J. Biomedical Science and Engineering, 2009, 2, 621-625 JBiSE
doi: 10.4236/jbise.2009.28090 Published Online December 2009 (http://www.SciRP.org/journal/jbise/).

FastCluster: a graph theory based algorithm for removing redundant sequences

Peng-Fei Liu[2], Yu-Dong Cai[1*], Zi-Liang Qian[3,4], Sheng-Yu Ni[5], Liu-Huan Dong[5], Chang-Hong Lu[6,7], Jin-Long Shu[6] , Zhen-Bing Zeng[2,5], Wen-Cong Lu[8*]

[1]Institute of Systems Biology, Shanghai University, Shanghai, China;
[2]Software Engineering Institute of East China Normal University, East China Normal University, Shanghai, China;
[3]Bioinformatics Center, Key Lab of Systems Biology, Shanghai Institutes for Biological Sciences, Chinese Academy of Sciences, Shanghai, China;
[4]Graduate School of the Chinese Academy of Sciences, Beijing, China;
[5]CAS-MPG Partner Institute for Computational Biology, Shanghai Institutes for Biological Sciences, Chinese Academy of Sciences, Shanghai, China;
[6]Department of Mathematics, East China Normal University, Shanghai, China;
[7]Institute of Theoretical Computing, East China Normal University, Shanghai, China;
[8]Department of Chemistry, Shanghai University, Shanghai, China.
Email: perphyliu@gmail.com; *cai_yud@yahoo.com.cn; zl_qian@yahoo.com.cn; sendru@gmail.com; dlh@picb.ac.cn; chluclear@gmail.com; jlshu@math.ecnu.edu.cn; zbzeng@sei.ecnu.edu.cn; *wclu@shu.edu.cn

Received 1 August 2009; revised 30 August 2009, accepted 3 September 2009.

ABSTRACT

In many cases, biological sequence databases contain redundant sequences that make it difficult to achieve reliable statistical analysis. Removing the redundant sequences to find all the real protein families and their representatives from a large sequences dataset is quite important in bioinformatics. The problem of removing redundant protein sequences can be modeled as finding the maximum independent set from a graph, which is a NP problem in Mathematics. This paper presents a novel program named FastCluster on the basis of mathematical graph theory. The algorithm makes an improvement to Hobohm and Sander's algorithm to generate non-redundant protein sequence sets. FastCluster uses BLAST to determine the similarity between two sequences in order to get better sequence similarity. The algorithm's performance is compared with Hobohm and Sander's algorithm and it shows that Fast- Cluster can produce a reasonable non-redundant pro- tein set and have a similarity cut-off from 0.0 to 1.0. The proposed algorithm shows its superiority in generating a larger maximal non-redundant (independent) protein set which is closer to the real result (the maximum independent set of a graph) that means all the protein families are clustered. This makes Fast-Cluster a valuable tool for removing redundant protein sequences.

Keywords: Blast; Graph Theory; Redundant Sequences; CD-HIT

1. INTRODUCTION

Recently, with the explosion of biological sequence data, many biological sequence databases have redundant sequences which can cause problems for data analysis. These redundant sequences cannot provide valuable information for analysis but detracts from the statistical significance of interesting hits. Moreover, processing these redundant sequences often requires more time and computational resources. Removing redundant sequences is undoubtedly very helpful for performing statistical analysis and accelerating extensive database searching [1]. And it is also a way to obtain the real protein families and their representatives from a large sequences dataset. Therefore, it is necessary to develop an appropriate algorithm to remove redundant sequences from a biological sequence database.

Hobohm and Sander's algorithm is a widely used algorithm in many redundant sequence removing programs. Hobohm and Sander's algorithm was firstly introduced by U.Hobohm *et al.* of EMBL laboratory in 1992. In 1998, Lissa Holm and Chris Sander developed a program based on this algorithm to generate a non-redundant protein database NRDB90 [2]. After that, other researchers developed some programs for removing redundant sequences on the basis of Hobohm and Sander's algorithm, such as CD-HIT and PISCES.

