



Tolerating some redundancy significantly speeds up clustering of large protein databases

Weizhong Li, Lukasz Jaroszewski and Adam Godzik*

The Burnham Institute, 10901 N. Torrey Pines Road, La Jolla, CA 92037, USA

Received on May 1, 2001; revised on July 6, 2001; accepted on July 20, 2001

ABSTRACT

Motivation: Sequence clustering replaces groups of similar sequences in a database with single representatives. Clustering large protein databases like the NCBI Non-Redundant database (NR) using even the best currently available clustering algorithms is very time-consuming and only practical at relatively high sequence identity thresholds. Our previous program, CD-HI, clustered NR at 90% identity in ~ 1 h and at 75% identity in ~ 1 day on a 1 GHz Linux PC (Li *et al.*, *Bioinformatics*, **17**, 282, 2001); however even faster clustering speed is needed because the size of protein databases are rapidly growing and many applications desire a lower attainable thresholds.

Results: For our previous algorithm (CD-HI), we have employed short-word filters to speed up the clustering. In this paper, we show that tolerating some redundancy makes for more efficient use of these short-word filters and increases the program's speed 100 times. Our new program implements this technique and clusters NR at 70% identity within 2 h, and at 50% identity in ~ 5 days. Although some redundancy is present after clustering, our new program's results only differ from our previous program's by less than 0.4%.

Availability: The program and its previous version are available at <http://bioinformatics.burnham-inst.org/cd-hi>

Contact: liwz@burnham-inst.org; adam@burnham-inst.org

INTRODUCTION

The sizes of protein databases are rapidly growing because of output from the large-scale genome projects, and as a result, it is becoming more difficult and time-consuming to manage these databases and to run routine searches. At the same time, many of the new sequences are similar or nearly identical to known proteins and may be splicing variants, proteins from related species or single nucleotide polymorphisms. This information may be crucial for many purposes, but not relevant in others such as the quick identification of newly sequenced proteins. A practical way to resolve this problem is to cluster protein sequences

into groups and then only use a representative sequence or consensus of each group.

There have been several attempts to remove excessive redundancy from protein sequence databases. For example, NCBI has compiled the Non-Redundant protein database (NR) from different databases with the program nrdb (<ftp://ftp.ncbi.nih.gov/pub/nrdb/>), which removes identical sequences. The NCBI's focus was to only remove identical entries, so the resulting NR database is still highly redundant.

NRDB90 (Holm and Sander, 1998) is a database where sequences with $\geq 90\%$ identities to a representative were removed. The program that produced NRDB90 was implemented with a lookup table and short-word filters, and took ~ 30 h to produce a database that was half the size of the original NR, speeding up all sequence searches by 50% with little or no loss in accuracy. Specific solutions used in the NRDB90 algorithm limited its use to sequence identity threshold larger than 90%.

The RSDB project (Park *et al.*, 2000) clustered the NR database at sequence identities from 20 to 99% (<ftp://ftp.ebi.ac.uk/pub/contrib/jong/RSDB/>). The RSDB databases were obtained by combining the NRDB90 algorithm with a large number of BLAST searches, which is very time-consuming; the reported CPU time for building RSDBs from NR in March 1999 was about 10 weeks.

Clustering saves time in database searching and simplifies the subsequent organizing of results. In some cases using a clustered database increased the accuracy of distant homology recognition for programs such as PSI-BLAST (Altschul *et al.*, 1997). In the RSDB study, it was found that even clustering the database at 50% identity did not compromise homology detection in comparison to the full NR. We have also tested clustered databases in fold recognition using PSI-BLAST and other search strategies, and have found that clustering followed by a few other simple steps can double the recognition rate and reduce the search time by 10-fold (manuscript in preparation). With many research groups devoting significant computational resources to BLASTing new sequences against databases of known proteins, there are significant savings to be realized with the smaller

*To whom correspondence should be addressed.

(clustered) databases and new searching strategies.

However, the existing clustering methods are too time-consuming for regular applications. Clustering can take weeks, making it difficult to keep up with the daily updates of major sequence databases. Inspired by NRDB90, we developed a program called CD-HI (Li *et al.*, 2001) that clusters the NR database at a high speed, consuming less than 1 day of CPU time to cluster NR at a threshold of 75% identity.

The techniques for speeding up clustering in CD-HI were short-word filters and an index table. We also employed a strict algorithm that guaranteed the removal of redundancy above a specified level. Subsequent analysis of the algorithm performance suggested that the program used a significant percentage of its running time to eliminate very rare situations. In this study, we show that permitting some redundancy allows for more efficient use of the short-word filter and makes clustering significantly faster. Our new program, CD-HIT, clusters NR at thresholds of 70 and 50% within 2 h and 1 day, respectively.

