A RAPID algorithm for sequence database comparisons: application to the identification of vector contamination in the EMBL databases

C. Miller¹, J. Gurd² and A. Brass¹

¹School of Biological Sciences, 2,205 The Stopford Building, University of Manchester, Oxford Road, Manchester M13 9PT, UK and ²Computer Science, Computer Building, University of Manchester, Oxford Road, Manchester, UK

Received on June 25, 1998, revised and accepted on November 11, 1998

Abstract

Motivation: Word-matching algorithms such as BLAST are routinely used for sequence comparison. These algorithms typically use areas of matching words to seed alignments which are then used to assess the degree of sequence similarity. In this paper, we show that by formally separating the word-matching and sequence-alignment process, and using information about word frequencies to generate alignments and similarity scores, we can create a new sequence-comparison algorithm which is both fast and sensitive. The formal split between word searching and alignment allows users to select an appropriate alignment method without affecting the underlying similarity search. The algorithm has been used to develop software for identifying entries in DNA sequence databases which are contaminated with vector sequence.

Results: We present three algorithms, RAPID, PHAT and SPLAT, which together allow vector contaminations to be found and assessed extremely rapidly. RAPID is a word search algorithm which uses probabilities to modify the significance attached to different words; PHAT and SPLAT are alignment algorithms. An initial implementation has been shown to be approximately an order of magnitude faster than BLAST. The formal split between word searching and alignment not only offers considerable gains in performance, but also allows alignment generation to be viewed as a user interface problem, allowing the most useful output method to be selected without affecting the underlying similarity search. Receiver Operator Characteristic (ROC) analysis of an artificial test set allows the optimal score threshold for identifying vector contamination to be determined. ROC curves were also used to determine the optimum word size (nine) for finding vector contamination. An analysis of the entire expressed sequence tag (EST) subset of EMBL found a contamination rate of 0.27%. A more detailed analysis of the 50 000 ESTs in est10.dat (an EST subset of EMBL) finds an error rate of 0.86%, principally due to two large-scale projects.

Availability: A Web page for the software exists at http://bioinf.man.ac.uk/rapid, or it can be downloaded from ftp://ftp.bioinf.man.ac.uk/RAPID

Contact: crispin@cs.man.ac.uk

Introduction

Database searching

Searching a single sequence against a database is a routine task. Software such as BLAST (Altschul et al., 1990) and FASTA (Pearson, 1990) exists to perform this task efficiently. Increasingly, however, there is a need to search a whole database against another. Such a need arises when batches of shotgun sequenced expressed sequence tags (ESTs) are to be screened for vector contamination, when ESTs are clustered before contig assembly, and in the study of complete genomes or large gene fragments.

These searches pose at least two significant problems: first, the time taken to perform the search can often be prohibitive; secondly, the large amount of data produced as a result of such a search must be carefully presented if a user is to make good use of it.

In this paper, we show how a formal split between word matching and alignment can be used to generate an algorithm which performs a database-against-database search extremely rapidly. Current search techniques use an initial word-matching step to ‘seed’ alignments, which are then scored and used to determine sequence similarity. For searches involving a small number of sequences, the time taken to compute the alignments is correspondingly short, and can safely be ignored. With a large search, the situation is rather different. In general, a query sequence is only similar to a small proportion of the database sequences. Computing alignments, which are then rejected because the sequences are considered too dissimilar, is an expensive process which contributes significantly to the time taken to conduct a search. An algorithm which is able to determine similarity at the word-searching phase rather than the alignment phase
should offer considerable gains in performance. Such an algorithm also changes the role of an alignment from a sequence-comparison technique to a user interface tool. Since the alignment no longer takes part in similarity prediction, different methods can be ‘plugged in’ to display similarity in the way which best suits the task in hand.

The algorithms described in this paper (RAPID, SPLAT and PHAT) are able to make this split without any apparent loss of sensitivity by using empirically determined word frequencies to increase the amount of information available at the word-searching phase.

RAPID is an extremely fast search tool based on word matching. It assesses the significance of a match against a probabilistic model which uses word frequencies to predict the number of matches to be expected by chance. RAPID is able to determine regions of similarity without any recourse to a computationally intensive alignment.

