BIOINFORMATICS

Vol. 14 no. 5 1998
Pages 423–429

# Removing near-neighbour redundancy from large protein sequence collections

Liisa Holm and Chris Sander

EMBL-EBI, Cambridge CB10 1SD, UK

Received on November 6, 1997; revised and accepted on January 15, 1998

## Abstract

***Motivation:*** *To maximize the chances of biological discovery, homology searching must use an up-to-date collection of sequences. However, the available sequence databases are growing rapidly and are partially redundant in content. This leads to increasing strain on CPU resources and decreasing density of first-hand annotation.*

***Results:*** *These problems are addressed by clustering closely similar sequences to yield a covering of sequence space by a representative subset of sequences. No pair of sequences in the representative set has >90% mutual sequence identity. The representative set is derived by an exhaustive search for close similarities in the sequence database in which the need for explicit sequence alignment is significantly reduced by applying deca- and pentapeptide composition filters. The algorithm was applied to the union of the Swissprot, Swissnew, Trembl, Tremblnew, Genbank, PIR, Wormpep and PDB databases. The all-against-all comparison required to generate a representative set at 90% sequence identity was accomplished in 2 days CPU time, and the removal of fragments and close similarities yielded a size reduction of 46%, from 260 000 unique sequences to 140 000 representative sequences. The practical implications are (i) faster homology searches using, for example, Fasta or Blast, and (ii) unified annotation for all sequences clustered around a representative. As tens of thousands of sequence searches are performed daily world-wide, appropriate use of the non-redundant database can lead to major savings in computer resources, without loss of efficacy.*

***Availability:*** *A regularly updated non-redundant protein sequence database (nrdb90), a server for homology searches against nrdb90, and a Perl script (nrdb90.pl) implementing the algorithm are available for academic use from http://www.embl-ebi.ac.uk/~holm/nrdb90.*

***Contact:*** *holm@embl-ebi.ac.uk*

## Introduction

Sequence similarity searching is the most powerful tool to predict the function of anonymous gene products that come out of genome sequencing projects. The idea is simple: functional information can be transferred between homologous proteins, which carry the memory of common ancestry in their amino acid sequences as a result of functional constraints that have persisted in evolution. Information transfer by homology has two practical limitations. The deduced information can only be as good as the original functional annotation in the search database, and can only be comprehensive if the search database covers all currently known sequences.

Interesting sequence similarities occur in the entire interval from 0 to 100% sequence identity, but the database has an overabundance of very closely related sequences (e.g. above 90% sequence identity). This uneven distribution of sequences in sequence space is due to research bias [leading to particular overrepresentation of, for example, HIV, the RUBISCO enzyme (alias ribulose-bisphosphate-carboxylase/oxygenase), immunoglobulins, cytochrome *b* and hepatitis virus] and the existence of clusters of closely related sequences from multigene families and from different species or organisms. As well as slowing down database searches using, for example, Blast (Altschul *et al.*, 1997) or Fasta (Pearson and Lipman, 1988), long lists of similar or identical alignments can obscure novel matches in the output. A more even sampling of sequence space would lead to short and clean output listings from database searches without repetition of redundant hits. Here, we use the concept of a representative set to obtain a complete covering of sequence space while hiding redundant sequences (but not their functional description) from view. A representative set is a subset of the sequence database that contains no neighbours (properly defined), and every sequence in the complement of this subset (redundant sequences) is a neighbour of at least one representative.

A partial solution to the redundancy problem is to merge sequences that are exact duplicates of one another. The nrdb program from NCBI (ftp://ncbi.nlm.nih.gov/pub/nrdb/) uses checksums computed from the sequence as addresses in a 32-bit hash table; a single pass of the sequence set and one look-up in the hash table suffice to determine whether an identical sequence has already been encountered. For example, the GeneQuiz system (Scharf *et al.*, 1994) searches against the union of Swissprot (full release), Swissnew (daily increments), Trembl (translation of EMBL nucleotide sequence database), Tremblnew, Genbank translations, PIR, Wormpep (nematode genome sequencing project) and PDB (known structures). Running the nrdb program on this combined database only takes 10 min on a single RISC CPU and reduces its size by a factor of 2.9, i.e. from 750 000 to 260 000.