MATERIALS AND METHODS

Algorithm

The basic algorithm for database clustering in this study is the greedy incremental algorithm (Hobohm *et al.*, 1992), which selects representative protein sequence sets, and is also used in NRDB90 and CD-HI.

According to this algorithm, a final clustered database is composed of representative sequences with pairwise similarities below a specified threshold. So a sequence having an identity greater than the threshold to a chosen representative is considered redundant and is removed from the database. Sequences are first sorted in order of decreasing length, and the longest sequence is selected as the first representative. Each remaining sequence is then compared to all the existing representatives, and if it is not redundant to all old representatives then it is added as a new representative. The sequence identity is computed as the number of identical residues in the alignment divided by the length of the shorter sequence.

This algorithm guarantees that all representatives are compared and that all deleted sequences are compared to their corresponding representatives. It doesn't care whether the non-representative sequences in the same cluster are similar or not. For example, two short sequences can be aligned at different regions of a long representative sequence.

Short-word filtering system

An explicit alignment is required to determine the sequence identity of two sequences. According to the clustering strategy described in the previous section,

the process needs sequence alignments between all representatives and also the alignments between redundant sequences and their representatives. With current computer power, such calculations are not practical for a large protein database like NR. The purpose of filters is to decide whether the identity between two sequences is above or below a threshold without aligning them, thereby saving CPU time.

A short-word filtering mechanism can be used to avoid the unnecessary alignments. In order for two sequences to have a certain sequence identity, it is necessary to have a certain number of identical words. If they don't, no alignment between them is needed. In NRDB90, the authors used a decapeptide as the primary filter, because two sequences with over 90% identity must have at least one identical decapeptide. In our previous work, we used short-word filters, from dipeptide to pentapeptide. The advantage of pentapeptides is that they can be easily indexed, because the index table has only 4 million entries (total number is 21^5 computed as each residue has 21 possibilities, 20 amino acids and 'X') and can be easily handled by a PC workstation.

Theoretically, each short-word filter is only suitable for certain thresholds: the decapeptide filter is only valid for thresholds greater than 90%; and the pentapeptide filter's lowest threshold is 80%, because two sequences having 80% sequence identity don't need to share a single pentapeptide: for instance when they differ at every 5th amino acid along the alignment. Our old program used pentapeptide and tetrapeptide filters for thresholds above 85 and 80% respectively, and used tripeptide and dipeptide filters for thresholds above 75 and 65% respectively. Unfortunately, when the length of the short-word filter decreases, the filtering efficiency drops significantly.

Statistically based filtering

Two sequences rarely differ at each 5th position along an alignment; usually aligned sequences share more identical short-words than the theoretically required number. So we inspected sequence alignments from a large set of BLAST searches at different identity levels and observed the numbers of identical short words between aligned pairs. We first used CD-HI to cluster the Pfam domain (Bateman *et al.*, 2000; <http://pfam.wustl.edu/>) and NR databases at 80% identity in order to avoid statistical bias from highly populated families, and named the 80% identity-clustered databases Pfam80 and NR80 (we will use this convention in this paper to name a clustered database). We then randomly selected 15 000 queries from Pfam80 for BLAST searches against NR80 and obtained 330 000 non-redundant alignments with sequence identities of 40–100%.

Using these alignments, we found that identical short-words were more common than required by CD-HI's filter

