Sequence analysis

Significant speedup of database searches with HMMs by search space reduction with PSSM family models

Michael Beckstette1,*,†, Robert Homann2,3,†, Robert Giegerich3 and Stefan Kurtz1

1Center for Bioinformatics, University of Hamburg, Bundesstrasse 43, 20146 Hamburg, ‡International NFR Graduate School in Bioinformatics and Genome Research, Center for Biotechnology (CeBiTec), Bielefeld University, 33594 Bielefeld and 3Group for Practical Computer Science, Technische Fakultät, Bielefeld University, Postfach 100 131, 33501 Bielefeld, Germany

Received on June 10, 2009; revised on October 2, 2009; accepted on October 9, 2009

Advance Access publication October 14, 2009

Associate Editor: Burkhard Rost

ABSTRACT

Motivation: Profile hidden Markov models (pHMMs) are currently the most popular modeling concept for protein families. They provide sensitive family descriptors, and sequence database searching with pHMMs has become a standard task in today’s genome annotation pipelines. On the downside, searching with pHMMs is computationally expensive.

Results: We propose a new method for efficient protein family classification and for speeding up database searches with pHMMs as is necessary for large-scale analysis scenarios. We employ simpler models of protein families called position-specific scoring matrices family models (PSSM-FMs). For fast database search, we combine full-text indexing, efficient exact p-value computation of PSSM match scores and fast fragment chaining. The resulting method is well suited to prefilter the set of sequences to be searched for subsequent database searches with pHMMs. We achieved a classification performance only marginally inferior to hmmssearch, yet, results could be obtained in a fraction of runtime with a speedup of >64-fold. In experiments addressing the method’s ability to prefilter the sequence space for subsequent database searches with pHMMs, our method reduces the number of sequences to be searched with hmmssearch to only 0.80% of all sequences. The filter is very fast and leads to a total speedup of factor 43 over the unfiltered search, while retaining >99.5% of the original results. In a lossless filter setup for hmmssearch on UniProtKB/Swiss-Prot, we observed a speedup of factor 92.

Availability: The presented algorithms are implemented in the program PoSSuMsearch2, available for download at http://bibiserv.techfak.uni-bielefeld.de/possumsearch2/.

Contact: beckstette@zhb.uni-hamburg.de

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

Profile hidden Markov models (pHMMs) are currently the most popular modeling concept for protein families. They provide very sensitive family descriptors, and sequence database searching with models from major pHMM collections (Finn et al., 2006; Haft et al., 2003) has become a standard task in sequence analysis. On the downside, database searching with pHMMs with well-known programs like hmmssearch or hmmpfam (Eddy, 1998) is computationally expensive. In particular, the long running times of pHMM-based methods and the time scaling behavior, which is linear in the length of the searched sequence, make them more and more demanding in today’s sequence database search scenarios. This problem will become even more severe as the continuing exponential growth of sequence databases will certainly be amplified by the increasing dispersal of next-generation sequencing technologies (Shendure and Ji, 2008). Nevertheless, pHMM-based database searches are indispensable for today’s genome annotation pipelines. For instance, the majority of member databases of the InterPro classification system (Hunter et al., 2009), a widely used system for protein annotation purposes, employ family information in form of pHMMs. The applied classification procedure InterProScan (Quevillon et al., 2005) includes searches with all pHMMs from the Pfam database. Some authors (Finn et al., 2006), TIGRFAM (Haft et al., 2003), Superfamily (Gough et al., 2001), PIRSF (Wu et al., 2004), Gene3D (Yeats et al., 2006), Smart (Letunic et al., 2006) and Panther (Mi et al., 2005) databases. These pHMM-based database searches render InterProScan a very compute intensive application whose employment on a large scale is challenging even on the largest cluster systems.

To solve this dilemma, much effort has been spent on improving the running time of pHMM-based database search tools. Some approaches for improvement use parallelism techniques and/or fast, extended, CPU-specific instructions sets, like SSE/SSE2 (Streaming Single Instruction/Multiple Data Extensions) (Walters et al., 2006). Hardware solutions implementing proprietary variants of hmmssearch on special field-programmable gate array (FPGA) boards are also available. Moreover, the application of machine learning techniques has been suggested (Lingner and Meinicke, 2008a, b). Very recently, Sun and Buhler (2009) described the design of patterns and profiles for speeding up hmmssearch using unordered sets of motifs in form of PROSITE-like patterns or position-specific scoring matrices (PSSMs) derived from a multiple alignment of a protein family. These motifs are then searched with standard regular expression matching and profile searching.
algorithms, respectively, to prefilter the search space for subsequent application of \textit{amphora}. The reported speedups over unfiltered search are in the range of 20-fold with almost 100% sensitivity and 30- to 40-fold with 90% sensitivity.