PHAT is a heuristic alignment tool that creates a spatially transformed dot plot which is used to determine regions of similarity. Matches are weighted according to the context a base occurs in so that low-complexity regions can be spotted and rejected at a glance. Examples of these alignments can be seen at http://www.bioinf.man.ac.uk/rapid.

SPLAT is a modification of the Smith–Waterman algorithm which uses probabilities to modify match/mismatch scores. The algorithm lets a user place an upper bound on the maximum possible gap size, allowing a trade-off to be made between speed and accuracy.

An initial implementation generates a tree of Web pages showing high-scoring hits, and the relevant alignments. Using PHAT to generate ungapped alignments, it is able to compare 50 000 ESTs against the vector database vector-ig, containing 1101 sequences, in 33 min on a 256 Mb P200 Pro running LINUX.

DNA sequence analysis can take a number of different forms, such as functional assignment by homology, clustering and vector screening. Each of these tasks is different, and requires a different test set to evaluate an algorithm’s performance properly. Since this is a lengthy process, this paper focuses on the ability of RAPID to perform one specific task: the identification of sequences containing vector contamination.

**Vector contamination**

At the time of writing, the EMBL DNA database contains 1.9 gigabases (Gb) and continues to grow exponentially, as it has done since its inception in 1987. Unfortunately, a small but significant proportion of the data is contaminated by vector sequences. Vector contamination can result in a number of errors, including incorrect contig assembly and false functional assignment due to spurious matches on vector sequence.

The problems of vector contamination have been studied by a number of researchers, including Lamperti et al. (1992) and Harger et al. (1998). Harger et al. found that of nearly 100 000 sequences from GSDB, 0.36% contained vector contamination. Although the overall level of vector contamination was found to remain constant over a 5 year period (at <1%), >50% of the contamination incorporated into the database came from EST and STS sequences. However, even though >50% of the contaminated sequences identified by Harger et al. were EST and STS sequences, 43.8% of the sequences added to EMBL between March 1996 and March 1998 were EST and STS sequences (see Table 1). This implies that EST and STS sequences are only marginally more contaminated than other sequences submitted to the database.

Clearly, the correct identification and annotation of vector contaminations is a task which is important if the integrity of sequence databases is to be improved. At present, this involves comparing each database entry against a database of vector sequences using a tool such as BLAST or FASTA, and examining any sequences which show similarity to a vector sequence; a time-consuming task.

The algorithms described in this paper are particularly well suited to the rapid identification of vector contamination.

**Theory**

In the following section, ‘word’ is defined to mean a \( k \)-long list of consecutive bases within a DNA sequence.

<table>
<thead>
<tr>
<th>Table 1. An analysis of the growth of the EMBL database showing the proportion of the last 2 years growth which can be attributed to EST and STS sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total size in nucleotides</strong></td>
</tr>
<tr>
<td>--------------------------------</td>
</tr>
<tr>
<td>EMBL release 46 March 1996</td>
</tr>
<tr>
<td>EMBL release 54 March 1998</td>
</tr>
</tbody>
</table>
RAPID (Rapid Analysis of Pre-Indexed Datastructures)

Word matching can be seen as a very rough approximation to computing an alignment: it is the number of \( k \)-long alignments that can be made between two sequences. By counting the number of short alignments without considering any kind of relative position, such an algorithm is able to reduce the amount of information it needs to consider and as a result make considerable gains in speed. Unfortunately, throwing so much information away also results in an algorithm which is rather insensitive.

In order to make a word-searching algorithm more sensitive, it is necessary to augment the raw matches with something else. Rather than use positional information to compute local alignments (as BLAST and FASTA do), RAPID uses statistical measures of word frequencies based on an \( n \)-gram analysis of a large amount of real data. This allows word matches to be scaled according to an estimate of the likelihood of them occurring by chance, and results in a surprisingly sensitive algorithm.

An analogy can be made to the task of comparing two newspaper articles: if they share a number of rare words such as ‘bug’, ‘millennium’ and ‘hospital’, they are likely to be referring to the same thing, but if they only share common words such as ‘the’, ‘and’ and ‘it’, they probably are not. The metric employed by RAPID for DNA searches is based on an analogous assumption.