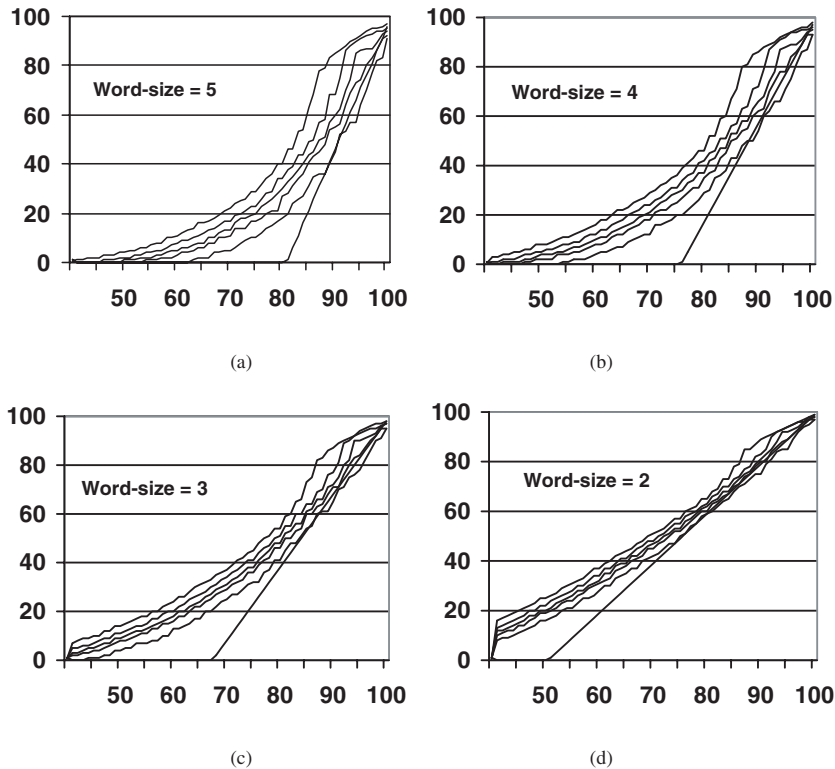


Fig. 1. The observed minimal short-word identity versus sequence identity of alignments. The lines from top to bottom are obtained from 60, 80, 90, 95 and 99% of all the alignments. The straight line is the theoretically required short-word identity (also see text).

for a given threshold (see the Results section for specific results and discussion). Based on this statistical data, we thought that short-word filters could be used for lower thresholds than in CD-HI.

One problem with using inadequate filters is that some redundant sequences would be missed and the final database would include more redundancy than the stated threshold. But because the longer word filters dramatically increases the speed we sought to define an optimal point that permits rapid clustering while minimizing redundancy.

RESULTS

Short-word statistics

The *short-word identity* of an alignment is defined as the number of identical short-words in two aligned sequences divided by the length of the alignment and is represented as a percentage. Figure 1 shows the short-word identities graphed versus the sequence identities of 330 000 non-redundant alignments.

We divided all of the alignments according to their sequence identities into 1% interval stacks, and sorted each stack in order of decreasing short-word identities.

The minimal short-word identity was then read for a certain percentage of the alignments from the top of the stack. In Figure 1, we read five series of data at 60, 80, 90, 95 and 99% of the alignments, and plotted them with the theoretically minimal short-word identity as a reference. This reference value should be lower than the five observed data sets. The one exception is at identities approaching 100%, because the reference value is calculated for alignments of infinite lengths, while the five observed sets are from real alignments.

Actual alignments between homologous sequences usually have conserved and variable parts. It is very rare that the differences are distributed evenly throughout the alignment. This effect is apparent in Figure 1: the short-word identities are higher than the reference value, and the differences become greater as the sequence identity lowers. For example, at 70% sequence identity, the short-word identity for a pentapeptide of 99, 95, 90, 80 and 60% of the alignments are above 5, 11, 13, 17, and 22% respectively (values not printed in Figure 1a).

Implementation

The short-word identities shown above can be used for filtering because they are sampled from a large set of non-

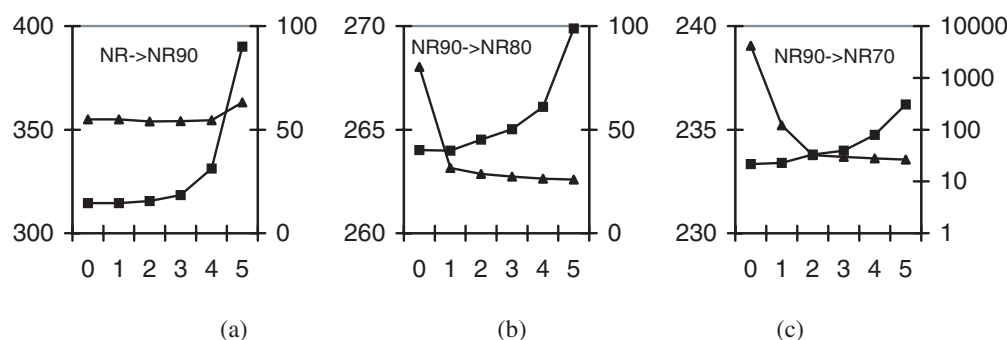


Fig. 2. Clustering of NR database with different tolerances. In (a–c), the NR database was clustered into NR90, NR80 and NR70. The x -axis is the tolerance level (see text). The two categories of data for the left and right y -axes are the size of the clustered database in 1000 s of sequences (■) and the CPU time in minutes (▲). The y -axis in (c) is logarithmic because of the large differences in CPU time.

redundant alignments. However, these filters may fail to recognize some redundant sequences because these five data sets represent 99, 95, 90, 80 and 60% of protein alignments. As a result, some redundancy in the final clustered database is inevitable. We denote this missed redundancy as *tolerance level* and tolerance levels of 1, 2, 3, 4 and 5 are related with the five data sets covering 99, 95, 90, 80 and 60% of protein alignments in Figure 1. The tolerance level of 0 is the theoretically required short-word identity.