CD-HIT [3,4] is a well-known program for processing large sequence databases efficiently. It is fast and flexible and can generate a representative set based on an incremental greedy algorithm introduced by Hobohm and Sander [5,6]. It uses short word filtering to determine the similarity between two sequences rather than performing an actual sequence alignment. However, the results generated by short word filtering are not accurate to some degree. The lowest threshold of CD-HIT is around 40% and it is not suitable for removing redundancy on lower threshold. PISCES [7] is a public server for culling sets of protein sequences from the Protein Data Bank. It determines sequence similarity by PSI-BLAST [8] alignments which are more accurate, and it also uses a structural quality criterion to cull sequences from a sequence database.

Hobohm and Sander's algorithm has the advantage of being simple and fast. But the result set generated by this algorithm is not large enough since some non-redundant sequences may also be removed.

FastCluster, introduced in this paper, uses BLAST [9] to determine sequence similarity, which is a general sequence alignment tool and can provide better sequence similarity than word filtering. FastCluster makes improvements to Hobohm and Sander's algorithm and can get a larger non-redundant protein dataset, which means more protein families can be clustered.

2. METHODS

2.1. Hobohm and Sander's Algorithm

Hobohm and Sander's algorithm sorts all sequences by length in descending order to generate an ordered sequence set S. Then similar sequences will be put together into the same cluster. The longest sequence is added into the first cluster (initially empty), which is also the representative of the cluster, and then all the other sequences are compared with the representative. If the similarity between a sequence and the representative is above a threshold then it will be included into the same cluster as the representative's, otherwise a new cluster will be created with it as the representative. Every remaining sequence will be processed in the same way, either as the representative of a new cluster if the similarity between it and any representative is below the threshold, or included into some existing cluster if it is similar to the cluster's representative.

2.2. Graph Theory Based Algorithm

In order to make some improvements to Hobohm and Sander's algorithm, a new algorithm using maximum independent set of graph theory to generate a representative set is developed. Firstly all the sequences are clustered simply and the first sequence of each cluster is the temporary representative sequence of the cluster. Then the maximum independent set of each cluster (excluding the representative sequence) is figured out. Finally the maximum independent set of each cluster is processed and the final maximal independent set can be generated.

Based on the algorithm above, FastCluster was written in C++ and tested on Linux platform. The input to the program is a protein sequence set in FASTA format. Three output files can be generated by FastCluster. One is a FASTA file containing a list of representative proteins free from redundancy. Another output file lists the clusters and their members and the third output file contains clusters and the size of each cluster's maximal independent set. FastCluster can be downloaded from http://pcal.biosino.org/FastCluster.html.

2.3. Blast-Based Similarity Score (BSS)

FastCluster uses BLAST to make pair wise sequence alignments. The similarity score between two sequences is determined by the identical percentage of their hits (homologous sequence segments). When a sequence alignment has more than one hit, the percentage of the sum of all hits' identical is calculated to represent the overall similarity score between the two sequences.

A Python script was used to parse BLAST output and construct a BLAST-BASED SIMILARITY SCORE (BSS) matrix. An expectation value parameter 1e-3 is set to filter BLAST output in which the expectation value of each hit is smaller than 1e-3. When calculating BSS, three cases have to be considered: 1) there are no hits found; 2) there is one hit found; 3) there are more than one hit found. Formula 1 below shows how to calculate the BSS.

$$BSS = \begin{cases} 0 & (No\ hit\ found) \\ I/L & (One\ hit\ found) \\ \sum_1^n I / \sum_1^n L & (N\ hits\ found) \end{cases}$$

Formula 1: The formula to calculate BSS. 'I' is short for the length of a hit's identities, and 'L' stands for the length of a hit's length. In case of no hits found, the BSS is regarded as 0. When there is one hit in the BLAST output, the identical percentage is taken as the BSS. On the occasion of more than one hit found, the percentage of the sum of all hits' identities is taken as the BSS.

