GENOMICS **38,** 179–191 (1996)
ARTICLE NO. 0614

Sensitivity and Selectivity in Protein Similarity Searches: A Comparison of Smith–Waterman in Hardware to BLAST and FASTA

EUGENE G. SHPAER,*,[1] MAX ROBINSON,† DAVID YEE,† JAMES D. CANDLIN,* ROBERT MINES,* AND TIM HUNKAPILLER†

*Perkin–Elmer, Applied Biosystems Division, 850 Lincoln Center Drive, Foster City, California 94404; and †University of Washington, Seattle, Washington 98195

Received May 13, 1996; accepted September 25, 1996

To predict the functions of a possible protein product of any new or uncharacterized DNA sequence, it is important first to detect all significant similarities between the encoded amino acid sequence and any accumulated protein sequence data. We have implemented a set of queries and database sequences and proceeded to test and compare various similarity search methods and their parameterizations. We demonstrate here that the Smith–Waterman (S-W) dynamic programming method and the optimized version of FASTA are significantly better able to distinguish true similarities from statistical noise than is the popular database search tool BLAST. Also, a simple "log-length normalization" of S-W scores based on the query and target sequence lengths greatly increased the selectivity of the S-W searches, exceeding the default normalization method of FASTA. An implementation of the modified S-W algorithm in hardware (the Fast Data Finder) is able to match the accuracy of software versions while greatly speeding up its execution. We present here the selectivity and sensitivity data from these tests as well as results for various scoring matrices. We present data that will help users to choose threshold score values for evaluation of database search results. We also illustrate the impact of using simple-sequence masking tools such as SEG or XNU. © 1996 Academic Press, Inc.

INTRODUCTION

The aim of searching protein sequence databases such as the Protein Identification Resource (PIR; National Biomedical Research Foundation; (George *et al.*, 1996)) or Swiss-Prot (Bairoch and Boeckmann, 1994) for similarity to some query sequence is to find homologues to that query. This is often the first step taken in analyzing the possible function of the query protein: a similarity between two sequences (a high Smith–Waterman score or low BLAST probability) is usually due to a common ancestry (homology), which, in turn, implies similar functions for these proteins (Bork *et al.*, 1994). The ability to perform these searches both rapidly and accurately has become critical to large-scale genomic and expressed sequence (cDNA) sequencing projects. The results often dictate the direction of further research and development. Therefore, accuracy is paramount.

A search that returns as matches all the sequences of the database can be said to be completely sensitive as no true positive (homologue) has been missed. However, there would be no way to distinguish the true positives from the noise. On the other hand, a very selective approach can be completely accurate in the matches it does return, but will miss homologous and distantly similar proteins. Quite often, these more obscure (weak) matches are the only ones to be found; in these cases they might provide key insights in understanding the proteins in question. Therefore, the trick is to balance the need for sensitivity with a useful degree of selectivity. Balancing selectivity and sensitivity of database searches was discussed before, for example for identification of tRNA in genomic sequences (Fichant and Burks, 1991). Issues important in the similarity searches have been recently presented in an excellent review (Altschul *et al.*, 1994).

There are three common algorithms used most frequently to search databases: BLAST (Altschul *et al.*, 1990), FASTA (Pearson, 1990), and dynamic programming methods based on Smith–Waterman (S-W) (Smith and Waterman, 1981). The FASTx family of programs represented the first rapid method to be widely adopted. FASTA prescreens the database for very short identical matches and then (if optimization of scores is used) extends these matches using the S-W algorithm. BLAST increases the speed of the search by building indexes for the database (as a separate step before the search) and not allowing gaps (insertions/deletions) in the alignments. In addition, BLAST can combine several nongapped alignments between the query and the database sequences and calculate probability for the similarity between the two using Karlin–

[1] To whom correspondence should be addressed. Telephone: (415) 638-5664. Fax: (415) 572-2743. E-mail: gene@apldbio.com.

Altschul statistics (Karlin and Altschul, 1993). The S-W algorithm has historically been considered the most sensitive. Unfortunately, its slow speed has made it impractical for most researchers. Also, in the past few years, improvements to both BLAST and FASTA have led many to believe that there was no measurable difference any longer between one or both rapid methods and S-W. Given the computational performance hit taken with S-W, the comparative argument has been largely moot anyway. However, as the Human Genome Project and other large-scale sequencing projects have begun to amass huge amounts of sequence data, researchers have once again become concerned about what they might be missing.

Commercial organizations in particular realize that they cannot afford to miss significant similarities that might lead in a timely fashion to new understanding as well as products. Consequently, considerable investment has been made in very large computers and special-purpose hardware designed to run S-W (Vogt and Argos, 1992; Wu *et al.*, 1993). But has this investment and traditional wisdom been correct: is S-W significantly better than the other two tools? At what cost to selectivity is the presumed advantage in sensitivity bought? Can special-purpose computer hardware perform as well as software versions while providing the practical operational speed economically?

To answer these questions we proceeded with three efforts. First, to assess the sensitivity and selectivity of the three algorithms, we established a large test set of protein sequence queries and a test database where we know what database sequences are homologous. Our goal was to establish a large and objective test set to avoid the reliance on anecdotal examples. Thus we had a way to measure false positives and false negatives, i.e., the accuracy of searches.

Second, to evaluate the worth of special-purpose computer hardware for this task, the Applied Biosystems Division of Perkin–Elmer, Paracel Inc., and the University of Washington have collaborated to develop a S-W implementation on the Fast Data Finder (FDF), a highly parallel pipeline array processor.

Finally we used our set of queries, representing essentially all diverse protein superfamilies (SF) classified in the PIR (George *et al.*, 1996), to compare the relative performances of BLAST, FASTA, and S-W (in software and with the FDF). We used these tests to also evaluate the consequences of various parametrizations of these algorithms as well as masking regions of low complexity in the query sequences.

We were able to demonstrate that the FDF version of S-W gives results similar to S-W in software that are much more accurate than BLAST. We demonstrate that normalized S-W scores improve the selectivity of searches compared to simple ("raw") S-W scores. Also, various simulation experiments with and without low-complexity masking (Claverie and States, 1993; Wootton and Federhen, 1993) have provided us with practical score thresholds (cut-off scores below which S-W scores are likely to be random). Finally, we have established a testing platform and data set resource for others to test new algorithms and methods in a manner that provides for direct comparison between different methods.

MATERIALS AND METHODS

Software and hardware. We used the BLASTP program Version 1.4 (Altschul *et al.*, 1990), distributed by the National Center for Biotechnology Information (NCBI), with the default parameters, except using different scoring matrices. FASTA Release 2.0 from the University of Virginia was used; suggestions on its optimal operation were solicited from the author. Optimization and ranking results according to probabilities were the recommended defaults. The software version of S-W was a publicly available version called SSEARCH from the FASTA distribution. These searches were run on various Digital, HP, and Sun UNIX workstations. A series of scripts and C-coded software known as the Biological Applications Testbed was used to generate and manage these searches.

The S-W searches in hardware were performed with the FDF—a systolic array designed for text string pattern matching. We used a "one-board" FDF system that has a physical pipeline of 480 cells (additional boards with 720 cells each can be installed), which is functionally expanded into a 45,000 "virtual" cell pipeline. The query sequence and scores from the matrix are loaded into the pipeline. The database sequences are streamed past the pipeline, and S-W scores are calculated at every hardware cycle (all scores in an anti-diagonal in parallel). The highest score for each database sequence is reported. The system used is part of GeneAssist, a sequence analysis package from Perkin–Elmer (Shpaer, 1996). The GeneAssist server operates on a Sun workstation running SunOS or Solaris. The speed of the search for a one-board system is 55 million cell updates per second, e.g., a search for a 176-amino-acid-long query against the 7.5-Mb database we used (see below) takes 24 s ($176 \times 7.5/24 = 55$); a similar search takes several hours on a Sparc workstation. The search speed for a five-board FDF is 230 million cell updates/s.

