Finding Homologous Sequences in Genomic Databases

Hsiao Ping Lee [1] Tzu Fang Sheu [2] Yin Te Tsai [3] Chuan Yi Tang [4,*]

[1] Department of Information Management Science
Chung Shan Medical University
Taichung City, Taiwan, ROC.
ping@csmu.edu.tw

[2] Institute of Communication Engineering
National Tsing Hua University
Hsinchu City, Taiwan, ROC.
sunnie@totoro.cs.nthu.edu.tw

[3] Department of Computer Science and Communication Engineering
Providence University
Taichung, Taiwan, ROC.
yttsai@pu.edu.tw

[4] Department of Computer Science
National Tsing Hua University
Hsinchu City, Taiwan, ROC.
cytang@cs.nthu.edu.tw

Received 10 August 2007; Accepted 10 September 2007

Abstract. Homologous sequences contain invaluable information about evolution. To search for homologous sequences in genomic databases, or homology search, is one of the most fundamental and crucial tasks in biological knowledge discoveries, and required in many biological applications. With exponentially increasing database size and number of queries, efficient algorithms and tools for homology search are required. In this paper, we will introduce some famous and important approaches in the context of finding homologous sequences.

Keyword: homologous sequences, homologous search, GenBank database, filter algorithms

1 Background

Based on the assumption of the theory of evolution and natural selection, almost all species shared a common ancestor at some point in time. Random mutations in the DNA of some organisms lead to differently structured proteins. If such a change gave rise to advantages in survival, the change prevailed in the gene pool. Over time, species with different needs and abilities evolved in this way. A consequences of evolution is that if we find sufficiently similar DNA sequences within different species, we can conclude that the corresponding proteins in the species have similar biological functions [1,2,3,4].

Sequences are called *homologous* if they have significant similarity and evolved from a common ancestral sequence. Given query sequences, *homology searches* are to find their similar, or homologous, sequences in genomic databases. To search for homologies is one of the most fundamental and essential tasks in biological knowledge discoveries. Homology searches are required in several applications, for example [5,6,7]:

1. Function determination. It is assumed that sufficiently high sequence similarity implies functional similarity. For example, the comparative analysis of the predicted proteins suggests that nearly 30% of the fly genes have putative orthologs in the worm genome [8]. Nearly 20% of the fly proteins have a putative ortholog in both worm and yeast; these shared proteins probably perform functions common to all eukaryotic cells.
2. Conserved region identification. Proteins are often different identifiable domains, and domains can occur in different combinations within different proteins. Short sequence similarities reflect conserved domains within proteins. The domains are likely to correspond to essential sites for the structure or function of the sequence.

* Correspondence author

3. EST clustering. EST clusters are produced by comparing every sequence in an EST database against all other sequences. Two sequences are assigned to the same cluster if they overlap with a sufficiently high degree of identity. We can try to assemble and align the sequences in an EST cluster, which come from the same gene. This makes it possible to infer the nucleotide sequence in the gene.

The term "Expressed Sequence Tag" (EST) was first introduced in 1991 by a landmark study from Venter and his colleagues [10]. After the initial demonstration of the utility and cost effectiveness of the EST approach, many similar projects were initiated, resulting in an ever-increasing number of human ESTs [11,12,13,14,15]. In addition, large-scale EST projects were launched for several other organisms of experimental interest. In 1992, a database called dbEST [10] was established to serve as a collection point for ESTs, which are then distributed to the scientific community as the EST division of GenBank [16]. The EST division continues to dominate GenBank, accounting for roughly two-thirds of all submissions. The 20 organisms with the largest numbers of ESTs in the public database (as of March 7, 2002) are shown in Table 1.

Accumulating sequence data among different species provide ample materials for comparative analysis. As the consequence of large-scale DNA sequencing activities, there are now more DNA sequences in GenBank than there are related publications in the literature [17]. The large-scale study of genes is the hallmark of the transition from 'structural' to 'functional' genomics [18]. With huge DNA sequence data, one of the biologists' desires is to understand the secrets of DNA sequences, the language of life. However, nothing is known about the coding potential of these stretches of DNA as they are being sequenced. As such, automated methods will become an important role in annotating the human and other genomes to increase the intrinsic value of these data as they are being deposited into the public databases.