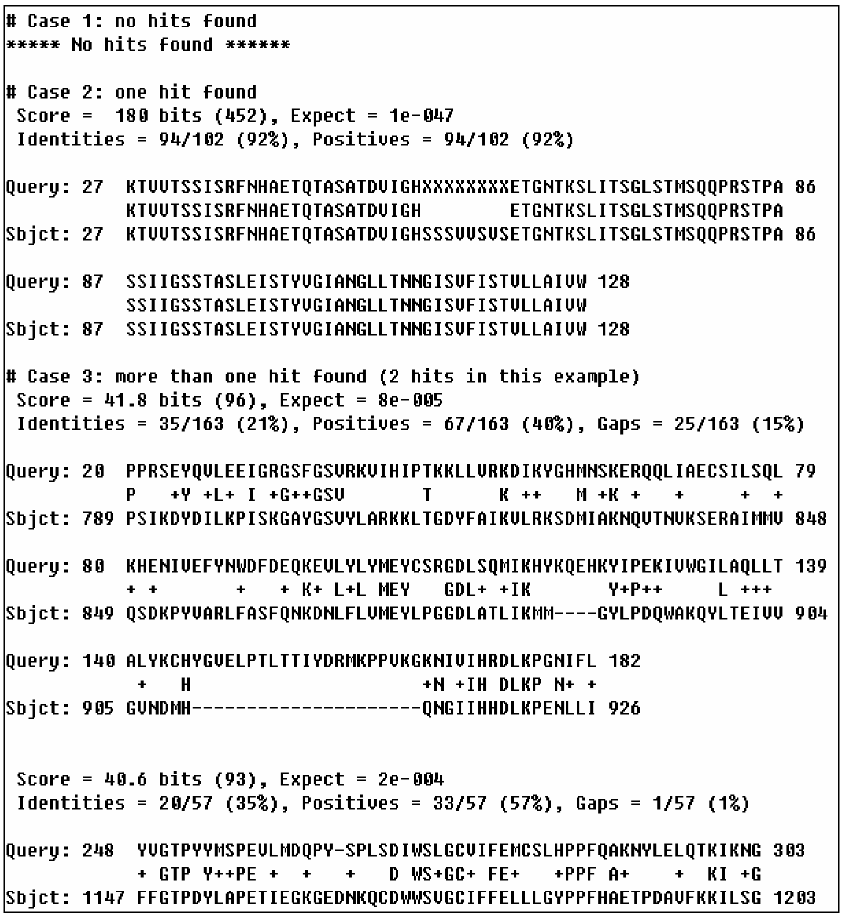
Figure 1 shows an example of how to calculate the BSS from BLAST output.

2.4. Graph Definition

A graph is a mathematical object which is composed of vertices and edges. It is usually used to represent relations between objects. Graphs can be categorized into four types: undirected graphs (or simple graphs), directed graphs, multigraphs and weighted graphs. FastCluster uses an undirected graph to represent relations between protein sequences.