A similar algorithm, EMBLSCAN, was proposed by Bishop and Thompson (1984); RAPID builds on that work in a number of ways. First, modern computer technology allows data structures to be built in memory which would have been too large a few years ago, allowing tables of longer words to be constructed. Secondly, DNA databases are now big enough to allow word frequencies to be determined empirically (EMBLSCAN assumed that the probability of a \( k \)-mer occurring was \( 4^{-k} \)). Other word search algorithms, such as FLASH (Rigoutsos and Califano, 1994), GCG’s wordsearch [which is similar to that of Wilbur and Lipman (1983)], Miropeats (Parsons, 1995) and the algorithm used to generate UniGene clusters (Boguski and Schuler, 1995) exist, but do not use word frequencies to influence the strength of the match (although FLASH uses probabilities to influence the database’s access pattern).

The algorithm

RAPID compares two sequences \( a \) and \( b \), by counting the number of words, \( N \), occurring one or more times in \( a \) which also occur one or more times in \( b \). This is compared to an estimate, \( E \), of the number of such ‘matches’ we would expect to occur by chance.

A DNA sequence of length \( L \) contains \( L - k + 1 \) overlapping words, which we consider as a list \( K_1, K_2 \ldots K_{L-k+1} \). Consecutive words in this list share \( K - 1 \) bases. The algorithm ignores words containing unknown bases (normally represented by the letter ‘n’ in sequence databases) with the result that a sequence containing a large number of unknowns has a relatively small number of unique words.

It is assumed that the probability of a word, \( K_i \), being \( w \) is simply \( P(w) \), the probability of \( w \) occurring next in an arbitrary DNA sequence, and we model the distribution of words within a DNA sequence as a Poisson distribution (Bishop and Thompson, 1984). Thus, the probability of a word, \( w \), occurring \( n \) times in a sequence of length \( L \) is given by a Poisson distribution with mean \( P(w)L \):

\[
P(w,n) = \frac{(P(w)L)^n e^{-P(w)L}}{n!}
\]  

(1)

So that the probability of a word occurring one or more times is:

\[
P(w,n \geq 1) = 1 - P(w,0) = 1 - e^{-P(w)L}
\]  

(2)

With typical values, \( e^{-P(w)L} \) is of the order \( e^{-1/500} \) so equation (2) can be approximated by expanding \( e^{-P(w)L} \) and ignoring all but the first two terms. Thus, the probability of a word occurring one or more times in a sequence of length \( L \) reduces to:

\[
P(w,n \geq 1) = P(w)L
\]  

(3)

The range over which this approximation holds is discussed in detail in Atteson (1998), which shows that it is reasonable to make such an approximation when \( k \) is of the order nine.

The number of matches to be expected by chance

Let \( W^a \) and \( W^b \), sizes \( L_a \) and \( L_b \), respectively, be the sets of words which occur one or more times in two sequences, \( a \) and \( b \).

The total number of matches \( E \) between a sequence, \( a \), and an unrelated sequence, \( b \), is estimated using equation (3):

\[
E = \sum_{i=0}^{L_a} L_b P(W^a_i) = L_a \sum_{i=0}^{L_a} P(W^a_i)
\]  

(4)

RAPID’s score

The significance of a match is estimated by taking the ratio \( S \) of the number of matches actually found to the number of matches expected by chance:

\[
S = N/E
\]  

(5)

\( S \) is highly dependent on the length of the sequences. With long sequences, small but significant matching regions are masked by chance matches from the rest of the sequences. Conversely, matches on very short sequences are assigned a higher score than they appear to warrant. For this reason, RAPID treats a long sequence as a set of independent, short,
overlapping fragments (typically 500 bp long), and adopts a modification of $S$ which normalizes it for the lengths of the subsequences:

$$S' = \frac{L_aL_b}{C_aC_b} \cdot S$$

(6)

where $C_a$ and $C_b$ are the sizes of the segments.

Note that $S$ and $S'$ are the same for all but very short sequences, where $S$ is considerably larger than $S'$. Whilst $S$ is a statistically robust estimate of significance, $S'$ is a pragmatic score which has been found to be more useful in practice.

Crucial to the calculation of $E$ are the chosen values of $P(w)$. The histogram of word frequencies in Figure 1 shows that it is not reasonable to assume that all words have equal likelihood. Rather, a small number of words are extremely common, whilst the majority of words occur with a probability less than $4^{-k}$.

