A New Method for Database Searching and Clustering

Antje Krause **Martin Vingron**
a.krause@dkfz-heidelberg.de m.vingron@dkfz-heidelberg.de
Deutsches Krebsforschungszentrum (DKFZ), Abt. Theoretische Bioinformatik
Im Neuenheimer Feld 280, D-69120 Heidelberg, Germany

Abstract

An iterative database searching method is introduced and applied to the design of a database clustering procedure. The search method virtually never produces false positive hits while determining meaningfully large sets of sequences related to the query. A novel set-theoretic database clustering algorithm exploits this feature and avoids a traditional, distance-based clustering step. This makes it fast and applicable to data-sets of the size of, e.g., the Swiss-Prot database. In practice we achieve unambiguous assignment of 80% of Swiss-Prot sequences to non-overlapping sequence clusters in an entirely automatic fashion.

1 Introduction

Grouping protein sequences into families as done, e.g., by M. Dayhoff [6] has proved useful in studying and understanding these sequences. With increasing numbers of homologous protein sequences known such a clustering can help in compressing the output produced by database search programs, can aid in the automatic derivation of multiple alignments and profiles, and can provide data for evolutionary analysis. Today, the PIR database provides a family classification [7, 5]. However, to our knowledge there is no generally accepted method that would be able to cluster automatically large amounts of data. In this paper we put forward a method addressing this problem that is able to reproduce to a large degree the family classification of the PIR database.

Searching a sequence database with a query sequence looking for homologues has become a routine operation. Programs like BLAST [1] or FASTA [13] output a list of similar sequences ranked by significance of the match. Several researchers have recently iterated such a search by choosing a sequence from the list as a new query [12, 16] or by generating a position specific scoring matrix from the list as input for a second search (PSI-BLAST[2]) to compute another list of hits. This new list may display overlaps with the first one or introduce new sequences, both of which provides valuable clues as to which sequences are really biologically related to the first query.

In this paper we take up the idea of iterating a database search to design an algorithm that identifies clusters of protein sequences related to a query in a conservative, reliable and yet informative way. Our procedure is called SYSTERS for "SYSTEmatic Re-Searching". We use it to circumvent the ranking of hits and instead identify a set of similar sequences without ranking. Based on the notion of "consistency" of a database search, we will demonstrate that SYSTERS virtually never produces false positive hits and also rarely misses a sequence.

In a next step we use the set of SYSTERS clusters generated when searching a database with each of its sequences to derive a clustering of an entire database. Here, by "clustering" we do not mean a hierarchical clustering but rather a biologically meaningful partitioning of the data. Neither do we attempt to delineate domains. The domain structure of proteins is, of course, the main obstacle to the application of traditional clustering procedures like single-linkage clusterings. This is the motivation for algorithms like DOMAINER [18] or the graph theoretical clustering algorithm of [10]. In contrast to these methods ours attempts only to cluster together sequences that share global, or at least very strong, similarity. It does so while at the same time maintaining a clear distinction between different

clusters. It is based on set operations instead of pairwise distances or graphs. The SYSTERS based clustering procedure is fully automatic, does not require pairwise comparisons between sequences, and is extremely fast. A comparatively small part of the database will not be assigned to disjoint clusters but to clusters which overlap each other. Thus the method is self-validating in the sense that errors would manifest themselves in further overlaps between clusters. SYSTERS based clustering distinguishes by itself between cases where it can make a decision and cases where it cannot.

2 Database searching by SYSTEmatic Re-Searching

2.1 Searching algorithm

SYSTEmatic Re-Searching, SYSTERS, is an algorithm that iterates traditional protein sequence database searches in a specific way in order to delineate for a given protein sequence a set of related ones. We call the sequence for which we want to extract its related sequences from a database the *seed* and the set of sequences related to this seed the *cluster*. SYSTERS starts with a database search e.g. BLASTP [1] or FASTA [13] using the seed sequence as a query. All hits in a search that are highly significant – we chose a cutoff of 10^{-30} – are accepted. This set of sequences is called *positive_set* and is included to the cluster. The lowest scoring sequence from the positive_set, which was not yet a member of the cluster, is used as query for a next database search. The procedure is iterated either until no not yet accepted sequence is above the cutoff, or until the positive_set has no overlap with the positive_set of the seed search (called *reference_set*) any more. Note that this procedure does not rank hits. The following sketch describes the algorithm of SYSTERS precisely.

