A pattern mining approach toward discovering generalized sequence signatures

Dietmar H. Dorr Anne M. Denton
Department of Computer Science
North Dakota State University
Fargo, ND 58105, USA
{Dietmar.Dorr, Anne.Denton}@ndsu.edu

Abstract

Typically, sequence signatures, such as motifs and domains, are assumed to be localized in one region of a sequence or are derived as combinations of the former. We generalize the concept of sequence signatures and introduce an algorithm for efficiently determining signatures based on subsequences that may be located anywhere on a sequence. In a preprocessing step, sequences are transformed into a feature space that is subsequently used to mine generalized signatures. We evaluate our signatures in relation to those in the InterPro database and highlight the differences between them. A second comparison with InterPro shows that our signatures can be used to derive sequence annotations with a higher confidence.

1 Introduction

In recent years, gene sequencing efforts have produced a wealth of sequence information, and the genomes of many species have been fully sequenced. In contrast, functional information is more difficult to obtain and often less reliable. Correspondingly, there is much incentive for transferring information from well-studied model species to others.

Protein domains and motifs [9, 19] are serving as a separate level of abstraction that hides sequence details while preserving similarities at the functional level. One of the most commonly used representations for protein domains is based on hidden Markov models [8]. Hidden Markov models allow representing the probability density of amino acids at each position in the domain, as well as modeling gaps. That means that a probability model for the occurrence of gaps is imposed by the design of the model. Domains that have been identified through one or more of these techniques may then be combined into superfamilies. Following the notation of the InterPro database [19] we will refer to domains, motifs and other sequence patterns as signatures.

This paper introduces an algorithm for deriving sequence signatures that makes no assumptions about the structure of gaps. The algorithm uses pattern mining concepts that are known from frequent itemset discovery in market basket research. Figure 1 summarizes the steps: A first alignment component reduces the size of input data, and ensures that each window has at least one match. Items are derived from the alignments using a sliding window approach (Windowing) and a subsequent grouping of windows (Linking). The grouping process is based on single linkage hierarchical clustering to allow capturing remote homologies [10]. Items are further processed using frequent itemset mining, regardless of their position on the sequence. It is this flexibility that allows finding complex signatures on the basis of frequently occurring windows alone, and regardless of the shape of the signature.

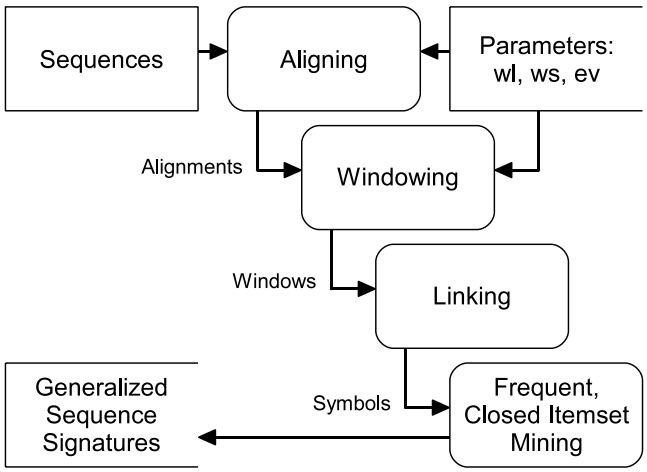
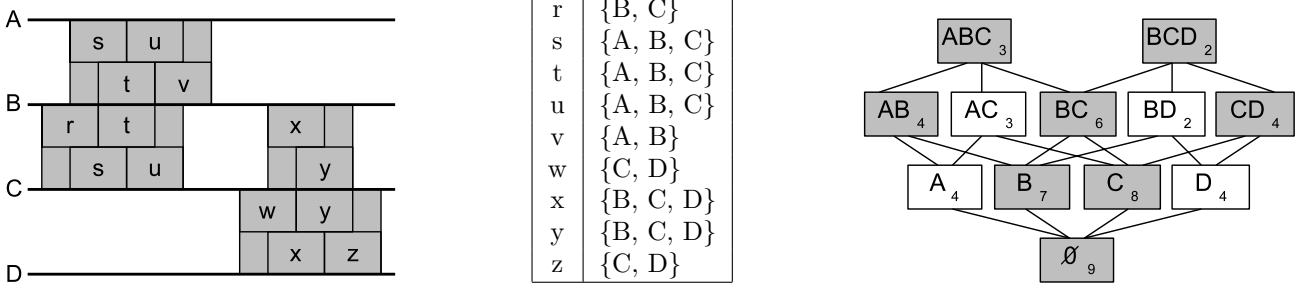


Figure 1: Data flow diagram of our approach. The first three processes establish a space of sequence features that is subsequently used to mine generalized sequence signatures.

A more detailed representation of our approach can be seen in Figure 2. The left panel shows how window-based alignments can be identified within local alignments, as they are commonly used in bioinformatics. The windows are given names based on a clustering step



T	Itemset
r	{B, C}
s	{A, B, C}
t	{A, B, C}
u	{A, B, C}
v	{A, B}
w	{C, D}
x	{B, C, D}
y	{B, C, D}
z	{C, D}

Figure 2: Conceptual representation of our approach. Left: The alignments (gray) among four sequences together with labeled windows (symbols) within them. Center: The itemsets (sequences) that share a symbol (transaction T). Right: The lattice of itemsets together with their support count; closed itemsets are highlighted in gray.

that is called Linking in Figure 1. The middle panel shows the transaction table that forms the basis of the frequent itemset mining. In this table, symbols form the transactions and sequences the items. This choice allows a user to focus on complex patterns by increasing the support threshold. We also require symbols to be supported by at least three sequences through an initial filtering step.

