ s(uw) := s'(uw) + f \cdot \text{Coverage}(uw). \]

Here \( s' : \left( \begin{array}{c} V \\ 2 \end{array} \right) \rightarrow \mathbb{R} \) denotes one of the previously presented similarity functions, BeH or SoH.

Parameter choices

The initial parameters obtained from the pre-processing training are \( R = 186, f_{att} = 1.245, f_{rep} = 1.687, M_0 = 633 \), and \( \rho = 200 \) for the protein clustering problem. Furthermore, we apply evolutionary training to each problem instance, as described in the Algorithms section.

Results

This section contains three different types of results. First we discuss the appropriateness of the WGCEP model for the detection of clusters of homologous proteins using the ASTRAL dataset described earlier. Next we show that the FORCE heuristic is fast in practice, and compares favorably against an exact (exponential-time) fixed-parameter algorithm in terms of solution quality. We show that FORCE is able to handle very large datasets efficiently, in particular the COG dataset described previously. Finally, we have integrated the clustering results of FORCE into the corynebacterial reference database CoryneRegNet [21,22].

Evaluation of the WGCEP model

To show that the WGCEP model is adequate for protein homology clustering, we evaluate our algorithm in the same way as Paccanaro et al. did in their article [7], using the so-called F-measure to quantify the agreement of FORCE’s result with the reference clustering provided by the ASTRAL dataset.

We first explain the F-measure, which equally combines precision and recall. Let \( K = (K_1, ..., K_n) \) be the clustering obtained from the algorithm and \( C = (C_1, ..., C_n) \) the reference clustering. Furthermore, we denote with \( n \) the total number of proteins and with \( n_i \) the number of proteins in the cluster \( K_i \) and \( C_j \) respectively. Following this, \( n_i^j \) is the number of proteins in the intersection \( K_i \cap C_j \). The F-measure is defined as

\[ F(K, C) := \frac{1}{n} \sum_{i=1}^{n} \frac{n_i^j}{n_i + n_j^i} \]

As mentioned earlier, Paccanaro et al. previously compared the most popular protein clustering tools against their own spectral clustering: GeneRAGE, TribeMCL, and Hierarchical clustering. Since there is no need to replicate existing results, we use the same data (ASTRAL95_1_161 and ASTRAL95_2_161). Table 1 summarizes the results: Using FORCE, we obtain slightly better agreements than with spectral clustering. The best similarity function parameters and score threshold for the ASTRAL95_1_161 dataset were Cov-scoring using \( f = 20 \) and BeH as a secondary scoring function, and \( t = -2.2 \). For the

| Dataset       | Method            | F-measure |
|---------------|------------------|-----------|
| ASTRAL95_1_161| FORCE            | 0.85      |
| ASTRAL95_1_161| Spectral clustering | 0.81   |
| ASTRAL95_1_161| Affinity Propagation | 0.65 |
| ASTRAL95_1_161| GeneRAGE         | 0.47      |
| ASTRAL95_1_161| TribeMCL         | 0.32      |
| ASTRAL95_1_161| Hierarchical clustering | 0.26 |

| Dataset       | Method            | F-measure |
|---------------|------------------|-----------|
| ASTRAL95_2_161| FORCE            | 0.89      |
| ASTRAL95_2_161| Spectral clustering | 0.82   |
| ASTRAL95_2_161| Affinity Propagation | 0.69 |
| ASTRAL95_2_161| GeneRAGE         | 0.54      |
| ASTRAL95_2_161| TribeMCL         | 0.52      |
| ASTRAL95_2_161| Hierarchical clustering | 0.42 |
For all test datasets, we provide F-measures of FORCE for a wide range of thresholds and coverage factors, for all used datasets and similarity functions. Good clustering quality is also reached by using other thresholds and similarity measures for all test datasets. In Additional File 5, we give F-measures for a range of thresholds, but with fixed coverage factor \( f = 20 \), for dataset ASTRAL95_1_161, and similarity function BeH. In Additional File 6, we provide F-measures for Affinity Propagation for a wide range of parameters and coverage factors, for all used datasets and similarity functions.