Input: *seed*
Output: set of related sequences (*cluster*)

cluster $\leftarrow \emptyset$
query $\leftarrow$ *seed*
`while` (*query* is defined) `do`
    search database with *query*
    positive_set $\leftarrow$ all hits above cutoff
    query $\leftarrow$ undefined
    `if` (*seed* search) `then`
        reference_set $\leftarrow$ *positive_set*
    `endif`
    `if` ($\exists$ sequence in *positive_set* which is not element of *cluster*) `and`
    (*positive_set* $\bigcap$ *reference_set* $\neq \emptyset$) `then`
    query $\leftarrow$ lowest scoring sequence in *positive_set*
            which is not element of *cluster*
    cluster $\leftarrow$ *cluster* $\bigcup$ *positive_set*
    `endif`
`endwhile`

Figure 1 gives a graphical representation of the two termination criteria of this procedure. Sequences are depicted as points in a space where a family forms a cloud. The first search identifies sequences within a given radius around the seed query. The next search chooses a distant but still related sequence as its query and draws another circle. This procedure continues as long as there are sequences within the circle, which are not yet incorporated to the cluster and as long as the circles still overlap with the one of the seed sequence. The iteration in the left part of Figure 1 stopped, because there is no not yet accepted sequence within the circle of query 3, which could be used for a next iteration. On

the right the circle of query 3 shows no overlap with the circle of the seed, so the hits of this query are "too far away" from the seed and therefore regarded as not suitable for further searches.

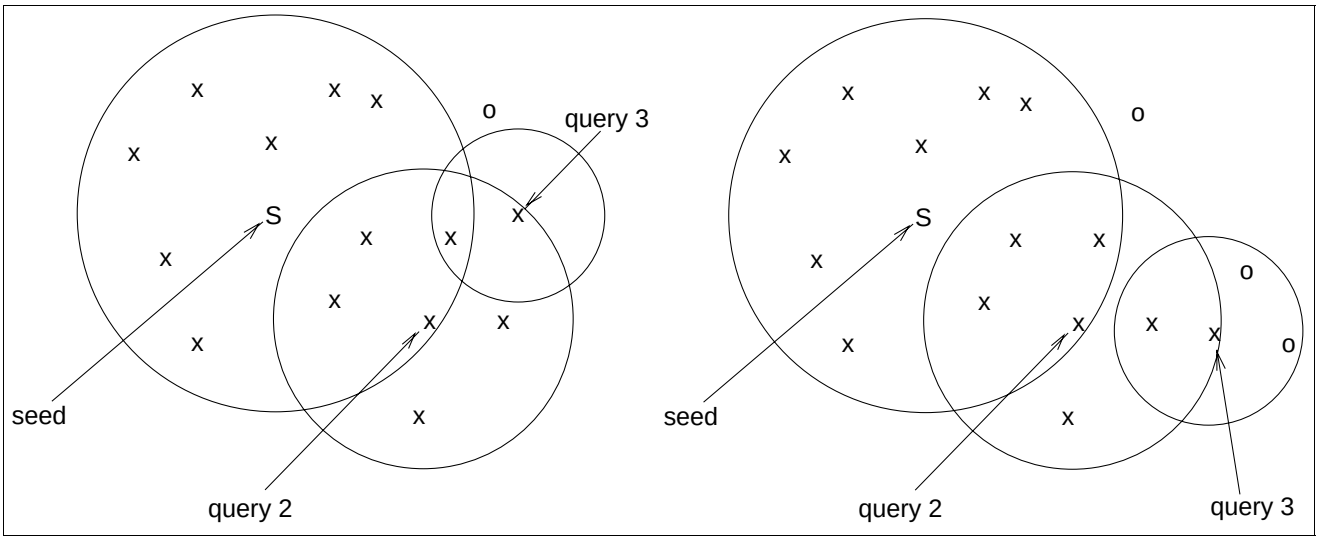

Figure 1: Graphical representation of the two termination criteria of the SYSTERS procedure (x: accepted sequence; o: not accepted sequence; S: seed).

Thus, the main advantage of SYSTERS lies in the fact that homologies are not all scored in relation to one sequence. The set of sequences to be identified supplies other queries that allow clear identification of other parts of a cluster of sequences. Note, however, that the image in the Euclidean plane is highly simplified and does not adequately represent relationships among protein sequences.

In terms of speed, most SYSTERS runs require 2 to 3 calls to a fast searching routine like BLAST or FASTA on the average, depending on the size of the cluster that a query is a member of. SYSTERS is therefore still very fast.

2.2 Description of clusters

In this subsection we briefly summarize some typical results of SYSTERS while in the next subsection we will introduce a means of systematic evaluation. Generally, sequences that do not share domains with other families pose no difficulty to SYSTERS. For example, searching the Swiss-Prot database (Release 34) [3] with the met tRNA synthetase sequence from yeast (SYMC_YEAST) identifies exactly all other met tRNA synthetase sequences from the database. Similar examples where exactly and all the sequences with the same Swiss-Prot description line are found are abundant.

One would expect common domains between proteins to create more of a problem. The homeobox is a domain shared by many different proteins. In one test we used the human engrailed homeobox gene sequence (HME1_HUMAN) as seed for SYSTERS. The resulting cluster identified all homeobox protein sequences from Swiss-Prot that contained the word "engrailed" in their annotation with the exception of two entries. These two were annotated "engrailed-like" and one of them was a fragment of only 60 amino acids length. In Figure 2 we compare the SYSTERS result to the search output generated by a rigorous Smith-Waterman ([17], SSEARCH program [14]) alignment of the seed to the database.