2.5. Clique and Independent Set of a Graph

An undirected graph is denoted by $G = (V, E)$, in which V is the set of vertices and E are the set of edges and every edge is composed of two adjacent vertices in V.



```
# Case 1: no hits found
***** No hits found ******

# Case 2: one hit found
 Score =  180 bits (452), Expect = 1e-047
 Identities = 94/102 (92%), Positives = 94/102 (92%)

Query: 27  KTVVTSSISRFNHAETQTASATDVIGHXXXXXXXXETGNTKSLITSGLSTMSQQPRSTPA 86
            KTVVTSSISRFNHAETQTASATDVIGH        ETGNTKSLITSGLSTMSQQPRSTPA
Sbjct: 27  KTVVTSSISRFNHAETQTASATDVIGHSSSVVSVSETGNTKSLITSGLSTMSQQPRSTPA 86

Query: 87  SSIIGSSTASLEISTYVGIANGLLTNNGISVFISTVLLAIVW 128
            SSIIGSSTASLEISTYVGIANGLLTNNGISVFISTVLLAIVW
Sbjct: 87  SSIIGSSTASLEISTYVGIANGLLTNNGISVFISTVLLAIVW 128

# Case 3: more than one hit found (2 hits in this example)
 Score = 41.8 bits (96), Expect = 8e-005
 Identities = 35/163 (21%), Positives = 67/163 (40%), Gaps = 25/163 (15%)

Query: 20  PPRSEYQVLEEIGRGSFGSVRKVIHIPTKKLLVRKDIKYGHMNSKERQQLIAECSILSQL 79
            P   +Y +L+ I +G++GSV       T      K ++   M +K +   +     +  +
Sbjct: 789 PSIKDYDILKPISKGAYGSVYLARKKLTGDYFAIKVLRKSDMIAKNQVTNVKSERAIMMV 848

Query: 80  KHENIVEFYNWDFDEQKEVLYLYMEYCSRGDLSQMIKHYKQEHKYIPEKIVWGILAQLLT 139
            + +       +   + K+ L+L MEY   GDL+ +IK       Y+P++     L +++
Sbjct: 849 QSDKPYVARLFASFQNKDNLFLVMEYLPGGDLATLIKMM----GYLPDQWAKQYLTEIVV 904

Query: 140 ALYKCHYGVELPTLTTIYDRMKPPVKGKNIVIHRDLKPGNIFL 182
             +   H                     +N +IH DLKP N+ +
Sbjct: 905 GVNDMH---------------------QNGIIHHDLKPENLLI 926


 Score = 40.6 bits (93), Expect = 2e-004
 Identities = 20/57 (35%), Positives = 33/57 (57%), Gaps = 1/57 (1%)

Query: 248  YVGTPYYMSPEVLMDQPY-SPLSDIWSLGCVIFEMCSLHPPFQAKNYLELQTKIKNG 303
             + GTP Y++PE +  +   +   D WS+GC+ FE+   +PPF A+    +  KI +G
Sbjct: 1147 FFGTPDYLAPETIEGKGEDNKQCDWWSVGCIFFELLLGYPPFHAETPDAVFKKILSG 1203
```

Figure 1. An example of parsing BLAST output. Three cases are considered: (1) Case 1, the BSS is 0; (2) Case 2, the BSS is 94/102=0.922; (3) Case 3, the BSS is (35+20)/(163+57) =0.25.

In FastCluster, an undirected graph is defined as follows: any vertex represents a protein sequence; and if two protein sequences have a BSS above the given threshold, there is an edge between them.

A clique C of a graph G is a subset of V, and every vertex in C is adjacent to all the other vertices in C, while an independent set of a graph is a set of vertices, none of which are adjacent. A clique is said to be *maximal* if it is not the subset of any larger clique, and *maximum* if there are no larger cliques in the graph [10]. The complement of a graph G is the graph G' with the same vertex set but whose edge set consists of the edges not present in G. [http://mathworld.wolfram.com/GraphComplement.html]. By taking the complement of a graph, the maximum independent set problem is transformed into the maximum clique problem.

The concept of clique and independent set is illustrated in **Figure 2**.

2.6. Algorithm Procedure

The algorithm used in FastCluster is described in the following steps.

1) Run local BLAST for sequence set S to make pair wise alignments.
2) Parse BLAST output and construct a BSS matrix for all the sequences.
3) Sort all sequences in descending order by length and construct the first cluster with the longest sequence as its representative.
4) Align each remaining sequence in S with the existing representatives. If the BSS (from BSS matrix in step 2) between a sequence with any representative is above a given threshold, then it is included into that cluster, otherwise a new cluster starts with it as representative.
5) Repeat Step 4 until S is empty. Thus, a list of clusters for set S is generated and every cluster has a temporary representative sequence (the first sequence of that cluster).
6) Compute the *maximum* independent set for each cluster.
7) Construct an empty set R. If the maximum inde-

pendent set of the last cluster has more than or equal to 2 sequences, then every sequence in the maximum independent set is put into R, otherwise the representative of the cluster is put into R.

8) Process each remaining maximum independent set from the last one to the first. If a set has more than or equal to 2 sequences which have no edge with any sequence in R, then put these sequences in the maximum independent set into R, otherwise put the cluster's representative into R.

9) Output the final representative non-redundant set R to a file in Fasta format.

2.7. Algorithm Comparison

An algorithm comparison is shown in **Figure 3**. As shown below, S is an ordered sequence set and R is the result sequence dataset. $S(r_i)$ is composed ofsequences which are similar to representative sequence r_i. In **Figure 3**, $C(S(r_i))$ is the maximum independent set of $S(r_i)$, in which no sequence is similar to any other one, and f is a procedure to process the maximum independent set of every cluster.

