BIOINFORMATICS

Vol. 15 no. 6 1999
Pages 463–470

Family pairwise search with embedded motif models

William Noble Grundy[1] and Timothy L. Bailey[2]

[1]Department of Computer Science, University of California, Santa Cruz, CA 95064 and [2]NPACI/SDSC, MC 0505, 9500 Gilman Drive, Bldg 109, La Jolla, CA 92093-0505, USA

Received on December 14, 1998; revised on March 10, 1999; accepted on March 12, 1999

Abstract

Motivation: *Statistical models of protein families, such as position-specific scoring matrices, profiles and hidden Markov models, have been used effectively to find remote homologs when given a set of known protein family members. Unfortunately, training these models typically requires a relatively large set of training sequences. Recent work (Grundy,* J. Comput. Biol., ***5****, 479–492, 1998) has shown that, when only a few family members are known, several theoretically justified statistical modeling techniques fail to provide homology detection performance on a par with Family Pairwise Search (FPS), an algorithm that combines scores from a pairwise sequence similarity algorithm such as BLAST.*

Results: *The present paper provides a model-based algorithm that improves FPS by incorporating hybrid motif-based models of the form generated by Cobbler (Henikoff and Henikoff,* Protein Sci., ***6****, 698–705, 1997). For the 73 protein families investigated here, this cobbled FPS algorithm provides better homology detection performance than either Cobbler or FPS alone. This improvement is maintained when BLAST is replaced with the full Smith–Waterman algorithm.*

Availability: *http://fps.sdsc.edu*

Contact: *bgrundy@cse.ucsc.edu*

Introduction

Science may be described as the process of building models to explain natural phenomena. Although every scientific theory implies a corresponding model, some models are less explicit than others. An explicit model with an exact interpretation is desirable, since it effectively summarizes the important features of the target phenomenon, rendering them easily explicable. In the case of protein family characterization, a statistical model with a probabilistic interpretation, in addition to being useful for tasks such as multiple alignment and homology detection, can provide biological insight into the important functional or structural features of the modeled family.

Unfortunately, the most elegant model is not always the most useful. For example, recent work (Grundy, 1998) has shown that, for small training sets, several complex and theoretically justified protein modeling techniques fail to provide homology detection performance on a par with a simple, non-model-based algorithm. The algorithm, called Family Pairwise Search (FPS), involves combining, for each sequence in the database being searched, the pairwise similarity scores of the sequences in the family of known homologs comprising the query. These similarity scores may be computed using a sequence search algorithm such as BLAST (Altschul *et al*., 1990). For small query sets, the FPS algorithm outperforms a full-sequence hidden Markov model approach (HMMER; Eddy, 1995) and a motif-based modeling approach [model construction by MEME (Bailey and Gribskov, 1998) followed by search with MAST (Bailey and Elkan, 1994)] to homology detection.

The explanations for the relatively poor performances of these model-based techniques differ. For HMMER, the difficulty lies in the large number of model parameters relative to the size of the training set. When only a few sequences are available for training, the number of parameters in the model is on the order of the total size of the training set. Consequently, even with strong prior information, training these models accurately is difficult. MEME, on the other hand, reduces the number of trainable parameters by focusing only upon the motif regions of the training set. The result is a set of relatively well-trained motif models. It is unlikely that the relatively poor performance of the MEME/MAST strategy results from a deficiency in the conserved regions identified by MEME, since a similar comparison of a motif-based and whole-sequence search method leads to similar results (Henikoff and Henikoff, 1997). Rather, MEME loses homology information by discarding the non-motif regions of the sequences (Pearson, 1997), and this loss affects MAST's search performance.

Cobbling (Henikoff and Henikoff, 1997) is a hybrid modeling scheme that addresses both of these problems. A cobbled profile model of a protein family is constructed by converting a single, representative family member (the template sequence) into a profile (Gribskov *et al*., 1990) and then replacing the motif regions with profile representations of the motifs. All gap-opening and extension penalties in the profile are set to the same values. The number of trainable parameters in the cobbled profile model is small, because