\textbf{Quality and running time of the heuristic} \\
After evaluating the WGCEP as a reasonable clustering paradigm, we address the performance of the FORCE heuristic. We compare the running time and solution quality against a slow exact algorithm on the large COG dataset. A recently developed fixed-parameter (FP) algorithm for the WGCEP was extended ideas of previously developed FP algorithms for the (unweighted) GCEP by Gramm et al. [23,24] and Dehne et al. [25], and has a running time of \( O(3^k |V|^3 \log |V|) \), if there exists a transitive projection of cost at most \( k \). This allows us to find the optimal solution for a WGCEP, given a graph \( G = (V, E) \) up to size \( |V| \approx 50 \) in appropriate time. To our knowledge, the implementation of this algorithm is the first available exact WGCEP solving program.

In order to compare the two approaches we use the COG dataset, split into connected subgraphs using similarity function SoH and a threshold of 10. We extracted 1244 connected components (with \( |V| \leq 3,387 \)). For the evaluation, we restricted the maximal run time to 48 hours. The FP algorithm thus could only be applied to 825 components with \( |V| \leq 56 \). For the remaining components, the FP algorithm was terminated unsuccessfully after 48 hours. Due to the large number of graphs, we abstained from applying FP to graphs with \( |V| \geq 100 \), because it is very likely that runtime would exceed 48 hours. Figure 2 illustrates a running time comparison of the FP (blue) and the heuristic algorithm (red). FORCE has been configured to use one generation of evolutionary parameter training for each graph, as described in the Algorithms section. All time measurements were taken on a SunFire 880 with 900 MHz UltraSPARC III+ processors and 32 GB of RAM.

One can see that for large graphs \( |V| \cdot |E| \geq 100,000 \), FORCE is much faster than the exact FP algorithm. Note that the axes are logarithmically scaled. We evaluate the quality of the FORCE heuristic by comparing the relative cost increase of the reported solution, with respect to the provably optimal solution. For 814 out of the 825 comparable components, the heuristic determines the optimal solution. The optimal cost over all 825 components is 171,986.8, while FORCE finds a solution with a total cost of 172,244.6, which is a difference of 0.15%. Figure 3 illustrates these numbers. Note that most of the data points lie on the x-axis and hence indicate that the optimal solution was found.

In addition to the direct running time and quality comparison, we make all connected components and clustering results of the COG dataset available on the FORCE website, using the following similarity functions and thresholds: BeH/10, BeH/20, SoH/10, SoH/20. These choices do not reproduce the original COG clustering; we obtained the following F-measures: 0.64 (BeH/10), 0.56 (BeH/20), 0.61 (SoH/10), and 0.53 (SoH/20). It should be noted that a) the COG clustering problem has very different properties than the SCOP clustering problem, and
Figure 1
Graphical summary of the obtained clustering results of FORCE for the two similarity functions (A) BeH and (B) SoH, and dataset ASTRAL95_1_161. We used MATLAB scripts provided by Paccanaro to create images similar to those of Figure 3 in [7]. Each row corresponds to a cluster. Green bars represent a protein assignment to a cluster; each protein is present in only one of the clusters. Boundaries between superfamilies are shown by vertical red lines, and boundaries between families within each superfamily are shown by dotted blue lines.
b) here we have not optimized in any way the scoring function and threshold. We discuss this further below.

CoryneRegNet

CoryneRegNet (online available at [26]) allows a pertinent data management of regulatory interactions along with the genome-scale reconstruction of transcriptional regulatory networks of corynebacteria relevant in human medicine and biotechnology, together with *Escherichia coli*. CoryneRegNet is based on a multi-layered, hierarchical and modular concept of transcriptional regulation and was implemented with an ontology-based data structure. It integrates the fast and statistically sound method PoSSuMsearch [27] to predict transcription factor binding sites within and across species. Reconstructed regulatory networks can be visualized on a web interface and as graphs. Special graph layout algorithms have been developed to facilitate the comparison of gene regulatory networks across species and to assist biologists with the evaluation of predicted and graphically visualized networks in the context of experimental results. To extend the comparative features, we need adequate data on gene and protein clusters. The integration of this information would widen the scope of CoryneRegNet and assist the user with the reconstruction of unknown regulatory interactions [21,22].