As part of the evaluation, we calculate the confidence of signature → annotation rules, but we intentionally do not filter using this confidence. Even for well-studied species, functional information cannot be expected to be complete and we, therefore, choose not to limit the set of derived signatures by requiring them to be predictive based on the existing functional data.

2 Related Work and Notation

2.1 Related Work A variety of algorithms have been proposed for the discovery of sequence signatures. The approaches differ in their definition of a targeted signature, or in the applied methods. Signatures with a particular length are addressed by some algorithms [5, 6], and others focus on signatures with a specific composition [28, 29]. Some approaches discover signatures by maximizing the number of subsequences associated with a signature [13, 24], or determining all maximal motifs [21, 15]. Methods used to identify signatures include hidden Markov models [9, 18, 12], Gibbs Sampling [16], expectation maximization [2], greedy methods [14], and clustering [7].

Techniques that focus on similarities among windows (l-mers, q-grams, shingles) have also been used to discover signatures [21, 15, 7], and to define measures of similarity between two sequences [26]. Our similarity measure among windows specifically addresses transitivity [10].

The signatures of some approaches, e.g. Pfam [9], PANTHER [18], and SUPERFAMILY [12], are incorporated into a single database, called InterPro [19]. Our definition of a signature is based on frequently co-occurring windows in sequences, which is similar to our proposal in [7]. In our previous work [7], we utilize clustering to identify signatures, and in this paper the frequent itemset mining algorithm FPclose [30] is applied. Frequent itemset mining is studied extensively, and many algorithms have been proposed [30, 11, 3]. FPclose is chosen, because its implementation is readily available (`http://fimi.cs.helsinki.fi/src/fimi06b.tgz`), and it performs well on our data set (Section 5.5).

A sequence annotation is typically derived based on commonalities between sequences with and without the annotations. A commonality can be a high similarity between entire sequences [22], or signatures [25, 4]. Our approach for annotation of sequences is similar to Biswas et al. [4], but the underlying signatures are different.

2.2 Notation While the evaluation in this paper is limited to biological data, the algorithm can, in principle, be used for any type of sequence. Correspondingly the notation is kept general. A sequence s_a is an ordered list of elements, where each element $s_a[i]$ is a member of an alphabet Σ. In the case of protein sequences, the alphabet Σ consists of 20 elements, each denoting one amino acid. A continuous segment of a sequence s_a is called a subsequence $s_{a,t:u}$, $t \leq u$, where $s_{a,t}$ is the first element, located at position t, and $s_{a,u}$ is the last element at position u.

The degree of similarity between two equally long subsequences is measured using a scoring function (score) that returns a value in $\mathbb{R}$. For biological sequences the score measure is typically calculated using a substitution matrix. The first Aligning step compares subsequences $s_{a,t:t+l}$, and $s_{b,v:v+l}$, $l \geq 1$ of the two se-

quences, where t and v may be arbitrary locations. This flexibility is typically required as a minimum in bioinformatics, and the result is called an ungapped local alignment [8]. The Windowing step is also based on subsequences, albeit fixed-length subsequences (Section 3.2).

3 Approach

This section provides details on the steps of the algorithm that is summarized in Figure 1. The first three steps, Aligning, Windowing, and Linking, are covered in Sections 3.1 through 3.3. They result in a symbolic representation of sequence features. Each symbol stands for a set of similar windows. Conceptually, symbols can be viewed as Boolean variables, in much the same way as items in market basked research. The subsequent mining process (Section 3.4) identifies frequent combinations of features that are interpreted as generalized sequence signatures.

The algorithm takes two parameters as input: the length of a window wl, and the minimum score ws a pair of windows has to satisfy in order to be considered similar. An E-value ev can be derived from the two parameters and is used to define alignments (Section 3.1) and to limit InterPro signatures in comparisons (Section 3.6).

3.1 Aligning The Aligning process identifies regions of similarity among the set of all sequences $S = \{s_1, s_2, \ldots, s_m\}$ to serve as basis for the subsequent sliding window extraction. We only consider ungapped alignments since the Windowing and Mining steps inherently produce gaps in signatures.

DEFINITION 3.1. *An ungapped,* ***local alignment*** *between two sequences s_a and s_b is given by two subsequences $s_{a,t:t+l-1}$ and $s_{b,v:v+l-1}$ of the same length l for which*

$$\text{score}(s_{a,t:t+l-1}, s_{b,v:v+l-1}) \geq ws.$$

It is represented as a tuple $a(s_{a,t}, s_{b,v}, l)$.

Conventionally, such as in the Smith-Waterman alignment algorithm [23], only the highest scoring alignment is returned. For performance reasons database searches are typically done using BLAST [1], which may return multiple results for the same two sequences. We use BLAST in the alignment step. Although BLAST is neither guaranteed to find all alignments, nor even all highest scoring alignments (due to its inherent heuristics) it provides an efficient means of limiting input sequence sections to those that are most likely to be relevant. BLAST is run with the option of returning only ungapped alignments.

