Gapped BLAST and PSI-BLAST: a new generation of protein database search programs

Stephen F. Altschul, Thomas L. Madden, Alejando A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller and David J. Lipman

National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, USA; 1Laboratory of Genetic Disease Research, National Human Genome Research Institute, National Institutes of Health, Bethesda, MD 20892, USA and 2Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA

Received June 23, 1997; Revised and Accepted July 16, 1997

ABSTRACT

The BLAST programs are widely used tools for searching protein and DNA databases for sequence similarities. For protein comparisons, a variety of definitional, algorithmic and statistical refinements described here permits the execution time of the BLAST programs to be decreased substantially while enhancing their sensitivity to weak similarities. A new criterion for triggering the extension of word hits, combined with a new heuristic for generating gapped alignments, yields a gapped BLAST program that runs at approximately three times the speed of the original. In addition, a method is introduced for automatically combining statistically significant alignments produced by BLAST into a position-specific score matrix, and searching the database using this matrix. The resulting Position-Specific Iterated BLAST (PSI-BLAST) program runs at approximately the same speed per iteration as gapped BLAST, but in many cases is much more sensitive to weak but biologically relevant sequence similarities. PSI-BLAST is used to uncover several new and interesting members of the BRCT superfamily.

INTRODUCTION

Variations of the BLAST algorithm (1) have been incorporated into several popular programs for searching protein and DNA databases for sequence similarities. BLAST programs have been written to compare protein or DNA queries with protein or DNA databases in any combination, with DNA sequences often undergoing conceptual translation before any comparison is performed. We will use the blastp program, which compares protein queries to protein databases, as a prototype for BLAST; although the ideas presented extend immediately to other versions that involve the translation of a DNA query or database. Some of the refinements described are applicable as well to DNA–DNA comparison, but have yet to be implemented.

BLAST is a heuristic that attempts to optimize a specific similarity measure. It permits a tradeoff between speed and sensitivity, with the setting of a “threshold” parameter, T. A higher value of T yields greater speed, but also an increased probability of missing weak similarities. The BLAST program requires time proportional to the product of the lengths of the query sequence and the database searched. Since the rate of change in database size currently exceeds that of processor speeds, computers running BLAST are subjected to increasing load. However, the conjunction of several new algorithmic ideas allow a new version of BLAST to achieve improved sensitivity at substantially augmented speed. This paper describes three major refinements to BLAST.

(i) For increased speed, the criterion for extending word pairs has been modified. The original BLAST program seeks short word pairs whose aligned score is at least T. Each such “hit” is then extended, to test whether it is contained within a high-scoring alignment. For the default T value, this extension step consumes most of the processing time. The new “two-hit” method requires the existence of two non-overlapping word pairs on the same diagonal, and within a distance A of one another, before an extension is invoked. To achieve comparable sensitivity, the threshold parameter T must be lowered, yielding more hits than previously. However, because only a small fraction of these hits are extended, the average amount of computation required decreases.

(ii) The ability to generate gapped alignments has been added. The original BLAST program often finds several alignments involving a single database sequence which, when considered together, are statistically significant. Overlooking any one of these alignments can compromise the combined result. By introducing an algorithm for generating gapped alignments, it becomes necessary to find only one rather than all the ungapped alignments subsumed in a significant result. This allows the T parameter to be raised, increasing the speed of the initial database scan. The new gapped alignment algorithm uses dynamic programming to extend a central pair of aligned residues in both directions. For speed, earlier heuristic methods (2,3) confined the alignments produced to a predefined strip of the dynamic
programming path graph (4). Our approach considers only alignments that drop in score no more than \( X_0 \) below the best score yet seen. The algorithm is able thereby to adapt the region of the path graph it explores to the data.

(iii) BLAST searches may be iterated, with a position-specific score matrix generated from significant alignments found in round \( i \) used for round \( i + 1 \). Motif or profile search methods frequently are much more sensitive than pairwise comparison methods at detecting distant relationships. However, creating a set of motifs or a profile that describes a protein family, and searching a database with them, typically has involved running several different programs, with substantial user intervention at various stages. The BLAST algorithm is easily generalized to use an arbitrary position-specific score matrix in place of a query sequence and associated substitution matrix. Accordingly, we have automated the procedure of generating such a matrix from the output produced by a BLAST search, and adapted the BLAST algorithm to take this matrix as input. The resulting Position-Specific Iterated BLAST, or PSI-BLAST, program may not be as sensitive as the best available motif search programs, but its speed and ease of operation can bring the power of these methods into more common use.

After describing these refinements to BLAST in greater detail, we consider several biological examples for which the sensitivity and speed of the program are greatly enhanced.

**STATISTICAL PRELIMINARIES**

To analyze the BLAST algorithm and its refinements, we need first to review the statistics of high-scoring local alignments. BLAST employs a substitution matrix, which specifies a score \( s_{ij} \) for aligning each pair of amino acids \( i \) and \( j \). Given two sequences to compare, the original BLAST program seeks equal-length segments within each that, when aligned to one another without gaps, have maximal aggregate score. Not only the single best segment pair may be found, but also other **locally optimal** pairs (3.5-7), whose scores cannot be improved by extension or trimming. Such locally optimal alignments are called “high-scoring segment pairs” or HSPs.

For the sake of the statistical theory, we assume a simple protein model in which the amino acids occur randomly at all positions with background probabilities \( P_i \). We require that the expected score for two random amino acids \( \sum P_i P_j \) be negative. Given the \( P_i \) and \( s_{ij} \), the basic theory (8,9) yields two calculable parameters, \( \lambda \) and \( K \), which can be used to convert nominal HSP scores to normalized scores thereby rendering all scoring systems directly comparable from a statistical perspective. The normalized score \( S' \) for an HSP is given by the equation:

\[
S' = \frac{2S - \ln K}{\ln 2}
\]

where \( N = mn \) is the search space size (8-10). If a protein is compared to a whole database rather than a single sequence, \( n \) is the database length in residues. Equation 2 may be inverted to yield \( S = \log_2(N/E) \), the normalized score required to achieve a particular \( E \)-value. In a typical current database search, a protein of length 250 might be compared to a protein database of 50,000,000 total residues. To achieve a marginally significant \( E \)-value of 0.05, a normalized score of \( \approx 38 \) bits is necessary.

While the theory just outlined has not been proved for gapped local alignments and their associated scores, computational experiments strongly suggest that it remains valid (3,12-15). The statistical parameters \( \lambda \) and \( K \), however, are no longer supplied by theory but must be estimated using comparisons of either simulated or real but unrelated sequences. To distinguish between whether a given set of parameters \( \lambda \) and \( K \) refer to gapped or ungapped alignments, we use the subscripts \( g \) and \( u \), respectively.

When gaps are not allowed, a further important theorem states that within HSPs the aligned pair of letters \( (y_j) \) tends to occur with the “target frequency”:

\[
q_{iy} = P_i P_j ^{\lambda} \ln 2
\]

The \( \delta_i \) of equation 3 sum to 1; indeed, \( \lambda_i \) is calculated to be the unique positive number for which this is the case (8,9). The scores \( s_{ij} \) are optimal for detecting alignments with these particular target frequencies (8,10), and by inverting equation 3 to \( s_{ij} = \ln(q_{iy}/P_j^{\lambda})/\ln 2 \), scores may be chosen, with arbitrary scale, that correspond to any desired set of \( q_{iy} \). The popular PAM (16,17) and BLOSUM (8) substitution matrices are constructed with explicit use of this log-odds formula. No corresponding result has been established for gapped alignment scoring systems. However, if the gap costs used are sufficiently large, it is expected that the target frequencies observed in high-scoring local alignments of random sequences will not differ greatly from those for the no-gap case.