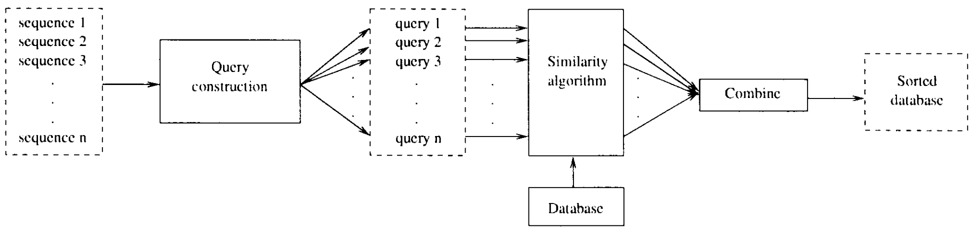

Fig. 1. The Family Pairwise Search algorithm.

models are only learned for the motif regions. The rest of the profile is constructed by simply replacing the letter in the template sequence with a column from a pairwise score matrix such as BLOSUM (Henikoff and Henikoff, 1992). Thus, the cobbled model retains useful homology information in the inter-motif regions by embedding the motif models into the profile constructed from the complete template sequence.

Cobbler thus provides a means of avoiding both of the problems that detract from the classification performance of the HMMER and MEME statistical modeling techniques. Therefore, in this paper, we extend the FPS algorithm to use cobbled profiles. We show that combining FPS with Cobbler yields a family-based homology detection algorithm with significantly better classification accuracy than either FPS or Cobbler alone.

The Family Pairwise Search algorithm

The FPS algorithm is illustrated in Figure 1. The input to the algorithm is a query set of sequences that are known to be homologous to one another, as well as a sequence database to be searched. FPS outputs a version of the database sorted in order of decreasing similarity with the query set. The algorithm proceeds in four steps. First, each sequence in the query set is converted into a query. Second, the queries are input to a similarity algorithm and compared to each sequence in the search database. Third, each sequence in the search database is assigned a similarity score by combining its scores relative to the sequences in the query set. Fourth, the search database is sorted according to the average similarity score.

Numerous variants of the FPS algorithm are possible, a number of which have been explored previously (Grundy, 1998). These variants include using the BLAST 'bit score' (Altschul *et al.*, 1997) and combining these scores by taking the average. However, the bit score is not normalized for the length of the sequence, and recent work has shown that length normalization improves the sensitivity of database searches (Pearson, 1998). It can be shown that a length-normalized bit score is proportional to the logarithm of the P value (Bailey and Grundy, 1999). Consequently, in this work, we compute the overall score of a sequence by averaging the logarithm of a statistical score (E value or P value). This works better in practice than the FPS variants examined previously (data not shown). Variants of the FPS algorithm that employ the best, rather than the average, score have also been explored (Grundy, 1998); however, this score-combination method is much more fragile in the presence of false-positive annotations and is therefore not as useful in general. In this work, therefore, we employ a version of FPS that averages the logarithm of a statistical score. For simplicity, we refer to this version simply as 'FPS'.

Here, we study the variations of the FPS algorithm outlined in Table 1. In previous work, the sequences in the query set were used directly to search the database using the BLAST algorithm. The current work improves FPS by using cobbled profiles of the sequences. We also examine variants of FPS that replace BLAST with the Profilesearch (Gribskov *et al.*, 1990) implementation of the Smith–Waterman algorithm (Smith and Waterman, 1981). For comparison, we also study searching with a representative sequence selected from the query set, as well as searching with a cobbled profile constructed using the representative sequence as a template. These last two search methods do not involve the averaging step of the FPS algorithm.

Table 1. Summary of homology detection methods investigated here. See the text for more complete descriptions

Method	Query format	Search algorithm	Combining
BLAST FPS	sequence	BLAST	Yes
Cobbled BLAST FPS	cobbled profile	BLAST	Yes
Profilesearch FPS	profile	Profilesearch	Yes
Cobbled Profilesearch FPS	cobbled profile	Profilesearch	Yes
BLAST	single sequence	BLAST	No
Cobbled BLAST	single cobbled profile	BLAST	No

To convert a sequence in the query set to a cobbled profile, we use a modified version of the Cobbler (Henikoff and Henikoff, 1997) algorithm to embed motif profiles into a profile constructed from the template sequence. Our modified version of the algorithm can output both log-odds profiles and frequency profiles. Log-odds profiles, for use with the Smith–Waterman algorithm, are built by replacing each letter in the sequence with the BLOSUM row for that letter; frequency profiles, for use with the BLAST algorithm, use the target letter frequencies corresponding to the BLOSUM row, rather than the log-odds scores. To convert a profile into a cobbled profile, motif models (built as described below) are converted either into log-odds position-specific score matrices (for log-odds profiles) or target frequency matrices (for frequency profiles) and are used to replace the appropriate positions in the profile, as in the original Cobbler algorithm. Log-odds profiles are built using BLOSUM55, whereas frequency profiles use BLOSUM62 in order to be comparable with the standard BLAST algorithm. All local gap-opening and extension penalties in the log-odds profiles are turned off, so that only the global penalties are used for scoring.