Procedure. The overall testing procedure was as follows:

1. Generate a test database with associated SF numbers from the PIR; randomly select a query sequence for each multirepresented SF in this database.
2. Refine the query test set to eliminate SF where all members are very similar and all searches correctly identify true SF members having higher scores than any other sequences in the database.
3. Run one of the Comparison Methods for a set of query sequences and generate a result file with 3000 top scores per query.
4. Calculate the "missed@equivalence" and "top-random cut-off" points (see Results) and generate a table for all queries in a set.
5. Normalize all scores in the result file using Eqs. [3] or [4].
6. Combine all tables with scores and compute better/worse SF counts as shown in Tables 3 and 4.

Test data. The query and database test sets were generated from PIR Release 40. We relied on the fact that the PIR, when possible, annotates its data with SF numbers: the same SF numbers indicate related (homologous) sequences. The test database included all entries of the PIR Release 40 database that had assigned SF numbers: all of the PIR1 (classified and annotated) and about half of the PIR2 (preliminary) sequences. About 5% of these entries were removed using the "nrdb" program from NCBI, which eliminated all but one copy of identical protein sequences. The resulting test database has 22,390 nonredundant entries (~7,500,000 amino acid residues).

To compile the query set, we first took the list of 3331 SFs represented in the test database and excluded the 1195 SF with just one database entry along with SFs with no entries in the PIR1 subset. From the remaining 1444 potentially useful SFs, we selected the first database occurrence of each as a query sequence. These 1444 queries were searched against the test database using FASTA with a k-tuple value of 2, the PAM250 scoring matrix, and no optimization.

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	4																			
C	-2	10																		
D	-1	-5	5																	
E	-1	-5	1	5																
F	-2	-2	-5	-4	7															
G	0	-4	-1	-2	-5	5														
H	-2	-5	-1	-1	-1	-2	8													
I	-2	-4	-4	-3	0	-5	-4	5												
K	-1	-5	-1	1	-3	-2	0	-3	5											
L	-2	-4	-5	-3	1	-4	-2	2	-2	5										
M	-1	-2	-5	-2	1	-4	-2	1	-1	2	7									
N	-1	-4	2	0	-3	-1	1	-4	0	-3	-2	5								
P	-1	-5	-1	-1	-4	-2	-2	-3	-1	-4	-4	-2	7							
Q	-1	-3	0	1	-3	-2	1	-3	1	-2	0	0	-2	6						
R	-2	-3	-2	0	-4	-3	0	-3	2	-2	-2	-1	-2	1	7					
S	0	-3	0	-1	-3	-1	-2	-4	-1	-3	-2	0	-1	0	-1	4				
T	-1	-3	-1	-1	-3	-3	-2	-2	0	-2	-1	0	-1	0	-1	1	5			
V	0	-2	-4	-2	-1	-4	-3	3	-2	1	0	-3	-3	-2	-3	-2	-1	5		
W	-3	-5	-5	-4	2	-4	-3	-2	-3	-1	0	-4	-5	-3	-1	-4	-5	-3	10	
Y	-2	-4	-3	-2	3	-4	0	-1	-2	-1	0	-2	-3	-2	-2	-2	-2	-2	2	7
-	-7	-9	-7	-6	-8	-7	-8	-9	-7	-8	-8	-7	-6	-7	-7	-7	-8	-8	-9	-8

FIG. 1. Structural (STR) matrix. The bottom row shows variable penalties for indels.

These parameterizations provided a rapid, but not very stringent, search. The rationale for this was to eliminate those query/SF pairs where any method would rank the results without error (all SF members rank above any other sequence in the database) and thus provide no potential for discriminating between algorithms. These searches ranked perfectly [no false positives (unrelated sequences) having a score larger than any true positive (SF members)] all members for 871 of the 1444 SF. By eliminating these 871 queries, we excluded "trivial" SFs from the test set. Finally, we replaced 15 queries that were only partial (incomplete) sequences with the first complete example of that SF in the database. Thus, we ended up with the set of 573 complete protein sequences (and one fragment) from PIR1 representing all "nontrivial" SFs in the PIR Release 40 database.

For some of our tests, we used SEG (Wootton and Federhen, 1993) filtering of the queries. SEG evaluates a sequence and replaces those regions that are simple (have a low information content, generally a simple composition) with "Xs." Using default SEG parameters, a number of the test query sequences were essentially replaced by Xs. After removing these along with several long (>850 residues) sequences, 502 queries remained in this subset.

Scoring matrix and insertion/deletion penalties. All algorithms were tested with a range of scoring matrices and relative insertion/deletion (indel) penalties. Percent amino acid mutation (PAM; Dayhoff *et al.*, 1978) and BLOSUM (Henikoff and Henikoff, 1992) matrices were tested along with a new matrix that is based on the structural (X-ray data) alignments of homologous proteins. Using amino acid substitution frequencies kindly provided by Dr. Mark Johnson (pers. comm., 1995, partially published in Johnson and Overington (1993)), we calculated the scores for amino acid substitutions as log-odd probabilities (Altschul, 1991):

$$S_{ij} = \log_2(Q_{ij}/(P_i \times P_j)) \times \text{Scale}, \qquad [1]$$

where S_{ij} is the score for amino acid pair "ij", Q_{ij} is the observed frequency of the "ij" pair in the alignments, P_i and P_j are the frequencies of residues i and j ($P_i \times P_j$ is the expected frequency for the "ij" pair based on the amino acid composition), Scale = 2 for scoring matrices in 1/2 bits, and Scale = 3 for matrices in 1/3 bits. Following Altschul (1991), we used logarithms of base 2 and multiplied log-odd probabilities by Scale = 2 (scaling factor) to get a scoring matrix in half bits. The structural (STR) matrix is shown in Fig. 1; its relative entropy (H, also average score in bits per aligned residue pair) is 0.84, which is approximately equivalent to the PAM140 and BLOSUM70 matrices.

Unlike the BLOCKS data used for the BLOSUM series, the structural data provide the frequencies of deletions for each amino acid. Using these data, we calculated variable indel scores for 20 amino acids, using the method described below. The structural alignments contain N_{aa} = 254,048 pairs of amino acid residues aligned with another amino acid residue and N_{ind} = 18,439 pairs with deletions. Indels, therefore, comprise ($N_{\text{ind}}/N_{\text{aa}}$ = 0.073) 7.3% of the alignments (there is, on average, one indel per 14 aligned amino acids). The average log-odd score for an indel (logarithm of the probability of finding an indel in a particular position) is

$$S_{\text{ind}} = \log_2(N_{\text{ind}}/N_{\text{aa}}) \times \text{Scale}$$

$$(S_{\text{ind}} = -7.5 \text{ for STR matrix in 1/2 bits}). \qquad [2]$$

The score for deleting a specific amino acid ("i") is the sum of the "amino acid specific" indel score calculated using Eq. [1] and the average (S_{ind}):

$$S_{i-\text{del}} = \log_2(Q_{i-}/(P_i \times P_-)) \times \text{Scale} + S_{\text{ind}}, \qquad [2a]$$

where Q_{i-} is the observed, and $P_i \times P_-$ the expected, frequencies for the deletion of amino acid "i".

For example, there are Q_{E-} = 1330 positions in the alignments where glutamic acid (E) is deleted in one of the sequences. The indel score for glutamic acid is

$$S_{\text{E}-\text{del}} = \log_2(1330/898) \times 2 + S_{\text{ind}} = -6.4. \qquad [2b]$$

Similarly, the indel score for the isoleucine (I) is: $S_{\text{I}-\text{del}} = \log_2(535/920) \times 2 + S_{\text{ind}} = -9.1$. The expected frequencies for Glu and Ile deletions are close (898 and 920, respectively) because the two amino acids have similar percentages in the amino acid composition. At the same time, the glutamic acid is deleted/inserted during evolution (as observed in the structural alignments of homologous sequences) 2.5 times more frequently than isoleucine (1330 vs 535). One can see in Fig. 1 that indel scores are the lowest for hydrophobic amino acids and cysteine (−8, −9) and the highest for charged amino acids and proline (−6, −7). Essentially, we treated indels as a 21st "residue"