**REFINEMENT OF THE BASIC ALGORITHM: THE TWO-HIT METHOD**

The central idea of the BLAST algorithm is that a statistically significant alignment is likely to contain a high-scoring pair of aligned words. BLAST first scans the database for words (typically of length three for proteins) that score at least \( T \) when aligned with some word within the query sequence. Any aligned word pair satisfying this condition is called a hit. The second step of the algorithm checks whether each hit lies within an alignment with score sufficient to be reported. This is done by extending a hit in both directions, until the running alignment's score has dropped more than \( X \) below the maximum score yet attained. This extension step is computationally quite costly; with the \( T \) and \( X \) parameters necessary to attain reasonable sensitivity to weak alignments, the extension step typically accounts for \( >90 \% \) of BLAST's execution time. It is therefore desirable to reduce the number of extensions performed.

Our refined algorithm is based upon the observation that an HSP of interest is much longer than a single word pair, and may therefore entail multiple hits on the same diagonal and within a relatively short distance of one another. (The **diagonal** of a hit involving words starting at positions \( (x_1,y_1) \) of the database and query sequences may be defined as \( x_1 - y_1 \). The **distance** between two hits on the same diagonal is the difference between their first coordinates.) This signature may be used to locate HSPs more
Figure 1. The sensitivity of the two-hit and one-hit heuristics as a function of HSP score. Using the BLOSUM-62 amino acid substitution matrix (18), and the target frequencies \( q_j \) implied by equation 3 and the background amino acid frequencies \( P_j \) of Robinson and Robinson (20), 100,000 model HSPs were generated for each of the nominal scores 37–42, corresponding to normalized scores 15.9–45.1 bits. It was determined by inspection whether each HSP failed to contain two non-overlapping length-3 word pairs with nominal score at least 11, and within a distance 40 of one another. The corresponding probabilities of missing an HSP, using the two-hit heuristic with \( T = 11 \), and the one-hit heuristic with \( T = 13 \), are plotted as a function of normalized HSP score. The two-hit method is more sensitive for HSPs with score at least 33 bits.

Because we require two hits rather than one to invoke an extension, the threshold parameter \( T \) must be lowered to retain comparable sensitivity. The effect is that more single hits are found, but only a small fraction have an associated second hit on the same diagonal that triggers an extension. The great majority of hits may be dismissed after the minor calculation of looking up, for the appropriate diagonal, the coordinate of the most recent hit, checking whether it is within distance \( A \) of the current hit’s coordinate, and finally replacing the old with the new coordinate. Empirically, the computation saved by requiring fewer extensions more than offsets the extra computation required to process the larger number of hits.

To study the relative abilities of the one-hit and two-hit methods to detect HSPs of varying score, we model proteins using the background amino acid frequencies of Robinson and Robinson (20), and use the BLOSUM-62 substitution matrix (18) for sequence comparison. Given these \( P_j \) and \( q_j \), the statistical parameters for ungapped local alignments are calculated to be \( \lambda_n = 0.3176 \) and \( K_n = 0.134 \). Using equation 3 above, we may calculate the \( q_j \) for which the scoring system is optimized, and employ these target frequencies to generate model HSPs. Finally, we evaluate the sensitivity of the one-hit and two-hit BLAST heuristics using these HSPs.

The one-hit method will detect an HSP if somewhere contains a length-\( W \) word of score at least \( T \), for \( W = 3 \) and \( T = 13 \), Figure 1 shows the empirically estimated probability that an HSP is missed by this method, as a function of its normalized score. The two-hit method will detect an HSP if it contains two non-overlapping length-\( W \) words of score at least \( T \), with starting positions that differ by no more than \( A \) residues. For \( W = 3 \), \( T = 11 \) and \( A = 40 \), Figure 1 shows the estimated probability that an HSP is missed by this method, as a function of its normalized score. For HSPs with score at least 33 bits, the two-hit heuristic is more sensitive.

To analyze the relative speeds of the one-hit and two-hit methods, using the parameters studied above, we note that the two-hit method generates on average \( \sim 3.2 \) times as many hits, but only \( \sim 0.14 \) times as many hit extensions (Fig. 2). Because it takes approximately one ninth as long to decide whether a hit need be extended as actually to extend it, the hit-processing component of the two-hit method is approximately twice as rapid as the same component of the one-hit method.

**TRIGGERING THE GENERATION OF GAPPED ALIGNMENTS**

Figure 1 shows that even when using the original one-hit method with threshold parameter \( T = 13 \), there is generally no greater than a 4% chance of missing an HSP with score \( \geq 38 \) bits. While this would appear sufficient for most purposes, the one-hit default \( T \) parameter has typically been set as low as 11, yielding an execution time nearly three times that for \( T = 13 \). Why pay this price for what appears at best marginal gains in sensitivity? The
reason is that the original BLAST program treats gapped alignments implicitly by locating, in many cases, several distinct HSPs involving the same database sequence, and calculating a statistical assessment of the combined result (21,22). This means that two or more HSPs with scores well below 38 bits can, in combination, rise to statistical significance. If any one of these HSPs is missed, so may be the combined result.

The approach taken here allows BLAST to simultaneously produce gapped alignments and run significantly faster than previously. The central idea is to trigger a gapped extension for any HSP that exceeds a moderate score $S_0$, chosen so that more than about one extension is invoked per 50 database sequences. (By equation 2, for a typical-length protein query, $S_0$ should be set at $\sim 22$ bits.) A gapped extension takes much longer to execute than an ungapped extension, but by performing very few of them the fraction of the total running time they consume can be kept relatively low.

By seeking a single gapped alignment, rather than a collection of ungapped ones, only one of the constituent HSPs need be located for the combined result to be generated successfully. This means that we may tolerate a much higher chance of missing any single moderately scoring HSP. For example, consider a result involving two HSPs, each with the same probability $P$ of being missed at the hit-stage of the BLAST algorithm, and suppose that we desire to find the combined result with probability at least 0.95. The original algorithm, needing to find both HSPs, requires $2P - P^2 < 0.05$, or $P$ less than $0.025$. In contrast, the new algorithm requires only that $P^2 < 0.05$, and thus can tolerate $P$ as high as 0.22. This permits the $P$ parameter for the hit-stage of the algorithm to be raised substantially while retaining comparable sensitivity—from $T = 11$ to $T = 13$ for the one-hit heuristic. (The two-hit heuristic described above lowers $T$ back to 11.) As will be discussed below, the resulting increase in speed more than compensates for the extra time required for the rare gapped extension.

In summary, the new gapped BLAST algorithm requires two non-overlapping hits of score at least $T$, within a distance $A$ of one another, to invoke an ungapped extension of the second hit. If the HSP generated by the ungapped extension of the second hit, then a gapped extension is triggered. The resulting gapped alignment is reported only if it has an $E$-value low enough to be of interest. For example, in the pairwise comparison of Figure 2, the ungapped extension invoked by the hit pair on the left produces an HSP with score 23.6 bits (calculated using $\lambda_G$ and $K_G$). This is sufficient to trigger a gapped extension, which generates an alignment with score 32.4 bits (calculated using $\lambda_D$ and $K_D$) and an $E$-value of 0.5 (Fig. 3). The original BLAST program locates only the first and last ungapped segments of this alignment (Fig. 3c), and assigns them a combined $E$-value $>50$ times greater.
THE CONSTRUCTION AND STATISTICAL EVALUATION OF GAPPED LOCAL ALIGNMENTS

The standard dynamic programming algorithms for pairwise sequence alignment perform a fixed amount of computation per cell of a path graph, whose dimensions are the lengths of the two sequences being compared (23–25). In order to gain speed, database search algorithms such as Fasta (2) and an earlier gapped version of BLAST (3) sacrifice rigor by confining the dynamic programming to a banded section of the full path graph (4), chosen to include a region of already identified similarity. One problem with this approach is that the optimal gapped alignment may stray beyond the confines of the band explored. As the width of the band is increased to reduce this possibility, the speed advantage of the algorithm is vitiated.