According to Figure 1, we believe it possible to use short-word filters at lower clustering thresholds than with CD-HI. For some cluster thresholds, longer short-words can replace CD-HI's shorter one so that the clustering speed is greatly increased. For example, at a threshold of 70%, CD-HI needs a tripeptide filter, but a pentapeptide filter can be employed based on statistical data.

We wrote a program in C++ implemented with the statistically based data on a RedHat Linux system. The program's input is a protein database in FASTA format, it outputs a FASTA file containing representative proteins. The members of each cluster are listed in another output file. The tolerance level and type of short-word can be specified to the program through command line.

Speed and quality

We tested this program on a PC with a 1 GHz PIII processor and 1 GB RAM running RedHat Linux 7.0. Three target databases were PDB (Berman *et al.*, 2000) from 20 September, 2000, Pfam (Bateman *et al.*, 2000) from 18 January, 2001, and NR from 20 September, 2000. Table 1 lists the clustering results for the three databases at a tolerance level of 2 when using statistically based filters. The results were compared with the old program CD-HI using strict filters, and for each threshold, the best short-word was used to yield the fastest speed.

This table clearly shows CPU time is a function of

threshold and length of short-word, and that gains occur when the filters are changed, and these gains are greatest at lower thresholds and for larger databases. A pentapeptide filter can be used effectively down to a 70% identity, a tetrapeptide filter to 55–60% identities, and a tripeptide to a 50% threshold. We did not attempt to cluster NR at levels below 50%, because the gain over the RSDb would be minimal.

In terms of the final size of the clustered database, the filter based on statistical data produces similar results to the strict filter, with differences below 0.4% for all the clustering tests.

This new filtering method is significantly faster in CPU time for thresholds below 75%. In some cases, we observed almost 100-fold increases in speed. More importantly, this new filter method pushes the threshold of clustering that could be applied routinely to the NR database down to 50%, a level that cannot be reached with the previous method.

We tested different tolerance levels for the NR database, and show the results of three clustering processes, NR->NR90, NR90->NR80 and NR90->NR70, in Figure 2. In each clustering, we used the short-word filter that yielded the fastest speed. At a 90% threshold, the CPU time fluctuates very little, while the number of redundant sequences rapidly increase as the tolerance level raises. At an 80% threshold, we observed a much longer CPU time for tolerance level 0 (strict filter) than the other thresholds (1–5), because the tetrapeptide had to be used in a strict algorithm while the pentapeptide was sufficient for the others. In Figure 2c, the change in CPU time is exponential, while the size of the database show a convergence trend.

Generally, a tolerance level of 2 is acceptable for overall situations above with minimal values in both CPU time and redundancy.

An interesting question is what sequences are missed

Table 1. Comparison of clustering of PDB, Pfam and NR using the old strict filter and the statistically based filter at tolerance level of 2

Database	Threshold (%)	W ^a	Statistically based filter		W ^a	Strict filter		Ratio ^c of time	Ratio ^d of size
			Clusters	Time ^b		Clusters	Time ^b		
PDB	90	5	4 194	7.6 s	5	4 187	8.2 s	1.1	0.998
	85	5	3 969	8.0 s	5	3 958	8.0 s	1.0	0.997
8732 sequences	80	5	3 798	7.8 s	4	3 785	8.1 s	1.0	0.997
	75	5	3 655	8.0 s	3	3 650	13.2 s	1.7	0.999
1850 235 letters	70	4	3 543	8.0 s	3	3 538	26.8 s	3.4	0.999
	6	4	3 423	7.7 s	2	3 419	113.9 s	14.8	0.999
60	55	3	3 282	10.4 s					
	50	3	3 155	18.5 s					
40	50	3	3 020	56.3 s					
	40	2	2 732	142.0 s					
Pfam	90	5	82 046	5.0 min	5	81 951	4.9 min	1.0	0.999
	85	5	77 913	4.8 min	5	77 829	5.2 min	1.1	0.999
111 934 sequences	80	5	74 886	4.7 min	4	74 747	10.2 min	2.2	0.998
	75	5	72 091	5.4 min	3	72 011	78.6 min	14.6	0.999
38 089 960 letters	70	5	69 514	5.8 min	3	69 458	371.4 min	64.0	0.999
	65	4	67 180	13.7 min	2	67 209	1093.4 min	79.8	1.000
60	60	4	64 616	48.7 min					
	55	3	62 072	73.7 min					
50	50	3	59 458	732.1 min					
	40	2	52 842	4819.5 min					
NR	90	5	315 502	54.0 min	5	314 512	54.9 min	1.0	0.997
	85	5	284 568	28.8 min ^e	5	283 673	30.6 min ^e	1.1	0.997
563 276 sequences	80	5	264 999	27.0 min ^e	4	264 011	80.3 min ^e	3.0	0.996
	75	5	248 587	29.7 min ^e	3	247 899	1129.8 min ^e	38.0	0.997
177 028 588 letters	70	5	233 767	32.1 min ^e	3	233 333	4151.0 min ^e	129.3	0.998
	65	4	221 259	89.0 min ^e					
60	60	4	207 898	490.7 min ^e					
	55	4	195 181	733.1 min ^e					
50	50	3	182 262	7055.7 min ^e					