As a result, RAPID determines word frequencies empirically. Ideally, these would be found by counting the occurrence of words in a large, representative and non-redundant set of DNA. One possible source would appear to be the yeast genome, but its highly skewed base composition (38% A+T) is reflected in an unusual word distribution. At present, we accept the problems of redundancy, and just count the number of words occurring in an EMBL subset. Redundancy has the effect of making words which occur in sequences repeated in the database appear more likely than they should.

It is necessary to put this into perspective: some words occur thousands of times more often than others, and it is matches due to these words that really need to be discounted. In practice, the program functions well with less than perfect word probabilities.

Intuition suggests that a search tool should scale down matches between low-complexity regions [such as a poly(A) sequence]. This would occur if $P(w)$ was inversely proportional to complexity. Figure 2 shows that this is indeed the case. The use of probabilities rather than entropy has the advantage that common regions which have a relatively high complexity (such as microsatellite repeats) are also scaled down.

Alignment generation

Whilst RAPID is sufficient to show that two sequences are similar, and to show to the nearest 500 bp where the similarity occurs, most tasks require more detailed information. RAPID is accompanied by a pair of alignment algorithms which can be used to examine one of these ‘hits’ more thoroughly.
**PHAT (Probabilistic Hough Alignment Tool)**

One of the earliest techniques used to determine sequence similarity was the dot plot (Gibbs and McIntyre, 1970). Given two sequences \(a\) and \(b\), a point \(p(x,y)\) is placed in the \(xy\) plane whenever \(a_x = b_y\). This results in regions of similarity appearing as diagonal lines angled at 45° to the axes.

Rather than plot such an image, it is possible to apply a spatial transformation and, for each point in \(xy\) space in which a large number of points occur, to plot lines in \(mc\) space which satisfy the equation \(c = y - mx\).

A set of points which lie on a line in conventional space, i.e. \(a\) and \(b\) lines in \(xy\) space which satisfy the equation \(c = y - mx\), appear as a set of intersecting lines in \(mc\) space. The point where these lines intersect gives the gradient and the \(y\)-axis crossing point of the original line.

This technique, known as a Hough transform (Hough, 1962), provides an efficient method for finding regions of similarity, when it is recognized that only lines with a gradient of one are of interest. These correspond to points in \(mc\) space on the line \(m = 1\). This is similar to the ‘diagonal method’ employed by FASTA.

When DNA sequences are compared in this way, the small number of symbols (a,c,g,t) result in a large number of points occurring by chance. In order to avoid this, PHAT only considers \(k\)-mer matches between two sequences.

PHAT plots histogram \(H_c\):

\[
H_c = \sum_x \sum_y S_{w_xw_y} \quad (7)
\]

where \(c = y - x\), and \(s\) is the score assigned to a match between the \(k\)-mers starting at \(a_x\) and \(b_y\). A large value of \(H_c\) corresponds to a significant match on the line \(y = x + c\).

Once an interesting diagonal has been determined, PHAT runs along it searching for the most significant alignment, offsetting the sequences by an amount determined by the point at which the diagonal crosses the \(y\)-axis. The alignment is found by computing a vector \(M\), where:

\[
M_c = \begin{cases} 
M_0 = 0 \\
M_{c-1} + m_{a,c} \cdot I_{a,b,c}/2 
\end{cases} \quad (8)
\]

The alignment ends where \(M_c\) is maximum, and starts at the first proceeding point where \(M_c\) is zero.

\(m_{a,c}\) is the score assigned to a match between the bases at \(a_x\) and \(b_y\).

\(I_{a,c}\) and \(I_{b,c}\) are ‘interest factors’ assigned to each nucleotide, determined as follows:

\[
I_i = \frac{\sum_{a < x} \left(1 - \frac{p(x)}{p_{\text{max}}(x)}\right)}{k} \quad (9)
\]

where \(k\) is the word length, \(p(w)\) is the probability of a word occurring (as used by RAPID) and \(p_{\text{max}}(w)\) is the probability of the most common word occurring.

Thus, a base in a common region such as a telomere repeat region is assigned an interest factor close to zero, whereas bases occurring in less common regions are assigned higher interest factors. The interest factor is used to weight the match and mismatch scores, both in the computation of the histogram and the alignment matrix. In both cases, the raw (mis)match score is multiplied by the average interest factor of the two bases being compared.