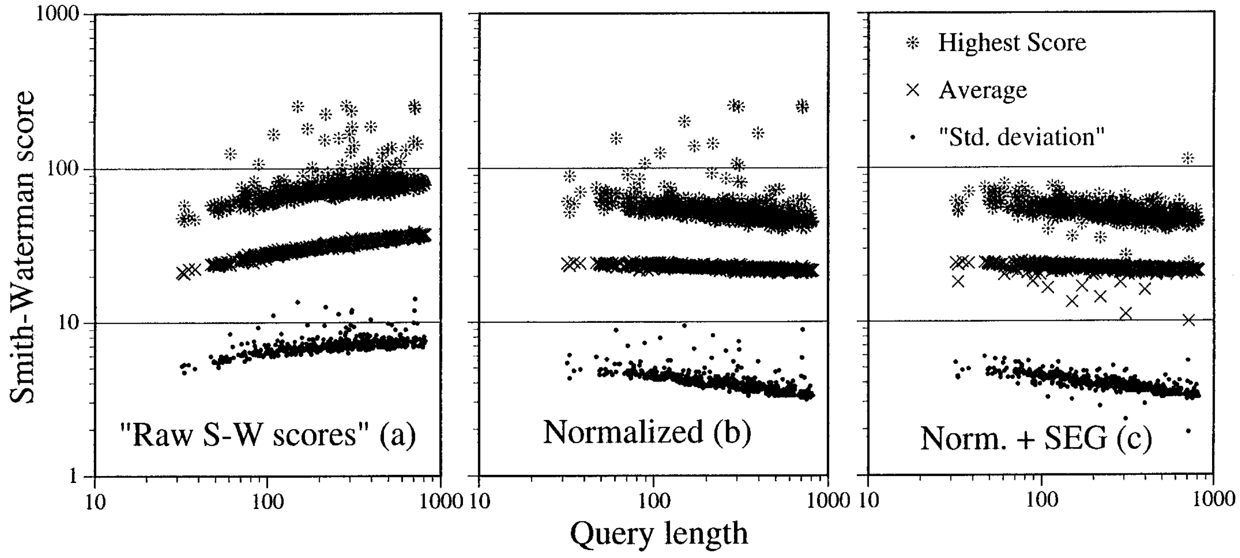

FIG. 2. Smith–Waterman (S-W) scores for 502 random query sequences depending on their lengths. Each panel shows 502 data points for the maximum, average, and "the width of the distribution." The latter was calculated by adding squares of the differences between the average S-W score and the 22,390 actual scores (had these been normal distributions the value would be the "standard deviation"). (**a**) Raw S-W scores; (**b**) scores after log-length normalization; (**c**) scores after both normalization and SEG filtering of query sequences.

and calculated scores for them in the same way as for other amino acids, with the exception that there is an additional penalty (S_{ind}) for all indels.

Next, we calculated variable indel scores for scoring matrices with different entropies/PAM values, extrapolating data from Benner *et al.* (1993). The number of indels in the alignments of homologous sequences approximately linearly correlates with the PAM values, e.g., $N_{\text{ind}}/N_{\text{aa}} = 0.03$ (3%) at PAM = 25, compared to 5.4% at PAM = 87 and 7.3% at PAM = 140 (STR data above). Extrapolation of these data predicts that indels would comprise 8.6% of all positions in the alignments at PAM = 160 (BLOSUM62), 9.7% at PAM = 185 (BLOSUM55), and 10.5% at PAM = 205 (BLOSUM50). The respective global indel penalties for these three matrices, as calculated using Eq. [2], are −7, −10, and −9.5, respectively. Because BLOSUM55 and BLOSUM50 are traditionally scaled in 1/3 bits, we used Scale = 3 in Eq. [2] for these two matrices.

Normalization of scores and filtering. The S-W scores for local alignments between distantly related sequences correlate only weakly, if at all, with sequence lengths. On the contrary, the S-W scores for random sequences increase when the lengths of the query and/or the database sequences increase. For example, the average S-W score for short random queries (50 amino acids) is ~20, while for 800-aa-long sequences it is 35 (Fig. 2a). Therefore, some form of normalization of raw S-W scores is needed to provide a better separation of S-W scores that occurred by chance from those due to sequence homology and thus increase the selectivity of the search.

Smith (Smith *et al.*, 1985) noticed that S-W scores for pairs of unrelated sequences increase proportionally to the logarithm of the product of the lengths of the sequences. Waterman and Vingron (1994) analyzed the distribution of S-W scores for random sequences. The "center" of the distribution of random scores is $\log_{1/p}(Q_{\text{len}} \times T_{\text{len}})$, where Q_{len} and T_{len} are the lengths of the query and target (database) sequences and p depends on the scoring matrix and indel penalties. This result is an extension of the work by Karlin and Altschul (1993), who analyzed nongapped alignments and have come up with the following normalization equation:

$$S_{\text{norm}} = \lambda \times S - \ln(\text{K} \times Q_{\text{len}} \times T_{\text{len}}), \qquad [3]$$

where S and S_{norm} are the raw and normalized S-W scores, respectively, and λ and K are constants that depend on the scoring matrix and amino acid composition of the database.

Independently from Pearson's recently published paper (Pearson, 1995), we noticed that empirical "logarithm" normalization function works better in our tests than the Karlin–Altschul (K-A) normalization (since K and λ have been calculated for nongapped alignments, we tried different values when normalizing S-W scores that include indels). A drawback of Pearson's normalization, however, is that it does not treat query and database sequence lengths symmetrically, and, consequently, the S-W scores for random sequences increase with the query length. Therefore, we decided to use a similar normalization function that: (1) uses both sequence lengths in a symmetrical way; (2) makes the average S-W score for random sequences independent of sequence lengths; (3) most improves the selectivity of the searches; and (4) does not change the raw S-W scores if both the query and the database sequences are 100 residues long. We used the following log-length normalization function:

$$S_{\text{norm}} = S \times 21.21/[\ln(Q_{\text{len}}) \times \ln(T_{\text{len}})]. \qquad [4]$$

This method of normalization would not change the raw score if $\ln(Q_{\text{len}}) \times \ln(T_{\text{len}}) = 21.21$, e.g., if both sequences are 100 residues long [$\ln(100) \times \ln(100) = 21.21$]. We also tried normalization using the logarithm of the product of two lengths (Smith *et al.*, 1985)—it seems to work as well as the log-length normalization (data not shown).

To test this normalization, we generated 502 random protein sequences, each having the length and composition as one of the queries in the 502 SF set. A S-W search of each was run against the test database (Fig. 2). After log-length normalization, the average S-W score for a random query sequence becomes nearly independent of the query length (~25 for BLOSUM62 matrix), and the "width" of the distribution decreases (compare Figs. 2a, 2b and 3a).

When analyzing the results of S-W searches, it is important to have in mind a cut-off score, below which the scores are likely to be random and above which scores indicate possible homology. Figure 3 shows the cumulative distribution of S-W scores (BLOSUM62 matrix) for 502 random query sequences. We applied SEG filtering (Wootton and Federhen, 1993) to evaluate the role of sequence complexity in signal-to-noise discrimination. As Fig. 3b indicates, SEG filtering dramatically decreases the number of scores at the "high end" of score distributions. For example, raw S-W scores >80 occur once in 4×10^3 pairwise sequence comparisons with random query sequences. This frequency decreases to one in 2×10^4 after score