We propose a new software-based method well suited: (i) for efficient and reliable protein family classification, and (ii) to speedup database searches with \textit{amphora}. Our approach employs a simpler model of protein families based on PSSMs in combination with exact p-value computation using lazy evaluation and full text indexing with enhanced suffix arrays (Abouelhoda et al., 2004) to filter the search space for subsequent database search queries with pHMMS corresponding to these families. The work is an extension of our PSSM search tool \textit{PosStaMearch} (Beckstette et al., 2006), so we briefly describe previous work on index-based PSSM matching and efficient p-value computation for PSSM matchescores (Sections 2.2 and 2.3) before describing the new concepts and algorithms used in the new version of \textit{PosStaMearch} (Sections 2.4–2.6), herein after referred to as \textit{PosStaMearch}2.

2 METHODS

2.1 Preliminaries

Let \( S \) be a sequence of length \( n \) over finite alphabet \( \mathcal{A} \), and let \( S[i..j] \), \( 0 \leq i \leq j \leq n-1 \), denote the substring of \( S \) of length \( j-i+1 \) starting at position \( i \) and ending at \( j \) (including \( i \) and \( j \) as included). Let \( S \) be a symbol in \( \mathcal{A} \), larger than all other symbols, which does not occur in \( S \). The suffix array \( SA[S] \) of \( S \) is a table in the range 0–\( n-1 \) that lists the starting positions of all \( n \) suffixes of \( S \) in lexicographical order (symbols \( S \) must be appended to \( S \) to obtain a well-defined order on suffixes). That is, \( SA[S][0] \leq SA[S][1] \leq \ldots \leq SA[S][n-1] \). As shown in Beckstette et al. (2006) by processing the suffix array \( SA[S] \) of \( S \), one simulates a depth first traversal of the suffix tree (cf. Abouelhoda et al., 2004) by processing the suffix array \( SA[S] \) of \( S \) and \( \text{leps} \) from left to right. To incorporate lookahead scoring, the search skips over certain ranges of suffixes \( SA[S] \) using the information stored in table \( \text{leps} \). Algorithmic details are given in Beckstette et al. (2006).

The practical speedup of \textit{ESAsearch} over methods that operate on the plain text is influenced by the choice of threshold \( \theta \). The larger the value of \( \theta \), the more likely it is to fall short of an intermediate threshold \( \theta' \) on average. This in turn means that the computation of the scores can be stopped earlier and more suffixes can be skipped by utilizing the information stored in tables \( \text{leps} \) and \( \text{shp} \). As shown in Beckstette et al. (2006), the expected runtime of \textit{ESAsearch} is sublinear in the text length, whereas its worst case runtime is \( O(n \cdot m) \) under the special condition that \( n = |\mathcal{A}|^{m-1} \) holds, independent of the chosen threshold \( \theta \). The high speed of \textit{ESAsearch} is the foundation for the speedup of database searches with pHMMS described in the sequel.

2.2 Fast database searching with single PSSMs

2.2.1 Algorithms for finding PSSM matches A naive \( O(mn) \) time algorithm solving the PSSM matching problem moves a sliding window of size \( m \) over the text to be searched of length \( n \) and is implemented in many programs facilitating PSSMs (Henikoff et al., 2000; Kel et al., 2003; Quandt et al., 1995; Scordis et al., 1999). Considerable practical speedups can be obtained with the lookahead search technique of Wu et al. (2000). It uses the implication \( \text{psuc}[w,M] < \theta = \text{sc}[w,M] < \theta \) as an early stop criterion for the calculation of \( \text{sc}[w,M] \). However, lookahead scoring does not improve the theoretical worst case time complexity of the naive algorithm.

2.2.2 Index-based searching with PSSMs

For fast database searching with PSSMs, \textit{PosStaMearch}2 employs the algorithm \textit{ESAsearch} (Beckstette et al., 2006), which in turn makes use of enhanced suffix arrays. To use enhanced suffix arrays for fast database searching with PSSMs, one simulates a depth first traversal of the suffix tree (cf. Abouelhoda et al., 2004) by processing the suffix array \( SA[S] \) of \( S \) and \( \text{leps} \) from left to right. To incorporate lookahead scoring, the search skips over certain ranges of suffixes \( SA[S] \) using the information stored in table \( \text{leps} \). Algorithmic details are given in Beckstette et al. (2006).

| \( i \) | \( \text{leps} \) | \( \text{shp} \) | \( \text{sc}[w,M] \) |
|----|----|----|----|
| 0  | 6  | 0  | 11 |
| 1  | 8  | 2  | 15 |
| 2  | 3  | 4  | 15 |
| 3  | 7  | 1  | 17 |
| 4  | 7  | 2  | 16 |
| 5  | 4  | 11 | 12 |
| 6  | 4  | 11 | 12 |
| 7  | 4  | 11 | 12 |
| 8  | 10 | 0  | 11 |