Many of these comparisons were studied in order to determine down to which significance level in the Smith-Waterman search cluster members were identified and also which higher scoring sequences were not included into the SYSTERS clusters. The statistical significances assigned by the SSEARCH program give an impression of how easy or difficult identification of these homologues is. In the particular case the lowest member sequence has a significance of only 0.00014 while several sequences

```
21210388 residues in 59021 sequences
 statistics extrapolated from 20000 to 58445 sequences
 15508 scores better than 63 saved                                          SYSTERS-
 BLOSUM50 matrix, gap penalties: -12,-2                                      Member
  scan time:  0:21:40
The best scores are:                                    s-w Z-score E(59021)
SPR|Q05925|HME1_HUMAN HOMEOBOX PROTEIN ENGRAILED-1 (391) 2681 1586.7       0            X
SPR|P09065|HME1_MOUSE HOMEOBOX PROTEIN ENGRAILED-1 (401) 2409 1427.4       0            X
SPR|Q05916|HME1_CHICK HOMEOBOX PROTEIN ENGRAILED-1 (333) 1461  873.6       0            X
SPR|P19622|HME2_HUMAN HOMEOBOX PROTEIN ENGRAILED-2 (332)  909  550.6 4.3e-24            X
SPR|P09066|HME2_MOUSE HOMEOBOX PROTEIN ENGRAILED-2 (324)  892  540.8 1.5e-23            X
SPR|Q05917|HME2_CHICK HOMEOBOX PROTEIN ENGRAILED-2 (288)  838  509.8   8e-22            X
SPR|P31538|HMEB_XENLA HOMEOBOX PROTEIN ENGRAILED-1 (171)  833  509.7 8.2e-22            X
SPR|P52729|HMEC_XENLA HOMEOBOX PROTEIN ENGRAILED-2 (265)  738  451.7 1.4e-18            X
SPR|P09015|HME2_BRARE HOMEOBOX PROTEIN ENGRAILED-2 (265)  729  446.5 2.7e-18            X
SPR|P52730|HMED_XENLA HOMEOBOX PROTEIN ENGRAILED-2 (265)  727  445.3 3.1e-18            X
SPR|P31533|HME3_BRARE HOMEOBOX PROTEIN ENGRAILED-3 (261)  706  433.1 1.5e-17            X
SPR|Q04896|HME1_BRARE HOMEOBOX PROTEIN ENGRAILED-1 (231)  701  430.8   2e-17            X
SPR|P02836|HMEN_DROME SEGMENTATION POLARITY PROTEI (552)  642  391.6 3.1e-15            X
SPR|P05527|HMIN_DROME INVECTED PROTEIN.            (576)  632  385.6 6.7e-15            X
SPR|P09145|HMEN_DROVI SEGMENTATION POLARITY PROTEI (584)  608  371.4 4.1e-14            X
SPR|P27609|HMEN_BOMMO SEGMENTATION POLARITY PROTEI (372)  586  361.0 1.6e-13            X
SPR|P27610|HMIN_BOMMO INVECTED PROTEIN.            (476)  573  352.1 4.9e-13            X
SPR|Q05640|HMEN_ARTSF HOMEOBOX PROTEIN ENGRAILED.  (349)  558  344.9 1.2e-12            X
SPR|P09532|HMEN_TRIGR HOMEOBOX PROTEIN ENGRAILED ( (154)  464  294.3 8.1e-10            X
SPR|P14150|HMEN_SCHAM HOMEOBOX PROTEIN ENGRAILED ( ( 93)  451  289.4 1.5e-09            X
SPR|P09076|HME3_APIME HOMEOBOX PROTEIN E30 (FRAGME (109)  447  286.2 2.3e-09            X
SPR|P09075|HME6_APIME HOMEOBOX PROTEIN E60 (FRAGME (109)  432  277.4   7e-09            X
SPR|P23397|HMEN_HELTR HOMEOBOX PROTEIN HT-EN (FRAG ( 98)  417  269.2   2e-08            X
SPR|P31537|HMEA_XENLA HOMEOBOX PROTEIN ENGRAILED-1 ( 60)  367  242.6 6.2e-07            X
SPR|P34326|HM16_CAEEL HOMEOBOX PROTEIN ENGRAILED-L (240)  372  238.1 1.1e-06                    engrailed-like
SPR|P50219|HB9_HUMAN HOMEOBOX PROTEIN HB9.         (401)  345  219.6 1.2e-05
SPR|P31535|HMEA_MYXGL HOMEOBOX PROTEIN ENGRAILED-L ( 60)  316  212.7 2.8e-05            X
SPR|P17277|HXA4_CHICK HOMEOBOX PROTEIN HOX-A4 (CHO (309)  328  211.0 3.5e-05
SPR|P06798|HXA4_MOUSE HOMEOBOX PROTEIN HOX-A4 (HOX (326)  327  210.1   4e-05
SPR|P22544|HM1D_DROAN HOMEOBOX PROTEIN OM(1D).     (606)  330  208.6 4.8e-05
SPR|P31536|HMEB_MYXGL HOMEOBOX PROTEIN ENGRAILED-L ( 60)  302  204.6 8.1e-05                    engrailed-like
SPR|P18488|HMES_DROME EMPTY SPIRACLES HOMEOTIC PRO (497)  315  200.9 0.00013
SPR|P31534|HMEN_LAMPL HOMEOBOX PROTEIN ENGRAILED-L ( 60)  295  200.5 0.00014            X
SPR|P31310|HXAA_MOUSE HOMEOBOX PROTEIN HOX-A10 (HO (399)  305  196.2 0.00024
SPR|P09077|SCR_DROME HOMEOTIC SEX COMBS REDUCED PR (415)  299  192.5 0.00038
SPR|P31314|HX11_HUMAN HOMEOBOX PROTEIN HOX-11 (TCL (330)  296  191.9 0.00041
SPR|Q00056|HXA4_HUMAN HOMEOBOX PROTEIN HOX-A4 (HOX (320)  294  190.9 0.00046
SPR|P50223|HMXX_CHICK HOMEOBOX PROTEIN GHOX-7 (CHO (288)  289  188.6 0.00063
SPR|P28357|HXD9_MOUSE HOMEOBOX PROTEIN HOX-D9 (HOX (339)  289  187.7  0.0007
SPR|P28356|HXD9_HUMAN HOMEOBOX PROTEIN HOX-D9 (HOX (342)  285  185.3 0.00095
SPR|P28360|HMX1_HUMAN HOMEOBOX PROTEIN MSX-1 (HOX- (297)  279  182.5  0.0014
                      :
                      :

Library scan:  0:21:40  total CPU time:  0:21:40                    CPU time: 0:01:25
```