### Table 2: Evaluation of the WGCEP model. The best F-measures for each dataset and each similarity function. ASTRAL95_1_161 and ASTRAL95_2_161 are as in Table 1. ASTRAL95_1_171 and ASTRAL95_2_171 refer to the updated ASTRAL95 data of SCOP v1.71. BeH or SoH denote the similarity function, while the coverage factor \(f\) represents the influence of the coverage to the similarity.

| Dataset          | Similarity | Factor \(f\) | Threshold | F-measure |
|------------------|------------|--------------|-----------|-----------|
| ASTRAL95_1_171   | SoH        | 18           | -3.0      | 0.91      |
| ASTRAL95_1_171   | BeH        | 15           | -3.4      | 0.90      |
| ASTRAL95_2_161   | SoH        | 19           | -1.6      | 0.89      |
| ASTRAL95_2_171   | SoH        | 15           | -3.2      | 0.88      |
| ASTRAL95_2_161   | BeH        | 14           | -2.4      | 0.87      |
| ASTRAL95_2_171   | BeH        | 13           | -2.6      | 0.85      |
| ASTRAL95_1_161   | BeH        | 20           | -2.2      | 0.85      |
| ASTRAL95_1_161   | SoH        | 20           | -1.8      | 0.83      |

### Figure 2

Comparison of the running times of FORCE against the exact fixed-parameter algorithm described in [2]. Plotted is the running time (y-axis in seconds) for different graph sizes (x-axis). Solely for visualization purposes, we describe the size of a graph on the x-axis as \(|V|\cdot|E|\). All graphs have been constructed from prokaryotic COG protein sequence comparisons using BeH as scoring function. Note that both axes are scaled logarithmically. The red points correspond to FORCE running times, and the blue points to the FP algorithm, respectively.

### Figure 3

Relative cost deviations (y-axis in %) of the FORCE solutions from the optimal solutions found by the exact fixed-parameter algorithm described in [2]. The x-axis is as in Figure 2 (logarithmically scaled).
Using FORCE, we calculated protein clusters for all organisms integrated in CoryneRegNet: *Corynebacterium diphtheriae, Corynebacterium efficiens, Corynebacterium glutamicum, Corynebacterium jeikeium, Escherichia coli, Mycobacterium tuberculosis CDC1551* and *Mycobacterium tuberculosis H37Rv* (altogether 22,797 proteins). Based on cluster size distribution, we empirically determined a comparatively high threshold of 30 (which can be explained by the relatively close evolutionary relationship of most organisms in CoryneRegNet) and similarity function SoH to create the FORCE input files based on the all-vs-all BLAST results that are generated during CoryneRegNet's data warehousing process.

The results computed by FORCE are parsed into the object oriented back-end and further on translated into the ontology based data structure of CoryneRegNet. We added a new concept class FORCECluster and a relation type b_fc (belongs to FORCECluster), which links the proteins to their clusters. Finally, we adapted the CoryneRegNet back-end to import the new data into the database and the web-front-end to present the clusters.

**Discussion and Conclusion**

We have shown that the WGCEP is an adequate model for remote protein homology clustering from sequence-based similarity measures and can outperform existing clustering approaches. Part of this effect is certainly attributable to the class of similarity functions that we consider. Nevertheless, in this particular application, the WGCEP paradigm (or rather, our implementation) even outperforms the Affinity Propagation approach, for which we use the same class of similarity functions and a similar parameter optimization as for our approach.

We described FORCE, a heuristic algorithm for the NP-hard weighted graph cluster editing problem. Compared to the currently most efficient exact (exponential-time) fixed-parameter algorithm for this problem, we have demonstrated empirically that FORCE regularly provides solutions that are optimal, although no guarantee is given by the algorithm. In contrast to the exact algorithm, FORCE can solve the problem for graphs with several thousands of nodes in reasonable time.

One of our motivations to develop a rapid and high-quality clustering algorithm arose from the need to extend the data warehouse CoryneRegNet with protein family information. Consequently, the clustering derived by FORCE has been integrated into the system.

We emphasize that FORCE can cluster any set of objects connected by any kind of similarity function using the concept of editing a graph into a transitive graph with minimum cost changes. The integrated evolutionary parameter training method ensures good performance on any kind of data.