Fig. 1. Enhanced suffix array for \( S = \text{cctacatcaacct} \), consisting of the suffix array \(\text{sa} \) and additional tables \(\text{leps} \) and \(\text{shp} \). The suffixes of \( S \) are sorted lexicographically (rightmost column).
An optimal chain of collinear non-overlapping matches is determined, by these alignment blocks are excised from a multiple alignment and the regions of a multiple alignment. Since \( L \) is a sequence of matches all from \( \{A_i\} \), \( PSSM \) can be computed from the blocks by several well-known algorithms like \((\text{Gribkovs et al., 1987}, \text{Hemsloff and Hemsloff, 1996}; \text{Tarunov et al., 1994})\). A match to \( M \) is a non-overlapping sequence of matches for some or all of the \( PSSM \) in \( M \) in their specified order. We will now make this more precise.

Consider a \( PSSM-FM \), \( M \) with total order \( < \). Let \( M \) be the set of all matches for all \( M \in M \) in sequence \( S \) of length \( n \). A match is represented by a triple \((M, p, s)\) such that \( M \) matches at position \( p \) in \( S \) and \( s > 0 \) is the corresponding match score. We say that matches \((M, p, s)\) and \((M', p', s')\) are collinear, written as \((M, p, s) \bowtie (M', p', s')\), if \( M, M' \in M \) and \( p = p' \) or \( p > p' \). A chain \( C \) for family model \( M \) is a sequence of matches

\[
C = (M_1, p_1, s_1), (M_2, p_2, s_2), \ldots, (M_k, p_k, s_k),
\]

all from \( M \), such that \((M_i, p_i, s_i) \bowtie (M_{i+1}, p_{i+1}, s_{i+1})\) for all \( 1 \leq i < k \). To incorporate a measure of match quality into \( PSSM-FM \) we associate with \((M, p, s)\) a \( p \)-value \( \pi(M, s) \) and a weight \( w(M, s) \) defined by

\[
w(M, s) = -\ln \left( 1 - (1 - \pi(M, s))^{y^{m+1}} \right).
\]

The chain score of a chain \( C \) is defined by

\[
\text{cs}(C) = \sum_{i} w(M_i, s_i).
\]

The motivation for Equation (1) is as follows. \( \pi = \pi(M, s) \) is the probability for the event that \( M \) matches a random string of length \( m = |M| \) for score threshold \( s \) by chance, i.e. \( \pi = \frac{1}{\text{pos}(M, s) + 1} \). Thus, \( 1 - \pi \) is the probability for the complementary event that \( M \) does not match a random string of length \( m \). The \( y^{m+1} \) is the probability that the event that there is at least one in \( n - m + 1 \) random strings that matches \( M \) with a score at least \( s \). We take the logarithm to obtain additive weights and divide by \( \ln(n) \) to account for sequence length.

The smaller the \( p \)-values of the matches in a chain (i.e. the more significant the matches of single \( PSSM \)s \( M \) are), the larger the weights get, and hence the overall chain score. Consequently, chains that consist of a number of significant matches are assigned larger chain scores than those with fewer, or many less significant matches. Equation (2) implicitly assumes independence of random strings, which is certainly an invalid assumption in our case as the ‘random strings’ are overlapping substrings of some longer sequence. Yet, our experiments confirm our chain scoring to work well in practice; it is significantly better than a more straightforward strategy that simply computes the product of raw \( p \)-values, i.e. one that sets \( w(M, s) = -\ln(\pi(M, s)) \) (see Fig. 2 in the Supplementary Material).

### 2.5 A specialized and improved PSSM chaining method

Thus far our description was based on a single sequence. However, the results described below are based on a large set of sequences \( S_1, \ldots, S_t \). To handle these, we concatenate the single sequences with separator symbols, and construct the enhanced suffix array for the concatenation. For a given \( PSSM-FM \), all \( M \in M \), \( 1 \leq i \leq L \), are matched one after the other against the enhanced suffix array. This gives match sets \( M(S_i) \) for \( PSSM \) \( M \).