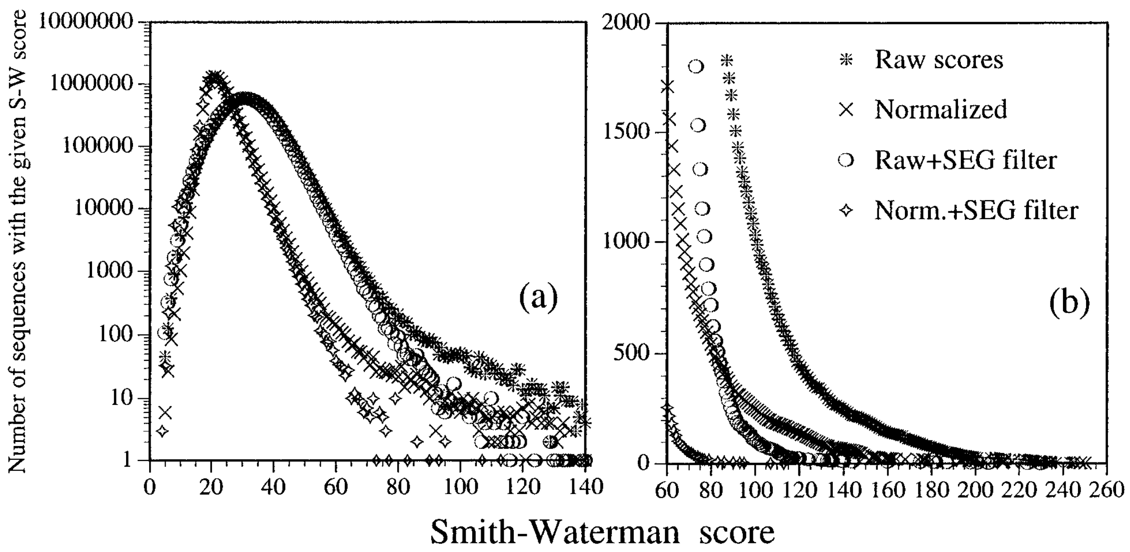

FIG. 3. Cumulative distribution of S-W scores for 502 random query sequences (BLOSUM62 matrix). Each curve in (**a**) shows distribution for 502 × 22,390 ~ 11 × 10^6 S-W scores. (**b**) The total number of S-W scores above the score on the *X* axis. For example, there are 2944 raw S-W scores higher than 80; normalization decreases this number to 544; SEG filtering alone reduces it to 717. When both normalization and SEG filtering are used, the number drops to just 9: the normalized S-W scores (BL62) exceeded 80 only 9 times of 11 × 10^6 pairwise comparisons between random SEG-filtered query sequences and PIR entries.

normalization. When SEG filtering is applied, the frequency drops to one in 10^6: i.e., there is a one in a million chance that a normalized S-W score exceed 80 for a random SEG-filtered query (BLOSUM62 matrix). The distribution in Fig. 3a shows a linear correlation between the logarithm of the number of high S-W scores and the score for the SEG-normalized scores. This is what one would expect if the probability to increase a S-W score does not depend on the score itself (Collins *et al.*, 1988).

RESULTS

The Accuracy of Different Comparison Methods

We used the criterion of "missed@equivalence" point, suggested by Pearson (1991, 1995), as one of the ways to compare the accuracy of different similarity search methods. Table 1 shows S-W and BLAST results for SF 26 "iron-sulfur proteins." The "missed@equivalence" point means that cut-off will always be at the SF size (in this case, there are nine "iron-sulfur" proteins in the database, so cut-off is after the ninth top score): the number of "missed" sequences (SF members that are below the cut-off) and the number of false positives (non-SF members above the cut-off) is equal. Hence, this is the "equivalence" point.

There are cases in the PIR database when unrelated sequences are described as belonging to the same SF. These cases will not affect our ranking of different methods, because it is likely that all methods will miss these unrelated sequences. On the other hand, the classification of protein SFs in the PIR database often does not take into account that certain SFs are related. The proteins from these related SFs are treated as "false positives" when we use the missed@equivalence criterion, potentially introducing a bias in our comparison of different search methods. For example, a more sensitive search method might detect weak real similarities and get penalized for this because they are treated as false positives. Therefore, we used the second criterion—"top-random cut-off"—to compare the accuracy of protein similarity search methods (an example is shown in Table 1). This method requires two searches: first, we run a search for a random query sequence with the same amino acid composition and length as the real query and record the highest score (lowest probability in the case of BLAST). Then we run a regular search and use the highest score for the random sequence as a cut-off to calculate the number of SF members missed—this is, generally, a more stringent method than using the missed@equivalence criterion because we do not allow any SF members with scores below that of the highest scoring random sequence (no false positives).

Comparing BLAST, S–W, and Scoring Matrices

Table 2 shows the missed@equivalence scores for the set of 36 protein SFs used previously (Pearson, 1991)—we present these results for illustration. The table shows the name of the SF, the number of sequences in the SF, and the query name in the PIR database.

There are 4636 PIR40 entries that belong to the 36 SFs. It is easy to notice in Table 2 that both FASTA and S-W perform much better (the number of missed sequences is smaller) than BLAST. It is important to note that the only difference between S-W in software and hardware was different scoring for indels: affine scoring for S-W in software (SSEARCH) had scores of "−10" for gap opening and "−2" for the extension, while "variable" scores (Fig. 1) were used for the hardware implementation of S-W. (SSEARCH and S-W on the FDF produced identical results when we used same

TABLE 1

Output for the Similarity Search for the Query Sequence IHKREV

	Smith–Waterman			BLAST				
Rank	Locus	SF No.	Score	Locus	SF No.	Score	$P(N)$	N
1	IHKREV	26	494	IHKREV	26	460	3.2e-60	1
2	IHTFER	26	416	IHTFER	26	388	3.1e-50	1
3	IHKREG	26	374	IHKREG	26	197	1.7e-44	2
4	IHTF	26	183	IHTF	26	98	1.4e-21	2
5	IHRFG	26	128	IHPC	26	78	1.9e-14	2
6	IHPC	26	104	IHRFG	26	84	1.1e-05	1
				Top-random		*37*	*4.1e-05*	*4*
7	IHQFT	26	72	ALBSMX	507	42	0.26	3
8	IHER1	26	68	IHER2	26	39	0.40	2
9	IHER2	26	63	ALBSGC	507	39	0.60	3
	Top-random		58					
10	D25866	1029	57	VGVN	2653	37	0.76	3
11	S08090	2795	57	IHER1	26	37	0.80	2

Note. The query sequence IHKREV belongs to the superfamily (SF) "high potential iron-sulfur proteins" (SF 26 in PIR40), which has nine members. The cut-off at the SF size (nine top scores shown by a horizontal line) was used. This cut-off is also called "missed@ equivalence" because at this cut-off the number of true SF members that are below cut-off (false negatives) is equal to the number of unrelated sequence above cut-off (false positives). All SF members are above the line for the S-W output—every "iron-sulfur protein" has a higher score than any other sequence in the database; no sequences have been "missed." BLAST output shows two unrelated sequences above the cut-off (false positives) and two members of SF 26 below the cut-off (false negatives; the sequence IHQFT was not even present among the top 100 in the output). So, BLAST missed two iron-sulfur proteins at the "equivalence" point. The "top-random cut-off" is also shown: the highest score for the S-W search and the lowest probability for BLAST for a random query sequence with the length and amino acid composition the same as the IHKREV query (in italics). All nine SF members have S-W scores higher than 58 (top-random); however, only six SF members have BLAST probabilities lower than 4.1e-05 (lowest probability for random query). All searches used the BLOSUM62 scoring matrix and BLASTP Version 1.4 with default parameters; S-W scores have been log-length normalized using Eq. [4].

"fixed" indel penalties for both, not shown.) We tried K-A and log-length normalization for both SSEARCH and S-W using the FDF. The K-A normalization improves results for 19 and 17 SFs for SSEARCH and the FDF S-W, respectively, compared to raw S-W scores. Log-length normalization further improves the selectivity of searches for 15 (SSEARCH) and 23 (FDF) SFs compared to K-A normalization (there are only three cases where log-length normalization increased the number of missed sequences by one compared to K-A normalization).

One can compare the total number of missed sequences in the bottom of Table 2 in the six columns that show results for S-W-BLOSUM62 searches in software and FDF; e.g., for FDF S-W this number decreases from 683, when raw scores are used, to 625 (K-A) to 543 (log-length) normalized searches. FASTA-BLOSUM62 without optimization of scores seems to be as accurate as BLAST. The optimization of FASTA scores, recommended by Pearson to be always used in searches, dramatically improves the accuracy (FASTA using optimized scores performs as well as S-W ranked by score). When ranked by probability, FASTA performs similarly to K-A normalized S-W searches. (Log-length normalized FASTA performed essentially as well as log-length normalized S-W; data not shown.) The total number of sequences missed by each method is shown at the bottom of Table 2. For example, BLAST-BLOSUM62 missed 756 sequences (16.0% of all SF members). S-W on the FDF with BLOSUM62 missed 543 SF members (12.0%).