Both of the algorithms share the same procedure to generate a list of clusters $S(r_1)$, $S(r_2)\ldots S(r_n)$. That is to say, both of them sort all sequences by length in decreasing order firstly to generate an ordered sequence set S and then partition them into different clusters. The difference lies in the way they generate result set R.

Hobohm and Sander's algorithm (**Figure 3(a)**) simply picks up the representative sequence of each cluster and put them together to compose result set R, while FastCluster behaves in a different way.

It searches the maximum independent set for each cluster and get n maximum independent set $C(S(r_1))$, $C(S(r_2))\ldots C(S(r_n))$. Then they are processed in a procedure f to generate the final result set R. f is such a procedure that if the number of sequences in $C(S(r_i))$ is less than 2, then r_i is added to R, otherwise every sequence in $C(S(r_i))$ is put to R on condition that the sequence has no edge with any sequence of current R which grows bigger in an incremental way.

Some $C(S(r_i))$ usually have more than 2 sequences, so after we replace r_i with $C(S(r_i))$, it is clear that R in FastCluster is larger than R in the algorithm of Hobohm and Sander. An example below is used to prove that.

2.8. An Example for Algorithm Comparison

As shown in **Figure 4**, vertices 1,2,3,4,5 and 6 represents 6 different protein sequences respectively. If two protein sequences have a BSS above the given threshold, there is an edge between them. Vertices 1,2,3,4,5 and 6

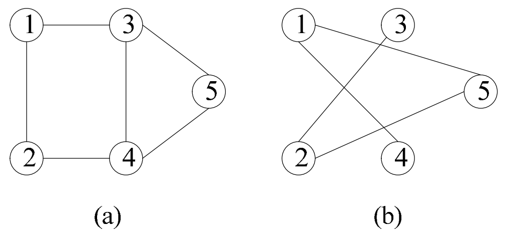


Figure 2. (a) is Graph G; (b) is its complementary graph G'. Vertices set $\{3,4,5\}$ is the maximum clique of G and the maximum independent set of graph G'. Vertices set $\{1,4\}$ or $\{1,5\}$ or $\{2,3\}$ or $\{2,5\}$ is the maximum clique of G' and is also the maximum independent set of graph G.

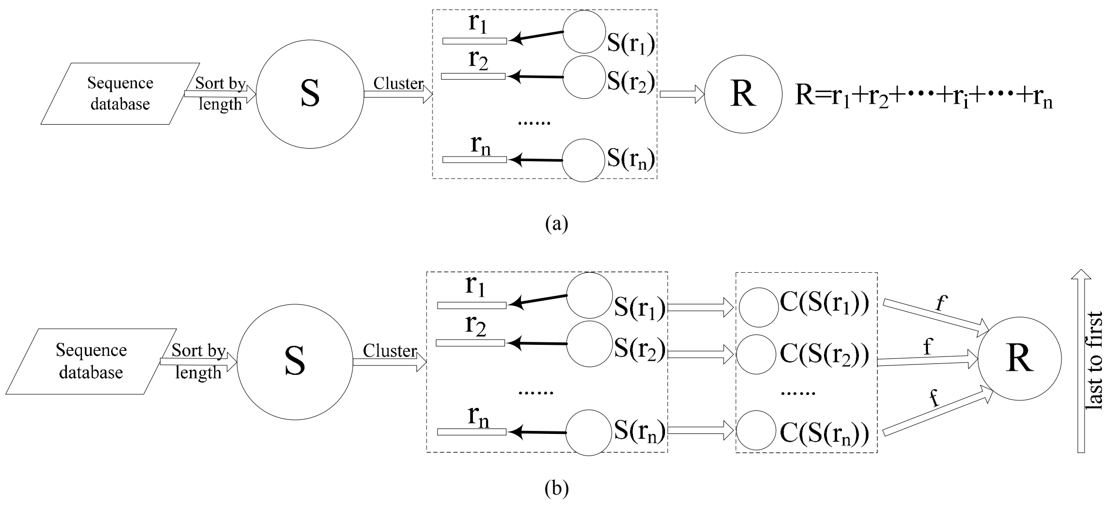


Figure 3. (a) Algorithm of Hobohm and sander; (b) Graph theory based algorithm by FastCluster.

