Exhaustive Peptide Searching Using Relations

Ela Hunt

Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland
hunt@inf.ethz.ch

Abstract. We present a new robust solution to short peptide searching, tested on a relational platform, with a set of biological queries. Our algorithm is appropriate for large scale scientific data analysis, and has been tested with 1.4 GB of amino-acids. Protein sequences are indexed as short overlapping string windows, and stored in a relation. To find approximate matches, we use a neighbourhood generation algorithm. The words in the neighbourhood are then fetched and stored in a relation. We measure execution time and compare the matches found to those delivered by BLAST. We report some performance gains in exact matching and searching within edit distance 1, and very significant quality improvements over heuristics, as we guarantee to deliver all relevant matches.

1 Introduction

Biological sequence comparison involves searching large repositories of string data for approximate matches to a string of interest. Strings use alphabets such as DNA={A,C,G,T} or the protein alphabet of over 20 letters. Such searches are now supported by database technologies [5,32] and are based on BLAST [1,2] and not on indexing. These solutions use the heuristic method, BLAST, and traverse the entire data set while searching for a string of interest. In relational terms this might be seen as equivalent of a full table scan, and is slow, as the complexity of searching is dominated by the size of the data set one searches against. For a database of size n, it will be $O(n)$ at least. An exhaustive search using dynamic programming (DP) which builds a comparison matrix aligning each query letter with each database letter, for a query of length m, has the complexity of $O(mn)$ [27,31], and is often impractical for that reason.

Since 1995 [7] we have experienced a dramatic increase in the amount of available biological sequence data, and a simultaneous desire to perform new forms of searching. This forces us to rethink the assumptions of such work. One of the new directions is persistent sequence indexing [25,23,13,14]. Another comes from new sequence analysis requirements involving micro array probe mapping [8], miRNA mapping[29], and motif searches [6,30,35], which all involve short query strings and very large databases, often larger than 1 GB. Indexing can reduce search times from linear in n, to, ideally, logarithmic, as an index tree depth is a logarithm of data size.

This work explores three issues we encountered in the execution of searches for short peptide strings (length 7) against the background of 1.4 GB of protein

R. Cooper and J. Kennedy (Eds.): BNCOD 2007, LNCS 4587, pp. 13–24, 2007.
© Springer-Verlag Berlin Heidelberg 2007

sequences. The first issue of interest is search complexity reduction from linear in n to logarithmic, achieved using indexing. The second issue is the need to deliver all matches fulfilling a given definition, as BLAST only offers statistical guarantees of quality, and is not exhaustive. The third is a simplification of the search algorithms described by Myers [25] and Baeza-Yates and Navarro [3] who partition a long query into short fragments and then assemble the result from fragments. Here, short queries directly use the index, with no post-processing.

The contributions of this paper are as follows. We build a protein sequence index in a commercial database. To our knowledge, this is the first report of index-based large-scale sequence comparison using a relational platform. We study the performance of approximate matching on a large repository of protein sequences and a set of experimentally derived short peptides, and we report on both time and quality comparison with BLAST.

Our findings are presented as follows. Section 2 motivates the need for exhaustive searches on short strings. Section 3 introduces the algorithms. Section 4 describes the implementation and Section 5 presents an experimental evaluation. A discussion is offered in Section 6, along with the context of other research, and conclusions are presented in Section 7.

2 Motivation

We briefly outline why short queries are of interest. The focus here is on an experiment called *phage display* [30,35]. This experiment can be used to understand which proteins have the capacity to interact, and what peptide sequences mediate their interaction. A phage is a small microorganism, that can be made to carry attached sequences on its outer surface, and therefore bind to tissues, using that extra sequence. In this work the sequences added to phages are 7-letter peptides, representing all the possible 7-letter combinations over the protein alphabet. After the experiments, the 7-letter peptides bound to the tissue are sequenced, and serve as queries. In this work we query for 26 peptides, each in two directions, so for peptide 'ABCDEFG' we also query in 'GFEDCBA'. We are looking for exact matches, matches with one mismatch, and with two mismatches. At a later stage all the matches will be clustered with regard to the sequences and organisms they match. They may also be visualised with regard to their position in a 3D protein structure.

The second scenario is approximate matching of short DNA or RNA sequences. RNA alphabet is {A, C, G, U}. Queries may be around 20 to 30 characters in length, and a small number of mismatches are allowed [29]. Such searches are performed on a genome scale (around 3 billion letters), and about 1 million queries may be sent to model a complete experiment [8].

BLAST is not a perfect choice for such queries, as it does not allow one to define the required match length or stringency, and is not exhaustive. In Section 5 we will come back to this point and show the types of matches returned by BLAST and by exhaustive matching. In the following we describe the technology that satisfactorily performs these approximate matching tasks. Our solution