The drawback of the comparison based on the total number of missed sequences is that a few SFs can make a big difference. For example, for the immunoglobulin variable region SF (IgV, Table 2), BLAST-BLOSUM62 missed 142 IgV sequences, compared to only 67 for S-W-BLOSUM62. Therefore, we compared methods against each other based on the number of SFs where the method was better or worse than others (Table 3). For example, S-W-BLOSUM62 performed better than BLAST in 28 SFs of 36 and worse in only 2 (FOVWH3, 2:3; O4HUD1, 79:80—the numbers show how many SF members have been missed by BLAST:S-W, respectively).

Table 3 shows these better/worse SF counts for the complete 573 SF test set. For example, the 28:2 better/worse numbers for S-W:BLAST-BLOSUM62 from Table 2 are now included in the 162:42 for these two methods for the 573 SF set. It is interesting to note that both BLAST and S-W did not miss any SF members in 117 SF of 573 (perfect separation of all SF members that have higher scores / lower probabilities than those for any other sequence in the database). BLAST worked perfectly for another 13 SF, but S-W was perfect for an additional 54 SF—one of them was the "iron-sulfur" protein SF shown in Table 1.

The better/worse SF count makes noticeable very small differences between various search methods. By

TABLE 2
"Missed at Equivalence Scores" for 36 Protein Superfamilies for BLAST, FASTA, and Smith–Waterman (Software and FDF)

										SW-bl62									
		BLAST					FASTA-bl62 ktup = 1			ssearch			FDF			S-W (FDF)			
Protein superfamily /number of members/	PIR locus	STR	bl70	bl62	bl55	bl50	No-opt.	Opt.	Prob.	Score	KA	Ln-N	Score	KA	Ln-N	STR	bl70	bl55	bl50
Glut.-ammonia ligase /47/	AJHUQ	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0
Plastocyanin /56/	AZBR	39	39	41	41	41	35	31	27	31	27	25	34	33	28	13	21	27	27
Cytochrome C /151/	CCHU	29	26	26	28	29	29	29	26	30	27	22	35	24	22	21	19	18	16
Alpha-crystalline /94/	CYBOA	13	12	14	13	13	7	5	3	5	4	3	7	4	3	4	3	3	4
Alc. dehydrogenase /72/	DEHUAA	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Ferredoxin /151/	FEPE	77	78	79	80	83	77	77	76	77	76	75	78	77	76	76	77	77	77
AIDS related gag /38/	FOVWH3	3	3	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Glucagon /63/	GCBO	9	8	6	6	8	9	9	6	11	7	2	16	11	4	4	9	4	6
Globin /635/	GGLMS	41	39	34	31	33	31	25	21	22	21	18	24	22	19	21	32	17	17
Pol polyprotein /84/	GNFV1R	17	20	20	20	20	19	19	19	17	17	17	17	17	17	17	17	17	17
Histocompatib. Ag /114/	HLHUB2	14	14	13	13	13	13	13	13	12	12	12	13	13	13	13	13	13	13
Influenzae hemagt. /118/	HMIVV	11	15	17	4	2	5	5	5	1	1	1	1	1	1	1	1	1	1
Histone H1 /55/	HSHU1B	3	3	3	5	4	4	4	4	3	2	2	4	4	2	2	2	2	2
Histone H2A /51/	HSHUA1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Insulin /118/	IPHU	10	9	10	9	6	11	11	10	11	10	8	12	11	10	9	9	8	7
IgV /908/	K1HUAG	124	135	142	164	141	197	97	78	95	85	59	93	77	67	62	74	64	67
IgC /122/	K3HU	41	44	46	45	44	39	39	39	38	38	35	38	35	30	28	34	34	33
Cytoskeletal keratin /124/	KRHUE	10	11	12	12	11	11	11	11	11	10	8	15	15	9	11	9	9	10
Hepatic lectin /24/	LNHU1	7	3	5	5	4	2	2	2	2	2	2	3	3	3	8	5	3	3
Influenza virus NS2 /22/	MNIV2K	2	2	2	2	2	2	2	1	2	1	2	2	2	2	2	2	2	2
Snake toxin /144/	N2KF1U	4	3	6	10	9	2	2	2	2	2	1	3	2	1	1	1	1	1
Scorpion neurotoxin /66/	NTSRIA	21	22	20	20	20	20	18	15	19	15	11	18	15	13	13	14	15	15
Cytochrome P450 /365/	O4HUD1	79	80	79	79	79	79	79	80	79	78	78	80	79	80	78	83	80	80
Phospholipase A2 /127/	PSHU	8	8	8	10	12	10	10	6	10	6	4	10	9	4	4	4	6	7
H + trans. ATPsynthase /41/	PWHU6	9	3	5	4	8	4	4	4	3	3	3	5	5	5	8	7	5	6
Inner membr. MalK /23/	QRECBD	6	7	6	5	6	7	7	7	13	13	5	11	11	4	5	6	4	4
Metallothionein /47/	SMHU2	9	9	10	10	12	8	8	8	7	6	3	8	7	3	3	3	3	3
Basic prot. inhibitor /39/	TIBO	15	14	16	15	14	16	16	16	16	16	11	16	16	11	10	11	11	11
Calmodulin /224/	TPHUCS	22	20	20	21	20	24	24	21	25	23	14	27	22	14	17	15	17	16
Trypsin /175/	TRRT1	53	53	53	53	53	52	52	51	53	53	52	53	53	51	53	50	52	51
Protamine Y2 /36/	TYTUY2	7	5	5	6	7	3	3	3	3	3	2	3	3	2	2	2	2	2
alpha-2u-globulin /79/	UART	30	30	27	28	29	23	22	22	24	22	22	28	28	26	26	25	23	23
Type C retrovir. env /69/	VCLJB	17	21	17	17	17	14	14	12	14	12	12	14	13	12	13	16	12	12
Papilomavirus E6 /68/	W6WL18	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
Papilomavirus E7 /29/	W7WLHS	2	3	1	0	2	1	1	0	1	0	1	0	0	0	0	0	0	0
Antithrombine-III /57/	XHHU3	7	7	7	7	7	7	7	7	7	7	5	7	7	5	5	5	5	5
Total: /4636/		742	749	756	770	756	767	652	601	650	605	521	683	625	543	536	576	541	544

Note. The table shows how many sequences are missed at the equivalence point for every superfamily (rows) and method of the similarity search (column; smaller numbers mean better accuracy). An example for missed@equivalence scores is shown in Table 1. BLAST was used with default parameters and five scoring matrices. FASTA2.0 was used with BLOSUM62 matrix, gap opening penalty "-10," gap extension "-2," and ktup = 1. The three columns under FASTA represent results without optimization ("-o" option in FASTA2.0), ranked by optimized scores, and ranked by probability (default output ranking in FASTA2.0). S-W searches with BLOSUM62 were run both in software [ssearch program from FASTA distribution, matrix (same gap opening/extension penalties as in FASTA)] and hardware (using FDF with variable gap penalties; Fig. 1 shows an example). The three columns each for ssearch and FDF-bl62 show: (1) results ranked by score; (2) using Karlin–Altschul (Eq. [3]), and (3) log-length normalization (Eq. [4]). The last four columns represent log-length normalized S-W searches in hardware with different scoring matrices and variable indel penalties.