Several issues remain to be resolved with the cluster editing or transitive projection approach. One disadvantage of the method is that it uses the same threshold for all clusters to determine the cost of adding or removing edges. The authors of SYSTERS [6] report an interesting approach to choose thresholds in a dynamical way. Finding a way of incorporating dynamic thresholds into cluster editing would certainly enhance its applicability.

The other issue we need to discuss is more global and applies to any clustering algorithm and concerns the choice of parameters. For evaluating the WGCEP model with the SCOP datasets, we have optimized similarity function and threshold (the "parameters") by using the known truth as a reference and thus determined that there exists a (reasonably simple) similarity function that models the truth rather well. In practice, given an unknown dataset, we do not know which parameters lead to the unknown truth. Therefore we need to find properties of the resulting clustering (beyond the target function) that tell us something about the quality of the clustering. For CoryneRegNet, we were able to use the cluster size distribution, as we had expert biological support. In other cases, it is an open challenge to find properties of the clustering that can be easily verified by knowledgeable experts in the field.

**Availability and Requirements**

Project name: FORCE

Project home page: http://gi.cebitec.uni-bielefeld.de/comet/force/

Operating system(s): Platform independent

Programming language: Java 6

License: Academic Free License (AFL)

Any restrictions to use by non-academics: License needed. User should contact Jan.Baumbach@CeBiTec.Uni-Bielefeld.DE.

Comment: Source code, all used datasets, and the clustering results can be obtained from the FORCE project website.

**Authors' contributions**

TW and JB developed and implemented the heuristic FORCE. Together with FPL, TW and JB evaluated the data with ASTRAL95. JB integrated FORCE into CoryneRegNet.
SR proposed to examine the clustering problem from the transitive graph projection viewpoint, modeled the similarity functions and supervised the whole project. All authors contributed to writing and all authors read and approved the final manuscript.

Appendix

Algorithm 1 – Graph layouting

Input: similarity matrix \((S_{ij})_{1 \leq i, j \leq n}\) with \(S_{ij} = \rho(ij) - t\); circular layout radius \(\rho\), attraction factor \(f_{\text{attr}}\), repulsion factor \(f_{\text{rep}}\), number of iterations \(R\)

Output: node positions \(\text{pos} = (\text{pos}[1], \ldots, \text{pos}[n])\); each \(\text{pos}[i] \in \mathbb{R}^2\).

1: \(\text{pos} = \text{arrangeAllNodesCircular}(\rho)\) ▶ initial layout
2: for \(r = 1\) to \(R\) do
3:  ▶ Compute displacements \(\Delta\) for iteration \(r\)
4:  initialize array \(\Delta = (\Delta[1], \ldots, \Delta[n])\) of displacement vectors to \(\Delta[i] = (0, 0)\) for all \(i\)
5:  for \(i = 1\) to \(n\) do
6:    for \(j = 1\) to \(i - 1\) do
7:      if \(S_{ij} > 0\) then
8:        \(f_{i \rightarrow j} = \log(d(i, j) + 1) \cdot S_{ij} \cdot f_{\text{attr}}\) ▶ attraction strength
9:      else
10:         \(f_{i \rightarrow j} = (1/\log(d(i, j) + 1)) \cdot S_{ij} \cdot f_{\text{rep}}\) ▶ repulsion strength
11:    \(\Delta[i] + = f_{i \rightarrow j} \cdot (\text{pos}[j] - \text{pos}[i])/d(i, j)\)
12:    \(\Delta[j] - = f_{j \rightarrow i} \cdot (\text{pos}[j] - \text{pos}[i])/d(i, j)\)
13:  ▶ Move nodes by capped displacement vectors
14:   for \(i = 1\) to \(n\) do
15:     \(\Delta[i] = (\Delta[i]/||\Delta[i]||) \cdot \min(||\Delta[i]||, M(r))\)
16:   \(\text{pos}[i] + = \Delta[i]\)
17: return \(\text{pos}\)

Algorithm 2 – Partitioning the layouted graph

Input: layout positions \(\text{pos}\), initial and maximal clustering distances \(\delta_{\text{init}}, \delta_{\text{max}}\), initial step size \(\sigma_{\text{init}}\), step size factor \(f_{\sigma}\), similarity matrix \((S_{ij})_{1 \leq i, j \leq n}\) to compute costs