We have accordingly taken a different heuristic approach to constructing gapped local alignments, which is a simple generalization of BLAST's method for constructing HSPs. The central idea is to consider only cells for which the optimal local alignment score falls no more than \( X_g \) below the best alignment score yet found. Starting from a single aligned pair of residues, called the seed, the dynamic programming proceeds both forward and backward through the path graph (Zhang, Zhang et al., manuscript in preparation) (Figs. 3a and 4). The advantage of this approach is that the region of the path graph explored adapts to the alignment being constructed. The alignment can wander arbitrarily many diagonals away from the seed, but the number of sequences expanded on each row tends to remain limited, and may even shrink to zero before a boundary of the path graph is encountered (Fig. 4). The \( X_g \) parameter serves a similar function to the band-width parameter of the earlier heuristic, but the region of the path graph it implicitly specifies is explored in general more productively chosen.

An important element for this heuristic is the intelligent choice of a seed. Given an HSP whose score is sufficiently high that it triggers a gapped extension, how does one choose a residue pair to force into alignment? While more sophisticated approaches are possible, the simple procedure we have implemented is to locate, along the HSP, the length-11 segment with highest alignment score, and use its central residue pair as the seed. If the HSP itself is shorter than 11, a central residue pair is chosen. For example, the first ungapped region in the alignment of Figure 3c constitutes the HSP that triggered the alignment. The highest-scoring length-11 segment of this HSP aligns leghemoglobin residues 55–66 with \( \beta \)-globin residues 57–67. Thus the \( \beta \)-globin residues at respective positions 60 and 62 are used as the seed for the gapped extension illustrated in Figure 3a. As discussed in the performance evaluation section below, this procedure is extremely good at selecting seeds that in fact participate in an optimal alignment.

Most gapped extensions are triggered by chance similarities, and are therefore likely to be of limited extent, as illustrated in Figure 4. The reverse extension in this example explores \( \sim 2000 \) path graph cells, so that a typical two-way gapped extension that does not encounter the end of either sequence is expected to involve \( \sim 4000 \) cells. Because \( S_p \) is set so that a gapped extension is invoked less than once per 50 database sequences, fewer than 80 cells need be explored per database sequence.

The execution time required for a gapped extension is \( \sim 500 \) times faster than for an ungapped extension. However, by triggering gapped extensions in the manner described, while simultaneously raising \( T \) for the single-hit version of BLAST from 11 to 13, approximately one gapped extension is invoked for every 4000 ungapped extensions avoided. Because the number of ungapped extensions is reduced by about two thirds, the total time spent on the extension stage of BLAST is cut by well over half. Of course, the two-hit strategy described above reduces the time needed for the ungapped extensions still further. Once program overhead is accounted for, the net speedup is a factor of about three.

For any alignment actually reported, a gapped extension that records "traceback" information (25) needs to be executed. To increase BLAST's accuracy in producing optimal local alignments, these gapped extensions use by default a substantially larger \( X_g \) parameter than employed during the program's search stage.

Table 1. Relative times spent by the original and gapped BLAST programs on various algorithmic stages

<table>
<thead>
<tr>
<th></th>
<th>Overhead: database scanning, output, etc.</th>
<th>Calculating whether hits quality for ungapped extension</th>
<th>Ungapped extensions</th>
<th>Gapped extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original BLAST</td>
<td>8 (8%)</td>
<td>12 (37%)</td>
<td>92 (92%)</td>
<td></td>
</tr>
<tr>
<td>Gapped BLAST</td>
<td>8 (24%)</td>
<td>12 (37%)</td>
<td>5 (15%)</td>
<td>8 (24%)</td>
</tr>
</tbody>
</table>
The times required by various steps of the BLAST algorithm vary substantially from one query and one database to another. Table 1 shows typical relative times spent by the original and the gapped BLAST programs on various algorithmic stages. The 'original BLAST program' is represented, here and below, by a variant form of blastp version 1.49, modified so that it uses the same edge-effect correction (22) and background amino acid frequencies as the 'gapped BLAST'. The times represent the average for three different queries, with the time for the original BLAST program normalized in each instance to 100 units.

More concretely, to search SWISS-PROT (26), release 34 (59,576 sequences; 21,219,450 residues), with the length-567 influenza A virus hemagglutinin precursor (27) as query, the original BLAST program requires 455.8 s, and the gapped BLAST program 15.8 s. This timing experiment, and others referred to below, was run on one 200 MHz R10000 cpu processor of a lightly loaded SGI Power Challenge XL computer with 2.5 Gbytes of RAM. This machine runs the operating system IRIX version 6.2, which is an implementation of UNIX. We used the standard SGI C compiler, with the -O flag for optimization, to compile all versions of the programs. The times reported are the user times given by the time command and are for the better of two identical runs.

A closely related type of gapped extension routine to that used here was developed by G. Myers during the evaluation of the original BLAST algorithm. It was not included in the publicly distributed version primarily because the then current strategy of extending every hit decreased the algorithm's speed unduly for the relatively small gain in sensitivity realized (1).

As discussed above, the statistical significance of gapped alignments may be evaluated using the two statistical parameters $\lambda_{g}$ and $K_{g}$. The current version of the Fasta program (2) estimates these parameters on each run, by analyzing the distribution of alignment scores produced by all the sequences in the database. BLAST gains speed by producing alignments for only the few database sequences likely to be related to the query, and therefore does not have the option of estimating $\lambda_{g}$ and $K_{g}$ on the fly. Instead, it uses estimates of these parameters produced beforehand by random simulation (3). A drawback of this approach is that the program may not accept an arbitrary scoring system, for which no simulation has been performed, and still produce accurate estimates of statistical significance. The original BLAST programs, in contrast, because they dealt only with ungapped local alignments, could derive $\lambda_{u}$ and $K_{u}$ from theory for any scoring matrix (8,9).

**Iterated Application of BLAST to Position-Specific Score Matrices**

Database searches using position-specific score matrices, also called profiles or motifs, often are much better able to detect weak relationships than are database searches that use a simple sequence as query (28–38). Employing these methods, however, frequently has involved the use of several different programs and a fair degree of expertise. Accordingly, to render the power of motif searches more readily available, we have written a procedure to construct a position-specific score matrix automatically from the output of a BLAST run, and modified BLAST to operate using such a matrix in the place of a simple query. The resulting PSIBLAST program often is substantially more sensitive than the corresponding BLAST program, but for each iteration takes little more than the same time to run. In related work, Henikoff and Henikoff (39) have described how, short of modifying BLAST so that it may operate on a position-specific score matrix, a single artificial sequence that approximates such a matrix may be used as a query with the original BLAST programs.

The construction of a position-specific score matrix is a multi-stage process, and at each stage a choice must be made among a number of alternative routes. We have been guided by the goals of automatic operation, speed of execution, and general simplicity. The issues discussed below are: (i) general architecture of the score matrix; (ii) construction of the multiple alignment from which the matrix is derived; (iii) weights for sequences within the multiple alignment, and evaluation of the effective number of independent observations it constitutes; (iv) estimation of target frequencies, and the construction of matrix scores; (v) applying BLAST to a position-specific matrix, and the statistical evaluation of search results. We do not claim our current implementation is optimal, and it is likely that over time some of its details will change.

**Score matrix architecture**

The alignment of a simple sequence with a pattern embodied by a position-specific score matrix is almost completely analogous to the alignment of two simple sequences. The only real difference is that the score for aligning a letter with a pattern position is given by the matrix itself, rather than with reference to a substitution matrix. For proteins, a query of length $L$ and a substitution matrix of dimension $20 \times 20$ are replaced by a position-specific matrix of dimension $L \times 20$. Position-specific gap costs may be defined as well (34, 40). As with pairwise sequence comparison, one may choose among finding the best global alignment of the matrix and the simple sequence (23), finding the best alignment of the complete matrix with a segment of the sequence (41), and finding the best local alignment of the matrix and sequence (24).