The PSSM chaining problem for a single sequence \( S_j \) can be considered a chaining problem for pairwise matches between sequence \( S_j \) and a virtual sequence \( V[1..L] \) such that a match for \( PSSM \) \( M \) in a match length \( m \) at position \( i \) in \( V \). The pairwise chaining problem can be solved in \( O(b\log b) \) time using an algorithm described in Abouelhoda and Ohlebusch (2005), where \( b = |M(S_j)| \) and \( M(S_j) \) is the set of \( PSSM \) matches in \( S_j \). This algorithm is called the general chaining algorithm. For the special case of the PSSM chaining problem, we have specialized and improved the general chaining algorithm to obtain a method with the following advantages:

- While the general chaining algorithm requires a dictionary data structure with insert, delete, predecessor and successor operations running in logarithmic time (e.g. an AVL-tree or a red-black tree), our approach only needs a linear list, which is much easier to implement and requires less space.
- While the general chaining algorithm requires an initial sorting step using \( O(b^2 \log b) \) time, our method only needs \( O(b + \sum_{i=1}^{n} \ln b_i) \) time for this step. Here, \( b_i \) is the total size of all sets \( M(S_i) \) and \( b = |M(S_1)| \) is the set of all \( PSSM \) matches of \( PSSM \) \( M \) in sequence \( S_j \).
- While the general chaining algorithm solves the chaining problem for \( M(S_j) \) in \( O(b \log b) \) time, our method runs in \( O(b \log b) \) time. If \( L \) is considered to be a constant, the running time becomes linear in \( b \).

The details of the improved chaining method are described in the Supplementary Material.

### 2.6 Using PSSM-FMs for sequence classification

To employ \( PSSM-FM \)s for protein family classification, we combine the three algorithms sketched in Sections 2.2–2.5. Namely (i) \( ESAsearch \) software. Building on these techniques for fast searching of single \( PSSM \)s, we now proceed to their generalization to \( PSSM \) family models (\( PSSM-FM \)s).
fast searching with single PSSMs; (ii) LazyDistrib for exact and efficient p-value computation; and (iii) chaining of single PSSM matches in the form of the chaining method sketched in Section 2.5. All three algorithms are implemented in PoSSuMsearch2 and, in combination provide, an efficient solution to the problem of protein family classification. In the first phase, PoSSuMsearch2 computes single PSSM matches for the PSSMs of a family model using algorithm ESsearch. In the second phase, PSSM matches obtained in phase one and their ordering information are used to compute optimal chains of PSSM matches according to the order given in the family model.

When classifying an unknown protein sequence into a known family, a sequence is searched with several PSSM-FMs, representing different protein families. The classification into a certain family should be based on the quality of the best match to the corresponding family model. Hence, the objective is to determine the best chain \( C_{M,S} \) of PSSM matches in a sequence \( S \) for a given family model \( M \) and their chain score \( \text{csc}(C_{M,S}) \).

\[
\text{csc}(C_{M,S}) := \text{csc}(C_{M,S}) \quad \text{with} \quad \text{csc}(C_{M,S}) = \max \{ \text{csc}(C_{M,S}) \mid C_{M,S} \text{ is a chain for } M \text{ on } S \}. \tag{3}
\]

We call such a chain an optimal chain. With the definition of optimal chains and their chain scores, we introduce a quantifiable, rankable criterion of match quality to our PSSM-FM concept, making it applicable for sequence classification. More precisely, let \( S \) be a sequence and \( F = \{ M_1, M_2, \ldots, M_T \} \) be a collection of \( T \) PSSM-FMs, representing \( T \) distinct protein families. Further, let \( \text{csc}_{F,S} := \max \{ \text{csc}(C_{M,S}) \mid C_{M,S} \text{ is a chain for } M \text{ on } S \} \) be the maximal score of all optimal chains in \( S \) over all family models in \( F \). We classify \( S \) into the family represented by \( M \in F \) if and only if \( \text{csc}_{F,S} = \text{csc}(C_{M,S}) \). That is, we classify the sequence under consideration into the family whose family model generates the highest scoring optimal chain. In practice, it is often useful to employ a threshold constraint, like a minimal necessary chain length, as a lower boundary for classification. That is, sequences not satisfying this constraint are not classified.

PoSSuMsearch2 can be used in two modes of operation:

- Mode modsearch allows sequence classification based on a typically small, library of PSSM-FMs. For each sequence the best matching chains for (up to) \( k \) different family models are reported.
- Mode seqclass allows sequence classification based on a typically large, library of PSSM-FMs. For each model, the best matching chains in (up to) \( k \) different sequences are reported.

Hence, mode modsearch mimics the modes operandi of program hmmsearch, whereas mode seqclass is comparable with program hmmpfam.

3 RESULTS

3.1 PSSM-FMs for protein classification

Detection of protein families in large databases is one of the principal research objectives in structural and functional genomics. To evaluate the suitability of PoSSuMsearch2 employing PSSM-FMs for fast and accurate protein family classification, we rigorously tested and validated our method using the evaluation system Phase4 (Rehmsmeier, 2002). We evaluated the sensitivity and specificity, addressing different database search scenarios at different levels of difficulty. That is, we measured our method’s ability to detect (A) very close, (B) close and (C) distant relationships.