3.2 Windowing The algorithm uses windows as atomic unit for the definition and mining of patterns. The length of a window wl is one of the parameters that are chosen by the user. Sequence similarity over the length of windows is the basis for the definition of symbols that are used in the Mining step:

DEFINITION 3.2. *A* ***window*** *$w(s_{a,t})$ is a subsequence $s_{a,t:t+wl-1}$ of sequence s_a with length wl.*

For efficiency reasons, we only consider windows that are represented within an alignment (Section 3.1). Each subsequence of length wl of an alignment is considered, corresponding to sliding window extraction. The similarity between two windows is defined by requiring that there is an alignment that 1) contains the two windows at the same relative position, and 2) the score of aligning the two windows is at least ws. The second criterion establishes that an alignment of two windows satisfies the minimum E-value threshold ev. More formally:

DEFINITION 3.3. *Two windows $w(s_{a,t'})$ and $w(s_{b,v'})$ are* ***similar****, $w(s_{a,t'}) \backsimeq w(s_{b,v'})$, if $\text{score}(s_{a,t':t'+wl-1}, s_{b,v':v'+wl-1}) \geq ws$ and there is an alignment $a(s_{a,t}, s_{b,v}, l)$ such that $t \leq t' \leq t + l \wedge v \leq v' \leq v + l \wedge t - t' = v - v'$ holds.*

3.3 Linking So far similarity is only defined for pairs of windows. The definition of features requires that more than two windows can be associated. In agglomerative hierarchical clustering a variety of prescriptions for combining multiple objects on the basis of pairwise similarity measures have been defined. We use the least restrictive, single linkage, as basis for the generation of features. The single linkage criterion only requires elements of a cluster to show transitive similarity. A recursive definition of transitive similarity is:

DEFINITION 3.4. *Two windows $w(s_{a,t})$ and $w(s_{c,x})$ are* ***transitively similar****, $w(s_{a,t}) \backsimeq_t w(s_{c,x})$, if either $w(s_{a,t}) \backsimeq w(s_{c,x})$, or $\exists w(s_{b,v}) : w(s_{a,t}) \backsimeq_t w(s_{b,v}) \wedge w(s_{b,v}) \backsimeq_t w(s_{c,x})$.*

Note that two (or more) different windows $w(s_{a,t})$ and $w(s_{a,u})$ of the same sequence s_a may be transitively similar to each other $w(s_{a,t}) \backsimeq_t w(s_{a,u})$, because of a third window $w(s_{b,v})$ of a second sequence that is similar to both windows $w(s_{a,t}) \backsimeq w(s_{b,v}) \wedge w(s_{b,v}) \backsimeq w(s_{a,u})$. However, the two similarities to the third window $w(s_{b,v})$ must originate from two different alignments between the sequences s_a and s_b. Using the transitive similarity of windows, we define disjoint sets of windows. Each set of transitively similar windows is labeled with a unique identifier, called a symbol.

DEFINITION 3.5. *A **symbol** represents a set of windows* $W = \{w_1, w_2, \ldots, w_n\}$, $n \geq 3$ *such that*

$$\forall w_i, w_j \in W : w_i \backsimeq_t w_j.$$

To reduce the chance of obtaining symbols due to biologically irrelevant matches, we require that each symbol represents windows of at least three different sequences. We denote the set of symbols associated with a particular sequence s_a by $\text{sym}(s_a)$. Furthermore, the number of symbols shared by a set of sequences S' is defined as the support count of S'.

DEFINITION 3.6. *The **support count** $\text{suppc}(S')$ of a set of sequences $S' \subseteq S$ is the number of shared symbols:*

$$\text{suppc}(S') = \mid \bigcap_{s_a \in S'} \text{sym}(s_a) \mid .$$

3.4 Frequent, Closed Itemset Mining The final step of the algorithm groups symbols using pattern mining concepts to derive complex patterns. We use an algorithm for mining frequent closed itemsets (FPclose [30]). Each symbol y is interpreted as a transaction, and the set of sequences associated with a symbol $\{s_a \mid y \in \text{sym}(s_a)\}$ is considered the corresponding itemset. In the evaluation we used a minimum support count (min_suppc) of 1, which means that even patterns that only consist of a single symbol will be discovered.

DEFINITION 3.7. *A frequent, closed **itemset** I is a set of sequences such that* $\text{suppc}(I) \geq min\_suppc \wedge \nexists S' : I \subset S' \subseteq S : \text{suppc}(I) = \text{suppc}(S')$.

We consider closed itemsets, because they represent the maximum number of sequences of any frequent itemset that contains the same symbols. Based on the frequent, closed itemsets, we introduce the following definition of a generalize sequence signature:

DEFINITION 3.8. *A **generalized sequence signature** $\text{sig}(I)$ is a set of symbols that is shared by the sequences of a nonempty, frequent, closed itemset I:*

$$\text{sig}(I) = \bigcap_{s_a \in I} \text{sym}(s_a).$$

By interpreting each symbol of a generalized sequence signature as a representative of a set of transitively similar windows, the signature describes a collection of windows for each sequence that is associated with it. The represented windows of a particular sequence may be located anywhere on that sequence such that any number of gaps between two windows is permissible.

A set of windows of a particular sequence can be converted to a collection of non-overlapping subsequences that are separated by at least one sequence element. We refer to subsequences reconstructed from sets of windows as intervals. An interval may be composed of a single window, or of a collection of overlapping or adjacent windows. More formally:

DEFINITION 3.9. *A **set of intervals** is a set of the longest subsequence SS that is based on a nonempty set of windows W, such that*

$$\forall s_{a,t:u} \in SS : \forall i : t \leq i \leq u : \exists w(s_{a,v}) \in W : v \leq i \leq v + wl - 1$$

holds.

Since a symbol is allowed to represent different numbers of windows on different sequences, the number and lengths of the intervals that can be reconstructed for the sequences of a particular signature may differ.