Motifs need only be discovered once for each query set. Ungapped motifs are discovered and modeled using MEME Version 2.2 (Bailey and Elkan, 1994) with the default parameter settings from the Web interface (Grundy *et al*., 1996). These defaults include empirical Dirichlet mixture priors weighted according to the megaprior heuristic (Bailey and Gribskov, 1996), a minimum motif width of 12 and a maximum of 55, and a motif model biased toward zero or one motif occurrence per sequence. A total of 10 motifs are discovered from each query set, and motif significance is judged using the majority occurrence heuristic (Grundy *et al*., 1997): motifs that do not appear in more than half of the query sequences are discarded. This heuristic excludes motifs that are specific to subfamilies of the given query set. For eight-sequence queries, the heuristic selects an average of 5.1 motifs. MEME outputs the motifs in BLOCKS (Henikoff and Henikoff, 1991) format for use as input to the modified Cobbler algorithm.

To evaluate the benefit of the averaging aspect of the FPS algorithm, we compare FPS to the use of a single, representative sequence from the query set. We choose this representative sequence using the same method as the original Cobbler algorithm. Essentially, the sequence which best matches the motifs for the family is chosen. This sequence is used to search the database, and the averaging step in the FPS algorithm is skipped. For comparison, we also include a test of the original Cobbler method. This involves using the same, representative sequence as the template for a cobbled profile. The database is searched using just this profile, and the averaging step is skipped.

In the second step of the FPS algorithm, the queries are input to a similarity algorithm and compared to each sequence in the search database. Any algorithm suitable for comparing the given type of query with protein sequences may be employed in this second step of FPS. The current work investigates using the BLAST and Smith–Waterman algorithms for computing query-to-sequence similarities. For BLAST searches, we use the negative logarithm of the *E* value as the similarity score. For Smith–Waterman searches, we use the negative logarithm of the *P* value of the Smith–Waterman score. We define the *P* value of the Smith–Waterman score of a sequence as the probability that the score of a random sequence of the same length as the given sequence would be at least as high as the observed score for the sequence.

We use gapped BLAST Version 2.0 (Altschul *et al*., 1990, 1997). In order to use cobbled profiles as BLAST queries, we use one iteration of a version of PSI-BLAST (Altschul *et al*., 1997) that is capable of storing and reading binary checkpoint files. Since these files contain a frequency matrix representation of the query, converting our cobbled frequency profiles to the BLAST checkpoint format is straightforward. PSI-BLAST is run for one iteration with its default parameters. The filtering of low-complexity regions in the query sequence is turned off because this option is unavailable in conjunction with reading checkpoint files. For BLAST searches using sequences as the queries, we use the BLOSUM62 score matrix.

For Smith–Waterman searches, we use the Profilesearch algorithm (Gribskov *et al*., 1990) as implemented on the Bioccelerator (Compugen Ltd, 1996). We set the global gap opening penalty to 8 and the extension penalty to 0.3. In order to calculate *P* values corresponding to Smith–Waterman scores, we calculate the score distribution by fitting the Karlin–Altschul (Karlin and Altschul, 1990) distribution to 10 000 random sequences of length 250 using linear regression. The estimated values of λ and K can then be used to calculate the *P* value of any score.

In the third step of the FPS algorithm, the similarity scores for a given database sequence with each of the queries are averaged together to give the score for comparing the sequence with the family. For convenience, we only include in this average the sequences most similar to the query. When BLAST is used as the similarity algorithm, we compute all similarity scores that correspond to an *E* value smaller than 1000 and assign an *E* value of 1000 to all other sequences. When Profilesearch is used, we compute scores for the 1000 highest-scoring sequences and assign all other sequences a *P* value of 1. Because all protein families in the database we search have far fewer than 1000 members, this approach should yield the same results as actually computing all similarity scores.

Comparing homology detection methods

We use a collection of 73 protein families (Bailey and Gribskov, 1997; Grundy, 1998) in our homology detection experiments. These families were selected from the PROSITE database (Bairoch, 1992) for their difficulty, based upon the number of false positives reported in the PROSITE annotations. The PROSITE IDs and sizes of these families are available on the Web (http://www.cse.ucsc.edu/bgrundy/75-families.html). The families range in size from 5 to 109 sequences, and from 949 to 58 015 amino acids. The associated release of SWISS-PROT (Bairoch, 1994), which contains 36 000 sequences and nearly 12.5 million amino acids, is used as the target database.