Table 1. Evaluation scenarios and number of models used in the experiments to assess method sensitivity and specificity

| Scenario (#models) | Description |
|--------------------|-------------|
| (A) Very close relationship (561) | For each superfamily: for each family, half of its sequences are chosen as test sequences, and the remaining ones are chosen as training sequences. The sequences of the surrounding superfamily are ignored in the evaluation. |
| (B) Close relationship (474) | For each superfamily, half of the sequences of each of its families are chosen as training sequences and the remaining ones are chosen as test sequences. From a superfamily, each family in turn is chosen to provide the test sequences. The remaining families within that superfamily provide the training sequences. |
| (C) Distant relationship (1221) | |

![Fig. 3. Construction of training and test sets for (A) very close, (B) close and (C) distant relationships.](image-url)

Table 1 and Figure 3 give more details on the three scenarios. The three scenarios used for our evaluations differ in how training and test sets are constructed from the data. Table 1 and Figure 3 give more details on the three scenarios. The task of the searching program in each case is to find, preferably, only protein sequences from the test sets in the whole SCOP database, while only providing the corresponding training sets to the searching program. That is, a perfect searching method would always find exactly the set of true positives, which is the test set.
Significant speedup of database searches with HMMs

3.1.3 Comparison of runtime and scalability. All benchmark experiments described in this article were performed on a single Intel Xeon CPU running at 2.3 GHz. For runtime experiments, we took the first 100 protein families from the Pfam Rel. 23.0 database (Finn et al., 2008), and computed PSSM-FMs from the Pfam seed alignments by excising alignment blocks as described above, but of width 5–8. This resulted in 100 models, consisting of 20%96 individual PSSMs. From the same alignments, we generated 100 calibrated HMMs using hmmbuild/hmmpcalibrate. We searched with these family descriptors in the UniProtKB/Swiss-Prot Rel. 57.5 database (Wu et al., 2006), containing 470,369 protein sequences in 167 MB. It took PosSMMsearch2 ~28.1 min to find all matches to the PSSM-FMs using a p-value threshold of 10−5 for each PSSM. For hmmssearch, we chose an E-value of 10−1 in order to find roughly the same set of matches. It took hmmssearch ~30 h to find matches to the pHMMs. This makes for a speedup factor of more than 64.8 for PosSMMsearch2 over hmmssearch. Along with these results, PosSMMsearch2 clearly showed sublinear time scaling when applied to subsets of UniProtKB/Swiss-Prot of different sizes, whereas hmmssearch showed linear scaling behavior due to the underlying Viterbi algorithm. For the results of this experiment, see Figure 3 in the Supplementary Material.

3.2 Acceleration of pHMM-based database searches

Here, we evaluate the performance of PosSMMsearch2 when it is used as a filter to reduce the search space for hmmssearch. The combination of PosSMMsearch2 and hmmssearch is called PSPosSMMSearch. The intention is to speedup the database search while keeping the sensitivity of hmmssearch.

As a prerequisite for reliable filtering, we have to calibrate p-value cutoffs for the PSSM-FMs such that they match the corresponding pHMMs trusted cutoff (tc) and noise cutoff (nc). That is, our calibrated PSSM-FMs operate on the same level of sensitivity as hmmssearch employing the pHMM, but with possibly reduced specificity. Hence, the determination of a proper family-specific p-value cutoff is crucial for the sensitivity as well as overall speedup of PSPosSMMSearch. A too stringent cutoff may reduce the search space too much and thus may have a negative effect on the sensitivity. On the other hand, a too relaxed cutoff may not sufficiently reduce the search space and lead to long running times. In the following, we evaluate two different strategies for cutoff calibration: cutoff calibration for lossless filtering and cutoff calibration based on training- and test-set separation.