3.5 Algorithm The pseudocode for our algorithm is shown in Algorithm 1. The input is the set of all sequences S as well as the window length wl and window score ws. As outlined in the data flow diagram (Figure 1), the algorithm is composed of four processes. The Aligning process (line 1) determines all alignments between pairs of sequences that satisfy the E-value threshold ev, which is derived from wl and ws for the given database size and scoring matrix. The second process, Windowing, is shown in lines 2 to 6. For each alignment, all window pairs of length wl are determined that have a score of at least ws. Once line 7 is reached, the set W contains all pairs of windows that satisfy the score criterion. The Linking process extends from line 7 to 21, and establishes the sets of all transitively similar windows. Using the *symnew* variable, each set is automatically labeled with a unique identifier, which also serves as the symbol. The Linking utilizes a bidirectional multimap (BDMMap) for keeping track of the relationships between symbols and sets of windows. The BDMMap data structure is multimap, because it associates a key (symbol) with multiple values (windows). The bidirectional characteristic of the BDMMap data structure refers to its ability to handle reverse lookups (line 10 and 11). The linking of windows is finished beginning with line 22, but its results in the BDMMap data structure have to be converted before the application of the mining algorithm. In lines 22 to 28, the sets of all transitively similar windows in the BDMMap data structure are converted to a set T of itemsets such that each itemset $I \in T$ contains sequences rather than windows. Since each symbol must represent windows

Data: S; // set of sequences
Input: wl; // window length
Input: ws; // window score
Input: ev; // E-value threshold
Result: $itemsets$; // set of frequent, closed itemsets
1 $A \longleftarrow \text{align}(S, ev)$; // Aligning
2 $W \longleftarrow \emptyset$; // Windowing
3 **foreach** $a(s_{a,t}, s_{b,v}, l) \in A$ **do**
4 **for** $i \longleftarrow 0$ **to** $l - wl$ **do**
5 **if** $\text{score}(s_{a,t+i:t+i+wl-1}, s_{b,v+i:v+i+wl-1}) \geq ws$ **then**
6 $W \longleftarrow W \bigcup (w(s_{a,t+i}), w(s_{b,v+i}))$;
7 $symnew \longleftarrow 1$; // Linking
8 BDMMap $SymXWin \longleftarrow$ new BDMMap();
9 **foreach** $(w(s_{a,t}), w(s_{b,v})) \in W$ **do**
10 $symA \longleftarrow SymXWin.\text{ReverseLookup}(w(s_{a,t}))$;
11 $symB \longleftarrow SymXWin.\text{ReverseLookup}(w(s_{b,v}))$;
12 **if** $symA \leq 0 \wedge symB \leq 0$ **then**
13 $SymXWin.\text{Add}(symnew, \{w(s_{a,t}), w(s_{b,v})\})$;
14 $symnew \longleftarrow symnew + 1$;
15 **else if** $symA \leq 0 \wedge symB > 0$ **then**
16 $SymXWin.\text{Reassign}(symB, SymXWin.\text{Lookup}(symB) \bigcup \{w(s_{a,t})\})$;
17 **else if** $symA > 0 \wedge symB \leq 0$ **then**
18 $SymXWin.\text{Reassign}(symA, SymXWin.\text{Lookup}(symA) \bigcup \{w(s_{b,v})\})$;
19 **else if** $symA \neq symB$ **then**
20 $SymXWin.\text{Reassign}(symB, SymXWin.\text{Lookup}(symB) \bigcup SymXWin.\text{Lookup}(symA))$;
21 $SymXWin.\text{Remove}(symA)$;
22 $T \longleftarrow \emptyset$; // creating the itemsets to be mined
23 **foreach** $sym \in SymXWin.\text{Keys}()$ **do**
24 $I \longleftarrow \emptyset$;
25 **foreach** $w(s_{a,t}) \in SymXWin.\text{Lookup}(sym)$ **do**
26 $I \longleftarrow I \bigcup \{a\}$;
27 **if** $\mid I \mid \geq 3$ **then**
28 $T \longleftarrow T \bigcup \{I\}$;
29 $itemsets \longleftarrow \text{FPclose}(T)$; // Frequent, Closed Itemset Mining
30 **return** $itemsets$;

Algorithm 1: Algorithm of our approach.

of at least 3 sequences (Section 3.3), only those itemsets I are added to the set of all itemsets T that have a cardinality of at least 3. The last process in line 29 utilizes the FPclose algorithm to mine all frequent, closed *itemsets*.

3.6 Comparison to InterPro's Signatures The first evaluation compares the resulting signatures with InterPro's signatures. An **InterPro signature** is considered to represent a set of subsequences. Each element of a subsequence set denotes a different instance of the corresponding signature. Since neither the InterPro nor our approach provides signatures on every sequence, we limit our comparisons to those sequences that have both an InterPro and generalized sequence signature. Otherwise, the comparisons would be affected by the degree to which the sequences are utilized for defining signatures. In addition, our comparisons only consider those subsequences of InterPro signatures that satisfy the minimum length wl and E-value ev thresholds.

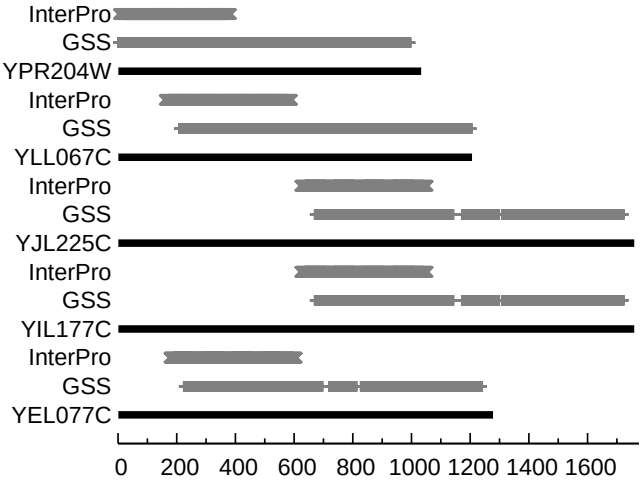


Figure 3: Sequential representation of a frequent, closed itemset consisting of five sequences (black lines). Above each sequence the segments represented by the generalized sequence signature of the itemset (GSS) and the InterPro signatures is shown.