Table 1. Top 20 organisms in dbEST (as of March 7, 2002).

Organism	EST
Homo sapiens (human)	4,070,035
Mus musculus (mouse)	2,522,776
Rattus norvegicus (rat)	326,707
Drosophila melanogaster (fruit fly)	255,456
Glycine max (soybean)	234,900
Bos taurus (cow)	230,256
Danio rerio (zebrafish)	197,630
Xenopus laevis (African clawed frog)	197,565
Caenorhabditis elegans (nematode)	191,268
Lycopersicon esculentum (tomato)	148,338
Zea mays (maize)	147,658
Medicago truncatula (barrel medic)	137,588
Arabidopsis thaliana (thale cress)	113,330
Chlamydomonas reinhardtii	112,489
Hordeum vulgare (barley)	104,803
Oryza sativa (rice)	104,284
Sus scrofa (pig)	103,321
Anopheles gambiae (mosquito)	88,963
Ciona intestinalis (sea squirt)	88,742
Sorghum bicolor (sorghum)	84,712

2 Approaches for Finding Homologous Sequences

In the literature, many algorithms have been proposed for homology searches. The algorithms can be simply categorized into three classes: exhaustive search, heuristic search and filter algorithms.

2.1 Exhaustive-Search Algorithms

An algorithm Needleman-Wunsch [19] is the first instance of dynamic programming being applied to biological sequence comparison. The algorithm performs, with an exhaustive search, an optimal global alignment on two sequences of similar lengths. It is guaranteed to find the alignment with the maximum score. However, the algorithm is unsuitable for searching databases because databases often contain sequences of different lengths.

Another algorithm Smith-Waterman [20] gives the local alignment of two sequences; that is, for determining similar regions between two nucleotide or protein sequences. Like the Needleman- Wunsch algorithm, the Smith-Waterman algorithm is also an exhaustive search approach based on dynamic programming. As such, it has the desirable property that it is guaranteed to find the optimal local alignment with respect to the scoring system being used (which includes the substitution matrix and the gap-scoring scheme).

The dynamic programming strategy employed in the Needleman-Wunsch and Smith-Waterman algorithms needs O(mn) time and space to find the best alignment between two strings of lengths m and n [19,20]. In general, the two technique is infeasible in terms of both time and space for large data and query sequences.

2.2 Heuristic Algorithms

Most heuristic homology searching algorithms have the same basic idea: first, they use a fast searching program to identify the regions that are likely to be homologous, and then, if possible, combine them to longer chains of sufficient sequence similarity. Finally, these chains are aligned using a slower but more accurate method, for example the Needleman-Wunsch algorithm. However, the techniques employed in the stages of the algorithms are quite different; for example, some of them use contiguous seed models, but the others use spaced models; some are hash-table-based algorithms, but the others employ suffix trees.

FASTA [21,22] is a well-known heuristic algorithm for sequence comparisons. When looking for an alignment, we might expect to find a few segments in which there will be absolute identity between the two compared sequences. FASTA is using this property and focuses on these identical regions. The algorithm consists of several processing stages. In the first stage, it seeks for matched segments between the two sequences, and combines the segments to find better alignments in the rest stages.

BLAST (Basic Local Alignment Search Tool) [23] is one of the most popular programs for searching biosequences against databases. The algorithm integrates the substitution matrix in the stage of finding hot spots. The idea results fewer and better hot spots found in the process, and thus increases the speed of the FASTA algorithm. The two stages, scanning and extension, in BLAST are as follows:

1. Specify a length parameter w and a threshold parameter t. It is recommended to set w to values of 3 to 5 for amino acids and 12 for nucleotide sequences. Finds all the w-tuples, called *words*, of database sequences, which align with words from the query with an alignment score higher than t. Such hot spots are referred to as *hits*.
2. Extend each hit to find out if it is contained in a segment pair with score above a predefined threshold s. The threshold must be chosen such that it is unlikely to find a random sequence that achieves a score higher than s when compared with the query sequence.