Even after the removal of exact duplicates, much redundancy remains. Algorithmically, the search for similarities is a much more complex problem than the detection of exact duplicates. Complete dynamic programming alignment is one of the most sensitive alternatives, but is costly in computer time. For speed, most database search algorithms (Pearson and Lipman, 1988; Califano and Rigoutsos, 1993; Altschul *et al.*, 1997) therefore first filter the database by matching short words (alias ktuples). The rationale is that overall similar sequences are likely to share a fair number of identical ktuples. The ktuple matching approach is ideally suited for the detection of very closely related sequences. For example, the OWL non-redundant composite protein database allows one mismatch in a segment of 30 residues between redundant sequences (Bleasby and Wootton, 1990), the CLEANUP program for removing redundancies from nucleic acids datasets uses a threshold of seven out of eight matching nucleotides (Grillo *et al.*, 1996), and the SWIR database aims at 95% sequence identity using a bit-vector representation of 2tuples as a heuristic filter (P.M.Rice and R.D.Durbin, personal communication).

We have developed an exhaustive search algorithm based on exact bounds that relate similarities in ktuple composition to pairwise sequence identity. Despite carrying out an exhaustive search, the algorithm is sufficiently fast for practical applications involving datasets of several hundred thousand protein sequences. The threshold between sequence neighbours and non-neighbours is somewhat subjective and depends on the application. For simplicity, we set the threshold to 90% sequence identity in the current implementation. For the nrdb database, application of our algorithm yields 140 000 clusters of sequences. The sequences in one cluster typically have a common biological function. Our algorithm uses a lower threshold than nrdb, OWL or SWIR, and consequently yields a smaller (yet complete) non-redundant protein database than these methods, and a larger speed-up for database searching. Looking ahead, the clustering of sequences into homologous families will also be a step towards resolving the problem of consistent annotation of anonymous sequences.

## Methods

### Representative set

There are many ways to select a representative set, and there are many selections that satisfy the criteria of a representative set. Here, we generate a representative set based on a greedy incremental algorithm (Hobohm *et al.*, 1992). Briefly, sequences from the database are processed in order of decreasing length. Each query sequence from the database is compared to all previous representatives. If a similarity exceeding the threshold of 90% identity is found, then the query is redundant with respect to the representative. If the query is unique, it is added to the representative set. Sequence identity is defined as seen from the query sequence. Owing to the length sorting, query sequences are only compared to longer representatives. Consequently, each redundant sequence is contained in the representative so that >90% of the residues are identically aligned.

The outcome of the selection process is deterministic for a given ordering of the sequences in the input database. A unique ordering of the input database is imposed by sorting the sequences first by length and then alphabetically within each length bin.

### All-on-all pairwise comparison

To build up the representative set, the algorithm needs effectively to perform an all-on-all comparison of the sequence database. Potential pairwise similarities above the threshold are detected using three filters. The filters are designed to be exhaustive (no false negatives), yet to be effective in pruning search space (eliminate plenty of obvious negatives). The filters are increasingly expensive, but are applied to a decreasing number of sequences. The first filter excludes any sequence pairs lacking a common decapeptide. The second filter further excludes any pairs that have <50% common pentapeptides. These peptide composition filters are implemented using look-up tables which makes them quite fast on large datasets. The third and final filter excludes pairs with <90% identity based on explicit sequence alignment. Details including informal proofs for the peptide composition filters follow.