Position-specific protein score matrices draw their power from two sources. The first is improved estimation of the probabilities with which amino acids occur at various pattern positions, leading to a more sensitive scoring system. The second is relatively precise definition of the boundaries of important motifs. By demanding the complete alignment of one or more motifs, rather than seeking an arbitrary local alignment, the size of the search space may be greatly reduced, thereby lowering the level of random noise. Unfortunately, there are many obstacles to automating well the delineation of a set of motifs from the output of a database search. The query sequence may contain a variety of different domains, and share different subsets of them with different proteins in the database. Furthermore, defining the proper extent of even a single motif may be challenging (42).

Accordingly, we have chosen to forgo the potential advantages of restricting the length of our derived matrices, and then demanding that they be completely aligned with segments of database sequences (41). Instead, each matrix we construct has length precisely equal to that of the original query sequence. When searching the database with such a matrix, we seek local alignments, in full analogy to those sought by BLAST when used for straightforward sequence-sequence comparison. Finally, we do not attempt to derive position-specific gap scores for use with our position-specific substitution scores. Instead, in each iteration
of PSI-BLAST, we employ the same gap scores that are used in the first, simple BLAST run. Our reasons are that there is no good theory for deriving gap costs from a multiple alignment and that, as will be discussed below, by eschewing position-specific gap costs we can make a reasonable estimate of the statistical significance of the resulting local alignments.

### Multiple alignment construction

To produce a multiple alignment from the BLAST output, we simply collect all database sequence segments that have been aligned to the query with E-value below a threshold, by default set to 0.01. The query is used as a master, or template, for constructing a multiple alignment. Any row (i.e., database sequence segment) identical to the query segment with which it aligns is purged, and only one copy is retained of any rows that are >98% identical to one another. Pairwise alignment columns that involve gap characters inserted into the query are simply ignored, so that M has exactly the same length as the query. Because we are dealing with local alignments, the columns of M may involve varying numbers of sequences, and many columns may include nothing but the query. We make no attempt to improve M by comparing database sequences with one another, or by any other true multiple alignment procedure.

As will be discussed, the matrix scores constructed for a given alignment column should depend not only upon the residues appearing there, but upon those in other columns as well. To make this dependency easy to formulate, however, we need to prune our raw multiple alignment M to a simpler "reduced" one. This pruning is done independently for each column, so the reduced multiple alignment M is in general vary from one column C to the next. To construct M, we first specify the set R of sequences it includes to be exactly those that contribute to a residue to column C. We then define the columns of M to be just those columns of M that include all the sequences of R are represented. By construction, the reduced multiple alignment M has residues or gap characters in every row and column (Fig. 5a), and is therefore amenable to the various manipulations described below.

### Sequence weights

When constructing a score matrix from a multiple alignment, it is a mistake to give all sequences the alignment equal weight. A large set of closely related sequences carries little more information than a single member, but its size alone may allow it easily to "outvote" a small number of more divergent sequences. One way past this difficulty is to assign weights to the various sequences, with those having many close relatives receiving smaller weight. The many sequence weighting methods that have been proposed (43–51) often produce roughly equivalent results. Because of its speed and simplicity, we have implemented a modified version of the sequence weighting method of Henikoff and Henikoff (47). Gap characters are treated as a 21st distinct character, and any columns consisting of identical residues are ignored in calculating weights. In speaking of a column's observed residue frequencies $f_i$, we shall henceforth mean its weighted rather its raw frequencies.

In constructing matrix scores, not only a column's observed residue frequencies are important, but also the effective number of independent observations it constitutes: a column consisting of a single valine and a single isoleucine carries different information than one consisting of five independently occurring instances of each. Accordingly, we need to estimate the relative number $N_C$ of independent observations constituted by the alignment $M_C$. A simple count of the number of sequences in $M_C$ is a poor measure, for 10 identical sequences imply fewer independent observations than do 10 divergent ones. We thus propose as a simple first estimate for $N_C$ the mean number of different residue types, including gap characters, observed in the various columns of $M_C$. This estimate is clearly not ideal as it saturates at 21 no matter how many independent sequences are contained in $M_C$. However, for the data we are likely to encounter, $N_C$ is typically much
smaller than 21, and therefore perhaps a good enough approximation for our purposes. As will be seen, it is not the absolute value of \( N_C \) that is important, but rather its relative value from one column to another. \( N_C \) is essentially the same measure of alignment variability as that proposed by Henikoff and Henikoff (52) for use in a different manner.

**Target frequency estimation**

Given a multiple alignment, many methods for generating score matrices have been advanced (28–37, 42, 52–54). The prescription with perhaps the best theoretical foundation is that the scores for a specific pattern position be of the form \( \log (Q_i/P_i) \), where \( Q_i \) is the estimated probability for residue \( i \) to be found in that column (29, 30, 32, 33, 36, 37, 42, 52–54). This leaves open the question how best to estimate the \( Q_i \).

Given a multiple alignment involving a large number of independent sequences, the estimate of \( Q_i \) for a specific column should converge simply to the observed frequency of residue \( i \) in that column. However, in addition to the sequence weighting issues discussed above, factors that complicate estimating the \( Q_i \) include small sample size (30), and prior knowledge of relationships among the residues (16, 37, 53). Various studies suggest that the best currently available method for estimating the \( Q_i \) is that of Dirichlet mixtures (52–56). However, because it often performs nearly as well (52), and due to its relative simplicity, we have implemented the data-dependent pseudocount method introduced by Tatusov et al. (37). This method uses the prior knowledge of amino acid relationships embodied in the substitution matrix \( S_9 \) to generate residue pseudocount frequencies \( g_i \) which are averaged with the observed frequencies \( f_i \) to estimate the \( Q_i \).

Specifically, for a given column \( C \), we construct pseudocount frequencies \( g_i \) using the formula:

\[
g_i = \frac{\sum_j f_j q_{ij}}{\sum_j f_j} \tag{4}
\]

where the \( q_{ij} \) are the target frequencies implicit in the substitution matrix, and given by equation 3. Intuitively, those residues favored by the substitution matrix to align with the residues actually observed receive high pseudocount frequencies. We then estimate \( Q_i \) by:

\[
Q_i = \frac{\alpha f_i + \beta g_i}{\alpha + \beta} \tag{5}
\]

where \( \alpha \) and \( \beta \) are the relative weights given to observed and pseudocount residue frequencies. So that the scores we construct will reduce to \( S_9 \) in columns where nothing has been aligned to the query sequence, we let \( \alpha = N_C - 1 \). \( \beta \) remains an arbitrary pseudocount parameter; the larger its value, the greater the emphasis given to prior knowledge of residue relationships via a visible observed residue frequencies. We have found empirically that, in conjunction with our method for calculating \( \alpha \), a reasonably good setting for \( \beta \) is 10.

**BLAST applied to position-specific score matrices**

The initial step of the BLAST algorithm is the construction of a list of words that align to query words with score at least \( T \). Only minor modifications to the code are necessary for this step to be performed on a query consisting of a position-specific matrix rather than a simple sequence. The same holds for the ungapped and gapped extension steps of BLAST. One important issue is whether key parameters such as \( T, \lambda_{AP} \) used at various heuristic stages of the algorithm and tuned to simple sequence comparison, can be applied unchanged to position-specific matrices without degrading unduly either the speed or sensitivity of database searches. We approach this problem by ensuring that the scale \( \lambda_{AP} \) of the matrix scores produced internally by PSI-BLAST corresponds to that of the substitution matrix \( S_9 \). In other words, we calculate the scores for a column of the matrix as \( \log (Q_i/P_i)/\lambda_{AP} \).

There is no analytic theory with which to estimate the statistical significance of a gapped alignment of a position-specific score matrix and a simple sequence. However, one may hypothesize that for a score matrix constructed to the same scale as \( S_9 \), a given set of gap costs should produce the same gapped alignment scale parameter \( \lambda_{AP} \) as for \( S_9 \). This would be convenient, because then PSI-BLAST could estimate statistical significance without incurring the substantial time required to estimate \( T, \lambda_{AP} \) and \( K_2 \) by random simulation. To test this hypothesis, we performed a number of statistical tests on PSI-BLAST generated score matrices, scaled to have \( \lambda_{AP} = 0.3176 \), the value applicable to previously published BLOSUM-62 simulations (3).

