Accelerating Search of Approximate Match on Large Protein Sequence Databases

Wei Wang [1], Jiong Yang [2], Yi Xia [3], Philip Yu [4]

Keywords: protein sequence index, approximate match.

Due to the significant advance in molecular biology, analysis of biological sequences has become an active research area in recent years. The amount of mapped sequences grows at an exponential pace, while the flat files have continued to be used as the standard format to store the huge sequence database. Searching on the a large sequence database is usually carried out by sequentially scanning the entire database using a screening approach to identify the set of desired sequences (e.g., BLAST). This approach inevitably suffers from a prolonged response time when dealing with a large amount of sequences, even though a biologist sometime may be able to localize the search to a relatively smaller scope. Alternative methods that can achieve a fast delivery (such as the q-grams based approaches) may only be suitable for search of high similarity matches. As a result, how to accelerate the search in a broader domain become a topic of great practical importance. Building an appropriate index structure is one of the possibilities to achieve such a goal, which is the focus of our investigation in this paper. The following observations provide the motivation and feasibility.

- An proper index may help to direct the search to the high potential portion of matching in the database and hence avoid the scan of the entire database. This is particularly important in improving the performance of searching a large database since it may take hours to evaluate every sequence in a large database and the saving in computation cost increases in proportion to the size of the database.

- Despite the rapid growth in the database size, the existing sequences in the database are relatively stable. (Most updates to the database are adding new sequences.) This justifies the potential benefit of supporting indexed search on large biological sequence databases since the construction cost of the index structure can be easily amortized over many searches. This also suggests that the index structure needs to support not only efficient search but also efficient insertion of new sequences.

Previous indexing structures are built for exactly match, i.e., symbols will not mutate to others. However, the amino acids in a protein sequence may mutate to others without changing its biological function significantly. To accommodate mutations, we need to build an index such that similar segments (according to some amino acid mutation matrix, e.g., BLOSUM50) are organized together. To address these issues, we propose a new index structure, namely SAM-tree, to support *approximate match* in sublinear time on a large sequence database. An example of SAM-tree built on sequence *FILIVSADLVFLM* is shown in Figure 1. The key idea behind the design is that all *position points* are grouped recursively into a fully balanced tree according to the similarities of the *subsequent* segments, where the similarities can be defined through a *substitution matrix.* Each node in the tree has a bounded size and is designed to be fit in a (pre-defined) disk block. Leading positions of similar segments would share a common ancestor and each node is labeled with a

[1]IBM T. J. Watson, 19 Skyline Drive, Hawthorne, NY 10532. E-mail: ww1@us.ibm.com
[2]IBM T. J. Watson, 19 Skyline Drive, Hawthorne, NY 10532. E-mail: jiyang@us.ibm.com
[3]UCLA Computer Science Department, Los Angeles, CA 90095 . E-mail: xiayi@cs.ucla.edu
[4]IBM T. J. Watson, 19 Skyline Drive, Hawthorne, NY 10532. E-mail: psyu@us.ibm.com

regular expression that describes the commonality of the segments indexed through the subtree. Each leaf node contains a collection of position points and each internal node holds links to a list of children nodes. The search of similar segments to a given pattern is then composed of a recursive traversal of relevant nodes in the SAM-tree. For example, all position points which are descendants of node X start with symbol A, D, S, or T.

When we try to find all similar segments of a pattern, the following procedure can be performed. Starting from the root, the given pattern is compared to the label of every child. A child is believed to be *relevant* if one of its leaf descendants may contain a pointer to a similar segment. The same procedure is then invoked again on each relevant child until the relevant leaf nodes are reached. The set of position points in these relevant leaf nodes are then returned as (highly potential) candidates for further evaluation. Many properties of the SAM-tree, including the balanced height and compactness, provide the theoretical foundation of the efficiency of the search as well as the maintenance. We have applied the SAM-tree index on a protein database and have been able to achieve an order of magnitude speed-up over the BLAST algorithm.

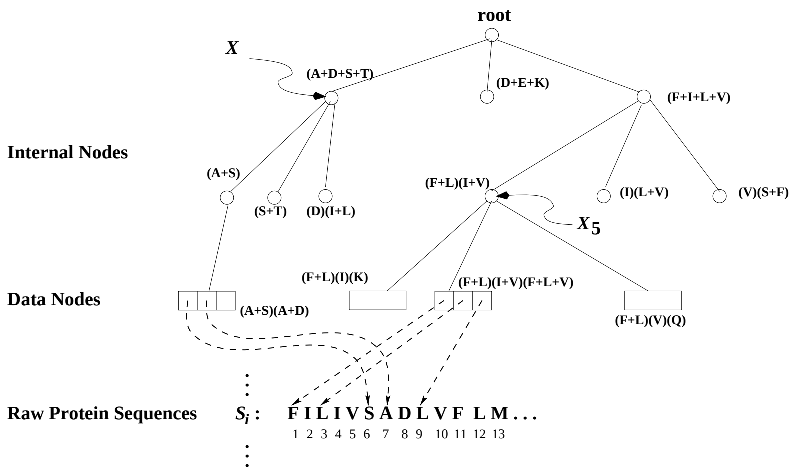

Figure 1: An Example of the SAM-Tree.

To better understand the benefits of employing the index structure on the protein sequence set, the SWISS-PROT protein database is used to test the SAM-tree. The data set consists of 100,304 protein sequences and 36,743,059 amino acids. Comparing with other methods, e.g., suffix tree and BLAST, the query time can be reduced by one order of magnitude when a SAM-tree index structure is used for the set of sequences.