3.2.1 Cutoff calibration for lossless filtering. We start by searching with a pHMM representing a protein family in a large protein database like UniProtKB/Swiss-Prot using hmmsearch with the model’s trusted and noise cutoffs, respectively, and tabulate all matching sequences. From the seed alignment of the employed pHMM, we construct a PSSM-FM with a block width of 6–12 and use this family model to iteratively search UniProtKB/Swiss-Prot using PosSMMsearch2. In each iteration, we relax the p-value cutoff until we find all the sequences also detected by hmmssearch using the model’s trusted and noise cutoffs respectively. With this procedure, we determine the p-value cutoffs denoted by πtc and πnc corresponding to the pHMM’s trusted and noise cutoffs in terms of sensitivity. Observe that the set of matching sequences detected by PosSMMsearch2 using cutoff πtc or πnc may be a superset of the set of sequences detected by hmmssearch employing the pHMM’s trusted and noise cutoffs. However, since we are interested in using PSSM-FMs searched with PosSMMsearch2 as a prefilter for search space reduction for hmmssearch, sensitivity is more important than specificity. Once πtc and πnc are computed on a large protein database, they are, together with the PSSM-FM, stored on file. That is, for further searches with hmmssearch using the model’s trusted or noise cutoffs, we can use PosSMMsearch2 using cutoff πtc or πnc as a filter and apply the compute intensive hmmssearch only on sequences that contain chains matching to the PSSM-FM. Sequences that contain no such chains are filtered out. Since sequences containing sufficiently long chains constitute only a small fraction of all sequences to be searched and since PosSMMsearch2 is much faster than...
We calibrated the Tables 2 and 3 for more detailed results for the first 20 matched by the pHMM. We employed these models and cutoffs in a database, training sets that consist of 20% using the pHMMs' trusted and noise cutoffs (for families' seed alignments with the procedure described in Section 3.1. TIGRFAM the first 200 families listed in For training- and test-set separation. how well the calibrated search with pHMMs and PSSM-FMs representing the experiment we searched with pHMMs and PSSM-FMs for the first 20 trusted as well as noise cutoffs with the iterative procedure described above. We measured the search space reduction (see Supplementary Fig. 4 for results for the first 20 TIGRFAM families) and the total runtimes of PfamSearch and compared them with hmmsearch operating on the unfiltered dataset. PSSM-FM-based filtering reduces the search space and hence the overall runtime considerably. For example, for family TIGR00001 only five sequences remain after the filtering step and are handed over to hmmsearch. Filtering with p-value cutoffs corresponding to the less-stringent noise cutoffs reduced, in the worst case (TIGR00001), the search space by ~80%. For all 200 tested families, the overall runtime is reduced from 4233 (4234) to only 46 (117) min when using trusted (noise) cutoffs. This is a speedup of factor 92 (36).

We emphasize that in this experimental setup, PfamSearch and direct hmmsearch obtain exactly the same results on the full sequence set. Hence, PSSM families’ seed alignments with the procedure described in Section 3.1. TIGRFAM family when searching with PSSM-FMs (pHMMs) representing the first 200 TIGRFAM families on UniProtKB/TREMBL Rel. 40.3. Different values of k represent different training set sizes. For further details see text.

3.2.2 Cutoff calibration based on training- and test-set separation. For the first 200 families listed in TIGRFAM, we built PSSM-FMs from the families’ seed alignments with the procedure described in Section 3.1. We calibrated the p-value cutoffs and minimal chain lengths to match all sequences of training sets of different sizes. Training sets contain every k-th sequence returned by direct hmmsearch on UniProtKB/TREMBL Rel. 40.3 TIGR00011 only five sequences searched afterwards employing these thresholds. This raises the question, since thresholds were trained/adjusted on the same set of sequences as were works as a perfect, lossless filter. This is not too surprising, obtain exactly the same results on the full sequence set. Hence, hmmsearch min when using trusted (noise) cutoffs. This is a speedup of factor 92 (36).

In an additional experiment we searched with pHMMs and PSSM-FMs representing the first 200 protein families in the TIGRFAM database in 26 publicly available Escherichia coli proteomes (see Supplementary Table 4 for further computational costs to reduce the search space for the subsequent application of the hmmsearch engine. We applied HMMERHEAD to the same experimental setup as described in the former paragraph. That is, we

3.2.3 Whole proteome annotation using PfamSearch. In an additional experiment we searched with pHMMs and PSSM-FMs representing the first 500 protein families in the TIGRFAM database and measured the overall runtime and true positives coverage per family and compared the running time with the time needed by direct hmmsearch using trusted cutoffs. See Figure 5 and Supplementary Table 1 for the results of this experiment.

PfamSearch returned >99.5% of the original results determined by hmmsearch, including their E-values and scores, when using half of the sequences matched by hmmsearch as training sets. Of 150 851, 523 matches (0.34%) were missed. With p-value cutoffs calibrated to match the sensitivity level of hmmsearch using noise cutoffs, PfamSearch detected 99.4%, while missing 649 out of 140 263 sequences. See Figure 6 and Supplementary Tables 2 and 3 for more detailed results for the first 20 TIGRFAM families. It took PfamSearch ~24 h to search with the first 200 TIGRFAM families, compared with ~45 days for direct hmmsearch using the models’ trusted cutoffs. That is, PfamSearch achieves a speedup of factor 43.8 over direct hmmsearch, while retaining >99.5% of the original results. In this experiment, the set of sequences to be searched with hmmsearch was reduced to only 0.80% of all sequences. Using the less-stringent noise cutoffs, PfamSearch reduces the search space to only 3.83% of the original search space size also with a sensitivity of 99.5% (see Supplementary Table 3 for more detailed results for the first 20 TIGRFAM families) and a speedup of factor ~14 over direct hmmsearch.