First, we searched SWISS-PROT using as query the length 567 influenza A virus hemagglutinin precursor (27), and captured the score matrix constructed by PSI-BLAST from the 128 local alignments with \( E \)-value \( < 0.01 \). We then compared this matrix to 10,000 random sequences of length 567, generated using the background amino acid frequencies of Robinson and Robinson (20). A gap of length 1 was charged a cost of 10 \( + K \). Counts of the optimal local alignment scores, calculated using an appropriately modified version of the Smith–Waterman algorithm (24), are plotted in Figure 6. Also shown is the best fitting extreme value distribution (3, 15) which, using the edge-effect correction described by Altschul and Gish (3), has statistical parameters \( \lambda_{AP} = 0.251 \) and \( K_2 = 0.031 \). It is apparent that the distribution fits the random trials reasonably well; a \( \chi^2 \) goodness-of-fit test with 34 degrees of freedom has value 41.8 which is lower than one would expect 20% of the time even were the theory precisely valid. This supports the idea that the statistical theory described above applies to local alignments of position-specific score matrices and simple sequences. Furthermore, the estimate \( \lambda_{AP} = 0.251 \pm 0.003 \) agrees to within experimental error with the value 0.255 previously published for these gap costs (3). Similar agreement was obtained with a number of other protein sequences as initial query (results not shown), and in all cases the much less important \( K_2 \) parameter could be estimated accurately as well. In general, values of \( \lambda_{AP} \) for the comparison of position-specific score matrices with simple sequences appear to differ by <2% from the values for simple pairwise sequence comparison. Using these precomputed values for \( \lambda_{AP} \) should thus entail an error of less than one bit for PSI-BLAST scores <50 bits, corresponding to a factor of less than two in the estimation of statistical significance.

**Performance Evaluation**

To test more directly the statistics used by PSI-BLAST, we compared query sequences from 11 large and well characterized protein families to the SWISS-PROT databases and then ran the position-specific score matrices generated against a shuffled version of the same database. For each query, we recorded the
Figure 6. The distribution of optimal local alignment scores from the comparison of a position-specific score matrix with 10,000 random protein sequences. The score matrix was constructed by PSI-BLAST from the 128 local alignments with E-value ≤ 0.01 found in a search of SWISS-PROT using as query the length-567 influenza A virus hemagglutinin precursor (27) (SWISSPROT accession no. P03455). The random sequences, each of length 567, were generated using the amino acid frequencies of Robinson and Robinson (20). Optimal local alignment scores were calculated using the position-specific matrix in conjunction with 10 × gap penalties. The extreme value distribution that best fits the data (31,5) is plotted. A χ² goodness-of-fit test with 34 degrees of freedom has value 41.8, corresponding to a P-value of 0.20.

The lowest E-value found, as well as the number of shuffled sequences yielding E-values ≤ 1 and 10. For comparison, we performed the identical shuffled-database test on the gapped and original versions of BLAST. To reduce the probability that high-scoring alignments were missed due to the heuristic nature of the algorithms, we performed these tests with T = 9 rather than the default value of 11. The results are given in Table 2. For the 11 queries, the median of the low PSI-BLAST E-values was 0.87, which corresponds to a median P-value of 0.58 (8.9). The mean numbers of shuffled database sequences with E-values ≤ 1 and 10 were 1.0 and 8.7, respectively, within 20% of the expected values of 1.0 and 10.0. The equivalent tests for the ungapped and gapped versions of BLAST also yielded results that diverged from theory by <50%.

The ability to estimate with reasonable accuracy the significance of gapped local matrix–sequence alignments permits us to automate the construction of position-specific score matrices during multiple iterations of the PSI-BLAST program. After each iteration, we generate a new multiple alignment simply by collecting those alignments with E-value lower than a defined threshold. An interactive version of PSI-BLAST allows the user to override either the inclusion or exclusion of specific local alignments. Once a given database sequence has been used in the generation of a position-specific score matrix, low E-values for this sequence are virtually guaranteed in future iterations, for the sequence is to a certain extent being compared with itself. The biological relevance of PSI-BLAST output thus depends critically on avoiding the inappropriate inclusion of sequences in the local alignment constructed. Specifically, the utility of the score matrix produced is immediately vitiated by the inclusion of any alignment involving a region of highly biased amino acid composition (57,58).

Table 2. The comparison of various query sequences with a shuffled version of SWISS-PROT

<table>
<thead>
<tr>
<th>Protein family</th>
<th>SWISS-PROT accession no. of query</th>
<th>Original BLAST E-value</th>
<th>No. of seqs with E-value ≤ 1</th>
<th>No. of seqs with E-value ≤ 10</th>
<th>Gapped BLAST E-value</th>
<th>No. of seqs with E-value ≤ 1</th>
<th>No. of seqs with E-value ≤ 10</th>
<th>PSI-BLAST E-value</th>
<th>No. of seqs with E-value ≤ 1</th>
<th>No. of seqs with E-value ≤ 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serine protease</td>
<td>P00762</td>
<td>0.86</td>
<td>1</td>
<td>7</td>
<td>3.0</td>
<td>0</td>
<td>4</td>
<td>0.94</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Serine protease inhibitor</td>
<td>P01008</td>
<td>3.9</td>
<td>0</td>
<td>4</td>
<td>0.078</td>
<td>1</td>
<td>9</td>
<td>1.5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Ras</td>
<td>P01111</td>
<td>3.4</td>
<td>0</td>
<td>8</td>
<td>3.4</td>
<td>0</td>
<td>7</td>
<td>1.1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Globin</td>
<td>P02232</td>
<td>2.4</td>
<td>0</td>
<td>7</td>
<td>2.8</td>
<td>0</td>
<td>5</td>
<td>8.2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Hemagglutinin</td>
<td>P03455</td>
<td>0.11</td>
<td>2</td>
<td>11</td>
<td>0.46</td>
<td>3</td>
<td>16</td>
<td>0.87</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Interferon α</td>
<td>P05013</td>
<td>2.4</td>
<td>0</td>
<td>6</td>
<td>0.27</td>
<td>2</td>
<td>4</td>
<td>0.11</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>Alcohol dehydrogenase</td>
<td>P07327</td>
<td>1.5</td>
<td>0</td>
<td>2</td>
<td>0.80</td>
<td>1</td>
<td>5</td>
<td>1.5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Histocompatibility antigen</td>
<td>P10318</td>
<td>0.91</td>
<td>1</td>
<td>10</td>
<td>0.13</td>
<td>1</td>
<td>7</td>
<td>0.0051</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Cytochrome P450</td>
<td>P10635</td>
<td>0.94</td>
<td>2</td>
<td>5</td>
<td>8.5</td>
<td>0</td>
<td>3</td>
<td>0.46</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>Glutathione transferase</td>
<td>P14942</td>
<td>1.0</td>
<td>1</td>
<td>10</td>
<td>3.3</td>
<td>0</td>
<td>3</td>
<td>0.30</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>H+-transporting ATP synthase</td>
<td>P20705</td>
<td>0.012</td>
<td>1</td>
<td>8</td>
<td>0.26</td>
<td>2</td>
<td>14</td>
<td>0.79</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td><strong>Average (mean)</strong></td>
<td><strong>1.0</strong></td>
<td><strong>0.7</strong></td>
<td><strong>68</strong></td>
<td><strong>0.80</strong></td>
<td><strong>0.9</strong></td>
<td><strong>7.0</strong></td>
<td><strong>0.87</strong></td>
<td><strong>1.0</strong></td>
<td><strong>8.7</strong></td>
<td></td>
</tr>
</tbody>
</table>

The original and gapped BLAST comparisons use BLOSUM-62 substitution scores (18). All three programs use threshold T parameter set to 9, but the gapped BLAST and PSI-BLAST programs use the two-hit method to trigger ungapped extensions. The original BLAST program has the X dropoff parameter set to nominal score 23. The gapped BLAST and PSI-BLAST comparisons change gaps of length k a cost of 10 + k. They have X₀ set to 16, and X₁ set to 40 for the database search stage, and to 67 for the output stage of the algorithms. Gapped alignments are triggered by a score corresponding to ~22 bits. For PSI-BLAST, the query is first compared to the SWISS-PROT database, and the position-specific score matrix generated is then compared to a shuffled version of SWISS-PROT. The median is used for the average of the low E-values, and the mean otherwise.
Table 3. The number of SWISS-PROT sequences yielding alignments with $E$-value $\leq 0.01$, and relative running times, for Smith–Waterman and various versions of BLAST.