### Decapeptide filter

The filters are based on the principle that similar sequences yield similar ktuple compositions. For generality, a ktuple is defined here as a subsequence of the query sequence that contains $k$ match positions in a window of $w$ residues ($1 \leq k \leq w$). For example, if $w = 3$ and $k = 2$, the sequence *ACE* generates the tuples *xCE*, *AxE* and *ACx*, where $x$ is a placeholder accepting any amino acid. We will first derive a lower bound on sequence identity when there is at least one matching ktuple.

Residue mismatches can be due to amino acid substitutions or insertions/deletions (gaps) in the optimal alignment (Figure 1). The length $L$ of the query sequence is not necessarily a multiple of $w$:

$$L = aw + b, \quad 0 \leq b < w \tag{1}$$

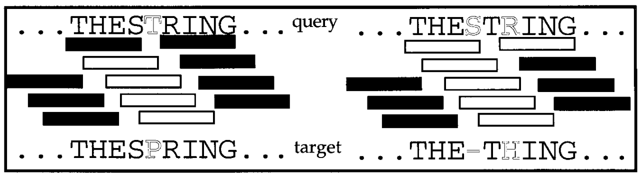

**Fig. 1.** Searching for close similarities by filtering on peptide composition. The generation of a representative set requires an effective all-on-all pairwise comparison of the database to identify neighbours and non-neighbours. Explicit sequence alignment is computationally expensive. Using table look-ups for comparing peptide compositions results in efficient pruning of search space. Here, two pairwise alignments are shown with matching residues or tetrapeptides (4tuples) in black and mismatching ones in white. The number of mismatches in terms of sequence positions (columns) is algebraically related to that in terms of peptides. This allows the formulation of filters for the exclusion of dissimilar pairs at a given sequence identity threshold (in this work: 90%) based on counts of matching peptides.

If no tuple matches, then every window $w$ must contain at least $(w - k + 1)$ mismatched residues. Let the number of mismatched residues in the query sequence be $n$. As the sequence is covered by non-overlapping windows, we have

$$n \geq (w - k + 1)a \qquad (2)$$

(If $b \geq k$, the lower bound on required residue mismatches is higher.) Let us define effective sequence identity $p$ as:

$$p = 1 - \frac{n}{L} \qquad (3)$$

where the mismatch count $n$ includes substitutions and gaps. Substituting equations (1) and (2) into equation (3) yields:

$$p \leq \frac{k}{w} - \frac{1}{w} + \frac{b}{w}\,\frac{w - k + 1}{L} < \frac{k}{w} - \frac{1}{w} + \frac{w - k + 1}{L} \qquad (4)$$

In the case $w = k$, equation (2) reduces to $n \geq a$, which yields $p \leq 1 - \frac{1}{k}$ when $n > a$ and $p \leq 1 - \frac{1}{k} + \frac{1}{L}$ when $n = a$.

The first filter of our algorithm is based on exact matches of decapeptides. Stated simply, a sequence identity >90% between two sequences requires the existence of at least one pair of aligned decapeptides where the two decapeptides are >90% identical to each other; this implies 100% identity for deca- and shorter peptides. More precisely, the first filter uses $w = k = 10$, yielding $p \leq 90\%$ when $n > a$. The equality $n = a$ requires that the mismatches are homogeneously distributed at every $k$th residue. In practice, this situation is only encountered with peptides shorter than 20 amino acids, for which $n = 1$ yields $p < 95\%$.

### *Pentapeptide filter*

Let us next derive a lower bound on the number of matching tuples at a given level of effective sequence identity $p$ between two aligned sequences. The query sequence is covered by $T = (L - w + 1)\binom{w-1}{k-1} + w - k$ ktuples. Each residue mismatch affects any ktuples spanning the position except those with a placeholder at that position. The number of mismatched ktuples $m$ is:

$$m \leq nk\binom{w-1}{k-1} \qquad (5)$$

Substituting equation (3) yields $T - m \geq w - k + [L - w + 1 - (1 - p)Lk]\binom{w-1}{k-1}$. The second filter of our algorithm uses $w = k = 5$ and $p = 90\%$, yielding the lower bound of $0.5L - 4$ matching tuples (where $L$ is the length of shorter sequence).