FASTA and BLAST are heuristic local alignment algorithms. They do not guarantee finding out the optimal alignment between the query and database sequences. However, the algorithms are much faster than the ordinary dynamic programming alignment algorithms, and, with respect to the optimal alignment, the resulting alignment scores are well. The detail description of the programs in the FASTA3 and BLAST packages can be found at http://helix.nih.gov/ and http://www.ncbi.nlm.nih.gov/ .

Gapped BLAST and Position-Specific Iterated (PSI) BLAST [24] are two improved BLAST algorithms. Gapped BLAST allows alignments with gaps. PSI BLAST constructs a profile, or position specific scoring matrix, *PSSM*, based on the results from the ordinary BLAST search. MegaBLAST [25], instead of using dynamic programming, employs a greedy method in the similarity search. It is one of the tools in the current NCBI BLAST package. The algorithm is very efficient for aligning DNA sequences that are extremely similar, for example the sequences that differ only by sequencing errors.

SAHA [26] preprocesses sequences in the database by breaking them into consecutive k-tuples, and then using a hash table to store the occurrence positions of the tuples. The similarity search is done by obtaining, from the hash table, the hits for each k-tuple in the query sequence, and then performing a sort on the matches to find possible extensions. BLAT [27] indexes all nonoverlapping k-tuples in the database. The algorithm uses the index to find regions in the database likely to be homologous to the query sequence, performs an alignment between the

candidate regions, and then combines these aligned regions (often exons) into larger alignments (typically genes). Finally, BLAT revisits small internal exons possibly missed at the previous stages and adjusts large gap boundaries that have canonical splice sites where feasible.

Unlike the above approaches that use tuples of contiguous bases, PatternHunter [28] finds short word matches under a spaced model in similarity searches. A spaced model can be represented as a binary string, for example 11010, where a 1 bit at a position means that a base match is required at the position and a 0 bit means either a base match or mismatch is acceptable. The number of 1 bits in a spaced model is referred to as the *weight* of the model. For examples, the two words `ACGTC` and `ACTTG` form a word match under the spaced model 11010 of weight 3. Because PatternHunter allows mismatches in seeds, or words, it is able to find more homologous regions than the algorithms, for example BLAST, which find matches under a contiguous model. However, it is very difficult to find the optimal spaced model for comparisons. A new version of BLASTZ also adapts the idea of spaced models [29].

In general, a single spaced seed can not achieve the sensitivity of exhaustive-search algorithms, for example Smith-Waterman algorithm [20]. It has been pointed out that more spaced seeds increase sensitivity [28]. PatternHunter II [30] extends the single spaced seed of original Pattern Hunter to multiple ones.

Some homology search algorithms use suffix trees and its derivatives (see [31]) to index the sequences, for example MUMmer [32], AVID [33] and CHAOS [34]. MUMmer builds suffix trees on both of the sequences to finds maximal matches that are unique, and orders these matches to form the basis of an alignment that can span even very long mismatch regions between the two input sequences. The algorithm assumes the sequences are closely related, and extracts information on single nucleotide changes, translocations and homologous genes under the assumption. AVID is a global alignment tool. The algorithm transforms the problem of finding all maximal matches between two sequences to the problem of finding maximal repeated substrings in one sequence, and then solves the problem using suffix trees. CHAOS uses a variation of trie (called a threaded trie), which is a cross between a suffix tree and a hash table in spirit, to index the k-tuples of one sequence. LAGAN [35] employed CHAOS to find global alignments; Shuffle-LAGAN [36] combines the CHAOS local alignment algorithm and the LAGAN global aligner to perform global alignments, a combination of global and local alignments.

2.3 Filters

As discussed in Section 2.1, the dynamic programming-based algorithms provide the best alignments. However, because of the quadratic time involved, the dynamic programming-based algorithms may not be directly or practically applied for searching on databases of large scale. Several heuristic algorithms, for example the methods discussed in Section 2.2, have been proposed to speed up homology search tasks. Nevertheless, these algorithms may not be efficient enough because they often need to inspect the entire database while only a very small part of the database might actually be of interest. Thus, another type of algorithms called *filter algorithms* was introduced. The sort of algorithms does not find the final alignments for the compared sequences, but the algorithms employ some techniques, for example domain transformation, to reduce the searching space of similarity search, and thus speed up the search operation. The algorithms can be used as a preprocessing phase on top of the exhaustive-search approaches, for example the Smith-Waterman algorithm, and heuristic approaches, for example the FASTA and BLAST algorithms.