PHAT displays an alignment as coloured text, with each nucleotide being assigned a colour temperature according to its interest factor. Thus, bases in rare regions are coloured yellow, whilst those in common regions are coloured blue.

**SPLAT (Smart Probabilistic Local Alignment Tool)**

SPLAT is a modification of the Smith–Waterman (Smith and Waterman, 1981) alignment algorithm. Match/mismatch scores are modified using interest factors, as described in the previous section, and alignments are displayed using a colouring scheme similar to that of PHAT. In an optimization similar to that employed by FASTA, SPLAT limits the maximum gap length which the program will attempt to identify.

**Methods and implementation**

**Implementation**

In order to evaluate the algorithms described above, an initial implementation was produced which uses ‘Hashing’ to locate word lists within a table. This limits the algorithm to DNA (because tables for proteins become too large to fit into memory). Other implementations could be produced which generalized to proteins, but so far this has not been done.

In advance of a search, the software builds a database of word lists which records, for each of the \(4^k\) possible words, the sequences in which that word occurs. The database is stored as a disk file. The use of this pre-computed data significantly reduces the amount of work required during a search.

For database-against-database searches, the low cost of memory would make it feasible to load the entire database into memory, where searches can be performed with no disk accesses. In the case of a single sequence search, where only a small proportion of the possible words need to be looked up, the time taken to load the entire database into memory would be prohibitive; a preferred solution would be to load a word list only when it was required. RAPID meets these conflicting demands by using memory-mapped IO to map the database file into its (virtual) address space. When the mapping takes place, no disk-to-memory transfer is initiated. Instead, page faults are generated each time the software ac-
cesses a word list which is on a page of the address space that has not been previously loaded into physical memory (RAM).

Memory-mapped IO has a number of advantages. First, only the sections of the database which are required are loaded into RAM. Secondly, the number of actual disk accesses is reduced to one-per-page as opposed to one-per-read with traditional file IO and, thirdly, traditional file operations perform a certain amount of buffering which results in repeated copying of the data being read. Direct memory access is, by its nature, unbuffered, and does not incur these costs.

**Input and output**

The software has a simple command line interface that unites the word-searching and alignment algorithms. The software accepts an EMBL or FASTA file as input. Different probability tables can be loaded, allowing the word weighting scheme to be changed if desired. One problem associated with a large search is the amount of data produced. We have tried to address this by providing a tree of Web pages containing the results of a search. The root page shows the top hit for each query sequence which has matched against at least one database sequence. Below this are a set of pages describing, for each query sequence, all the hits that have been found. This page intentionally resembles the list produced at the top of a BLAST output file, and links through to a set of pages, each containing an alignment for a query/database pair.

**Evaluation**

In order to assess the software’s efficacy, we produced a test set by artificially introducing progressive amounts of vector sequence into a set of uncontaminated DNA sequences. The test set was used to determine RAPID’s ability to classify correctly sequences as being contaminated or uncontaminated, and to determine the optimum score threshold and word size for identifying vector contamination. The test set was also used to compare the performance of raw matches versus statistically weighted ones. The specific task addressed in this paper is that of identifying vector contaminations; a different task is likely to demand different parameters.

The test set was produced by taking an uncontaminated EST and replacing progressive amounts of the sequence with that of a vector, resulting in 21 sequences containing between zero and 200 bp of contamination. This process was repeated with six ESTs and five different vectors, producing 630 entries. A total of 612 uncontaminated sequences were added to this, to give a test set containing 1242 sequences. The test set does not mimic sequencing errors (such as mutations or indels) when the test set was constructed. Since sequencing error rates are generally ~3%, and RAPID can identify matches of 30 bp, we do not consider this to be a significant flaw in the test set.

A sequence was considered to be contaminated if it contained over 30 bp of contamination. Figure 3 shows RAPID scores resulting from a comparison of the artificial test set against vector-ig.

Receiver Operator Characteristic (ROC) curves (Figure 4) can be used to determine a search tool’s ability to classify sequences correctly by calculating the tool’s sensitivity and selectivity for different score thresholds.