### *Pairwise sequence alignment*

The third filter uses the Smith and Waterman (1981) algorithm which delineates biologically meaningful local alignments even if sequence lengths differ greatly. Briefly, the alignment score is accumulated along a trace in the $N*M$ alignment matrix (where $N$ and $M$ are the lengths of the sequences). A trace starts at a positive score and ends when the alignment score no longer increases. In other words, trailing ends with a negative net contribution to the alignment score are cut off. The alignment parameters used were +1 for identities, –0.15 for amino acid substitutions, gap opening and elongation penalties of (0, –1) for insertions in the query sequence and (–1,0) for deletions in the query sequence. (This detects query sequences which are concatenated from fragments of the representative such as S20898 in Table 3.) The amino acid ambiguity codes BXZ always counted as a mismatch. For speed, we used a heuristic that restricted the pairwise alignment to a small number of diagonals (sequence offsets) with a high number of matching pentapeptides.

Given the optimal alignment, sequence identity was computed as the number of identities divided by the length of the query sequence so that at least 90% of any redundant sequence must be identical with the representative sequence. This is more permissive than the effective sequence identity $p$ of equations (1–5) which includes gaps. However, as any gaps are rare between close homologues, the difference is not important in practice.

## Implementation

The algorithm was implemented in the Perl5 language (Wall *et al.*, 1996). The input is a sequence database. The output is a non-redundant sequence database (representatives) and a list of cluster memberships (redundant sequences may be neighbours of several representatives, but are listed with only one). The sequence clusters centred around the representatives are described by a simple functional summary using similar principles as in GeneQuiz (Scharf *et al.*, 1994). Incremental updates of the non-redundant database (nrdb90) start from a previous clustering.

The program, regular updates of the database, and a server for similarity searches are available electronically for academic use at http://www.embl-ebi.ac.uk/~holm/nrdb90. Commercial users should apply for a licence.

## Results and discussion

We have developed a complete and fast algorithm (nrdb90) for removing near neighbours and fragments from large sequence collections. The algorithm is not based on heuristics, but performs an exhaustive all-on-all comparison to generate representative sets at an effective 90% sequence identity threshold. The algorithm is efficient because the use of look-up tables to compare protein peptide compositions eliminates the vast majority of dissimilar pairs before they are ever compared by explicit alignment.

The algorithm was applied to four databases (Table 1). Removing fragments and near neighbours, the nrdb90 algorithm is more general but slower than the nrdb program from NCBI (which only removes exact duplicates). Datasets of a few thousand sequences are processed in a matter of minutes (e.g. yeast; Table 2). Cleaning the nrdb dataset showed an overabundance of close variants. A total of 260 000 non-identical sequences were reduced to 140 000 clusters, where each cluster member is contained in the representative at >90% sequence identity (see Table 3 for an example). Since popular similarity search tools (e.g. Fasta, Blast) scan the database sequentially, the smaller search database will lead to substantial time savings in large-scale sequence analysis and compute services for sequence searches (e.g. GeneQuiz).

**Table 1.** Benchmarks for generation of representative subsets at 90% sequence identity threshold

| Database | Sequences[a] | Clusters | Redundancy (%) | Pair alignments | CPU time (h)[b] |
|---|---|---|---|---|---|
| PDB | 4 575 | 2 125 | 54 | 2 793 | 0.12 |
| Yeast | 6 216 | 5 983 | 4 | 345 | 0.33 |
| Swissprot 34.0 | 58 804 | 47 580 | 19 | 18 166 | 5.2 |
| nrdb[c] | 260 754 | 141 338 | 46 | 308 780 | 50 |

[a]Number of sequences excluding duplicates and sequences shorter than 10 amino acids.
[b]Using one MIPS R10000 processor (194 MHz) with 1 Gbyte main memory.
[c]Union of Swissprot, Swissnew, Trembl, Tremblnew (Bairoch and Apweiler, 1996), Genpept, PIR, Wormpep and PDB (Bernstein *et al.*, 1977).