Output: best found \(n \times n\) adjacency matrix \(E^*\) describing a clustering, associated cost \(c^*\)

1: \(\delta = \delta_{\text{init}}, \sigma = \sigma_{\text{init}}, c^* = \infty, E^* = (0)^{n \times n}\)
2: while \(\delta \leq \delta_{\text{max}}\) do
3:   construct auxiliary graph \(G_{\delta} = (V, E_{\delta})\) with \(E_{\delta}: = \{uv : d(u, v) \leq \delta\}\)
4:   detect connected components of \(G_{\delta}\)
5:   compute transitively closed adjacency matrix \(E'\) from \(E_{\delta}\)
6:   if \(\text{cost}(E') < c^*\) then
7:      \(E^* = E'; c^* = \text{cost}(E')\)
8:   \(\sigma = \sigma \cdot f_{\sigma}; \delta = \delta + \sigma\)
9: return \((E^*, c^*)\)

Additional material

Additional file 1

Graph layout I. This file is an image illustrating the layout process of a graph with 41 nodes after (A) 3, (B) 10, and (C) 40 iterations. Click here for file [http://www.biomedcentral.com/content/supplementary/1471-2105-8-396-S1.jpeg]

Additional file 2

Graph layout II. This file is an image illustrating the layout process of a graph with 10 nodes after (A) 3, (B) 10, and (C) 40 iterations. Click here for file [http://www.biomedcentral.com/content/supplementary/1471-2105-8-396-S2.jpeg]

Additional file 3

Graphical clustering summary. This zipped file contains images summarizing the FORCE clustering results for the two similarity functions BeH and SoH, and all four datasets, similar to our Figure 1. We used MATLAB scripts provided by Paccanaro [7] to create these images. Click here for file [http://www.biomedcentral.com/content/supplementary/1471-2105-8-396-S3.zip]
Acknowledgements

The authors wish to thank Andreas Dress for helpful discussions. Anke Truss, Sibylle Briesemeister, and Sebastian Böcker generously made available an implementation of their fixed-parameter algorithm for comparison. We are grateful to Marcel Martin and Ralf Nolte for expert technical support. Further thank to Alberto Paccanaro, for providing the MATLAB scripts used for the painting of the graphical clustering summary. Additionally, we would like to thank Constantin Bannert for helpful advice.