QUASAR [37] builds a suffix array, a derivative of suffix trees, on the grams, short substrings of certain length, from the sequence. The algorithm works based on the q-gram lemma (see [38]), and achieves proximity filtration. QUASAR partitions the database into several blocks and maintains a counter for each of them. During query time, the counter value of a certain block reflects the amount of the grams in the query sequence, which occur exactly within the block. The blocks having counter values satisfying the q-gram lemma are chosen as candidates, and searched using BLAST. [39] is a study on gapped q-grams.

Chavez et al. [40] translate the problem of approximate string search into a range query or proximity search in a metric space. The algorithm selects k pivots randomly, maps each sequence with a k-dimensional vector, and further uses triangle inequality to prune non-relevant sequences. The method employs Suffix Trees as index structures. SST [41] partitions the database sequences into a set of overlapping w-segments, and translates them into 4^k-dimensional vectors, where $k \leq w$. Each element of the vectors indicates the number of occurrences of each k-tuple in the w-segment. The vectors are clustered hierarchically using a k-means clustering algorithm. The nonoverlapping segments are extracted from the query sequence, and compared against the clusters to prune the segments which are far from the query segments.

Aghili et al. [42] study on vector transformations, and conduct the application of discrete fourier transformation (DFT) and discrete wavelet transformation (DWT) dimensionality reduction techix niques for DNA sequence proximity search to reduce the search time of queries. And, Aghili et al. [43] also map the problem of

whole-genome sequence homology search into an approximate vector comparison in the well-established multi-dimensional vector space.

Kahveci et al. [44] proposed a frequency vector-based algorithm for aligning large genome sequences. They define the frequency vector of a sequence as *5*-dimensional vector (for the alphabet set = {A, C, G, T, N}, where N stands for unknown) in which entries are the number of occurrences of the different letters in the sequence. The algorithm computes the frequency vectors for all subsequences in the database sequence, which are of a certain length, and aggregates the vectors into a index structure called *F-index*, in which an entry records the boundries of a certain amount of consecutive frequency vectors. Then, the algorithm constructs a boolean match table by partitioning the query sequence into subsequences and searching these subsequences in the F-index of the database sequence. The subsequence pairs containing potential similar patterns are given to BLAST (or any other alignment tool) for further processing.

3 Conclusions

In this paper, we introduce some approaches for finding homologous sequences. On some popular databases, such as the large GenBank nucleotide collection, the number of access has increased to over 120,000 queries per day. Moreover, the GenBank collection is doubling in size almost yearly, query lengths are steadily increasing, and the number of users continues to grow. It is an essential issue to develop efficient homology search approaches continuingly.

Acknowledgement

This work was supported in part of the National Science Council of Republic of China under grant NSC-94-2213-E-126-002. We also thank Mr. Chia Jung Chang for this help in typesetting.

References

[1] R. F. Doolittle, “WhatWe Have Learned andWill Learn from Sequence Databases,” in G. Bell and T. Marr, editors, *Computers and DNA*, pp.21–31. Addison-Wesley, Reading, MA, USA, 1990.

[2] C. Caskey, R. Eisenberg, E. Lander, and J. Straus, “Hugo Statement on Patenting of DNA,” *Genome Digest*, Vol.2, pp.6–9, 1995.

[3] A. M. Lesk, “Computational Molecular Biology,” in A. Kent and J. G.Williams, editors, *Encyclopedia of Computer Science and Technology*, Vol.31, pp.101–165. Marcel Dekker, New York, 1994.

[4] W. R. Pearson, “Protein Sequence Comparison and Protein Evolution,” *Tutorial of Intelligent Systems in Molecular Biology 2000*, 2000.

[5] A. Krause and M. Vingron, “A Set-Theoretic Approach to Database Searching and Clustering,” *Bioinformatics*, Vol.14, pp.430–438, 1998.