**Table 2.** Redundant genes at 90% sequence identity threshold in the yeast genome (Goffeau *et al.*, 1996)

| Representative | Length of representative | Cluster size | Protein family |
|---|---|---|---|
| YHR214C-B | 1793 | 61 | TY1A/B |
| N2715 | 1810 | 27 | TY2A/B |
| YGR296W | 1859 | 22 | Probable membrane protein |
| YAL068C | 120 | 13 | PAU1/2/4/6 |
| YOR394W | 164 | 10 | PAU3/5 |
| G5984 | 1547 | 5 | TY3A/B |
| YJL113W | 1803 | 4 | TY4B |
| YGR287C | 589 | 4 | FSP2, alpha-d-glucosidase homolog |
| YDL245C | 567 | 4 | HXT13/15/16/17 hexose transporter |
| YNL336W | 381 | 4 | Hypothetical protein |
| YLR155C | 362 | 4 | ASP3 |
| YDL244W | 340 | 4 | THI5 multigene family |

The cleaning procedure yielded, additionally, 11 clusters with 3 members, 122 clusters with 2 members, and 5838 singlets. The TY proteins are transposable elements.

**Table 3.** Cluster of redundant sequences centred on human titin

| *p* (%) | Length | nrdb entry | Species |
|---|---|---|---|
| 100 (repr) | 26926 | /:trembl\|X90568\|HSTITIN2B_1 gene: “titin”; H.sapiens mRNA for titin protein (clone hh1-hh54) //:gp\|X90568\|1212992 Protein sequence and annotation available soon via Swiss-Prot; available at present via e-mail from LABEIT@EMBL-Heidelberg.DE [Homo sapiens] | human |
| 100 | 26926 | /:pironly\|I38344\|I38344 titin - human | human |
| 94 | 6875 | /:trembl\|X64696\|OCTITINR_1 product: “titin”; O.cuniculus mRNA for titin //:gp\|X64696\|1208922 titin [Oryctolagus cuniculus] | rabbit |
| 95 | 6805 | /:pironly\|S20901\|S20901 titin - rabbit (fragment) | rabbit |
| 100 | 4731 | /:pironly\|S20898\|S20898 titin - human (fragments) | human |
| 99 | 4650 | /:trembl\|X69490\|HSTITIN_1 gene: “titin”; product: “titin”; H.sapiens mRNA for titin //:gp\|X69490\|407139 titin [Homo sapiens] | human |
| 99 | 3100 | /:trembl\|X64697\|HSTITINC3_1 product: “titin”; H.sapiens mRNA titin (subclone C37) //:gp\|X64697\|37195 titin [Homo sapiens] | human |
| 95 | 1802 | /:trembl\|U28657\|OC28657_1 product: “titin”; Oryctolagus cuniculus cardiac titin mRNA, Z-line region, partial cds. //:gp\|U28657\|1145880 titin [Oryctolagus cuniculus] | rabbit |
| 99 | 1100 | /:trembl\|X64698\|HSTITINC2_1 product: “titin”; H.sapiens mRNA titin (subclone C26/C18) //:gp\|X64698\|37193 titin [Homo sapiens] | human |
| 96 | 1016 | /:trembl\|X17329\|OCTTNAB_1 product: “Titin (1016 AA)”; Rabbit mRNA for titin (partial) (A-band) //:gp\|X17329\|930251 Titin (1016 AA) [Oryctolagus cuniculus] | rabbit |
| 95 | 1000 | /:trembl\|X17330\|OCTTNIB_1 product: “titin (1000 AA)”; Rabbit mRNA for titin (partial) (A-band) //:pironly\|I46521\|I46521 titin - rabbit (fragment)//:gp\|X17330\|930252 titin (1000 AA) [Oryctolagus cuniculus] | rabbit |
| 95 | 572 | /:trembl\|M97767\|SSTITIN_1 product: “titin”; Pig titin mRNA, partial cds. //:gp\|M97767\|164693 titin [Sus scrofa] | pig |
| 94 | 541 | /:trembl\|M98338\|OCTITN_1 product: “titin”; Oryctolagus cuniculus cytoskeletal titin mRNA, partial cds. //:gp\|M98338\|165745 titin [Oryctolagus cuniculus] | rabbit |
| 97 | 531 | /:trembl\|X64699\|HSTITIN27_1 product: “titin”; H.sapiens titin gene (subclone hum278+279) //:gp\|X64699\|37191 titin [Homo sapiens] | human |
| 95 | 531 | /:trembl\|X64700\|MMTITIN_1 product: “titin”; M.musculus titin gene //:pironly\|S20900\|S20900 titin - mouse (fragment)//:gp\|X64700\|54808 titin [Mus musculus] | mouse |
| 90 | 393 | /:trembl\|X59596\|OCTITIN_1 product: “titin”; Rabbit titin mRNA //:pironly\|S16844\|S16844 titin - rabbit (fragment)//:gp\|X59596\|1723 titin [Oryctolagus cuniculus] | rabbit |
| 99 | 214 | /:trembl\|X98114\|HSTITS_1 product: “titin Z-disc”; H.sapiens mRNA for cardiac titin, clone ZisS | human |
| 100 | 213 | /:gp\|X98114\|1508778 titin Z-disc [Homo sapiens] | human |
| 94 | 207 | /:trembl\|L38717\|RNTITI_1 product: “titin”; Rattus norvegicus (clone 1) titin mRNA, 5 end of cds. //:pironly\|A56190\|A56190 titin - rat (fragment)//:gp\|L38717\|793757 titin [Rattus norvegicus] | rat |
| 96 | 164 | /:trembl\|M95596\|OCTITINA_1 product: “titin”; Oryctolagus cuniculus titin muscle protein mRNA, partial cds. //:gp\|M95596\|495066 titin [Oryctolagus cuniculus] | rabbit |
| 100 | 98 | /:pdb\|1NCT\|1NCT titinfragment: n-terminally extended module m5; (connectin, nextm5)//:pdb\|1NCU\|1NCU titinfragment: n-terminally extended module m5; (connectin, nextm5) | human |