Bias within the families is minimized via sequence weighting. Since many weighting schemes perform almost as well as one another (Henikoff and Henikoff, 1994), all the experiments reported here employ a simple, binary weighting scheme based upon BLAST similarity scores (Lawrence *et al*., 1993). This approach is simple, since the highly similar sequences can be removed at once before any analysis is performed, and leads to faster execution, since the sizes of the weighted training sets are reduced. For these experiments, a BLAST similarity threshold of 200 is used. The sizes of the weighted PROSITE families range from 2 to 73 sequences with an average of 10.7 sequences, and from 394 to 18 702 amino acids with an average of 4202.

For each family, the query set is the largest possible set of size 2, 4, 8, 16 or 32 sequences randomly selected from the weighted sequence set. This results in 16 query sets of size 2, 22 sets of size 4, 19 of size 8, 13 of size 16, and three query sets of size 32. In addition, for each family, an independent test set is constructed, consisting of all family members not contained in the query set.

Each homology detection experiment returns a sorted version of the target database. Each sequence in the sorted database is then marked with a '1' or a '0', indicating whether that sequence appears in the PROSITE listing for the current family. In order to test the ability of the homology detection algorithms to generalize from the query set, all family members that do not appear in the independent test set are eliminated from the sorted list. The resulting, purged sequence of bits represents the homology detection algorithm's ability to separate novel family members from non-family members. Perfect performance corresponds to a series of ones followed by a series of zeros.

This bit sequence is subjected to two forms of analysis. The first is a modified version of the Receiver Operating Characteristic, called ROC50 (Gribskov and Robinson, 1996). The ROC score is the area under a curve that plots true positives versus false positives for varying score thresholds. ROC analysis combines measures of a search's sensitivity and selectivity. The ROC50 score is the area under the ROC curve, up to the first 50 false positives. This value has the advantages of yielding a wider spread of values, requiring less storage space, and corresponding to the typical biologist's willingness to sift through only ~50 false positives. ROC50 scores are normalized to range from 0 to 1, with 1 corresponding to the most sensitive and selective search.

In addition to ROC50 analysis, each homology detection method is evaluated using the equivalence number (Pearson, 1995). The equivalence number is the number of false positives given by a database search when the threshold is set so that the number of false positives equals the number of false negatives. To compute the equivalence number from the sequence of bits described above, a mark is moved along the sequence until the number of zeros to the left of the mark equals the number of ones to the right. The equivalence number is the number of zeros to the left of the mark. Perfect separation corresponds to an equivalence number of 0, and the maximum possible equivalence number is the size of the family. In the results reported here, equivalence numbers are scaled to range from 0 to 1 by dividing by the size of the family. This allows equivalence numbers from homology searches for variously sized families to be compared.

Results

Our experiments show that cobbled FPS performs significantly better than both the original Cobbler algorithm and the original FPS algorithm. This improvement occurs regardless of whether the BLAST or Smith–Waterman algorithm is used for pairwise sequence comparisons. On the other hand, replacing the BLAST sequence similarity algorithm in FPS with the full Smith–Waterman algorithm does not lead to a significant performance difference.

Figure 2a shows the performance improvement, as measured by a two-tailed signed rank test (Snedecor and Cochran, 1980; Henikoff and Henikoff, 1997; Salzberg, 1997) on ROC50 scores, offered by the cobbled version of FPS relative to Cobbler. The topmost comparison in the figure verifies the effectiveness of the cobbled BLAST method by comparing it with BLAST. The next comparison shows the improvement that FPS brings to cobbled BLAST. The final comparison shows the large improvement that cobbled BLAST FPS offers relative to BLAST alone. All of the differences shown here are significant when computed with ROC50 scores or normalized equivalence numbers. A summary of the signed rank results for these and other methods is given in Table 2.

Just as FPS improves the performance of Cobbler, so Cobbler improves the performance of FPS. Figure 2b shows the improvement that embedding motif models in the query sequences brings to FPS. Using either BLAST or Profilesearch, cobbling improves FPS at the 1% significance level. Similar analyses with normalized equivalence numbers corroborate the improvement of Profilesearch FPS (1% significance level), but not that of BLAST FPS. This difference in statistical significance is not surprising: equivalence number analysis is less sensitive than ROC analysis, since for a family of size n, the equivalence number can take on one of only n values. Rank comparisons of equivalence numbers therefore tend to result in more ties than similar comparisons based upon ROC scores. Overall, therefore, cobbling improves the performance of FPS.