<table>
<thead>
<tr>
<th>Protein family</th>
<th>Query</th>
<th>Smith–Waterman</th>
<th>Original BLAST</th>
<th>Gapped BLAST</th>
<th>PSI-BLAST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serine protease</td>
<td>F00702</td>
<td>273</td>
<td>273</td>
<td>275</td>
<td>106</td>
</tr>
<tr>
<td>Serine protease inhibitor</td>
<td>F10100</td>
<td>106</td>
<td>106</td>
<td>106</td>
<td>108</td>
</tr>
<tr>
<td>Ras</td>
<td>P01111</td>
<td>255</td>
<td>249</td>
<td>252</td>
<td>252</td>
</tr>
<tr>
<td>Globin</td>
<td>P02232</td>
<td>28</td>
<td>26</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Hemagglutinin</td>
<td>P03435</td>
<td>128</td>
<td>114</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>Interferon α</td>
<td>P05013</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>Alcohol dehydrogenase</td>
<td>P07327</td>
<td>138</td>
<td>128</td>
<td>137</td>
<td>160</td>
</tr>
<tr>
<td>Histocompatibility antigen</td>
<td>P10318</td>
<td>262</td>
<td>241</td>
<td>261</td>
<td>261</td>
</tr>
<tr>
<td>Cytochrome P450</td>
<td>P10635</td>
<td>211</td>
<td>197</td>
<td>211</td>
<td>224</td>
</tr>
<tr>
<td>Glutathione transferase</td>
<td>P14942</td>
<td>83</td>
<td>79</td>
<td>81</td>
<td>81</td>
</tr>
<tr>
<td>H^+ transporting ATP synthase</td>
<td>P20786</td>
<td>198</td>
<td>191</td>
<td>197</td>
<td>207</td>
</tr>
<tr>
<td>Normalized running time</td>
<td></td>
<td>36</td>
<td>1.0</td>
<td>0.34</td>
<td>0.87</td>
</tr>
</tbody>
</table>

To score and evaluate the significance of the alignments found, the original BLAST program uses BLOSUM-62 substitution scores [18] and gap statistics [21,22]. The Smith–Waterman and gapped BLAST programs use BLOSUM-62 substitution scores, $10 + k$ gap costs, and the statistics of equations 1 and 2, in conjunction with the experimentally determined parameters $λ_g = 0.255$ and $λ_e = 0.055(3)$. PSI-BLAST uses the same gap costs and $λ_g$, but applied to the position-specific score matrix constructed from the output of the gapped BLAST run. Only one PSI-BLAST iteration is executed. All three BLAST programs use the same parameters settings as in Table 2, except that $r$ is set to 11. Normalized running times are the mean ratio of program running time to that for the original BLAST. The time for PSI-BLAST includes the time for the initial BLAST search.

To compare the performance of the new gapped version of BLAST and its PSI-BLAST extension to that of the Smith–Waterman algorithm (24) and the original ungapped BLAST algorithm, we employed the same 11 query sequences that were used above to investigate the accuracy of PSI-BLAST statistics. Because, as shown, these statistics are quite accurate, we may use the number of statistically significant sequences found in a database search as a reasonable measure of algorithm sensitivity. We employed the search program, version 2.0.x.54, from the FastA package (2) as our implementation of the Smith–Waterman algorithm. Using each of the 11 queries, we searched SWISS-PROT with each of the four programs. We show in Table 3 the numbers of sequences found with $E$ value $\leq 0.01$, as well as the average ratio of running time to that for the original BLAST program. Based upon SWISS-PROT annotation, all sequences recorded in Table 3 appear to be true family members, with the exception of one of the lowest-scoring alignments found by Smith–Waterman when applied to the histocompatibility antigen query, and the lowest-scoring alignment found by the original BLAST applied to the hemagglutinin query. While some alignments involve hypothetical proteins, the pattern of conserved residues in all such cases suggests a true positive.

As can be seen, the gapped BLAST program runs on average three times faster than the original, and in all but one case examined finds a greater number of statistically significant alignments. It runs >100 times faster than Smith–Waterman, but for the combined 11 queries misses only eight of the 1739 significant similarities found by the rigorous algorithm. Of these eight, only one has an $E$-value $<0.001$, and another appears to be a random as opposed to a biologically meaningful similarity. The scores produced by gapped BLAST for the 1731 similarities it finds differ from those produced by the Smith–Waterman algorithm in only two instances. The discrepancy arises in both cases from an $X_e$ parameter that is too low rather than from an incorrect choice of seed. Thus, despite its simplicity, the seed-selection heuristic is extremely accurate.

A search that includes a single PSI-BLAST iteration still runs faster than the original BLAST, and 40 times faster than Smith–Waterman, but in many cases is much more sensitive. It finds every true positive returned by Smith–Waterman, but frequently many others as well. Here only a single PSI-BLAST iteration has been considered, but as will be seen below, multiple iterations can yield even better results. Furthermore, we have found PSI-BLAST to perform better on searches of the non-redundant protein sequence database maintained by the NCBI (59) than on searches of SWISS-PROT, because of the greater number of significant similarities that are found by the initial BLAST run.

For the particular examples in Table 3, the PSI-BLAST iteration takes noticeably longer than the gapped BLAST iteration, due primarily to the time needed to construct the position-specific score matrix from the large number of significant local alignments found by BLAST. For queries that return a small number of significant alignments, each PSI-BLAST iteration requires more nearly the same time as BLAST.

**PSI-BLAST EXAMPLES**

In many instances, PSI-BLAST is able to automatically uncover biologically interesting similarities that elude simple database searches. Multiple iterations of PSI-BLAST are sometimes required to recognize the more distantly related protein family members. We here consider two representative cases in greater detail.

**HIT proteins**

Holm and Sander (60) describe how a comparison of three-dimensional structures is able to identify significant similarity between histidine triad (HIT) proteins and galactose-1-phosphate uridylyltransferase (Galt) proteins. Indeed, using the human HIT protein (61) as query, a BLAST search of SWISS-PROT reveals hits with $E$-value $<0.01$ only to other HIT proteins (Fig. 5a). An alignment to
the rat GaIT protein (62) has the only marginally significant E-value of 0.012. A PSI-BLAST search, using the score matrix generated from the six alignments illustrated in Figure 5a, can immediately cement confidence in the biological relevance of this similarity. The E-value of the similarity with rat GaIT drops to 2 × 10^-4, and an alignment to Haemophilus influenzae GaIT (63) (Fig. 5b) receives the even more significant E-value of 4 × 10^-5. These similarities, of course, are uncovered using no structural information. In addition, on the next iteration, PSI-BLAST finds a strongly significant alignment (Fig. 5c; E-value 2 × 10^-6) to yeast S. pombe SPAC69g.12 (64), for which no structure is available.