^aW means the length of short word filter. ^bCPU times of clustering are in seconds (s) or minutes (min). ^cRatio of CPU times (strict filter/statistically based filter). ^dRatio of size of clustered databases (strict filter/statistically based filter). ^eIn order to save time, these databases are clustered from NR90.

by the statistically based filters. We compared a clustered database NR80 prepared with the statistically based filter at a tolerance level of 2 (the optimal value) with the NR80 calculated with the old program. Figure 3 shows that most of the extra redundant sequences missed by the statistically based filter approach are very short sequences: about half are shorter than 20 amino acids. Because the statistical data is derived from a large number of alignments, therefore it reflects the averages for alignments of various lengths. A shorter sequence pair has a larger probability that differences will be evenly distributed. So the short-word identity is lower than that from a longer alignment of the same sequence identity.

DISCUSSION AND CONCLUSIONS

If we tolerate a little redundancy in an output database, then clustering protein databases requires much less CPU time at sequence identity thresholds below 75%. Such extra redundancy will not make any practical difference

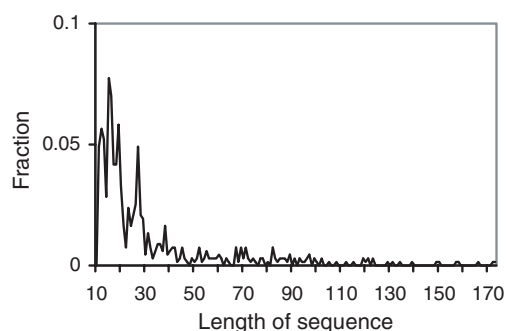


Fig. 3. Distribution of the redundant sequences missed in the statistically based filters.

because the amount of redundancy is very limited.

With our new method it is feasible to do daily updates for clustered versions of large databases, since clustering NR at a 60% threshold takes less than 10 h. However,

for thresholds below 50%, it is still too slow to regularly update the NR database; clustering at or below this threshold requires more advanced algorithms. Another improvement regarding clustering of databases is to handle multi-domain proteins. The sequence identity measured in the current method is based on a global similarity, so two multiple-domain proteins cannot be grouped into a single cluster even if they share an identical domain. This conserves the information about domain arrangements while keeping the redundancy between domains. It is better to handle this type of redundancy in many situations. We are currently developing algorithms for clustering at lower thresholds and for treating multi-domain sequences.

The source code of the program, along with its previous version CD-HI, is available from <http://bioinformatics.burnham-inst.org/cd-hi>.

ACKNOWLEDGEMENT

The research described in this manuscript was supported by NIH grant GM60049.

REFERENCES

- Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
- Bateman,A., Birney,E., Durbin,R., Eddy,S.R., Howe,K.L. and Sonnhammer,E.L. (2000) The Pfam protein families database. *Nucleic Acids Res.*, **28**, 263–266.
- Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat,T.N., Weissig,H., Shindyalov,I.N. and Bourne,P.E. (2000) The protein data bank. *Nucleic Acids Res.*, **28**, 235–242.
- Hobohm,U., Scharf,M., Schneider,R. and Sander,C. (1992) Selection of representative protein data sets. *Protein Sci.*, **1**, 409–417.
- Holm,L. and Sander,C. (1998) Removing near-neighbour redundancy from large protein sequence collections. *Bioinformatics*, **14**, 423–429.
- Li,W., Jaroszewski,L. and Godzik,A. (2001) Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, **17**, 282–283.
- Park,J., Holm,L., Heger,A. and Chothia,C. (2000) RSDB: representative protein sequence databases have high information content. *Bioinformatics*, **16**, 458–464.