Figure 2: Comparison of the SYSTERS result to a Smith-Waterman search output for the human engrailed homeobox gene sequence (HME1_HUMAN) compared against the Swiss-Prot database.

of higher significance have been rejected. In other instances, SYSTERS correctly retrieved sequences only down to a very stringent significance level. In particular, this shows that SYSTERS is capable of circumventing the problems associated to fixing a significance cutoff in traditional database searching.

2.3 Consistency of cluster identification

While the SYSTERS clusters in the above examples all make sense, establishing their biological validity is a process which is difficult to automate. Automatic schemes for checking database search sensitivity are usually based, e.g., on PROSITE [4] assignments of certain motifs. These allow to decide automatically whether a sequence identified in the search contains the same motif as the query or not. As seen from the examples in the prior section, SYSTERS clusters tend to agree with the annotation in Swiss-Prot. This information is not standardized and thus difficult to use for automatic validation. In particular, there may be description lines stating that a sequence is a "hypothetical protein" or that the information was derived by similarity.

Instead of using database annotations for validation of the clustering we introduce a new, formal criterion for the quality of SYSTERS searches. The focus of this criterion is on internal consistency of a search. If one SYSTERS seed identifies a certain cluster, then every other cluster member, when used as a seed should identify the same, or at least a very similar cluster. If a set of sequences indeed forms a cluster, e.g. because they constitute the same protein from different species, this implies that

the same cluster is identified independent of which cluster member is used as seed of the search. To check this criterion we ran SYSTERS searches seeded by all sequences in a database. This identifies a cluster in the data as many times as the number of sequences contained in the cluster. The resulting, very large set of SYSTERS clusters provides the information to check internal consistency.

Denote the set of SYSTERS clusters for all queries from a database as the *complete cluster set* (CCS). For every sequence in the database we computed the following quantities: First the set theoretic union and set theoretic intersection of all clusters in the CCS that contain this sequence are identified. We compare the cardinality of the union of clusters containing a specific sequence and the cardinality of the intersection of clusters containing this sequence. Ideally all members of a family would identify the family in exactly the same way, i.e., produce the same SYSTERS cluster. Then for each of its sequences the union and the intersection of the clusters containing a sequence from this family would coincide and thus their cardinalities would agree. However, if a sequence constitutes a false positive for a specific search, then it is not only contained in the CCS cluster for its own biological family but also in one or more other clusters where it appeared erroneously. Thus, for a false positive the union will be significantly greater than the intersection of clusters that contain this sequence. On the other hand, suppose a sequence is a false negative in some search. Then of the CCS clusters that try to describe the biological family, one or more will lack this one sequence. This in turn makes union and intersection of clusters where other family members are contained differ. For such a sequence the union will exceed the intersection by at least the false negative.

This criterion of internal consistency was systematically tested by performing SYSTERS searches with all sequences of the Swiss-Prot and the PIR1 databases (Release 51). In summary we observed a surprising degree of consistency in SYSTERS searches with 59% of the 59,021 Swiss-Prot sequences and 72% of the 13,489 PIR1 sequences having equally large union and intersection of the clusters containing the sequence.