**Table 3.** *Continued*

| *p* (%) | Length | nrdb entry | Species |
|---|---|---|---|
| 100 | 97 | /:pironly\|S63665\|S63665 titin protein - human (fragment) | human |
| 100 | 91 | /:pdb\|1TNM\|1TNM Titin module m5 (connectin) (NMR, minimized average structure)//:pdb\|1TNN\|1TNN Titin module m5 (connectin) (NMR, 16 structures) | human |
| 96 | 89 | /:pdb\|1TIT\|1TIT titin, i27(connectin i27, titin ig repeat 27)//: pdb\|1TIU\|1TIU titin, i27(connectin, i27, titin ig repeat 27) | human |
| 94 | 72 | /:trembl\|U89530\|RNU89530_1 product: “titin protein homolog”; Rattus norvegicus titin protein homolog mRNA, partial cds. //: gp\|U89530\|1881715 titin protein homolog [Rattus norvegicus] | rat |
| 92 | 63 | /:trembl\|U89531\|RNU89531_1 product: “titin protein homolog”; Rattus norvegicus titin protein homolog mRNA, partial cds. //: gp\|U89531\|1881717 titin protein homolog [Rattus norvegicus] | rat |
| 93 | 60 | /:trembl\|D85838\|BTCON1_1 product: “connectin (titin)”; Bovine DNA for connectin (titin), partial cds. //:gp\|D85838\|1395147 connectin (titin) [Bos taurus] | cow |
| 97 | 60 | /:trembl\|D85840\|SSCON3_1 product: “connectin(titin)”; Pig muscle DNA for connectin(titin), partial cds. //:gp\|D85840\|1395166 connectin (titin) [Sus scrofa] | pig |
| 98 | 51 | /:trembl\|D85841\|SSCON4_1 product: “connectin(titin)”; Pig muscle DNA for connectin(titin), partial cds. //:gp\|D85841\|1395168 connectin (titin) [Sus scrofa] | pig |
| 96 | 51 | /:trembl\|D85839\|BTCON2_1 product: “connectin (titin)”; Bovine DNA for connectin (titin), partial cds. //:gp\|D85839\|1395149 connectin (titin) [Bos taurus] | cow |
| 100 | 38 | /:trembl\|X83270\|HSTITINMP_1 gene: “titin”; H.sapiens mRNA for titin,partial CDS //:gp\|X83270\|602580 titin gene product [Homo sapiens] | human |