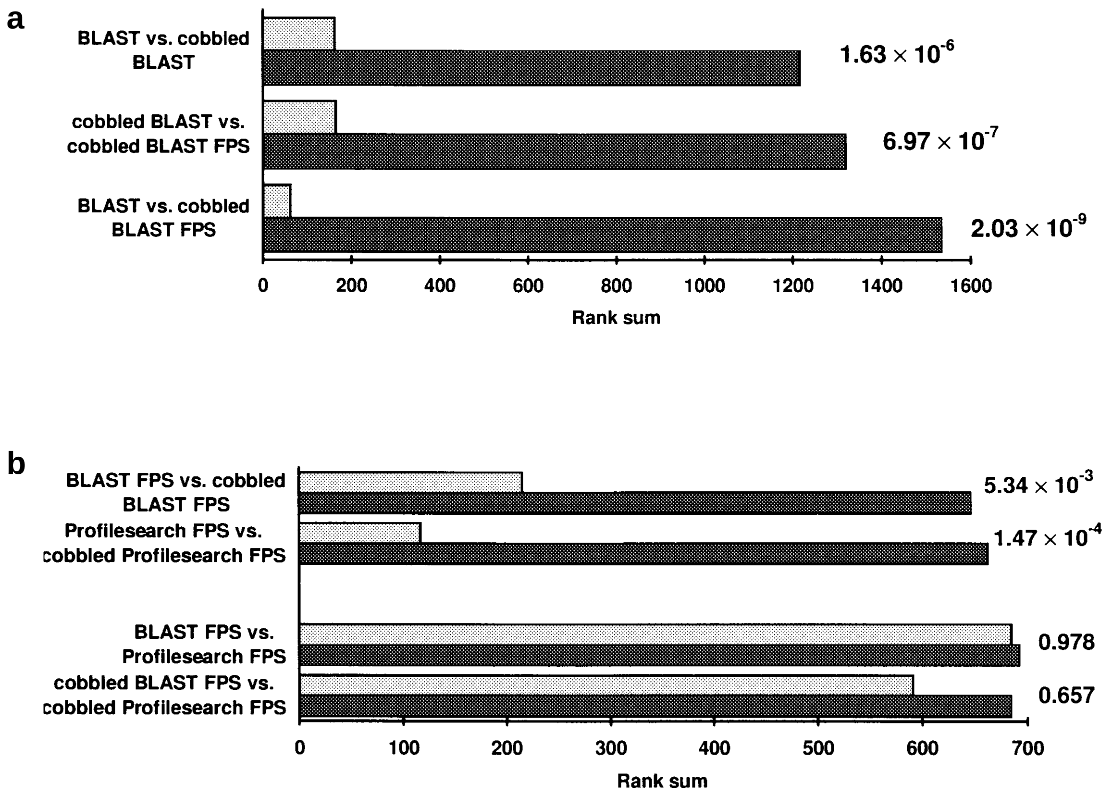

Fig. 2. Improvement offered by cobbled Family Pairwise Search. (**a**) The benefits of adding Family Pairwise Search to Cobbler; (**b**) benefits of adding cobbling to Family Pairwise Search. For each pair of methods A and B, the differences in ROC50 scores are computed with respect to all 73 families in the study. The resulting differences are sorted ignoring the sign of the difference, and the ranks of the differences for which method A scored higher than method B are summed. Each bar represents this rank sum for one method with respect to another. The label on a pair of bars is the significance level at which the null hypothesis that the two methods are equivalent can be rejected.

Table 2. Summary of signed rank comparisons of homology detection methods. For the cell in row A and column B, method A received a higher rank sum that method B, and the cell contains the *P* value at which the null hypothesis that the two methods are equivalent can be rejected

	Method	2	3	4	5	6
1	cobbled Profilesearch FPS	0.657	1.47×10^{-4}	0.140	6.42×10^{-5}	2.16×10^{-8}
2	cobbled BLAST FPS		0.306	5.34×10^{-3}	6.97×10^{-7}	2.03×10^{-9}
3	Profilesearch FPS			0.978	9.90×10^{-5}	2.73×10^{-8}
4	BLAST FPS				1.88×10^{-5}	2.36×10^{-8}
5	cobbled BLAST					1.63×10^{-6}
6	BLAST					

Figure 2b also shows the statistical equivalence of versions of FPS using BLAST and Profilesearch. For both the cobbled and uncobbled versions of the algorithm, BLAST FPS performs slightly worse than Profilesearch FPS, but neither difference is significant. This is somewhat surprising, since BLAST is a heuristic approximation algorithm that runs in $O(n)$ time, whereas the Smith–Waterman algorithm is an $O(n^2)$ dynamic programming algorithm.