3 SYSTERS based database clustering

3.1 Clustering Swiss-Prot

The algorithm for database clustering was applied to the Swiss-Prot database (Release 34) containing 59,021 sequences. The above analysis led us to sorting database entries into their respective clusters based on the SYSTERS generated cluster system. Here, we do not aim at a hierarchical clustering. We only wish to compute disjoint clusters, i.e. a biologically meaningful partitioning of the data set.

The first observation is that a CCS cluster all of whose members have identical union and intersection are already perfectly defined. It cannot have any overlap with another cluster and each of its member sequences identifies the cluster in the exact same way. These *perfect clusters* (corresponding to *1-quasi complete graphs* in the nomenclature of [10]) may of course be trivial in the sense that they contain only 1 sequence. Even in this case, though, one knows in particular that the union of clusters containing it is a one-element set which implies that there are no other CCS clusters that contain this one sequence. In the case of the Swiss-Prot database, there are 14,362 such perfect, *single sequence clusters* (24% of all sequences). Further 4,710 perfect clusters contain 19,463 sequences (33%) altogether. Thus, 1,003 sequences with union and intersection of equal cardinality are not elements of perfect clusters.

Since a perfect cluster is disjoint from any other CCS cluster, one may consider this part of the database as perfectly sorted into clusters. Among the remaining clusters, accounting for 25,196 sequences (43%) there exist inclusions and overlaps. One cluster being included by another one typically is the consequence of false negatives in a search. When the same cluster is identified using another seed the (formerly) false negative may be found and the resulting cluster contains the other cluster. Consequently, one wishes to use the larger cluster for the partitioning. However, there may be another cluster containing this one, and so on. Therefore, one needs to check for chains of inclusions among

clusters. Only the final, largest cluster in such a chain is a candidate for our database clustering. However, this set of maximal clusters falls into two groups again. One group, we call them *separate maximal clusters*, are those maximal clusters that are disjoint from any other maximal cluster. The final, residual group of clusters are maximal clusters that overlap with other maximal clusters. These we call the *overlapping maximal clusters*.

Obviously neither the separate maximal clusters overlap each other nor can a separate maximal cluster overlap with a perfect cluster. For the Swiss-Prot database there are 735 separate maximal clusters comprising 13,337 sequences (23%). As an example of the inner structure of a separate maximal cluster, Figure 3 shows a *set-membership matrix* for such a cluster and all the clusters it contains. A column of the matrix corresponds to the cluster found with the seed named on top and a row lists all clusters that the sequence named on the left is a member of. As an example consider the family of engrailed homeobox genes discussed earlier. In the final clustering 27 engrailed or engrailed-like genes form one separate maximal cluster while one fragment (P31536) builds a single sequence cluster. 7 sequences, when used as seed, identify the cluster that is also the separate maximal cluster and the remaining 20 sequences in the SYSTERS search identify a smaller subset.

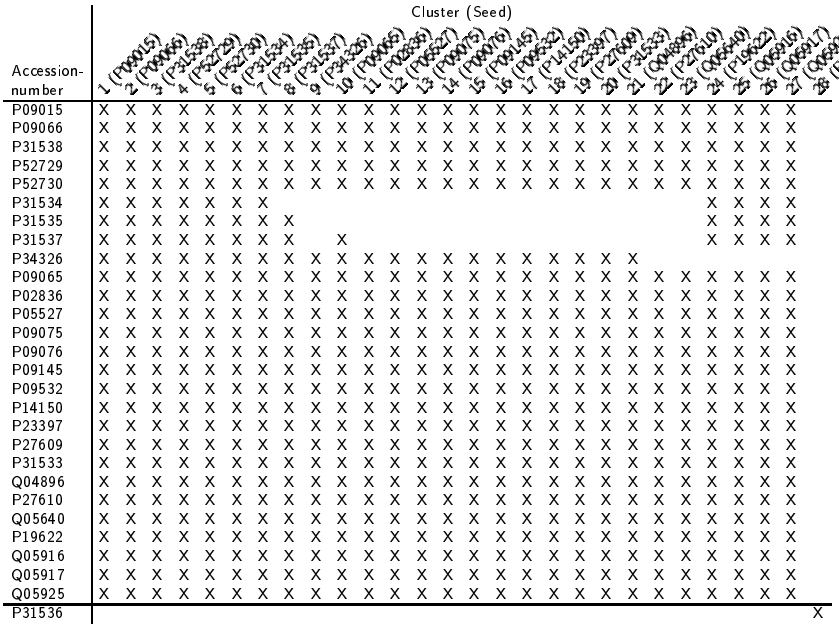

Figure 3: Set-membership matrix for all engrailed homeobox gene sequences contained in the Swiss-Prot database and clustered together by SYSTERS. Columns represent the clusters found with the seed mentioned on top. Rows show all clusters where the sequence on the left is a member of.