Sequence identity (*p*) is with respect to the representative (human titin gene at the top). Merged duplicate entries from various databases are shown as ‘/:database|accession|identifier definition/’.

## *Limitations*

A main purpose of having a reduced size database is to speed up searches with new query sequences. The speed-up is valuable whenever there is a match to the query sequence in the database, which will be most of the time. Whenever there is not a match, one cannot be certain whether a marginal hit has been overlooked which would be picked up in the complete database, and users are advised to rerun the search against a complete database in these cases. However, due to natural selection, proteins do not drift randomly in sequence space and, in practice, we expect the detection of marginal hits to be triggered by certain highly conserved regions present in every member of a protein family.

The representative set at the 90% sequence identity threshold can be used as a starting point to further clustering of the sequence database which need not necessarily be in terms of sequence identity. Using the present algorithm, it is impractical to lower the threshold much from 90%. Equations (4–5) show that complete peptide composition filters can only be obtained for relatively high values of *p*. The sequence identity threshold could be lowered slightly by using gapped tuples (at the cost of increased memory consumption) or, if compositionally biased segments were masked [12], shorter than decapeptides might be sufficiently selective in the first filter [equation (4)]. The selectivity of the second filter [equation (5)] could be increased by taking order constraints (sequence offsets) into account, as is done in Fasta (Pearson and Lipman, 1988), Flash (Rigoutsos and Califano, 1994) and Blast (Altschul *et al.*, 1997).

Depending on the application, one may want to use a higher or lower threshold to define redundancy. For example, SWIR13 (ftp://ftp.sanger.ac.uk/pub/databases/swir/) aims at 95% sequence identity and contains 154 000 sequences (with 16% redundancy remaining; release of 27 August 1997). The OWL database is generated using more stringent segment mismatch criteria and contains 190 000 sequences (release of 17 August 1997). The curators of the OWL database particularly emphasize the validation of accepted sequence entries in order to create a high-quality replacement database (Bleasby and Wootton, 1990). Our method makes no effort to distinguish genuine biological variation from artefacts, as illustrated by the cluster of mammalian titin sequences (Table 3). However, with well-defined datasets such as genomes, our algorithm can be used to scan quickly for evolutionarily recent gene duplications, as in the yeast example (Table 2), or to align the genomes of microbial strains.

In building the representative set, we required that redundant sequences must be fully contained in the representative sequence. This ensures that the representative set contains at least one example of all 'unique' sequence segments that occur in the original database. However, many representatives may share regions of extremely high similarity, but embedded in different sequences so that overall sequence identity falls below the 90% threshold. In some cases, we see abrupt changes between dissimilar and nearly identical segments which are suggestive of possible frameshifts in the conceptual translation of one or other sequence (Brown *et al.*, 1997). In other cases, we see multiple conserved domains that recur in different contexts in other proteins. The problem of domains is accentuated if the concept of a representative set is used at larger sequence distances. For an economic yet complete description of the protein universe, it will be necessary to break up the sequences into domains.