The improvement that cobbling adds to BLAST FPS is shown in more detail in Figure 3. The figure compares the ROC50 scores for each of the query sets. Out of 73 queries, cobbled BLAST FPS outperforms BLAST FPS 26 times,

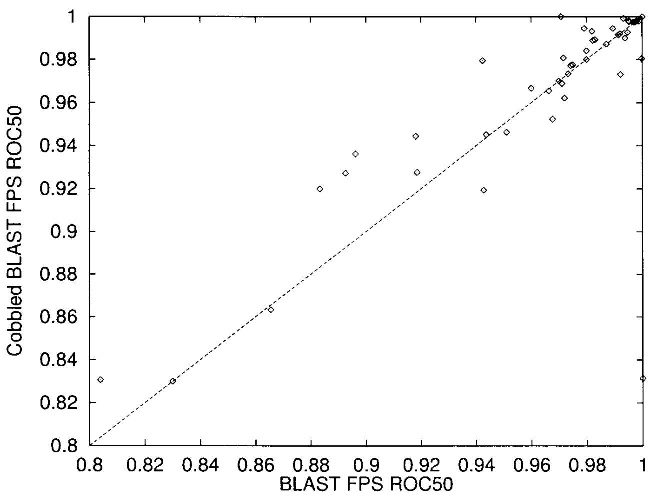

Fig. 3. Query-by-query comparison of cobbled and uncobbled BLAST FPS. Each point represents the ROC50 scores from a single query set. Points above the line $y = x$ are queries for which cobbled BLAST FPS performs better than BLAST FPS, and vice versa. Six of the 73 queries were left off the figure because one or both ROC50 scores fall below 0.80.

and the opposite occurs 14 times. The remaining 33 queries lead to ties, 21 of which involve a perfect ROC50 score for both methods. Similar results hold for the corresponding Profilesearch comparison: the cobbled version wins 23 times and loses six times. The remaining 34 queries result in ties, of which 23 are for perfect scores.

Table 2 summarizes the pairwise comparisons of all six methods evaluated in this study. Of the six methods, BLAST using a single, uncobbled sequence performs worst, followed by cobbled BLAST using a single sequence. The two uncobbled versions of FPS are next, and the two cobbled versions of FPS perform best overall. The pattern of statistical significances in Table 2 does not change if a similar table is generated using normalized equivalence numbers, except for the difference noted previously for methods 1 and 2.

Discussion

Our results show the benefits of building models of protein families. Previous experiments indicated that, for small query sets, a non-model-based algorithm (BLAST FPS) outperforms both sequence-level and motif-based models on the homology detection task (Grundy, 1998). The experiments reported here, however, show that statistical models, used appropriately, can be helpful even for very small query sets. The Cobbler approach (Henikoff and Henikoff, 1997) is an effective means of reducing the size of the models being trained while retaining homology information in noisy regions of the query sequence. The FPS algorithm takes Cobbler one step further by retaining the noisy regions of all query sequences, rather than a single representative. The result is an intelligent compromise, an algorithm that models only the regions of the sequence that are effectively modelable while retaining all of the information from the noisier regions.

In contrast, incorporating the full Smith–Waterman algorithm into FPS yields a small improvement in performance. The computational complexity of the Smith–Waterman algorithm is $O(n^2)$ in the length of the input sequences, whereas BLAST is $O(n)$. Thus, the small improvement offered by the Smith–Waterman algorithm comes at considerable computational cost.

The improved performance of both cobbled BLAST and BLAST FPS relative to BLAST can be explained in terms of the use of homology information in the query sequences. When multiple query sequences are available, searching for homologs using BLAST with a single representative sequence obviously discards important homology information from the rest of the query set. Cobbled profiles remedy this problem somewhat, since they include in the motif regions information from all the query sequences. BLAST FPS furthers the improvement by including all of the information from all of the query sequences.