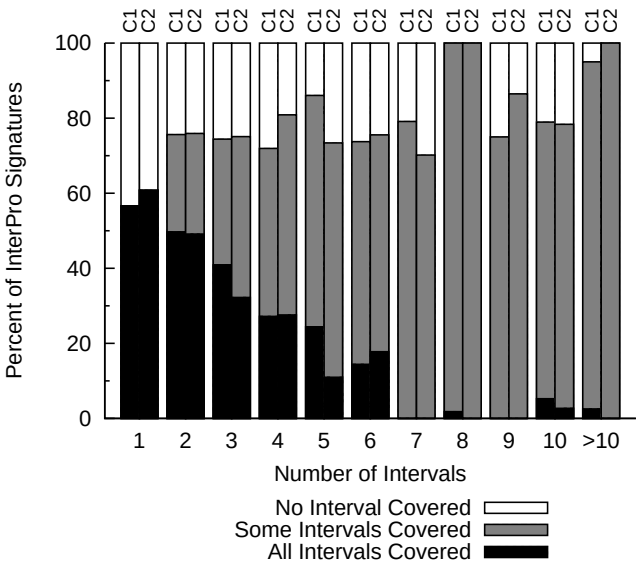


Figure 4: Comparison between InterPro signatures and the sequence segments (intervals) defined by our symbols. For the two parameter combinations $C1$ and $C2$, the percentages of InterPro signatures are shown that cover all, some, or none of the intervals.

3.6.1 Symbols vs InterPro Each sequence is associated with a particular set of symbols, which in turn represents a particular set of windows as well as intervals. For each sequence, we compare its intervals to each InterPro signature that is located on the sequence. It is determined whether the InterPro signature covers all, some, or none of the intervals of the sequence. An interval is covered by an InterPro signature, if at least 50% of the interval's elements are part of the signature. A subsequence $s_{a,t:u}$ **covers** another subsequence $s_{a,v:w}$, if

$$\frac{\min(u,w) - \max(t,v) + 1}{w - v + 1} \geq 50\%.$$

3.6.2 Class Association Rule Mining Our second evaluation utilizes signatures in order to derive annotations. The underlying assumption is that if some sequences share a signature, then it is likely that these sequences have the same function (annotation). We model **annotations** in the form of a binary relation $\text{anno}(o,t)$ between an object o and a GO (Gene Ontology) term t. The object can represent a sequence or a signature. Using the data set of original sequence annotations (Section 4) we first derive annotations for each signature, and then determine two additional sets of sequence annotations. The derivation of signature annotations is achieved by determining class association rules. The sequences of a signature represent the antecedent, and the consequent is a GO term t. Each association rule (signature annotation) is associated with a confidence value denoting its strength.

DEFINITION 3.10. *The* ***confidence*** $\text{conf}(S' \longrightarrow t)$ *of an association rule that associates a set of sequences* S' *with a GO term* t *is:*

$$\text{conf}(S' \longrightarrow t) = \frac{|\{s|\text{anno}(s,t)\}|}{|S'|}.$$

Using the above signature annotations, we determine two additional sets of sequence annotations (IP and MA). One set of sequence annotations (IP) is based on the InterPro signatures, and the second set (MA) is based on our generalized sequence signatures. A sequence is assigned a GO term t, if there is a signature on the sequence that is assigned the term t. The same sequence annotation may be derived using different InterPro or generalized sequence signatures. We define the confidence value for a sequence annotation as the highest confidence of a signature annotation that provides the sequence annotation. In order to eliminate trivial inferences, we only consider those signature annotations that are based on at least two sequences. The **confidence** $\text{conf}(s_a \longrightarrow t)$ of an association rule that associates a sequence $s_a \in S$, with a GO term t is:

$$\text{conf}(s_a \longrightarrow t) = \max_{\substack{S' \longrightarrow t \\ |S'| \geq 2 \\ s_a \in S'}} \{\text{conf}(S' \longrightarrow t)\}.$$

Our approach for annotating signatures is similar to the one taken by InterPro. InterPro automatically annotates its signatures based on a confidence threshold

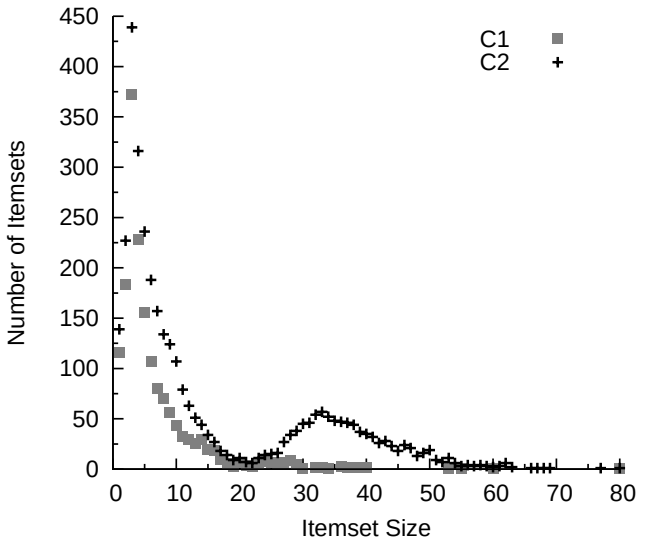


Figure 5: The number of itemsets (signatures) in relation to the itemset size (number of sequences associated with a signature) for both parameter combinations $C1$ and $C2$.