### *Resolving the 'annotation catastrophe' by family classification*

The non-redundant search database speeds up homology searches which aim at assigning probable functions to anonymous new sequences. Annotation by similarity typically is based on examining the list of hits produced by database search programs such as Blast or Fasta, and picking the first informative hit. A number of researchers have expressed concern about the quality and integrity of functional annotation by similarity. In particular, with current sequence databases it is not possible to trace back a chain of annotations by similarity, nor to propagate changes in annotation to dependent sequence entries. Fortunately, family classification provides a way out of this conceptual muddle. We propose to use sequence data for similarity searching, with the goal of placing the query sequence in a homologous family; each member of the family will then have pointers to what experimental information is available on individual members.

Although we only cluster very similar sequences here, there are numerous examples of clusters with specific functional annotation for some member sequences while others are merely labelled hypothetical proteins. To avoid loss of functional information in the non-redundant database compared to the original, we describe the clusters around the representative sequences by a brief functional summary derived from the annotations of all member sequences. The idea is to select, from a list of alternative sequence definitions, the most relevant one(s) while ignoring non-informative definitions. Here, we used a negative list of words for exclusion (e.g. putative proteins or gene products composed of letters and numbers). The high-quality annotations of Swissprot were given precedence over those from other databases. Any known structure (from the PDB) is also reported. We look forward to applying more sophisticated methods for automatic annotation (Andrade and Valencia, 1997).

## Acknowledgements

We thank Michael Scharf for discussion and earlier efforts along these lines, and Christophe Leroy for database updates.

## References

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

Andrade,M.A. and Valencia,A. (1997) Automatic annotation for biological sequences by extraction of keywords from MEDLINE abstracts. Development of a prototype system. *ISMB*, **5**, 25–32.

Bairoch,A. and Apweiler,R. (1996) The SWISS-PROT protein sequence data bank and its new supplement TrEMBL. *Nucleic Acids Res*., **24**, 21–25.

Bernstein,F.C., Koetzle,T.F., Williams,G.J.B., Meyers,E.F., Brice,M.D., Rodgers,J.R., Kennard,O., Shimanouchi,T. and Tasumi,M. (1977) The protein data bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol*., **112**, 535–542.

Bleasby,A.J. and Wootton,J.C. (1990) Construction of validated, non-redundant composite protein sequence databases. *Protein Eng*., **3**, 153–159.

Brown,N.P., Sander,C. and Bork,P. (1998) Frame: detection of genomic sequencing errors. *Bioinformatics*, **14,** 367–371.

Califano,A. and Rigoutsos,I. (1993) FLASH: A fast look-up algorithm for string homology. *ISMB*, **1**, 56–64.

Goffeau,A. *et al.* (1996) Life with 6000 genes. *Science*, **274**, 546–567.

Grillo,G., Attimonelli,M., Liuni,S. and Pesole,G. (1996) CLEANUP: a fast computer program for removing redundancies from nucleotide sequence databases. *Comput. Applic. Biosci*., **12**, 1–8.

Hobohm,U., Sander,C., Scharf,M. and Schneider,R. (1992) Selection of representative protein datasets. *Protein Sci.*, **1**, 409–417.

Pearson,W. and Lipman,D. (1988) Improved tools for biological sequence comparison. *Proc. Natl Acad. Sci. USA*, **85**, 2444–2448.

Scharf,M., Schneider,R., Casari,G., Bork,P., Valencia,A., Ouzounis,C. and Sander,C. (1994) GeneQuiz: a workbench for sequence analysis. *ISMB*, **2**, 348–353.

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

Wall,L., Christiansen,T. and Schwartz,R.L. (1996) *Programming Perl*, 2nd edn. O'Reilly and Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.

Wootton,J.C. (1994) Non-globular domains in protein sequences: automated segmentation using complexity measures. *Comput. Chem*., **18**, 269–285.