Perfect and separate maximal clusters together comprise 80% of the database sequences. There remain 1,383 overlapping maximal clusters accounting for the missing 11,859 sequences. Since overlapping maximal clusters do not constitute a partitioning of the data we sort them into *connected components* where by a connected component we mean a set of clusters which are all linked by overlaps. Thus, a first cluster in this connected component might overlap another one which in turn overlaps a third one and so on. The overlapping maximal clusters for the Swiss-Prot database fall into 147 connected components. The connected components are precisely those cases where SYSTERS cannot delineate separate clusters. Typical members of this group were kinases and proteins that contain a kinase domain or coiled-coil containing proteins like myosin. The largest of these connected components comprises 4,000 sequences contained in 683 overlapping maximal clusters. Several other connected components were trivial in the sense that the overlap between the clusters contained most of the sequences in the connected component. Figure 4 shows the overlapping maximal clusters for

all proteins containing the word "chaperone" in their description line as an example. The last two sequences are not members of the connected component; each of them builds a single sequence cluster. For such trivial cases of connected components one may safely choose the union of the clusters in the connected component as a new cluster to use instead. Such a decision, however, should be taken by an expert and is not part of our algorithm.

Accession-number	Cluster 1	2	3	4	5	6	7	Identifier	Description
P31697	X	X	X	X	X			FIMC_ECOLI	CHAPERONE PROTEIN FIMC PRECURSOR.
P37923	X	X	X	X	X			FIMC_SALTY	CHAPERONE PROTEIN FIMC PRECURSOR.
P46008	X	X	X	X	X			FOCC_ECOLI	CHAPERONE PROTEIN FOCC PRECURSOR.
P53516	X	X	X	X	X			AFAB_ECOLI	CHAPERONE PROTEIN AFAB PRECURSOR.
P46004	X	X	X	X	X			AGGD_ECOLI	CHAPERONE PROTEIN AGGD PRECURSOR.
P43661	X	X	X	X	X			LPFB_SALTY	CHAPERONE PROTEIN LPFB PRECURSOR.
P21646	X	X	X	X	X			MRKB_KLEPN	CHAPERONE PROTEIN MRKB PRECURSOR.
P42914	X	X	X	X	X			YRAI_ECOLI	HYPOTHETICAL 25.7 KD FIMBRIAL CHAPERONE IN AGAI-MTR INTERGENIC REGION PRECURSOR (O231).
P35757	X	X	X	X				HFB1_HAEIN	CHAPERONE PROTEIN HIFB PRECURSOR.
P45991	X	X	X	X				HFB2_HAEIN	CHAPERONE PROTEIN HIFB PRECURSOR.
P15319	X	X	X	X				PAPD_ECOLI	CHAPERONE PROTEIN PAPD PRECURSOR.
P53520	X	X	X	X				PMFD_PROMI	CHAPERONE PROTEIN PMFD PRECURSOR.
P33409	X	X		X	X			FIMB_BORPE	CHAPERONE PROTEIN FIMB/FHAD PRECURSOR.
P33407	X	X	X		X			MYFB_YEREN	CHAPERONE PROTEIN MYFB PRECURSOR.
P46738	X	X	X		X			NFAE_ECOLI	CHAPERONE PROTEIN NFAE PRECURSOR.
P31523	X	X	X		X			PSAB_YERPE	CHAPERONE PROTEIN PSAB PRECURSOR.
P33387	X	X	X		X			SEFB_SALEN	CHAPERONE PROTEIN SEFB PRECURSOR.
P26926	X	X	X		X			CAFM_YERPE	CHAPERONE PROTEIN CAF1M PRECURSOR (CAPSULE PROTEIN FRACTION 1).
P15483	X	X	X		X			CS31_ECOLI	CHAPERONE PROTEIN CS3-1 PRECURSOR.
P53518	X	X	X					CSC1_ECOLI	CHAPERONE PROTEIN CSSC PRECURSOR.
P33128	X	X		X				ECPD_ECOLI	CHAPERONE PROTEIN ECPD PRECURSOR.
P33342	X	X		X				YEHC_ECOLI	HYPOTHETICAL 26.6 KD FIMBRIAL CHAPERONE IN MRP 5'REGION PRECURSOR.
P42183	X		X					PRSD_ECOLI	CHAPERONE PROTEIN PRSD (FRAGMENT).
P53519		X	X					CSC2_ECOLI	CHAPERONE PROTEIN CSSC PRECURSOR.
P25401				X	X			FAEE_ECOLI	CHAPERONE PROTEIN FAEE PRECURSOR.
P25402				X	X			FANE_ECOLI	CHAPERONE PROTEIN FANE PRECURSOR.
Q05433				X	X			CLPE_ECOLI	CHAPERONE PROTEIN CLPE PRECURSOR.
P40876						X		YCBF_ECOLI	HYPOTHETICAL FIMBRIAL CHAPERONE IN PEPN-PYRD INTERGENIC REGION (FRAGMENT).
P28722							X	YHCA_ECOLI	HYPOTHETICAL 25.3 KD FIMBRIAL CHAPERONE IN GLTF-NANT INTERGENIC REGION PRECURSOR (O224).