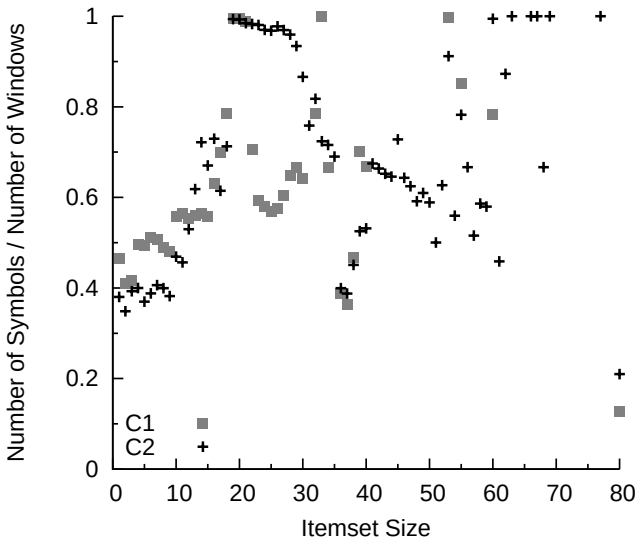


Figure 6: The ratio of the average number of symbols and the average number of windows in relation to the itemset size for both parameter combinations $C1$ and $C2$.

of 100% [4]. For the evaluation we rederive InterPro signatures for a spectrum of confidence values.

4 Data and Parameter Choices

We apply our approach to 5883 yeast (*Saccharomyces cerevisiae*) protein sequences (`ftp://genome-ftp.stanford.edu/pub/yeast/data_download/sequence/genomic_sequence/orf_protein/orf_trans.fasta.gz`) and utilize GO sequence annotations (`ftp://genome-ftp.stanford.edu/pub/yeast/literature_curation/gene_association.sgd.gz`) that assign a GO term to the aforementioned yeast sequences. In order to avoid addressing ontologically related GO terms and affecting the confidence of a derived annotation due to different but very similar GO annotations, we convert the sequence annotations such that only a subset of all GO terms is used. This conversion is achieved by using a maintained GO slim for yeast (`http://www.geneontology.org/GO_slims/goslim_yeast.obo`) and the Perl script 'map2slim' (`http://search.cpan.org/~cmungall/go-perl-0.06/scripts/map2slim`). Furthermore, we only focus on those sequence annotations that are associated with the evidence code IDA (Inferred from Direct Assay) or IMP (Inferred from Mutant Phenotype). The sequence signatures for the above protein sequences originate from the InterPro database (`ftp://genome-ftp.stanford.edu/pub/yeast/sequence_similarity/domains/domains.tab`) and are determined by an application of the InterProScan tool [20].

Table 1: Parameter Combinations

Combination	wl	ws	E-value	ev
$C1$	25	96	8.34E-6	1E-5
$C2$	30	96	8.23E-6	1E-5

Our approach is applied to the above data sets with two different combinations of parameters as shown in Table 1. In order to investigate the effects of different window lengths, wl is chosen to be 25 in the parameter combination $C1$ and 30 in $C2$. Using the median K and λ values provided by BLAST ($K = 0.135$ and $\lambda = 0.318$), the corresponding minimum window scores ws are determined such that they represent the smallest possible score resulting in an E-value below the threshold ev.

5 Results

5.1 Example An example of a generalized sequence signature is depicted in Figure 3, which is determined using the parameter combination $C1$. The figure shows all sequences with the signature as well as all InterPro signatures that are located on those sequences. The InterPro signatures for each sequence are projected to one line. Each sequence is assigned at least five of the InterPro signatures shown in Table 2. None of InterPro's signature covers every interval of our generalized sequence signature. Furthermore, there are extensive regions on each sequence that are not covered by any one of InterPro's signatures.

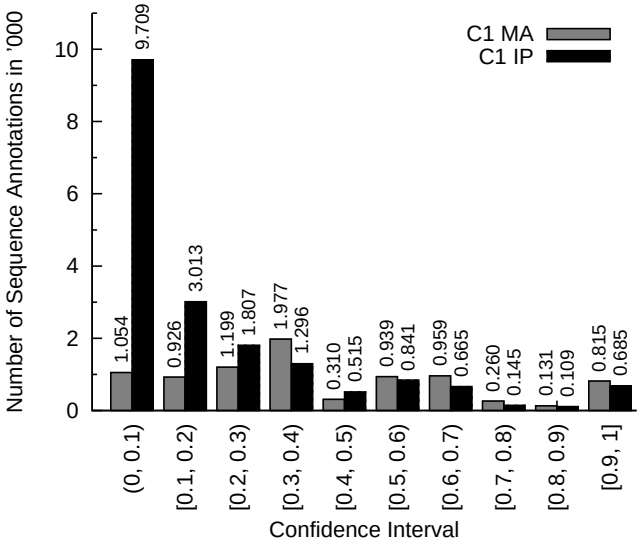


Figure 7: Histogram showing the number of derived sequence annotations obtained by using InterPro's (IP) and our generalized sequence signatures (MA) for the parameter combination $C1$.

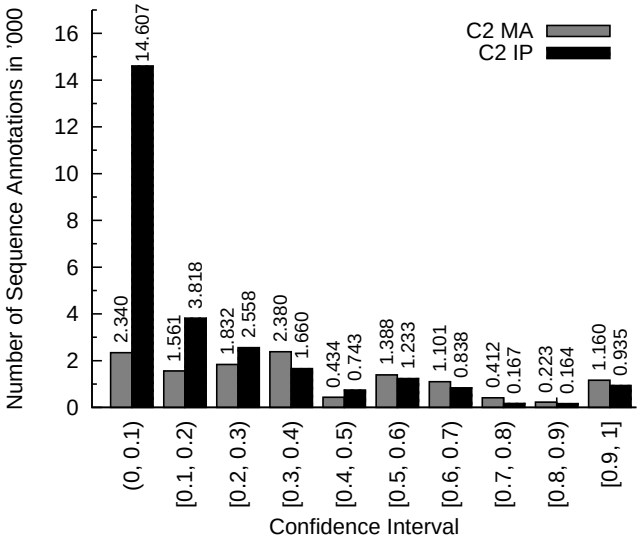


Figure 8: Same histogram as Figure 7 but for the parameter combination $C2$.