Given a score between a test sequence \( Q \) and a vector sequence, \( \Theta \), and a score threshold \( \Theta_c \), the test sequence can be assigned to one of four sets:

- \( t^+ \): true positive; \( \Theta > \Theta_c \), \( Q \) is contaminated.
- \( t^- \): true negative; \( \Theta < \Theta_c \), \( Q \) is not contaminated.
- \( f^+ \): false positives; \( \Theta > \Theta_c \), \( Q \) is not contaminated.
- \( f^- \): false negatives; \( \Theta < \Theta_c \), \( Q \) is contaminated.

The number of sequences in each of these sets \( (T^+, T^-, F^+, F^-) \) can be determined for a particular value of \( \Theta_c \), allowing \( P^+ \) and \( P^- \), sensitivity and selectivity, to be determined for different score thresholds:

\[
P^+ = \frac{T^+}{T^+ + F^-}
\]
A RAPID algorithm for sequence database comparisons

Fig. 4. Received Operator Curves (ROC) for searches against vector-ig with an artificially contaminated test set. A contaminated sequence contained over 30 bp of contamination. Filled circles represent weighted 9-mers, triangles 8-mers. Unfilled circles represent unweighted 9-mers; 10-mers are overlaid on the graph, and are therefore not visible.

With a sufficiently low value of $\Theta_c$, every sequence is judged to be contaminated (they have a score above $\Theta_c$), resulting in $P^+ = 1$ and $P^- = 0$. Conversely, for a sufficiently high threshold, every sequence is judged to be uncontaminated, so that $P^+ = 0$ and $P^- = 1$. An ideal tool would have a score threshold which allowed it to identify correctly all contaminated sequences without misclassifying any uncontaminated ones ($P^+ = 1$ and $P^- = 1$). In reality, such a tool does not exist, and it is useful to investigate the relationship between $P^+$ and $P^-$ for different score thresholds. Such a curve is known as a ROC and ideally should have an area of 1.0 (Swets, 1982; Shah and Hunter, 1997).

ROC curves of RAPID scores were produced for $k = 8$, 9, 10 and for the raw number of matches for $k = 9$ (i.e. without any probabilistic weighting). The area under the curve was 0.96 for $k = 8$-mers, 1.00 for 9-mers and 0.99 for 10-mers. Unweighted 9-mers also produced a curve with area 1.00. However, the test set did not contain any sequences with significant low-complexity regions. When these are considered, it is evident that probabilistic weighting significantly reduces the scores due to these matches (see Figure 5).

Although 9-mers and 10-mers have similar discriminatory ability, 9-mers place smaller demands on memory. Thus, $k = 9$ with probabilistic weighting was selected for spotting vector contaminations. The optimal value for the score threshold $\Theta_c$ was determined to be 10 for 9-mers. This is in keeping with the results in Figure 3.

Comparison with BLAST

Having established RAPID’s ability to classify sequences from the test set correctly, we compared the scores produced by RAPID and BLAST for a number of real sequences.

This was done by taking five randomly selected ESTs which showed a varying amount of similarity to vector sequences and using them to search against vector-ig with both algorithms. Each EST hit against a number of vector sequences, resulting in a total of 1803 pairwise comparisons. A graph of RAPID versus BLAST scores for each EST/vector pair was plotted (see Figure 6). The approximate straight line demonstrates that RAPID and BLAST identify the same set of matches with a given probe sequence, and that the matches are ordered in an equivalent way.

Speed, memory and disk usage

Two factors which contribute to the time and space performance of the algorithm are the query database size (which affects the number of times the algorithm needs to look up a word in the target database) and the target database size (which affects the length of the word lists the algorithm has to process).

The number of table look-ups made by the RAPID algorithm is proportional to the number of $k$-mers in the query database, which in turn is approximately proportional to the number of nucleotides it contains.
Fig. 6. A comparison of RAPID and BLAST scores for sequences with different levels of similarity to those in vector-ig. Each point represents a hit between a query sequence and a particular database sequence. Filled circles, C15000; unfilled circles, X93604; filled triangles, C14014; unfilled triangles, C15706; filled squares, C14077.

The time taken to process a table look-up is proportional to the number of sequences which contain that $k$-mer. If it is assumed that the $k$-mer composition of the query database is uniform, then the time taken to perform a search should be proportional to the number of $k$-mers in the query.