References

1. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997, 25(17):3389-3402.
2. Rahmann S, Wittkop T, Baumbach J, Martin M, Tru ß A, Böcker S: Exact and Heuristic Algorithms for Weighted Cluster Editing. Comput Syst Bioinformatics Conf 2007, 6(1):391-401.
3. Delvaux S, Horsten L: On best transitive approximations of simple graphs. Acta informatica 2004, 40(9):637-655.
4. Shamin R, Sharan R, Tsur D: Cluster graph modification problems. Discrete Applied Mathematics 2004, 144:173-182.
5. Pipervachi P, Schliep A, Schneckenberg S, Schoenhuth A, Schomburg D, Schrader R: ProClust: improved clustering of protein sequences with an extended graph-based approach. Bioinformatics 2002, 18(Suppl 2):S182-S191.
6. Krause A, Stoye J, Vingron M: Large scale hierarchical clustering of protein sequences. BMC Bioinformatics 2005, 6:155.
7. Paccanaro A, Carbon JA: Spectral clustering of protein sequences. Nucleic Acids Research 2006, 34(5):1571-1580.
8. Enright AJ, Dongen SV, Ozouzis CA: An efficient algorithm for large-scale detection of protein families. Nucleic Acids Res 2002, 30(7):1575-1584.
9. Everitt BS: Cluster Analysis 3rd edition. Edward Arnold, London; 1993.
10. Enright AJ, Ouzounis CA: GeneRAGE: a robust algorithm for sequence clustering and domain detection. Bioinformatics 2000, 16(5):451-457.
11. Frey BJ, Dueck D: Clustering by Passing Messages Between Data Points. Science 2007, 315(5814):972-976.
12. Tatsuova RL, Fedorova ND, Jackson JD, Jacobs AR, Kyrutin B, Koonin EV, Krylov DM, Mazumder R, Mekhedov SL, Nikoliskaya AN, Rao BS, Smirnov S, Sverdlow AV, Vasudevan S, Wolf YI, Yin J, Natale D: The COG database: an updated version includes eukaryotes. BMC Bioinformatics 2003, 4:41.
13. Fruchterman TMJ, Reingold EM: Graph drawing by force-directed placement. Software – Practice and Experience 1991, 21(11):1129-1164.
14. King AD, Przulj N, Jurisica I: Protein complex prediction via cost-based clustering. Bioinformatics 2004, 20(17):3013-3020.
15. Andreeva A, Howorth D, Brenner SE, Hubbard TJ, Chothia C, Murzin AG: SCOP database in 2004: refinements integrate structure and sequence family data. Nucleic Acids Research 2004, 32:D226-D229.
16. Chandonia JM, Hon G, Walker NS, Conte LL, Koehl P, Levitt M, Brenner SE: The ASTRAL Compendium in 2004. Nucleic Acids Research 2004, 32:D189-D192.
17. SCOP website [http://scop.mrc-lmb.cam.ac.uk/scop/]
18. ASTRAL website [http://astral.berkeley.edu/]
19. COG website [http://www.ncbi.nlm.nih.gov/COG/]
20. COG sequences (FTP) [ftp://ftp.ncbi.nih.gov/pub/COG/]
21. Baumbach J, Brinkkof K, Czaia L, Rahmann S, Tauch A: CoryneRegNet: An ontology-based data warehouse of corynebacterial transcription factors and regulatory networks. BMC Genomics 2006, 7:24.
22. Baumbach J, Wittkop T, Rademacher K, Rahmann S, Brinkkof K, Tauch A: CoryneRegNet 3.0: An interactive systems biology platform for the analysis of gene regulatory networks in corynebacteria and Escherichia coli. J Biotechnol 2007, 129(3):279-289.
23. Gramm J, Guo J, Hufnuff F, Niedermeier R: Automated generation of search tree algorithms for hard graph modification problems. Algorithmica 2004, 39(4):321-347.
24. Gramm J, Guo J, Hufnuff F, Niedermeier R: Graph-modeled data clustering: Exact algorithm for clique generation. Theor Comput Syst 2005, 38(4):373-392.
25. Dehne F, Langston MA, Luo X, Pitre S, Shaw P, Zhang Y: The Cluster Editing Problem: Implementations and Experiments. In Proc of International Workshop on Parameterized and Exact Computation (IWPEC 2006) Volume 4169 LNCS, Springer; 2006:13-24.
26. CoryneRegNet website [http://www.CoryneRegNet.DE]
27. Beckstette M, Homann R, Gigerich R, Kurtz S: Fast index based algorithms and software for matching position specific scoring matrices. BMC Bioinformatics 2006, 7:389.

Additional file 4

Quality evaluation for different scoring themes and datasets. This file is tab-delimited and stores F-measures for a wide range of thresholds and coverage factors, for all used datasets and similarity functions. Column 1: F-measure, column 2: coverage factor l, column 3: threshold, column 4: dataset/similarity function.

Click here for file [http://www.biomedcentral.com/content/supplementary/1471-2105-8-396-S6.txt]

Additional file 5

Quality evaluation for different thresholds and fixed coverage factor, dataset and similarity function. This file is tab-delimited and stores F-measures for a range of thresholds, and fixed coverage factor l = 20, dataset ASTRAL95:1_161, and similarity function BeH. Column 1: threshold, column 2: coverage factor l, column 3: F-measure.

Click here for file [http://www.biomedcentral.com/content/supplementary/1471-2105-8-396-S5.txt]

Additional file 6

Quality evaluation of Affinity Propagation for different scoring themes and datasets. This file is tab-delimited and stores F-measures for a wide range of parameter constellations and coverage factors, for all used datasets and similarity functions. Column 1: F-measure, column 2: coverage factor l, column 3: preference pre, column 4: damping factor dfl, column 5: dataset/similarity function.

Click here for file [http://www.biomedcentral.com/content/supplementary/1471-2105-8-396-S4.txt]