TABLE 3

Comparison of BLAST, FASTA, and S-W Based on "Missed@Equivalence Scores" for 573 Protein Superfamilies

	BLAST					FASTA/bl62, ktup= 1		Smith−Waterman (FDF)				
Method	STR	bl70	bl62	bl55	bl50	Opt.	Prob.	bl62	STR	bl70	bl55	bl50
BLAST-stR	0	56	53	63	57	79	94	152	159	161	145	147
BLAST-bl70	44	0	36	50	48	73	84	142	152	155	139	138
BLAST-bl62	65	68	0	46	40	82	95	***162***	178	173	160	159
BLAST-bl55	67	76	40	0	26	83	94	152	165	165	154	149
BLAST-bl50	82	87	62	53	0	97	106	164	176	177	164	160
FASTA-ol	95	98	82	85	81	0	66	157	170	170	154	148
FASTA-o1P	69	74	58	61	58	10	0	123	142	141	122	118
SW-bl62	48	48	***42***	43	42	32	32	0	57	56	29	28
SW-STR	49	52	52	52	50	45	49	51	0	48	53	55
SW-bl70	51	49	51	54	51	53	54	55	49	0	56	55
SW-bl55	48	51	48	50	50	36	39	42	68	71	0	30
SW-bl50	56	58	53	54	53	38	44	48	72	72	34	0
Better	674	717	577	611	556	628	757	1248	1388	1389	1210	1187
Worse	1166	1061	1228	1171	1328	1306	976	457	556	578	533	582
Total	**−492**	**−344**	**−651**	**−560**	**−772**	**−678**	**−219**	**719**	**832**	**811**	**677**	**605**

Note. The numbers in the table show for how many protein SF the method in the column performed better than the one in the row, based on the missed@equivalence scores (Table 1 and 2). All BLAST searches were with default parameters; FASTA was run with BL62, gap opening/extension penalties "-10/-2," ranked by optimized score or probability. S-W scores on FDF (variable indel penalties) were log-length normalized. For example, SW-bl62 was more accurate than BLAST-bl62 in 162 of 573 SF; the opposite was true for 42 SF (both numbers are shown in boldface italics). The "Better" row shows how many times the method in the column performed better than the other 11 methods (sum of all numbers in the column); "Worse" shows how many times the method was worse (sum of all numbers in the row for the given method). The "Total" shows the difference between "Better" and "Worse" values and may be regarded as a numerical characterization of the accuracy of the search compared to the 11 other methods (negative "Total" numbers show that the method was worse than average; positive, better than average).

adding all numbers in the column we calculated how many times the comparison method was Better than all other methods in the Table; similarly, by adding all values in a row we calculated how many times the method was Worse. Finally, the Total score for the method is the difference between the two. The three numbers (Better, Worse, Total) are shown in the bottom three rows of Table 3. A positive Total number means that the method was better than the average among 12 methods in Table 3; a negative number means that the method was worse than the average.

FASTA, ranked by probability, performs worse than S-W with log-length normalization, but much better than BLAST (Table 3). S-W with STR, BLOSUM70, or BLOSUM62 scoring matrices shows very similar performance. There is a tendency for higher relative entropy matrices, (e.g., BLOSUM70, STR) to work better than low-entropy BLOSUM50 matrix for both BLAST and S-W.

The missed@equivalence criterion has certain limitations discussed above. Therefore, we used the top-random cutoff criterion (see an example in Table 1) to compare Better/Worse SF counts for BLAST, FASTA, and S-W with different scoring matrices (Table 4). According to the results shown in Table 4, BLAST performs significantly worse than S-W: there are four times as many SF members for which S-W-BLOSUM62 performed better than BLAST-BLOSUM62 than for which the reverse was true (112 vs 29). FASTA ranked by probability again ranks between BLAST and S-W with log-length normalization. There is a very good correlation between the relative entropy of the matrix (*H*, shown in parentheses) and Better/Worse/Total scores for BLAST: BLOSUM70 ($H = 0.84$) is better than BLOSUM62 ($H = 0.7$), which is better than BLOSUM55 ($H = 0.55$), which, in turn, is better than BL50 ($H = 0.48$). On the other hand, BLOSUM55 and BLOSUM62 matrices with intermediate *H* values are the best for S-W searches, while both higher (BLOSUM70, STR) and lower (BLOSUM50) entropy matrices perform worse.

Our results comparing different matrices are consistent with those of Henikoff and Henikoff (1992, 1993), who found that the BLOSUM62 matrix is the best overall and the STR matrix (Henikoff used a smaller structural alignment set than in this paper) is very close to BLOSUM62. It should be noted that the cut-off criterion in these papers was at 0.5% of top scoring unrelated to query database sequences (~125 unrelated sequences); this is a much higher number of false positives than either of the criteria we used. Also, we had the benefit of knowing that BLOSUM62 seems to be the matrix with the optimal relative entropy (Henikoff and Henikoff, 1993); therefore we included close matrices (BLOSUM55 and 70) in our comparison, rather than BLOSUM50 and −80, which are further away from BLOSUM62.

Figures 4 and 5 show how many random sequences have high S-W scores by chance and how many sequences that belong to the same SF as the query will

TABLE 4

Comparison of BLAST (BL), FASTA (FA), and S-W-FDF Based on the "Top-Random" Cut-Off for 502 Protein Superfamilies

Method	BLAST-STR	BLAST-bl170	BLAST-bl62	BLAST-bl55	BLAST-bl50	FASTA-bl62	SW-bl62	SW-STR	SW-bl70	SW-bl55	SW-bl50
BLAST-STR	0	50	49	42	46	70	103	98	103	100	98
BLAST-bl70	40	0	35	33	37	64	94	93	94	94	88
BLAST-bl62	51	48	0	27	30	71	***112***	104	106	112	107
BLAST-bl55	59	59	38	0	28	77	117	108	115	115	110
BLAST-bl50	71	74	53	40	0	88	126	116	126	125	117
FASTA-bl62	55	56	50	49	46	0	87	90	93	88	84
SW-bl62	36	37	**29**	27	26	29	0	42	32	28	25
SW-STR	40	43	46	44	43	51	60	0	42	55	55
SW-bl70	41	42	43	40	39	51	46	35	0	51	50
SW-bl55	30	30	28	25	26	28	23	36	39	0	21
SW-bl50	31	30	28	22	22	31	32	41	40	28	0
Better	454	469	399	349	343	560	800	763	790	796	755
Worse	759	672	768	826	936	698	311	479	438	286	305
Total	**−305**	**−203**	**−369**	**−477**	**−593**	**−138**	**489**	**284**	**352**	**510**	**450**

Note. The numbers in the table show for how many SF the method in the column performed better than the one in the row, according to "top-random" cut-off criterion. FASTA was run with bl62 matrix, gap opening/extension penalties "-10/-2"; scores were optimized and ranked by probability. For example, SW-bl62 was more accurate than BLAST in 112 SF of 502; the opposite was true for only 29 SF (both numbers are shown in boldface italics). The "Better" row shows how may times the method in the column performed better than the other 10 methods; "Worse" shows how many times it was worse. The "Total" shows the difference between the "Better" and the "Worse" numbers.

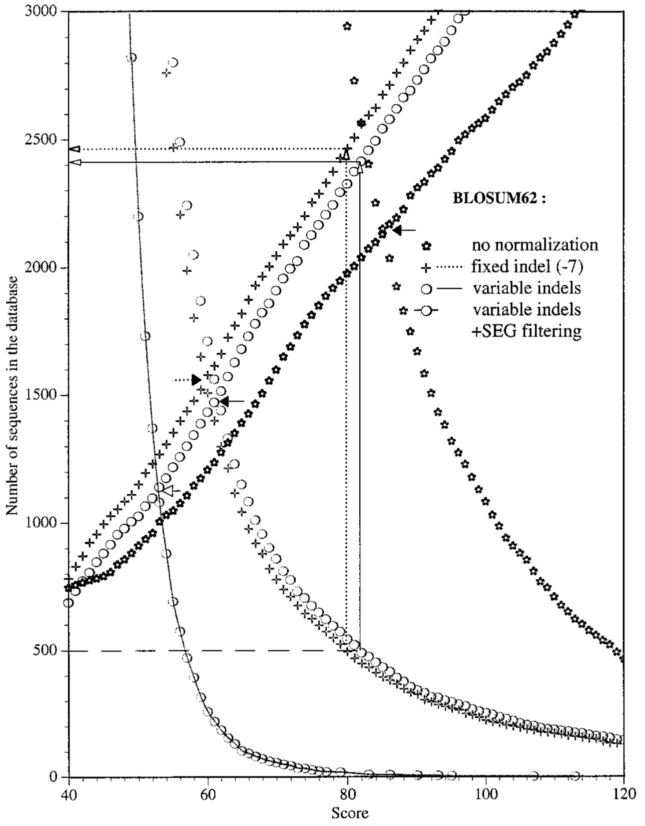

FIG. 4. The number of false negatives and false positives in S-W searches depending on the score. All data are cumulative for 502 queries from 502 different SFs. There are 12,511 database entries that belong to one and only one of the 502 protein SFs. The lines that go up when the score (*X* axis) increases show the results for 502 S-W searches with real protein sequences as queries, the number of SF members missed as a function of the cut-off score (SF members that have scores below the cut-off, i.e., false negatives). The descending lines (gray shaded symbols) show how many times S-W scores were higher than the score on the *X* axis for 502 S-W searches with random queries (false positives; all random sequences had lengths and amino acid composition identical to real SF queries). The short filled horizontal arrows show "missed@equivalence" points (the number of false positives is equal to the number of false negatives). The vertical and the long horizontal lines show the number of true sequences that were missed at 1-false cut-off. For example, for BLOSUM62 with variable indel penalties (open circles, solid lines with arrows) there would be 500 false positives (on average one random S-W score $\geqslant$82 per query or "1-false cut-off") with score 82 or higher. At the same cut-off of 82 approximately 2400 true SF members will be missed; i.e., 2400 of 12,511 SF members have normalized S-W scores less than 82 (false negatives). Similarly, for BLOSUM62 with a fixed indel penalty of "-7" (crosses, dotted lines) the 1-false cut-off would be ~80, and 2470 true SF members will score below. The continuous gray line with open circles shows the number of false positives for SEG-filtered random sequences.