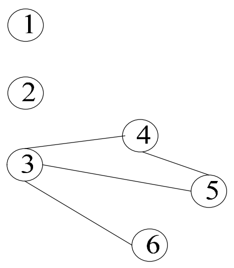


Figure 4. A graph example.

Table 1. Number of non-redundant sequences obtained using FastCluster and HSCluster.

Database	Threshold (%)	Number (FastCluster)	Number (HSCluster)
65718 Enzyme proteins	10	3167	2824
	20	3185	2845
	30	5256	4753
	40	12940	10058
	50	18177	17275
	60	25854	25020
	70	33400	32758
	80	41049	40555
	90	49986	49669

are assumed to be ordered in descending order by length. This example can be used to prove that FastCluster can produce more results than Hobohm and Sander's algorithm.

Hobohm and Sander's algorithm and FastCluster share the same way to generate three clusters, and they are $\{1\}, \{2\}$ and $\{3,4,5,6\}$ (**Figure 4**). But the way they generate the result set is different. According to Hobohm and Sander's algorithm, the representative vertex from each cluster is picked up to compose the result vertices $\{1,2,3\}$. While FastCluster computes the maximum independent set for each cluster and they are $\{1\}$, $\{2\}$ and $\{4,6\}$, and then it collects $\{4,6\}$, $\{2\}$ and $\{1\}$ together to compose the final result vertices $\{1,2,4,6\}$. Obviously, $\{1,2,4,6\}$ is larger than $\{1,2,3\}$. From this example, it is proved that FastCluster can produce more results than Hobohm and Sander's algorithm.

3. RESULTS AND DISCUSSION

FastCluster introduces a graph theory algorithm to remove redundant protein sequences and runs in a flexible and user-friendly way. It can also be changed to process other types of biological sequences easily.

A protein sequence set (65718 enzymes, from http://expasy.org/sprot/) was selected to test the performance of FastCluster. Another program HSCluster which implements the algorithm of Hobohm and Sander is also developed. The results generated by these two algorithms are shown in **Table 1**. It is clear that FastCluster can produce more results than HSCluster.

4. CONCLUSIONS

This paper makes an investigation on removing redundant biological sequences, which is modeled as a mathematical problem of finding a maximal independent set from a graph. Based on this model, FastCluster has made an improvement to Hobohm and Sander's algorithm in finding a larger independent set of a graph and thus generate more result sequences, which mean that more protein families can be clustered in a protein dataset. In a word, FastCluster provides an alternatively improved way to remove redundancy from a biological database and is also a computational tool to find more protein families and their representatives.

5. ACKNOWLEDGEMENTS

This research was supported by the grants from Shanghai Commission for Science and Technology (KSCX2-YW-R-112), and Shanghai Leading Academic Discipline Project (J50101).

REFERENCES

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

[2] L. Holm and C. Sander. (1998) Removing near-neighbour redundancy from llarge protein sequence collections. Bioinformatics, **14,** 423–429.

[3] W. Li and A. Godzik. (2006) Cd-Hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics, **22,** 1658–1659.

[4] W. Li, J L. aroszewski, A. Godzik. (2001) Clustering of highly homologous sequences to reduce the size of large protein databases. Bioinformatics, **17,** 282–283.

[5] U. Hobohm, M. Scharf, R. Schneider, C. Sander. (1992) selection of representative protein data sets. Protein Sci, **1,** 409–417.

[6] U. Hobohm and C. Sander. (1994) Enlarged representative set of protein structures. Protein Sci, **3,** 522–524.

[7] G. Wang and R. L. Jr. Dunbrack. (2003) PISCES: a protein sequence culling server. Bioinformatics, **12,** 1589–1591.

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

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

[10] S. Niskanen and P. R. J. Östergård. (2003) Cliquer user's guide, Version 1.0, Communications Laboratory, Helsinki University of Technology, Espoo, Tech. Rep. T48. http://users.tkk.fi/~pat/cliquer.html.