[6] J. C. Venter, M. D. Adams, G. G. Sutton, A. R. Kerlavage, H. O. Smith, and M. Hunkapillar, “Shotgun Sequencing of the Human Genome,” *Science*, Vol.280, pp.1540–1542, 1998.

[7] A. D. Baxevanis and B. F. Francis Ouellette, *Bioinformatics: a Practical Guide to the Analysis of Genes and Proteins*, Wiley Interscience, New York, USA, second edition, Apr. 2001.

[8] G. M. Rubin et al., “Comparative Genomics of the Eukaryotes,” *Science*, Vol.287, pp.2204–2215, 2000.

[9] M. D. Adams, J. M. Kelley, J. D. Gocayne, M. Dubnick, M. H. Polymeropoulos, H. Xiao, C. R. Merril, A. Wu, B. Olde, and R. F. Moreno, et al., “Complementary DNA sequencing: expressed sequence tags and human genome project,” *Science*, Vol.252, pp.1651–1656, 1991.

[10] M. S. Boguski, T. M. Lowe, and C. M. Tolstoshev, “dbEST: database for ”expressed sequence tags”. *Nat. Genet.*, Vol.4, pp.332–333, 1993.

[11] M. D. Adams, M. B. Soares, A. R. Kerlavage, C. Fields, and J. C. Venter, “Rapid cDNA sequencing (expressed sequence tags) from a directionally cloned human infant brain cDNA library,” *Nat. Genet.*, Vol.4, pp.373–380, 1993.

[12] R. Houlgatte, R. Mariage-Samson, S. Duprat, A. Tessier, S. Bentolia, B. Larry, and C. Auffray, “The Genexpress Index: a resource for gene discovery and the genic map of the human genome,” *Genome Res.*, Vol.5, pp.272–304, 1995.

[13] L. D. Hillier, G. Lennon, M. Becker, M. F. Bonaldo, B. Chiapelli, S. Chissoe, M. Dietrich, T. DuBuque, A. Favello, and W. Gish, “Generation and analysis of 280,000 human expressed sequence tags,” *Genome Res.*, Vol.6, pp.807–828, 1996.

[14] D. B. Krizman, L. Wagner, A. Lash, R. L. Strausberg, and M. R. Emmert-Buck, „The Cancer Genome Anatomy Project: EST sequencing and the genetics of cancer progression,” *Neoplasia*, Vol.1, pp.101–106, 1999.

[15] K. Okubo, N. Hori, R. Matoba, T. Niiyama, A. Fukushima, Y. Kojima, and K. Matsuba, “Large scale cDNA sequencing for analysis of quantitative and qualitative aspects of gene expression,” *Nat. Genet.*, Vol.2, pp.173–179, 1992.

[16] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, B. A. Rapp, and D. L. Wheeler, “GenBank,” *Nucleic Acids Res.*, Vol.30, pp.17–20, 2002.

[17] O. Ermolaeva et al., “Data Management and Analysis for Gene Expression Arrays,” *Nature genetics*, Vol.20, pp.19–23, 1998.

[18] P. Hieter and M. Boguski, “Functional Genomics: It’s All How You Read It,” *Science*, Vol.278, pp.601– 602, 1997.

[19] S. B. Needleman and C. D.Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal of Molecular Biology*, Vol.48, pp.443–453, 1970.

[20] T. F. Smith and M. S. Waterman, “Identification of Common Molecular Subsequences,” *Journal of Molecular Biology*, Vol.147, pp.195–197, 1981.

[21] D. J. Lipman and W. R. Pearson, “Rapid and sensitive protein similarity searches,” *Science*, Vol.227, pp.1435–1441, 1985.

[22] W. R. Pearson and D. J. Lipman., “Improved tools for biological sequence comparison,” *Proceedings of the National Academy of Sciences*, Vol.85, No.8, pp.2444–2448, 1988.

[23] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, “Basic Local Alignment Search Tool,” *Journal of Molecular Biology*, Vol.215, pp.403–410, 1990.

[24] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang,W. Miller, and D. J. Lipman, “Gapped blast and psi-blast: a new generation of protein database search programs,” *Nucleic Acids Res.*, Vol.25, No.17, pp.3389–3402, 1997.