Figure 4: Set-membership matrix for all chaperone protein sequences contained in the Swiss-Prot database.

A simple BLAST clustering without iteration similar to the first clustering step performed for the analysis of the E. coli proteins by [8] was compared to our clustering. We accepted all sequences above the same conservative cutoff of 10^{-30} as the cluster of sequences belonging to this query. This leads to the following results: As expected the number of perfect, single sequence clusters (14,377 sequences, 24% of all sequences) remains nearly the same, but the total number of perfect clusters decreases (4,175 clusters containing 15,217 sequences, 26% of all sequences). After removing all inclusions, there are 894 separate maximal clusters comprising 10,285 sequences (17%), so the average size of a maximal cluster is much smaller than in the SYSTERS clustering. The number of connected components increased significantly to 518 with 19,142 sequences (the remaining 33%) contained in 3,869 clusters.

3.2 Clustering algorithm, implementation, performance

From the above description of the set-theoretic features of the CCS we extract the following SYSTERS based clustering algorithm for a protein sequence database. Input to the algorithm is a set of sequences. Its output is a collection of sets of sequences, the clusters.

1. Compute SYSTERS clusters for all sequences in a database.
2. Extract a subset of clusters that partition the database:
   (a) For all identical clusters eliminate all but one.
   (b) For any two pairs of clusters where one includes the other, eliminate the smaller one. Repeat this until no inclusions are left.

(c) From this set accept the clusters which do not overlap any other cluster. These are the perfect and the separate maximal clusters.

3. Compute the connected components of the remaining overlapping maximal clusters.

In this description perfect clusters are generated in one step together with the separate maximal clusters. Overlapping maximal clusters are grouped into connected components and possible decisions about merging the clusters from a connected component into one cluster are left to an expert.

The run time for the database clustering is dominated by the BLAST runs performed for the SYSTERS searches. In our implementation all BLASTP searches are first done in parallel on a workstation cluster. 59,021 searches for the Swiss-Prot database took about 5 days, the results are compressed and stored for further working. Then, based on the BLASTP output, the CCS of SYSTERS clusters is derived by a script written in Perl. A program written in C++ using the LEDA library [11] then executes the above procedure extracting the database clustering. It is worth noting that the test for overlaps among clusters does not involve comparing each cluster with each other one. Instead, it suffices to build a list of sequences annotated with the clusters that they are a member of. Then the test for overlaps will require time linear in the number of sequences instead of quadratic in the number of clusters. The overall execution time for processing the CCS using those two programs is on the order of a few hours. No decisions on the part of the user are necessary during the entire process.

3.3 Clustering PIR1

The above analysis was also done for the PIR1 database [7]. This release of PIR1 contains 13,489 sequences. 20% of these sequences were grouped into single sequence clusters, 50% into perfect clusters, 21% into separate maximal clusters and only 9% of the database sequences are contained in overlapping maximal clusters. The PIR1 database is very carefully annotated with superfamilies [5] that the sequences are grouped into. Therefore we compared the SYSTERS based clustering of PIR1 to the PIR1 superfamily annotation. 51% of the sequences were sorted into clusters that are identical to a PIR1 superfamily, 46% are involved in inclusions of PIR1 superfamilies into SYSTERS clusters or vice versa. In only 22 cases (3% of the sequences) did we observe that a PIR1 superfamily and a SYSTERS cluster overlapped without one containing the other. Further details of this clustering will be described elsewhere.

3.4 Web interface

For the Swiss-Prot and PIR1 databases the set of all disjoint clusters and a manually chosen collection of connected components are being made available for browsing over the World Wide Web:

`http://www.dkfz-heidelberg.de/tbi/services/modest/browsesysters.pl`

The complete database annotations of the sequences in the clusters are searchable using the indexing and query scheme GLIMPSE [9]. This leads to a collection of clusters where for every cluster at least one sequence annotation satisfies the search pattern. The sequences in every cluster have been multiply aligned using ClustalW 1.7 [19] and there are hypertext links from the sequence identifier to the complete database entry in the Swiss-Prot or PIR1 database. With each cluster its set-membership matrix (cf. Figure 3) is available as well as an unrooted tree (computed using Neighbor-Joining [15]).

4 Summary and Conclusions

We introduced a novel database search method, SYSTERS, that iterates a traditional search procedure like BLAST and produces clusters of sequences related to the query. We showed that this procedure is to a large degree internally consistent in the sense that the resulting cluster shows little dependence

on the specific query. This has provided the foundation for a database clustering method that sorts sequences into clusters of which a large fraction is pairwise disjoint.