BRCT proteins

Proteins containing one or multiple copies of the BRCT domain form a superfamily many of whose members are involved in DNA damage-responsive cell cycle checkpoints (65–67). While detailed analysis is needed to delineate completely this diverse set of proteins, PSI-BLAST is able to automatically identify most of the superfamily. We used the C-terminal 215 residues of human BRCA1 (68), which includes two BRCT domains (63), as the initial query for a search of NCBI's non-redundant protein sequence database. Using the default cutoff E-value of 0.01, the initial BLAST search recognized as significant only alignments to other BRCA1 sequences, and the previously described BRCT protein BARD (69) (Table 4). Subsequent PSI-BLAST iterations, however, retrieved all the proteins recorded in Table 4; additional close homologs are omitted from the table. Almost all the BRCT proteins described by Bork et al. (66) were recognized. Not found were the retinoblastoma family, whose putative BRCT domain is particularly divergent, worm R13A1.13, which was not in the database searched, and human DNA-ligase III. PSI-BLAST did report yeast RAD9 and YHR102w, the Kluyveromyces lactis RAP1 homolog, worm ZK675.2, and various terminal deoxynucleotidyltransferases and poly(ADP-ribose) polymerases, but all with E-values >0.01 (Table 4). Detailed examination of the alignments produced suggests that the only likely false positives involved a trypanosome EST (70) and the Methanococcus jannaschii mifl protein (71), the latter despite its involvement in DNA repair (Table 4).

Seven recent additions to the protein databases are reported here as members of the BRCT superfamily (Table 4). (i) Arabidopsis T10M13.12 (72), is the first plant protein observed to contain BRCT domains. (ii) KIAA0259 (73) is a large human protein of unknown function with eight BRCT domains, the greatest number so far observed within a single protein. (iii) T13F2.3 (74) is a worm protein with a 500-residue low-complexity (57) N-terminus. (iv) SPAC69g.12 (75) is a fission yeast protein strongly similar to the previously recognized (66) yeast BRCT protein LS543.18 (76). (v) C36A4.8 (74) is a worm protein whose C-terminus contains a single BRCT domain, and whose N-terminus, containing a RING finger domain, is strongly similar to that of BRCA1. The similar organization to BRCA1 makes this protein of particular interest. (vi) Synechocystis sp. D90004 (77) is the first bacterial BRCT protein that is not a bacterial ligase. While it failed to pass the cutoff E-value of 0.01, its C-terminal BRCT domain is very similar to that of several bacterial ligases, which presumably led to its incorrect classification as such in the databases. Most of the protein N-terminal to its BRCT domain consists of a coiled-coil domain. The actual Synechocystis sp. DNA ligase (77,78) is found by PSI-BLAST on the 13th iteration, with an E-value of 0.02.

Figure 7. The location of BRCT domains within human BRCA1 (68). (ii) Arabidopsis T10M13.12 (72), human KIAA0259 (73), worm T13F2.3 (74), fission yeast SPAC69g.12 (75), worm C36A4.8 (74), Synechocystis sp. D90004 (77) and human Pescadillo (79), BRCA1 and C36A4.8 each have, in addition, an N-terminal RING finger domain. The near identity to other worm sequences of a short region directly preceding the BRCT domain of C36A4.8 suggests the possibility that this protein has been misassembled.

(vii) Pescadillo is a human protein whose zebrafish ortholog is essential for embryonic development (79), and whose yeast ortholog YGR102w (80) has been previously recognized as a BRCT protein (66,67). It failed to pass the cutoff E-value of 0.01, but appeared with near-significant E-values in PSI-BLAST output from the 5th iteration onward. The approximate positions of the BRCT domains within BRCA1 and the seven newly identified BRCT proteins are illustrated in Figure 7.

DISCUSSION AND CONCLUSION

In addition to the major algorithmic changes described above, we have modified an aspect of the original BLAST program's output routine that on occasion caused important similarities to be overlooked. When a very large number of statistically significant alignments was found, BLAST would typically report only the top scoring 500. These alignments, however, might all involve one domain of the query that occurred frequently within the database. Interesting but weaker relationships to other regions of the query might simply be forced off the bottom of the list. Accordingly, following the general idea of Sonnhammer and Durbin (81), we have limited the number of alignments reported that involve each region of the query, but set no overall upper limit.

The BLAST programs are unlikely to remain static, and there are many possible avenues for future improvement. We discuss three of them briefly here.

Gap costs

Gapped alignments may be constructed using a variety of different types of gap cost. Because a single mutational event may insert or delete a large number of residues, it has been argued that long gaps should not cost much more than short ones, and affine gap costs, which assess a score -(a + bk) for a gap of length k (82–85), have become the most widely used. A generalization of these costs has been proposed, that allows a gap to involve residues in both sequences rather than just one (86). Specifically, a gap in which k residues are inserted or deleted and j pairs of residues are left unaligned receives the score -(a + bk + cj). The algorithm necessary
for using such costs is only a minor variant on that for traditional affine gap costs. In many cases, the new gap costs generate local alignments that are both more accurate and more statistically significant (86). These costs are potentially of particular value for use with PSI-BLAST, because by imposing alignment only where it is justified, they may lead to the construction of more sensitive position-specific score matrices. Whether it is desirable to use generalized affine gap costs as the default for general purpose database searches awaits detailed empirical study.

Table 4. PSI-BLAST protein database search results using the C-terminus of BRCA1 as query