3.2.4 Comparison of PfamSearch with other hmmsearch acceleration solutions. Another approach to accelerate hmmsearch is the HMMERHEAD program (Poster presentation at RECMB2007). HMMERHEAD uses a filtration approach that employs four filtering stages with increasing computational costs to reduce the search space for the subsequent application of the hmmsearch engine. We applied HMMERHEAD to the same experimental setup as described in the former paragraph. That is, we
searched with pHMMs representing the first 200 TIGRFAM families on the complete UniProtKB/TremBL database and measured the running time of HMMERHEAD and the coverage using the models’ trusted (noise) cutoffs. In this experiment, HMMERHEAD was able to reach the running time compared with direct hmmsearch from 3088.05 h to 626.08 h, while retaining 100% of the original results (for details, see Supplementary Fig. 5). This makes for a speedup of factor ~1.7, with per model speedups in the range of 4·1.9.

*psblast* (Marchler-Bauer and Bryant, 2004) may be seen as an alternative to hmmsearch employing pHMMs. It uses psi-blast’s (Altschul et al., 1997) checkpoint files which can be seen as models for protein families, much like pHMMs and our PSSM-FMs. *psblast*-compatible models representing TIGRFAM protein families are part of the CDD database (Marchler-Bauer et al., 2009). To test the ability of *psblast* to obtain the same results as hmmsearch we offer an alternative to hmmsearch and PSfamSearch respectively, we compared the classification performance of *psblast* with that of PSfamSearch employing PSSM-FMs for the first 200 TIGRFAM families with *p*-value cutoffs \( \pi_0 \) and \( \pi_1 \), respectively. As a state of truth we use the results returned by hmmsearch using trusted (noise) cutoffs. In this experiment, *psblast* achieved an averaged coverage in the range of 85.6–98.6% (84.7–95.5%) compared with hmmsearch using trusted (noise) cutoffs. Using the same experimental setup, PSfamSearch achieved a uniform coverage of 99.54% (99.47%) when using cutoffs \( \pi_0 \) (\( \pi_1 \)). See Supplementary Figure 6 for further details on this experiment. It took PSfamSearch 1490 (4476) min using cutoffs \( \pi_0 \) (\( \pi_1 \)) to perform this task (Supplementary Table 1), while *psblast* needed only 1341.96 min. Hence, by application of *psblast* one obtains an additional speedup of factor 1.1 (3.4) at the price of reduced sensitivity.

Recently, Lingner and Meinicke (2009a) described an approach for search space reduction applicable to speedup database searches with pHMMs based on machine learning techniques. Although the described method and presented results seem to be promising, up to now only a prototype implemented in MATLAB and a web server for interactive usage (Meinicke, 2009) are available.

### 4 DISCUSSION AND CONCLUDING REMARKS

We have presented a new database search method based on PSSM-FMs. It is well suited for fast and reliable protein family classification. Moreover, it can serve as a filter to considerably speedup database searches with pHMMs, while retaining almost 100% sensitivity. Our method combines fast index-based searching of PSSMs, an efficient algorithm for exact \( p \)-value computation for PSSM score thresholds, and a fast fragment chaining algorithm. The methods described here are implemented in the robust and well-documented software tool PSfamSearch2.

We carefully evaluated the performance of PSSM-FMs in terms of sensitivity and specificity by using *PosStaSearch2* in two different modes of operation, i.e. for direct sequence classification, and as a prefilter for *hmmsearch*. We have shown that PSSM-FMs are only marginally inferior to pHMMs when used for sequence classification. The FPP50 value (the average coverage achieved when tolerating 50 false positives) for PSSM-FMs never dropped below the FPP50 value for pHMMs by more than ~6 percentage points in all of our three evaluation scenarios (Fig. 4). This renders PSSM-FMs a fast alternative to pHMMs: for example, we have observed that *PosStaSearch2* is more than 64 times faster than *hmmsearch* for the same classification task.

We also demonstrated that PSSM-FMs are well suited for prefiltering sequence databases to be searched by *hmmsearch*. Using PSfamSearch (the combination of *PosStaSearch2* and *hmmsearch*), we observed dramatic search space reductions for UniProtKB/TremBL to 0.80% and 3.83%, respectively, when filtering with 200 PSSM-FMs built from the TIGRFAM database using the pHMMs’ trusted and noise cutoffs, respectively. The reduction of the sequence database resulted in speedups of ~43.8 and 14 over original, unfiltered *hmmsearch*, respectively, while retaining 99.5% of the original results in both cases. Extrapolated to all 3603 families in TIGRFAM (Ref 8.0), we estimate a runtime of ~18.6 days for PSfamSearch, and 2.23 years for direct *hmmsearch* using the models’ trusted cutoffs. Notably, the observed speedups come from an algorithmic as well as a conceptual advancement: the speed of index-based PSSM searching, and the astonishing fact that pHMMs can be approximated well by the much simpler PSSM-FMs. This is also consistent with the finding that protein classification works well with word correlation matrices (Lingner and Meinicke, 2008b).