The resulting set of clusters is self-validating since problems become manifest in overlaps between clusters. The real success of the method lies in automatically delineating the subset of sequences that can be sorted into non-overlapping clusters. Since no special procedure is applied to enforce the disjointness, it may be taken as an indicator that this information is in fact correct. Problematic cases reveal themselves through overlaps and are collected in the connected components. Thus, the limit of automation is pushed as far as possible without enforcing arbitrary decisions. At the same time this allows for a model of updating the clustering that uses the same logic as the clustering itself: A new sequence can be searched against the ungrouped set of sequences using SYSTERS. It will generate its clusters of related sequences. If the sequences identified in this way happen to constitute a cluster already, this is strong evidence for membership of the new sequence in this cluster.

The granularity of this clustering is determined by the data and not by a user-supplied cutoff. Mostly, the clustering is conservative and clusters do not comprise evolutionarily diverse families. Detailed inspection of the clusters, however, provides exceptions to the rule and diverse members can be found in some clusters. Since the perfect and separate maximal clusters are non-overlapping they can by definition not represent common domains between different protein families. Rather, knowledge about common protein folds or common domains needs to be referenced on top of this clustering. Having established an automatic and efficient method for the conservative clustering should provide a sound basis for the automation of the second level of analysis.

Acknowledgments

We thank Marc Rehmsmeier for the creation of the World Wide Web interface. Financial support by the German Ministry of Research BMBF (Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie) is gratefully acknowledged.

References

[1] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J., "Basic Local Alignment Search Tool", *J. Mol. Biol.*, 215:403–410, 1990.

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

[3] Bairoch, A. and Apweiler, R., "The SWISS–PROT protein sequence data bank and its supplement TrEMBL", *Nucleic Acids Research*, 25(1):31–36, 1997.

[4] Bairoch, A., Bucher, P. and Hofmann, K., "The PROSITE database, its status in 1997", *Nucleic Acids Research*, 25(1):217–221, 1997.

[5] Barker, W.C., Pfeiffer, F. and George, D.G., "Superfamily Classification in PIR–International Protein Sequence Database", *Methods in Enzymology*, 266:59–71, 1996.

[6] Dayhoff, M.O. and Eck, R.V. and Chang, M.A. and Sochard, M.R., *Atlas of Protein Sequence and Structure*, National Biomedical Research Foundation, Silver Spring, MD, 1965

[7] George, D.G., Dodson, R.J., Garavelli, J.S., Haft, D.H., Hunt, L.T., Marzec, C.R., Orcutt, B.C., Sidman, K.E., Srinivasarao, G.Y., Yeh, L.-S.L., Arminski, L.M., Ledley, R.S., Tsugita, A. and Barker, W., "The Protein Information Resource (PIR) and the PIR–International Protein Sequence Database", *Nucleic Acids Research*, 25(1):24–27, 1997.

[8] Koonin, E.V. and Tatusov, R.L. and Rudd, K.E., "Sequence similarity analysis of Escherichia coli proteins: Functional and evolutionary implications", *Proc. Natl. Acad. Sci. USA*, 92:11921–11925, 1995.

[9] Manber, U. and Wu, S., "GLIMPSE: A Tool to Search Through Entire File Systems", *Usenix Winter 1994 Technical Conference, San Francisco*, 23–32, 1994.

[10] Matsuda, H. and Ishihara, T. and Hashimoto, A., "A Clustering Method for Molecular Sequences based on Pairwise Similarity", *Proceedings of the Seventh Workshop on Genome Informatics, Universal Academy Press, Tokyo*, 1996.

[11] Mehlhorn, K. and Näher, S., "LEDA: A Platform for Combinatorial and Geometric Computing", *Communications of the ACM*, 38(1):96–102, 1995.

[12] Neuwald, A.F., Liu, J.S., Lipman, D.J. and Lawrence, C.E., "Extracting protein alignment models from the sequence database", *Nucleic Acids Research*, 25(9):1665–1677, 1997.

[13] Pearson, W.R. and Lipman, D.J., "Improved tools for biological sequence comparison", *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.

[14] Pearson, W.R., "Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith–Waterman and FASTA algorithms", *Genomics*, 11(3):635–650, 1991.

[15] Saitou, N. and Nei, M., "The neighbor-joining method: A new method for reconstructing phylogenetic trees", *Mol. Biol. Evol.*, 4:406–425, 1987.

[16] Schultz, J., Ponting, Ch.P., Hofmann, K. and Bork, P., "SAM as a protein interaction domain involved in developmental regulation", *Protein Science*, 6:249–253, 1997.

[17] Smith, T.F. and Waterman, M.S., "The identification of common molecular subsequences", *J. Mol. Biol.*, 147:195–197, 1981.

[18] Sonnhammer, E.L.L. and Kahn, D., "Modular arrangement of proteins as inferred from analysis of homology", *Protein Science*, 3:482–492, 1994.

[19] Thompson, J.D., Higgins, D.G. and Gibson, T.J., "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice", *Nucleic Acids Research*, 22(22):4673–4680, 1994.