[25] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller, “A greedy algorithm for aligning dna sequences,” *J. Comput. Biol*., Vol.7, pp.203–214, 2000.

[26] Z. Ning and J. C. Mullikin, “SSAHA: A Fast Search Method for Large DNA Databases,” *Genome Research*, Vol.11, No.10, pp.1725–1729, 2001.

[27] W. J. Kent, “BLAT: The BLAST-Like Alignment Tool,” *Genome Research*, Vol.12, No.4, pp.656–664, 2002.

[28] B. Ma, J. Tromp, and M. Li, “PatternHunter: Faster and More Sensitive Homology Search,” *Bioinformatics*, Vol.18, pp.440–445, 2002.

[29] S. Schwartz, W. J. Kent, A. Smit, and W. Miller, “Human-mouse alignments with blastz,” *Genome Research*, Vol.13, pp.103–107, 2003.

[30] M. Li, B. Ma, D. Kisman, and J. Tromp, "Patternhunter ii: Highly sensitive and fast homology search," *Journal of Bioinformatics and Computational Biology*, 2004.

[31] D. Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*, Cambridge University Press New York, NY, USA, 1997.

[32] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg, „Alignment of whole genomes," *Nucleic Acids Res.*, Vol.27, pp.2369–2376, 1999.

[33] N. Bray, I. Dubchak, and L. Pachter, "Avid: a global alignment program," *Genorne Res.*, Vol.13, pp.97–102, 2003.

[34] M. Brudno and B. Morgenstern, "Fast and sensitive alignment of large genomic sequences," *BMC Bioinformatics*, Vol.4, No.66, 2002.

[35] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, E. D. Green, A. Sidow, and S. Batzoglou, "Lagan and multi-lagan: efficient tools for large-scale multiple alignment of genomic dna," *Genome Research*, Vol.13, No.4, pp.721–731, 2003.

[36] M. Brudno, S. Malde, A. Poliakov, C. Do, O. Couronne, L. Dubchak, and S. Batzoglou, "Glocal alignment: finding rearrangements during alignment," *Bioinformatics*, Vol.19, No.(Suppl.) 1, pp.154– 162, 2003.

[37] S. Burkhardt, A. Crauser, P. Ferragina, H. P. Lenhof, E. Rivals, and M. Vingron, "q-Gram Based Database Searching Using a Suffix Array (QUASAR)," *Proceedings of the third Annual International Conference on Computational Molecular Biology (RECOMB99)*, pp.77–83, 1999.

[38] P. Jokinen and E. Ukkonen, "Two Algorithms for Approximate String Matching in Static Texts," *Proc. of the 16th Symposium on Mathematical Foundations of Computer Science* (Lecture Notes in Computer Science 520), pp.240–248, 1991.

[39] S. Burkhardt and J. Karkkainen, "Better Filtering with Gapped q-Grams. to appear in Fundamenta Informaticae," *special issue on Computing Patterns* in Strings, 2003.

[40] E. Chavez and G. Navarro, *A Metric Index for Approximate String Matching*, Lecture Notes in Computer Science 2286 (LATIN 2002), pp.181–195, 2002.

[41] E. Giladi, M. G. Walker, J. Z. Wang, and W. Volkmuth, "SST: an Algorithm for Finding Near-Exact Sequence Matches in Time Proportional to the Logarithm of the Database Size," *Bioinformatics*, Vol.18, No.6, pp.873–877, 2002.

[42] S. A. Aghili, D. Agrawal, and A. E. Abbadi, *Using Transformation Techniques Towards Efficient Filtration of String Proximity Search of Biological Sequences*, Technical Report 2003-19, Department of Computer Science, University of California, Santa Barbara, 2003.

[43] S. A. Aghili and O. D. Sah, *Efficient filtration of sequence homology search through singular value decomposition*, Technical Report 2003-19, Department of Computer Science, University of California, Santa Barbara, 2003.

[44] T. Kahveci, V. Ljosa, and A. K. Singh, "Speeding up whole-genome alignment by indexing frequency vectors," *Bioinformatics*, Vol.20, No.13, pp.2122–2134, 2004.