be missed depending on the cut-off score. All graphs are cumulative for 502 protein SFs. Missed@equivalence points (intersection of ascending and descending curves are shown by short filled horizontal arrows) are close to ~1500 sequences for all five scoring matrices in Figs. 4 and 5. The only exception is "no normalization" curves that intersect much higher—the missed@equivalence point is 2150 (Fig. 4). Indeed, the normalization contributes significantly to the accuracy of the search by decreasing the number of high-scoring random sequences (Fig. 4). The reason that normalized missed@ equivalence points are similar for all five scoring matrices is that the "false" curves are very steep when the number of false positives is close to 1500. Therefore, we examined two other criteria to compare the selectivity of different scoring matrices.

First, we used the cut-off scores at 500 random high scores (on average, one random high score per query): 82 (BLOSUM62), 74 (STR), 73 (BLOSUM70), 114 (BLOSUM55), and 113 (BLOSUM50)—these points are the intersection of horizontal lines at 500 sequences and random (false positive, descending) curves for respective matrices. Figure 4 compares BLOSUM62 without normalization, with a "fixed" indel penalty of "-7" (crosses and dotted lines) and with variable indel penalties (circles and solid lines). Without normalization, the cut-off score is 119, and ~3500 SF members score below this cut-off. The cut-off score for fixed indel penalties is 80; ~2470 SF members score below 80. The cut-off for BLOSUM62 with variable indel scores is 82, and ~2400 members score below that; this is better by 70 sequences than the result for "fixed indels." There is a similar difference between fixed and variable indels when comparing missed@equivalence points. Using the "500 false positives" criterion only 2200 sequences are missed when the STR matrix is used (the best): 2350 for BLOSUM55, 2440 for BLOSUM70, and 2510 for BLOSUM50. The difference between the best and the worst matrix (310 SF members) is quite substantial.

Finally, we compared five different scoring matrices (Figs. 4 and 5) using the random (false positive) curves for SEG-filtered queries. At the cut-off equal to 100 random high scores, BLOSUM62 and BLOSUM55 are better than the other three matrices by about 150 (~1700 missed vs 1850).

DISCUSSION

The goal of this work was to carry out an unbiased (in terms of protein SF choice) and statistically relevant comparison of the accuracy of several similarity search methods and to use empirical score normalization to improve the selectivity of the searches. We included essentially all protein SFs from the PIR database. One can always find a few protein SFs where one algorithm or a specific scoring matrix performs better than others. We are interested, however, in identifying those methods and scoring matrices that consistently work better for the majority of proteins.

The BLAST algorithm does not produce the most accurate possible results for protein similarity searches: there is a significant chance that BLAST will miss a distant similarity between sequences that can be readily detected by S-W or FASTA.

In some sense, our comparison of BLAST to S-W indicates that there are more proteins whose evolution includes indels (and therefore S-W works better for these SFs, see Tables 1–4) than proteins that evolve without

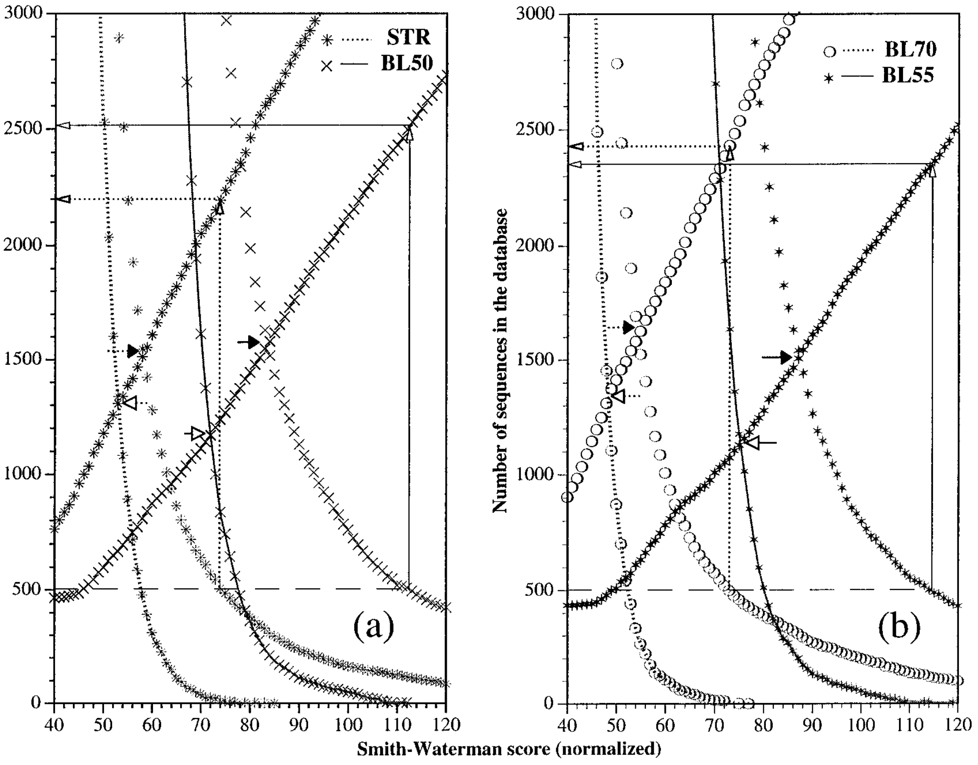

FIG. 5. The number of false negatives and false positives in S-W searches depending on the score. The data are shown for BLOSUM70, 55, 50, and STR matrices. All the data are analogous to those in Fig. 4; nonnormalized scores are not shown.

indels and have several regions of nongapped homology (called high scoring pairs). Indeed, the data we used to calculate the STR matrix show that there is, on average, one indel per 14 aligned amino acids at PAM = 140 (140 amino acid substitutions per 100 residues). In other words, the BLAST model of protein evolution is less accurate than the S-W model, which allows indels within a single high scoring match (for the majority of protein SFs).

The log-length normalization of Smith–Waterman scores (Eq. [4]) dramatically improves the accuracy of the searches (Tables 2 and 3; Fig. 4). As evident from Fig. 3a, normalization makes the distribution of random scores much narrower, and the number of sequences with high scores decreases (Fig. 3b, compare the "raw" and "normalized" curves). Figure 2a shows that the average S-W score for random sequences depends on the sequence length and that the width of the distribution, approximated by "standard deviation," is close to 7. After normalization, the average score is close to 25 (BLOSUM62) and does not depend on the query length. The standard deviation varies between 6 for short and 3.5 for longer query sequences (Fig. 2b). Also, the highest random S-W scores decrease in Fig. 2b compared to Fig. 2a. Figures 3 and 4 show that normalization dramatically decreases the number of high scoring random sequence pairs, but has little effect on the distribution of true S-W scores (Fig. 4).

The Karlin–Altschul normalization (Eq. [3], which describes nongapped alignments (it was not intended for use with S-W) did not decrease random S-W scores for long queries sufficiently so that they became independent of the query length. Both missed@equivalence (Table 2) and top-false cut-off tests showed worse results for K-A than log-length normalization (data not shown).