<table>
<thead>
<tr>
<th>Protein</th>
<th>Species</th>
<th>GenBank ID number</th>
<th>PSI-BLAST Iteration</th>
<th>E-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BARD</td>
<td>Homo sapiens</td>
<td>1710475</td>
<td>0</td>
<td>2e-06</td>
</tr>
<tr>
<td>T10M13.12</td>
<td>Arabidopsis thaliana</td>
<td>2104545</td>
<td>1</td>
<td>4e-06</td>
</tr>
<tr>
<td>F260D2.b</td>
<td>Caenorhabditis elegans</td>
<td>1914176</td>
<td>1</td>
<td>4e-04</td>
</tr>
<tr>
<td>KIAA0294</td>
<td>H. sapiens</td>
<td>1665785</td>
<td>1</td>
<td>2e-07</td>
</tr>
<tr>
<td>F372D4.c</td>
<td>Celegans</td>
<td>1418521</td>
<td>2</td>
<td>6e-07</td>
</tr>
<tr>
<td>C19G1.0.7</td>
<td>Schizosaccharomyces pombe</td>
<td>1723501</td>
<td>2</td>
<td>6e-07</td>
</tr>
<tr>
<td>KIAA0170</td>
<td>H. sapiens</td>
<td>1136400</td>
<td>2</td>
<td>2e-07</td>
</tr>
<tr>
<td>S38P1</td>
<td>Celegans</td>
<td>488892</td>
<td>2</td>
<td>2e-07</td>
</tr>
<tr>
<td>T13F2.3a</td>
<td>Celegans</td>
<td>1667234</td>
<td>3</td>
<td>2e-07</td>
</tr>
<tr>
<td>KHC2.4</td>
<td>Celegans</td>
<td>470351</td>
<td>3</td>
<td>3e-07</td>
</tr>
<tr>
<td>T19E10.1</td>
<td>Celegans</td>
<td>1067605</td>
<td>3</td>
<td>3e-07</td>
</tr>
<tr>
<td>RHA5/Cat5</td>
<td>S. pombe</td>
<td>730470</td>
<td>4</td>
<td>1e-05</td>
</tr>
<tr>
<td>REV1</td>
<td>Saccharomyces cerevisiae</td>
<td>132409</td>
<td>4</td>
<td>1e-05</td>
</tr>
<tr>
<td>ECT2</td>
<td>Mus musculus</td>
<td>423597</td>
<td>5</td>
<td>1e-04</td>
</tr>
<tr>
<td>XRCC1</td>
<td>Mu. musculus</td>
<td>652867</td>
<td>5</td>
<td>6e-04</td>
</tr>
<tr>
<td>Cby2</td>
<td>S. pombe</td>
<td>1490177</td>
<td>5</td>
<td>1e-04</td>
</tr>
<tr>
<td>RAP1</td>
<td>S. cerevisiae</td>
<td>1723588</td>
<td>5</td>
<td>1e-06</td>
</tr>
<tr>
<td>TeEST130F</td>
<td>Trypanosoma cruzi</td>
<td>1536887</td>
<td>6</td>
<td>4e-01</td>
</tr>
<tr>
<td>DPB11</td>
<td>S. cerevisiae</td>
<td>1352299</td>
<td>6</td>
<td>1e-04</td>
</tr>
<tr>
<td>L543.18</td>
<td>S. cerevisiae</td>
<td>1078575</td>
<td>6</td>
<td>4e-04</td>
</tr>
<tr>
<td>SPAC009G12</td>
<td>S. pombe</td>
<td>1644324</td>
<td>7</td>
<td>4e-04</td>
</tr>
<tr>
<td>YM8021.03</td>
<td>S. cerevisiae</td>
<td>1078533</td>
<td>7</td>
<td>4e-04</td>
</tr>
<tr>
<td>YHR154w</td>
<td>S. cerevisiae</td>
<td>731729</td>
<td>7</td>
<td>2e-08</td>
</tr>
<tr>
<td>CSK64.12</td>
<td>Celegans</td>
<td>1657667</td>
<td>7</td>
<td>2e-08</td>
</tr>
<tr>
<td>UN5452</td>
<td>S. cerevisiae</td>
<td>1151400</td>
<td>8</td>
<td>8e-05</td>
</tr>
<tr>
<td>DNA ligase IV</td>
<td>H. sapiens</td>
<td>1706982</td>
<td>8</td>
<td>8e-05</td>
</tr>
<tr>
<td>CDC9</td>
<td>Candida albicans</td>
<td>1706983</td>
<td>9</td>
<td>4e-00</td>
</tr>
<tr>
<td>DNA ligase</td>
<td>Thermus aquaticus</td>
<td>1352933</td>
<td>10</td>
<td>1e-00</td>
</tr>
<tr>
<td>GNF1</td>
<td>Dictyostelium melanogaster</td>
<td>544404</td>
<td>11</td>
<td>1e-04</td>
</tr>
<tr>
<td>mutF</td>
<td>Myxococcus</td>
<td>2129134</td>
<td>15</td>
<td>1e-08</td>
</tr>
<tr>
<td>RAD9</td>
<td>S. cerevisiae</td>
<td>131817</td>
<td>7</td>
<td>0.74</td>
</tr>
<tr>
<td>RAP1 homolog</td>
<td>Klocis</td>
<td>422387</td>
<td>9</td>
<td>0.21</td>
</tr>
<tr>
<td>ZK675.2</td>
<td>Celegans</td>
<td>599712</td>
<td>13</td>
<td>35</td>
</tr>
<tr>
<td>D9090.4</td>
<td>S. cerevisiae sp.</td>
<td>1652399</td>
<td>15</td>
<td>0.17</td>
</tr>
<tr>
<td>TTD</td>
<td>Mus domesticus</td>
<td>2140634</td>
<td>15</td>
<td>0.46</td>
</tr>
<tr>
<td>YOR105w</td>
<td>S. cerevisiae</td>
<td>1723693</td>
<td>16</td>
<td>0.017</td>
</tr>
<tr>
<td>Pespadillo</td>
<td>H. sapiens</td>
<td>2194203</td>
<td>16</td>
<td>0.017</td>
</tr>
<tr>
<td>PPOL</td>
<td>Saxophaga peregrina</td>
<td>1729741</td>
<td>16</td>
<td>4e-00</td>
</tr>
</tbody>
</table>

Iteration zero refers to the initial BLAST run, using the 215 C-terminal residues of BRCA1 (66) (SWISS-PROT accession no. P88086) as query. Subsequent PSI-BLAST iterations use derived position-specific score matrices in place of the query. The score matrix for iteration i + 1 is constructed from alignments achieving an E-value <0.01 for iteration i. For each protein, the E-value is that returned during the PSI-BLAST iteration indicated, and precedes the protein's use for score matrix construction. Only one representative is listed for families of closely related proteins. On its 16th iteration PSI-BLAST uncovered no new proteins with E-value <0.01, and therefore ceased iteration. At the end of the table are shown BRCT proteins returned by PSI-BLAST with E-value >0.01 but ≤10, listed for the iteration in which they achieved their lowest E-value.

- Recent additions to the databases, first identified as BRCT proteins here.
- The Celegans F260D2.b protein (74) with a recent addition to the databases, is a close homolog of the previously recognized (66,67) family of Celegans BRCT proteins containing, for example, FSTAA4.9 proteins (93).
- The trypanosome EST (70) and the Myxococcus mutF protein (71) are the only likely false positives.
Position-specific score matrices as input to PSI-BLAST

PSI-BLAST performs three distinct operations; it constructs a multiple alignment from BLAST output data; it processes this alignment into a position-specific score matrix; and it uses this matrix to search the database. A researcher may wish, however, to bypass the first two of these operations, and provide a score matrix as query directly to PSI-BLAST. The central difficulty is retaining the ability to calculate reliable statistics, as described above. PSI-BLAST imposes strict scaling rules on the matrices it generates, permitting the use of precomputed $A_0$ matrices to assess significance. Three possible routes are open. (i) One may permit the specification of 20 target frequencies $Q_j$ for each position rather than 20 scores. These can then be converted internally to log-odds scores with the appropriate scale for precomputed statistical parameters to apply. (ii) One may estimate by random simulation the statistical parameters for an input score matrix $(3)$. An advantage is the applicability to a greater range of scoring systems, including the possible use of position-specific gap costs. A disadvantage is the computational cost of obtaining reasonably accurate estimates of the relevant statistical parameters may increase the program's execution time unduly. (iii) One may abandon the statistical assessment of the alignments produced. This gives the greatest scope to input scoring systems, but precludes any reasonable scheme for automatic iteration of PSI-BLAST. Much experimentation will be required to determine which of these approaches is the most fruitful.

Realignment

After the initial BLAST run, or a later PSI-BLAST iteration, the multiple alignment used for subsequent iterations can be constructed in a more sophisticated manner than described above. Rather than coalescing all the pairwise alignments that pass the threshold immediately into a multiple alignment, a more significant among them can be used to build an initial multiple alignment and associated position-specific score matrix, which can then be used to resolve and realign database sequences that received lower scores. This step can be iterated several times before another full-scale database search is executed. There are several potential advantages to this procedure. (i) Weaker pairwise alignments, that may be somewhat inaccurate, can be improved and perhaps extended before they are incorporated into the evolving multiple alignment. (ii) Unrelated sequences that receive chance high scores can have their scores downgraded by an improved matrix, and perhaps be rejected before they are included in the alignment. (iii) Related sequences that received relatively high alignment scores, but missed the threshold for inclusion, can have their scores increased, and perhaps be included in the multiple alignment. In short, the realignment procedure can prevent inaccurate pairwise alignments from corrupting the evolving multiple alignment, and can accelerate the recognition of related sequences, all for very little computational cost. Preliminary studies suggest this line of development will be fruitful.

In conclusion, the new gapped version of BLAST is both considerably faster than the original one, and able to produce gapped alignments. While the relevant statistical parameters can no longer be calculated from theory, random simulation allows them to be estimated beforehand for commonly used amino acid substitution matrices and gap costs. For many queries, the PSI-BLAST extension can greatly increase sensitivity to weak, but biologically relevant sequence relationships. PSI-BLAST retains the ability to report accurate statistics, per iteration runs in times not much greater than gapped BLAST, and can be used both iteratively and fully automatically. These developments should enhance significantly the utility of database search methods to the molecular biologist.

Note

Source code for the new BLAST programs is available by anonymous ftp from the machine ebisu.nih.gov, within the directory `blast', and the programs may be run from NCBI's web site at http://www.ncbi.nlm.nih.gov/

ACKNOWLEDGEMENTS

W.M. and ZZ. are supported by grant LMS5110 from the National Library of Medicine. We thank Dr Warren Gish for helpful discussions, Dr Eugene Koonin for assistance with the examples, and Dr Gregory Schuler for producing several of the figures.

REFERENCES

3402 Nucleic Acids Research, 1997, Vol. 25, No. 17

86 Alachau.S.E., Proteins, in press.