5.2 Symbols vs. InterPro Figure 4 shows the results of the comparison between InterPro signatures and our symbols as described in Section 3.6.1. The identified symbols represent subsequences (intervals) that are only partially covered by InterPro's signatures. The number of InterPro signatures that cover all intervals is overall decreasing with the number of intervals.

5.3 Frequent, Closed Itemsets For the two parameter combinations, the number of itemsets in relation to the itemset size is shown in Figure 5. An itemset I can also be interpreted as a generalized sequence signature $\text{sig}(I)$ (Definition 3.8), and the size of an itemset $|I|$ is equivalent to the number of sequences associated with the corresponding signature. In closed itemset mining the number of itemsets may increase, plateau, or decrease between two different itemset sizes. Remarkably, the histogram for the parameter combination $C2$ has a local maximum at an itemset size of 33. The parameter combination $C1$ shows a similar, albeit less pronounced, characteristic at an itemset size of 28.

In Figure 6 the ratio between the average number of symbols and the average number of corresponding windows is depicted for itemsets (signatures) of different sizes (number of associated sequences). For small itemsets (≤ 20), the ratio increases with the itemset size. For larger itemsets, the ratios of the parameter combination $C1$ and $C2$ differ strongly. This is likely to be the result of the differences in the number of itemsets (Figure 5).

Table 2: InterPro signatures shown in Figure 3. The origin column indicates the database that defines the signature and implicitly also the methodology used to identify the signature.

Name	Origin
G3DSA:3.40.50.300	Gene3D [27]
PF00270	Pfam [9]
PF00271	Pfam
PTHR14074	PANTHER [18]
SM00490	SMART [17]
SSF52540	SUPERFAMILY [12]

5.4 Class Association Rules Based on the signature annotations, we derive two sets of sequence annotations (Section 3.6.2). One set (IP) only uses InterPro's signatures, and the other set (MA) is solely based on our generalized sequence signatures. The histograms in Figures 7 and 8depict the number of sequence annotations of each set (IP and MA) that are derived for different confidence intervals, and parameter combinations ($C1$ and $C2$). The sequence annotations obtained by using our signatures outnumber those obtained by using InterPro's signatures in the confidence intervals of 50% and above. Furthermore, until a confidence threshold as low as $\geq 20\%$ the annotations obtained by using our signatures also outnumber those obtained by using InterPro's signatures.

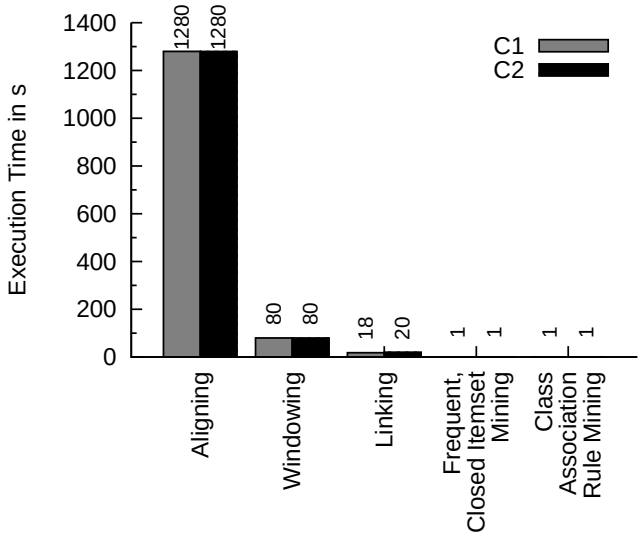


Figure 9: Overview of the execution times for the different processes shown in Figure 1 and the class association rule mining.

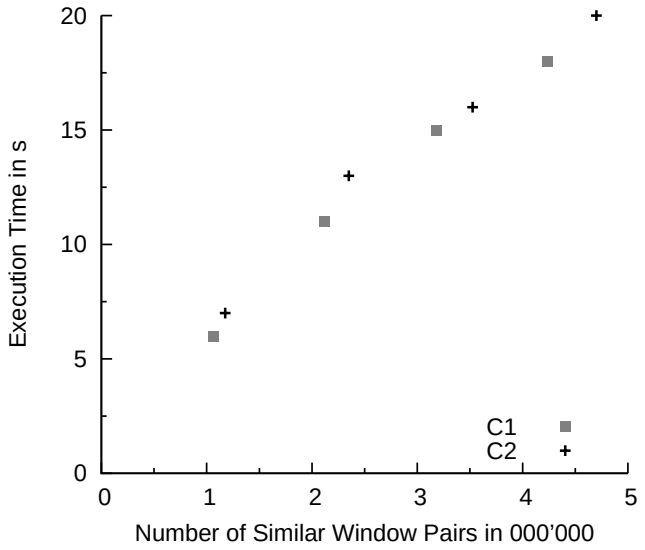


Figure 10: Performance analysis of the Linking step (Section 3.3) for all similar window pairs determined by using the two parameter combination $C1$ and $C2$. Additional data points are obtained by only considering a quarter, one half, and three quarters of the input.

5.5 Performance Figure 9 depicts the execution times for the different processes shown in Figure 1 and our class association rule mining approach (Section 3.6.2). Overall, the Aligning step has the largest contribution to the execution time. Since BLAST is commonly considered as a fast tool, it can be concluded that the performance is sufficient for practical purposes.