This kind of explanation, however, fails to account for cobbled FPS's strong performance relative to FPS. Since the FPS algorithm already considers all of the information in the query sequences, the improvement that embedded motif models add to the algorithm must derive from the models themselves, rather than because cobbled FPS considers more information in the query set.

Statistical models of the type built by MEME offer two important advantages over direct pairwise sequence similarity algorithms. First, a position-specific scoring matrix entails the assumption that amino acid occurrences at one position in a protein are statistically independent of amino acid occurrences at other positions. This site-independence assumption allows a candidate protein to receive a high score even if that protein does not closely resemble a single query sequence, but instead is comprised of a mix of sites similar to several proteins in the query set. Second, a statistical model can incorporate prior knowledge that effectively augments the information provided in the query set. For this purpose, MEME employs a set of empirically derived Dirichlet mixture priors (Brown *et al.*, 1995). These priors allow MEME to guess from very little evidence a biologically plausible amino acid distribution for each position in the motif model. Thus, cobbled FPS's improved homology detection performance relative to FPS illustrates the positive effect of the site-independence assumption and of the use of prior information in detecting homologs.

Ultimately, the decision to use a more effective but more expensive homology detection method depends upon the resources—time and money—available to the experimenter. Since the FPS algorithm involves comparing each of the *n* sequences in the protein family to the sequences in the search database, the algorithm requires approximately *n* times as long as searching with a single representative sequence. In practice, however, if binary weighting of the family is employed, *n* is fairly small. In this study, for example, the average weighted family size is 10.7. In most cases, the large performance improvement offered by FPS over single-sequence BLAST will justify FPS's use. However, adding motif modeling to FPS incurs considerable overhead. In this study, MEME requires an average of 9.3 CPU min on a 167 MHz Sparc Ultra and MAST requires 70 s. By comparison, the average BLAST FPS query requires 69 s on the same hardware.

Although the experiments described here use families of sequences as queries, cobbled FPS can also operate in a single-sequence query fashion, scanning a database of sequence families. This mode would allow FPS to take advantage of existing motif databases such as BLOCKS (Henikoff and Henikoff, 1996) and Prints (Attwood *et al*., 1998). In single-sequence query mode, cobbled FPS would be similar to the SSMAL algorithm (Nicodéme, 1998). Like cobbled FPS, SSMAL differentiates between highly conserved motif regions and noisier inter-motif regions. SSMAL requires as input a multiple alignment of the entire family, rather than the unaligned sequences used by cobbled FPS. SSMAL then operates on an alignment graph that reflects the motif-based structure of the family. SSMAL is a heuristic algorithm that incorporates portions of the BLAST algorithm. Therefore, SSMAL is likely to be faster than cobbled FPS at the cost of possibly missing some similarities.

Several recent papers have described algorithms that embed a pairwise sequence similarity algorithm such as BLAST in an iterative procedure for finding homologs. The PSI-BLAST algorithm (Altschul *et al*., 1997) employs a non-motif-based, position-specific scoring matrix representation; Probe (Neuwald *et al*., 1997), on the other hand, employs motif models. The current results indicate that querying using cobbled FPS, rather than merging the queries into a single model, may offer superior performance. We therefore intend to investigate the use of the cobbled FPS algorithm within an iterative framework.

Acknowledgements

The authors would like to thank Heidi Sofia at NCBI for providing a pre-release version of PSI-BLAST capable of performing checkpointing, and Michael Gribskov for helpful discussion. W.N.G. is supported by a Sloan/DOE Fellowship in Computational Molecular Biology. T.L.B. is supported by the National Biomedical Computation Resource, an NIH/NCRR-funded research resource (P41 RR-08605), NBCR grant (NIH P41 RR-08605) and the NSF through cooperative agreement ASC-02827.

References

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) A basic local alignment search tool. *J. Mol. Biol*., **215**, 403–410.

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

Attwood,T.K., Beck,M.E., Flower,D.R., Scordis,P. and Selley,J.N. (1998) The PRINTS protein fingerprint database in its fifth year. *Nucleic Acids Res*., **26**, 304–308.

Bailey,T.L. (1999) MEME—Multiple EM for Motif Elicitation. http://www.sdsc.edu/MEME.

Bailey,T.L. and Elkan,C.P. (1994) Fitting a mixture model by expectation-maximization to discover motifs in biopolymers. In Altman,R., Brutlag,D., Karp,P., Lathrop,R. and Searls,D. (eds), *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, Menlo Park, CA.