The size of a given word list, assuming even $k$-mer composition, is slightly sub-linear with respect to database size. This is because words which occur more than once in a sequence are only recorded once by the algorithm. Thus, search time should be linear with respect to target database size, and the overall memory usage should also be linear. This is confirmed by the results shown in Figure 7, produced by clustering varying numbers of ESTs.

If it is assumed that the databases are of even composition, then the total number of query–target matches which score above a threshold should also be proportional to query and target database sizes. Thus, if alignments are required, the number of calls to the alignment algorithm should also be proportional to query and target database sizes.

Fig. 7. The time taken and memory usage when clustering 10k, 20k, 40k and 80k ESTs on a P200 Pro running RAPID. Clustering is performed by comparing a database against itself. Memory usage is proportional to database size, as expected. As the database gets bigger, both the number of query sequences and the number of target sequences increase. Thus, the time taken to cluster the database is proportional to the square of the database size, as expected.

An estimate based on the results in Figure 7 suggests that the EMBL DNA database (currently containing ~1.9 Gb) could be clustered in ~6 Gb of RAM.

Index files
Since the implementation uses memory-mapped IO, the index file on disk is similar in size to the program’s memory image. In addition to the index file, the implementation stores the sequence description lines, a probability table, a compressed representation of the DNA sequences in the database.
A RAPID algorithm for sequence database comparisons

Contamination bounded by EcoR1 site

SPLAT - Smart Probabilistic Local Alignment Tool

FORWARD strand of C06135 (similar to none.) from 1 to 392
against:

U03443 (Saccharomyces/E.coli phagemid vector pRS403 - complete.) from 2001 to 3000

<table>
<thead>
<tr>
<th>Length</th>
<th>Gaps</th>
<th>SPLATScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>5</td>
<td>34.1</td>
</tr>
</tbody>
</table>

Fig. 8. A run of vector sequence bounded by an EcoRI restriction site (underlined).

and their constituent k-mers (the last two files are used by the alignment tools). For est10.dat, this totals 282 Mb, ∼30% bigger than the original EMBL file.

Indexing times are also fast: est10.dat is indexed in 187 s on a P200Pro running LINUX.

Comparison with BLAST

Timings were taken using the UNIX time command. In order to compare the speed of RAPID and BLAST, we searched est10.dat for vector contaminations by using each tool to compare it against vector-ig. NCBI BLAST Version 2.05 was used for all the comparisons. On our platform, it used memory-mapped IO, as used by RAPID. Parameters were set so that each tool only identified and aligned the top hit for each matching EST, and BLAST was set to use the blastn program for comparison. RAPID generated ungapped alignments using PHAT. On a Sun Ultra 5 with 256 Mb, RAPID takes ∼33 min to perform this search; NCBI BLAST Version 2.05 takes 493 min.

Results

Contamination in the EST subset of EMBL

A search of the entire EST subset of EMBL against vector-ig identified 4061 sequences, giving an estimated error rate of 0.27%, which is broadly in keeping with Lamperti et al. (1992) and Harger et al. (1998). Parameters were as determined in the previous section.

Analysis of EST10

A search of est10.dat (one of the EST subsets in EMBL) against vector-ig using a score threshold of 10.0 (as determined from searches with the artificial test set) identifies 412 sequences with significant similarity to vector. This gives an estimated contamination rate of 0.82%.

A total of 66% of the sequences identified were submitted as part of two batch sequencing projects (Nathans J., 1996, unpublished; Lanfranchi et al., 1997). This proportion was greatly in excess of their overall contribution to the subset (14.5%).

Two per cent of Lanfranchi’s sequences and 2.8% of Nathans’s sequences showed significant match to vector.

Of the sequences identified, 171 (41.5%) contained <100 bp of vector, and are likely to be simple editing errors where regions flanking the insert have not been removed before submission. If a restriction site is present in one of these sequences, the vector/insert junction can be identified and the vector cleanly removed (see Figure 8).

A total of 241 (58.5%) of the sequences contain >100 bp of vector. One hundred and thirty-one of these were submitted by Nathans. Fifty-five of these showed significant similarity to the OP region of λ phage (λgt10 was used as the cloning vector). Figure 9 shows one of these matches.