Figure 9 shows that a difference between the two parameter combinations used in the evaluation, $C1$ and $C2$, can be observed for the Linking step. The input to the Linking step is larger for the parameter combination $C2$ than for $C1$ explaining the increase in execution times. We provide a more detailed analysis of this process in Figure 10. Besides the performance for all similar window pairs of $C1$ and $C2$, the execution times for processing a quarter, one half, and three quarters of the input are also provided. The Linking process shows near linear performance.

6 Conclusion

We have presented an algorithm for deriving complex sequence signatures using frequent itemset mining techniques. The approach is based on sliding window extraction of fixed-length subsequences. The windows are grouped into disjoint sets using sequence clustering. The sets, to which we refer as symbols are combined in a frequent-item-set-mining step that results in complex sequence signatures. We demonstrate the effectiveness and efficiency of the approach based on all yeast sequences and gene ontology annotations. Using an example we demonstrate the different nature of our signatures. We also show that the more complex signatures are typically not covered well by InterPro signatures. Finally, we are able to demonstrate that annotations based on our sequence signatures, generally have a higher confidence than those based on InterPro-like annotations.

7 Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. DM-0415190. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the supporting agency.

References

[1] S. F. Altschul, T. L. Madden, A. A. Schaffer, 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] T. L. Bailey and C. Elkan. The value of prior knowledge in discovering motifs with MEME. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 21–29, 1995.

[3] R. Bayardo, B. Goethals, and M. J. Zaki, editors. *Workshop on Frequent Itemset Mining Implementations*, volume 126. CEUR-WS.org, 2004.

[4] M. Biswas, J. F. O'Rourke, E. Camon, et al. Applications of interpro in protein annotation and genome analysis. *Briefings in Bioinformatics*, 3(3):285–295, 2002.

[5] J. Buhler and M. Tompa. Finding motifs using random projections. In *Proceedings of the Fifth International Conference on Computational Biology*, pages 69–76, 2001.

[6] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining*, pages 493–498, 2003.

[7] D. Dorr and A. Denton. Generalized sequence signatures through symbolic clustering. In *Proceedings of the Sixth International Conference on Machine Learning and Applications, Workshop on Machine Learning in Biomedicine and Bioinformatics*, 2007.

[8] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1999.

[9] R. D. Finn, J. Mistry, B. Schuster-Bckler, et al. Pfam: clans, web tools and services. *Nucleic Acids Research*, 34(Database Issue):D247–D251, 2006.

[10] M. Gerstein. Measurement of the effectiveness of transitive sequence comparison, through a third 'intermediate' sequence. *Bioinformatics*, 14(8):707–714, 1998.

[11] B. Goethals and M. J. Zaki, editors. *Workshop on Frequent Itemset Mining Implementations*, volume 90. CEUR-WS.org, 2003.

[12] J. Gough, K. Karplus, R. Hughey, and C. Chothia. Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. *Journal of Molecular Biology*, 313(4):903–919, 2001.

[13] J. Gouzy, F. Corpet, and D. Kahn. Whole genome protein domain analysis using a new method for domain clustering. *Computers & Chemistry*, 23:333–340, 1999.

[14] G. Z. Hertz and G. D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7/8):563–577, 1999.

[15] K. L. Jensen, M. P. Styczynski, I. Rigoutsos, and G. N. Stephanopoulos. A generic motif discovery algorithm for sequential data. *Bioinformatics*, 22(1):21–28, 2006.

[16] C. E. Lawrence, S. F. Altschul, M. S. Boguski, et al. Detecting subtle sequence signals: A gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, 1993.

[17] I. Letunic, R. R. Copley, B. Pils, et al. SMART 5: domains in the context of genomes and networks. *Nucleic Acids Research*, 34(Database issue):D257–D260, 2006.

[18] H. Mi, N. Guo, A. Kejariwal, and P. D. Thomas. PANTHER version 6: protein sequence and function evolution data with expanded representation of biological pathways. *Nucleic Acids Research*, 35(Database issue):D247–D252, 2007.

[19] N. J. Mulder, R. Apweiler, T. K. Attwood, et al. New developments in the InterPro database. *Nucleic Acids Research*, 35(Database issue):D224–D228, 2007.

[20] E. Quevillon, V. Silventoinen, S. Pillai, et al. InterProScan: protein domains identifier. *Nucleic Acids Research*, 33(Web Server Issue):W116–W120, 2005.

[21] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics*, 14(1):55–67, 1998.

[22] O. Sasson, N. Kaplan, and M. Linial. Functional annotation prediction: All for one and one for all. *Protein Science*, 15(6):1557–1562, 2006.

[23] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

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

[25] T. Tao, C. Zhai, X. Lu, and H. Fang. A study of statistical methods for function prediction of protein motifs. *Applied Bioinformatics*, 3(2–3):115–124, 2004.

[26] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191–211, 1992.

[27] C. Yeats, M. Maibaum, R. Marsden, et al. Gene3D: modelling protein structure, function and evolution. *Nucleic Acids Research*, 34(Database issue):D281–D284, 2006.

[28] Y. Zhang and M. J. Zaki. ExMotif: Efficient structured motif extraction. *Algorithms for Molecular Biology*, 1(21), 2006.

[29] Y. Zhang and M. J. Zaki. SMOTIF: efficient structured pattern and profile motif search. *Algorithms for Molecular Biology*, 1(22), 2006.

[30] J. Zhu and G. Grahne. Reducing the main memory consumptions of FPmax* and FPclose. In *Proceedings of the Workshop on Frequent Itemset Mining Implementations*, volume 126. CEUR-WS.org, 2004.