Bailey,T.L. and Gribskov,M. (1996) The megaprior heuristic for discovering protein sequence patterns. In States,D.J., Agarwal,P., Gaasterland,T., Hunter,L. and Smith,R. (eds), *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, Menlo Park, CA, pp. 15–24.

Bailey,T.L. and Gribskov,M. (1997) Score distributions for simultaneous matching to multiple motifs. *J. Comput. Biol*., **4**, 45–59.

Bailey,T.L. and Gribskov,M. (1998) Combining evidence using p-values: Application to sequence homology searches. *Bioinformatics*, **14**, 48–54.

Bailey,T.L. and Grundy,W.N. (1999) Classifying proteins by family using the product of correlated p-values. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology.* To appear.

Bairoch,A. (1992) PROSITE: A dictionary of sites and patterns in proteins. *Nucleic Acids Res*., **20**, 2013–2018.

Bairoch,A. (1994) The SWISS-PROT protein sequence data bank: Current status. *Nucleic Acids Res*., **22**, 3578–3580.

BLAST (1999) NCBI BLAST search. http://www.ncbi.nlm.nih.gov/BLAST.

Brown,M., Hughey,R., Krogh,A., Mian,I., Sjolander,K. and Haussler,D. (1995) Using Dirichlet mixture priors to derive hidden Markov models for protein families. In Rawlings,C. (ed.), *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, Menlo Park, CA, pp. 47–55.

Compugen Ltd (1996) Bioccelerator Manual. http://www.compugen-us.com.

Eddy,S.R. (1995) Multiple alignment using hidden Markov models. In Rawlings,C. (ed.), *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, Menlo Park, CA, pp. 114–120.

Gribskov,M. and Robinson,N.L. (1996) Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Comput. Chem*., **20**, 25–33.

Gribskov,M., Lüthy,R. and Eisenberg,D. (1990) Profile analysis. *Methods Enzymol*., **183**, 146–159.

Grundy,W.N. (1998) Homology detection via Family Pairwise Search. *J. Comput. Biol*., **5**, 479–492.

Grundy,W.N., Bailey,T.L. and Elkan,C.P. (1996) ParaMEME: A parallel implementation and a web interface for a DNA and protein motif discovery tool. *Comput. Applic. Biosci*., **12**, 303–310.

Grundy,W.N., Bailey,T.L., Elkan,C.P. and Baker,M.E. (1997) Meta-MEME: Motif-based hidden Markov models of protein families. *Comput. Applic. Biosci*., **13**, 397–406.

Henikoff,J.G. and Henikoff,S. (1996) Blocks database and its applications. *Methods Enzymol*., **266**, 88–105.

Henikoff,S. and Henikoff,J.G. (1991) Automated assembly of protein blocks for database searching. *Nucleic Acids Res*., **19**, 6565–6572.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA*, **89**, 10915–10919.

Henikoff,S. and Henikoff,J.G. (1994) Position-based sequence weights. *J. Mol. Biol*., **243**, 574–578.

Henikoff,S. and Henikoff,J.G. (1997) Embedding strategies for effective use of information from multiple sequence alignments. *Protein Sci*., **6**, 698–705.

Karlin,S. and Altschul,S.F. (1990) Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl Acad. Sci. USA*, **87**, 2264–2268.

Lawrence,C.E., Altschul,S.F., Boguski,M.S., Liu,J.S., Neuwald,A.F. and Wootton,J.C. (1993) Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, **262**, 208–214.

Neuwald,A.F., Liu,J., Lipman,D. and Lawrence,C. (1997) Extracting protein alignment models from the sequence data database. *Nucleic Acids Res*., **25**, 1665–1677.

Nicodéme,P. (1998) SSMAL: similarity searching with alignment graphs. *Bioinformatics*, **14**, 508–515.

Pearson,W.R. (1995) Comparison of methods for searching protein sequence databases. *Protein Sci*., **4**, 1145–1160.

Pearson,W.R. (1997) Identifying distantly related protein sequences. *Comput. Applic. Biosci*., **13**, 325–332.

Pearson,W.R. (1998) Empirical statistical estimates for sequence similarity searches. *J. Mol. Biol*., **276**, 71–84.

Salzberg,S.L. (1997) On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Min. Knowl. Discov*., **1**, 371–328.

Smith,T. and Waterman,M. (1981) Identification of common molecular subsequences. *J. Mol. Biol*., **147**, 195–197.

Snedecor,G.W. and Cochran,W.G. (1980) *Statistical Methods*. Iowa State University Press, Iowa.