Discussion

The use of word frequencies to weight word matches results in an algorithm which is fast and sensitive. Direct comparison with BLAST shows that RAPID produces scores which are very similar to those produced by BLAST approximately an order of magnitude faster.

ROC analysis allows the optimum word size and score threshold for identifying vector contamination to be determined. It is possible to determine a score threshold which successfully identifies contaminations >30 bp long. The fact
Example contamination from lambda phage

SPLAT – Smart Probabilistic Local Alignment Tool

REVERSE strand of W21863 (57G5 Human retina cDNA Tsp509I-cleaved sublibrary Homo sapiens cDNA) from 203 to 701

gainst:

V00636 (E. coli phage vector lambda (Styloviridae) - complete.) from 38001 to 38999

<table>
<thead>
<tr>
<th>Length</th>
<th>Gaps</th>
<th>Identity</th>
<th>SPLATScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>399</td>
<td>2</td>
<td>98.0</td>
<td>377.1</td>
</tr>
</tbody>
</table>

Fig. 9. One of the 55 sequences with high similarity to the OP coding domains of λ phage.

that matches <30 bp occur by chance has implications for EST clustering.

Analysis of the scores produced by low- and high-complexity sequences confirms that probabilistic weighting successfully scales down matches to low-complexity sequences.

An analysis of the entire EST subset of EMBL, performed by comparing it to sequences in the vector database vector-ig, found a contamination rate of 0.27%. This is in keeping with the results of Harger et al. (1998) and Lamperti et al. (1992). This approach relies on the vector database having sufficient coverage to hit against every contaminated sequence in the EST subset. As a result, it is possible that a number of contaminated sequences were missed. However, given that sequences are often submitted without the cloning vector and restriction enzyme sites being included in their annotation, it is not possible to assess exactly how many. In many cases, a similar, but not identical, sequence will exist in the database. The λgt10 contaminations discussed earlier are an example of this: the hits found were matches against λ phage, a similar, but not identical sequence.

It is worth noting the role that annotation plays in the whole process: we are placed in the position of using computational approaches to try to deduce information which was known at the time of sequencing. Thus, it is necessary to compare each sequence in the contaminated data set against a few thousand possible vector sequences, rather than just the relevant one.

Analysis of 50 000 public domain ESTs suggests that the majority of contaminations arise from a small number of projects which have submitted data without the appropriate quality control. Although the increasing use of well-designed cloning kits should help reduce the number of errors,
A RAPID algorithm for sequence database comparisons

quality control is still necessary and should involve routinely scanning all ESTs for vectors before submission.

It can be argued that a sequence database acts as a repository for experimental results, allowing an experimental scientist to submit their data to a public site for peer review and analysis. If this view is taken, then it is correct to submit contaminated sequences. However, sequence databases are used as information resources in their own right; it is unreasonable to consider a database such as EMBL as a mere repository for experimental results. Thus, if contaminated sequences are submitted to a database, it is imperative that they are clearly and unambiguously annotated as such.

The algorithms and tools described in this paper provide a mechanism which can be used to identify vector contamination quickly before sequences are submitted to a database, and to identify offending sequences which have already made their way into a database.

It is envisaged that the tools described in this paper will be suitable for other projects which involve comparing large data sets against each other, such as functional genomics, clustering and function prediction by similarity. However, such an application requires serious consideration to be made about handling the results of such a comparison: comparing two databases to produce a third one is not necessarily a desirable course of action.

Formally separating word searching from alignment allows alignment generation to be viewed as a user interface problem rather than a part of a similarity search. This allows a client server architecture to be developed where similarity searching is performed by the RAPID algorithm on a server, whilst alignments are generated by the client on the fly, as and when they are needed. Such an architecture exploits the power of modern workstations to reduce the load on a server significantly.

Acknowledgements

This work was carried out as part of a PhD CASE studentship funded by Pfizer UK Ltd and the Biotechnology and Biological Sciences Research Council (BBSRC). The authors would like to thank Anne Westcott, Paul Higgs, Graham Riley, Isobel Anderson and Richard Moore for their valuable discussions. The software was compiled using GNU GCC and run on a computer using the LINUX operating system and the KDE desktop environment.

References