There are 13 random sequences (of 502) with the highest S-W score above 100 (these sequences also have "abnormally" high standard deviations; Fig. 2b). When SEG filtering is applied, these high scores go down, indicating that high S-W scores occur in the "simple" regions of sequences that are removed by filtering. The importance of SEG filtering is even more evident in Figs. 3–5. The high scoring range of random S-W scores shifts dramatically to the left (lower scores) when SEG filtering is applied.

It is well known that higher relative entropy matrices, such as BLOSUM70, are better at detecting short stretches of relatively strong similarities, while lower entropy matrices, like BLOSUM50, are more sensitive in cases of longer, but weaker similarities (Altschul, 1991). BLAST, by excluding gaps, tends to find one or several shorter regions of similarity, while S-W can extend the alignment across the indels. Not surprisingly, we found that lower relative entropy matrices (BLOSUM70 and STR—H = 0.84) are better for BLAST (Tables 3 and 4). Table 4 provides a very good illustration of this: every higher relative entropy BLAST worked better than the lower one (STR being between BLOSUM70 and BLOSUM62). For example, BLOSUM70 was better than BLOSUM50 in 74 SFs, whereas the reverse was true for only 37 SFs.

BLOSUM62/55 (intermediate values for H; relative entropy: $H = 0.7$ and $H = 0.55$) are usually the best choices for S-W searches, if one scoring matrix has to be chosen (Table 4; Figs. 4 and 5). The STR matrix, which has an origin different from that of the BLOSUM series, is the best in Table 3, and according to "1-false cut-off" (Fig. 5a). It should be noted that, depending on the protein SF, one matrix can be much better than others. For example, STR is dramatically better than any of the BLOSUMs for the plastocyanin SF (AZBR query; second line in Table 2), but BLOSUM50 is the best for the cytochrome C SF (CCHU query, third line in Table 2). The most accurate searches might, therefore, include several matrices with various relative entropies (e.g., STR, BLOSUM62, BLOSUM50) and combine the results (Altschul, 1993; Henikoff and Henikoff, 1993).

We propose here a new scoring method for indels—"variable gap penalties"—for S-W searches; this method has been implemented in the FDF. It should be noted that affine scoring (assigning a separate penalty for gap opening and extension) is generally a better evolutionary model for indels: according to missed@ equivalence criterion, S-W in software with log-length normalization is more accurate than S-W with variable gap scoring for 86 SFs in the 573 query set (the opposite was true for only 39 SFs). The variable indel penalties shown in Fig. 1 are consistent with different structural roles for amino acids: charged amino acids are more frequently found on the surface of proteins, and, one might expect, have a higher chance of being deleted/ inserted during evolution than do hydrophobic amino acids that are often buried inside the protein core. Proline, most frequently found in coil or turn structures, has the highest indel score.

Pearson has recently published a paper that compares the accuracy of BLAST, FASTA, and S-W protein similarity searches (Pearson, 1995). Several findings and conclusions from Pearson are similar to what we found: (1) Log-length normalization dramatically improves the selectivity of the searches. (2) Karlin–Altschul normalization (Pearson calls this method "regression scaled scores") improves the selectivity, but less than log-length normalization. (3) Both FASTA, in its accurate mode, and S-W are more sensitive than BLAST.

Pearson studied a wider range of scoring matrices (PAMs, Gonnet92, JTT series) and various combinations of indel penalties. He also included in his comparison methods that are known to be less accurate, e.g., FASTA ktup = 2 and BLAST ranked by score. We used a different equation for log normalization than Pearson, which makes S-W scores for random sequences independent of query lengths (Fig. 2b). Because of the latter, we are able to compare several scoring matrices "graphically" using normalized S-W scores, as shown in Figs. 4 and 5. Also, we include more protein superfamilies (502 or 573 in this manuscript compared to 67 in Pearson's paper) and twice as many entries in the database. In addition to missed@equivalence criterion we used top-random cut-off (Table 4) and 1-false (Figs. 4 and 5) criteria to compare different methods—both are not sensitive to imperfect superfamily classification in the PIR database and are more strict (allow a smaller number of unrelated sequences above cut-off) than missed@equivalence.

We have also provided to users a way to choose cut-off scores for several matrices (Figs. 4 and 5) and looked at the importance of query filtering. The data in Figs. 4 and 5 allow users to choose cut-off scores in real-life applications of S-W searches. The cut-off depends on: (1) the number of false scores permitted; (2) whether the query sequence has "simple" regions, as determined by SEG or XNU filtering programs; and (3) the choice of scoring matrix. The difference between BLOSUM55/ 50 and BLOSUM70/62/STR is due to the use of a different scaling factor. In practical terms, all matches for SEG-filtered query sequences with scores above 74 (STR), 70 (BLOSUM70), 80 (BLOSUM62), or 106 (BLOSUM55/50) have a very small likelihood of occurring by chance (approximately one in a million) and might indicate biologically important similarity.

REFERENCES

Altschul, S. (1991). Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.* **219:** 555–565.

Altschul, S. (1993). A protein alignment scoring system sensitive at all evolutionary distances. *J. Mol. Evol.* **36:** 290–300.

Altschul, S., Boguski, M., Gish, W., and Wootton, J. (1994). Issues in searching molecular sequence databases. *Nature Genet.* **6:** 119–129.

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

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

Benner, S., Cohen, M., and Gonnet, G. (1993). Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J. Mol. Biol.* **229:** 1065–1082.

Bork, P., Ouzounis, C., and Sander, C. (1994). From genome sequences to protein function. *Curr. Opin. Struct. Biol.* **4:** 393–403.

Claverie, J.-M., and States, D. (1993). Information enhancement methods for large scale sequence analysis. *Computers Chem.* **17:** 191–201.

Collins, J., Coulson, A., and Lyall, A. (1988). The significance of protein sequence similarities. *Comput. Appl. Biosci.* **4:** 67–71.

Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1978). A model of evolutionary change in proteins. *In* "Atlas of Protein Sequence and Structure 1978," pp. 345–352. Vol. 5, Suppl. 3. National Biomedical Res. Foundation, Washington, DC.

Fichant, G., and Burks, C. (1991). Identifying potential tRNA genes in genomic DNA sequences. *J. Mol. Biol* **220:** 659–671.

George, D., Barker, W., Mewes, H., Pfeiffer, F., and Tsugita, A. (1996). The PIR-international protein sequence database. *Nucleic Acids Res.* **24:** 17–20.

Henikoff, S., and Henikoff, J. (1993). Performance evaluation of amino acid substitution matrices. *Prot. Struct. Funct. Genet.* **17:** 49–61.

Henikoff, S., and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* **89:** 10915–10918.

Johnson, M., and Overington, J. (1993). A structural basis for sequence comparisons. An evaluation of scoring methodologies. *J. Mol. Biol.* **233:** 716–738.

Karlin, S., and Altschul, S. (1993). Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc. Natl. Acad. Sci. USA* **90:** 5873–5877.

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

Pearson, W. R. (1990). Rapid and sensitive sequence comparison with FASTP and FASTA. *In* "Methods in Enzymology" (R. F. Doolittle, Ed.), Vol. 183, pp. 63–98, Academic Press, New York.

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

Shpaer, E. (1996). GeneAssist: Smith–Waterman and other database similarity searches and identification of motifs. *In* "Methods in Molecular Biology: Sequence Data Analysis Guidebook" (S. Swindell, Ed.), Humana Press, Totowa, NJ.

Smith, T., Waterman, M., and Burks, C. (1985). The statistical distribution of nucleic acid similarities. *Nucleic Acids Res.* **13:** 645–656.

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

Vogt, G., and Argos, P. (1992). Searching for distantly related protein sequences in large databases by parallel processing on a transputer machine. *Comput. Appl. Biosci.* **8:** 49–55.

Waterman, M., and Vingron, M. (1994). Rapid and accurate estimates of statistical significance for sequence data base searches. *Proc. Natl. Acad. Sci. USA* **91:** 4625–4628.

Wootton, J. C., and Federhen, S. (1993). Statistics of local complexity in amino acid sequences and sequence databases. *Comput. Chem.* **17:** 149–163.

Wu, C., Berry, M., Fung, Y., and McLarty, J. (1993). Neural networks for molecular sequence classification. *Proc. Int. Sys. Mol. Bio.* **1:** 429–437.