In our experiments, PSfamSearch also showed a >25-fold speedup over the program HMMERHEAD. Compared with the well-known *psblast* tool, PSfamSearch is slightly slower. PSfamSearch, however, achieved a significantly higher sensitivity in our experiment. In the experiment showing the ability of PSfamSearch for efficient annotation of E.coli proteomes, PSfamSearch returned >99.99% of the original *hmmsearch* results and showed a speedup over direct *hmmsearch* of ~30.

PSfamSearch is twice as fast as the previously fastest software-based acceleration method for *hmmsearch* (Sun and Buhler, 2009). Note that Sun and Buhler (2009) focus on the problem of designing unordered sets of motifs with good filtering characteristics while searching them with straightforward algorithms, whereas our work focuses on efficient index-based searching in sublinear expected time while keeping the derivation of motifs rather simple. This raises the question whether a future combination of both approaches might lead to further improvements in software-based pHMM database search methods.

### ACKNOWLEDGEMENTS

The authors thank Andrew Torda and four anonymous reviewers for comments on the manuscript.

*Conflict of Interest*: none declared.

### REFERENCES

Abouelhoda,M. and et al. (2005) Chaining algorithms for multiple genome comparison. *J. Discrete Algorithms*, 3, 321–341.

Abouelhoda,M. et al. (2006) Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorithms*, 2, 53–86.

Altschul,S. et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25, D483–D487.

Beckstette,M. et al. (2006) Fast index based algorithms and software for matching position specific scoring matrices. *BMC Bioinformatics*, 7, Article 289.

Bradley,P. (1994) Profile hidden Markov models. *Bioinformatics*, 10, 755–763.

Finn,R. et al. (2008) Pfam: clans, web tools, and services. *Nucleic Acids Res.*, 36, D247–D251.

Finn,R. et al. (2008) The Pfam protein families database. *Nucleic Acids Res.*, 36, D281–D288.

Gough,J. et al. (2001) Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure. *J. Mol. Biol.*, 313, 903–919.

Gribskov,M. et al. (1987) Profile analysis: detection of distantly related proteins. *Proc. Natl Acad. Sci. USA*, 84, 4355–4358.
Marchler-Bauer, A. and Bryant, S. (2004) CD-Search: protein domain annotations on the

Lingner, T. and Meinicke, P. (2008b) Word correlation matrices for protein sequence

Lingner, T. and Meinicke, P. (2008a) Fast target set reduction for large-scale protein

Kasai, T. et al. (2006) SMART 5: domains in the context of genomes and networks.

Henikoff, S. et al. (1994) Using substitution probabilities to improve position-specific

Letunic, I. et al. (2006) SMART 5: domains in the context of genomes and networks.

Kärkkäinen, J. and Sanders, P. (2003) Simple linear work suffix array construction. In

Hunter, S. et al. (2009) InterPro: the integrative protein signature database. Nucleic Acids Res., 37, D281–D284.

Mi, H. et al. (2005) The PANTHER database of protein families, subfamilies, functions

Quandt, K. et al. (1995) MatProf and MatInspector: new fast and versatile tools for
detection of consensus matches in nucleotide data. Nucleic Acids Res., 23, 4876–4884.

Quevillon, E. et al. (2005) InterProScan: protein domains identifier. Nucleic Acids Res., 33, W116–W120.

Rahmann, S. (2005) Dynamic programming algorithms for two statistical problems in

computational biology. In Benson, G. and Page, R. D. M., eds. Proceedings of the 1st Workshop on Algorithms in Bioinformatics (WABI). Vol. 2812 of Lecture Notes in Computer Science. Springer, Budapest, Hungary, pp. 151–164.

Rehm, M. et al. (2002) Automatic evaluation of database search methods. Brief. Bioinform., 3, 342–352.

Scheid, P. et al. (1999) FingerPRINTScan: intelligent searching of the PRINTS motif database. Bioinformatics, 15, 799–806.

Shendure, J. and H. H. (2008) Next-generation DNA sequencing. Nat. Biotechnol., 26, 1135–1145.

Staden, R. (1990) Searching for patterns in protein and nucleic acid sequences. In

Stoeckert, R., ed., Methods in Enzymology, Vol. 183. Academic Press, San Diego,

CA, pp. 193–211.

Sun, Y. and Buhler, J. (2009) Designing patterns and profiles for faster HMM search. IEEE/ACM Trans. Comput. Biol. Bioinform., 6, 252–263.

Tatusov, R. et al. (1994) Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks. Proc. Natl Acad. Sci. USA, 91, 12091–12095.

Toutzé, R. and Vant,J.-S. (2007) Efficient and accurate P-value computation for position

weight matrices. Algori...