Efficient mapping of short peptides on whole proteome database for biomarker discovery

Thomas Hume, Hayssam Soueidan, Fabienne Wong Jun Tai, Antoine Vekris, Klaus Petry, Macha Nikolski

Efficient mapping of short peptides on whole proteome database for biomarker discovery

Thomas HUME[1,2], Hayssam SOUEIDAN[3], V. Fabienne WONG JUN TAI[2], Antoine VEKRIS[4], Klaus PETRY[4] and Macha NIKOLSKI[1,2]

[1] Univ. Bordeaux, CNRS / LaBRI, F-33405 Talence, France
{thomas.hume, macha.nikolski}@labri.fr
[2] Univ. Bordeaux, CBiB, F-33000 Bordeaux, France
virginie.wongjuntai@u-bordeaux2.fr
[3] NKI-AVL, Plesmanlaan 121, 1066 CX Amsterdam, Netherlands
h.soueidan@nki.nl
[4] Univ. Bordeaux, INSERM U1049, F-33000 Bordeaux, France
avec@mac.com, klaus.petry@inserm.fr

Abstract In vivo *phage display can be used to simultaneously screen peptide repertoires for peptides presenting specificity for certain interaction and recognition sites, and thus enable biomarker identification. In this work we assume the hypothesis that certain small peptides mimic physiological interactions of proteins that contain a sub-sequence similar to these peptides. A possible solution then involves mapping of the peptide repertoire against the proteome of interest, which is computationally challenging. We formulate this problem as simultaneous matching of multiple patterns against multiple strings and propose an algorithmically efficient solution. It is currently accessible at* http://services.cbib.u-bordeaux2.fr/spack/pepteam.php.

Keywords mapping, string matching, peptide, phage display

1 Introduction

The aim of this work is to streamline the biomarker discovery based on the hypothesis of mimed proteins in the context of selection of phage displayed peptides. Phage display is an *in vitro* or *in vivo* technique used to identify relevant protein interaction and recognition sites, first described in 1985 by Smith [1] ; it was later shown that antibody fragments could be successfully displayed on phage [2]. The technique is based on the insertion of DNA fragments into bacteriophage genes to create fusion proteins with the foreign sequence in the middle. Viral particles contain the encoding DNA and display the encoded peptide on the surface of their capsides. Combinatorial libraries of peptides can be produced with complexities of 10^9 and screened to isolate peptides that bind to biological samples ranging from purified macromolecules to *in vivo* pathological tissues.

In vivo screening, with targets on endothelial cells or on tissues where the accessibility is the product of the pathological condition (e.g. tumoral cells), produces large repertoires of peptides for which the use of NGS technologies made possible a global overview. A typical setup for biomarker discovery study is the differential analysis of two repertoires corresponding to two different conditions (say, healthy vs. pathological).

Previous studies for screening of peptide libraries for organ-specific binding [3] studied sequence composition and similarity in this context. Arap et al. has shown that the distribution of the ligand-receptor pairs is non-random regarding organ specificity [4]. However, in order to prevent the problem related to the sparce count of the 20^7 possible 7-mers, they were broken into 3-mers. The work of Kolonin [5] is based on the presence of those 3-mers inside the candidate peptide-bound receptors, which identifies these targeted receptors and sites of ligands involved in receptor interaction.

In this paper we assume the working hypothesis that certain peptides mimic physiological interactions of proteins that contain a sub-sequence similar to these peptides. However, sequence similarity of peptides with a protein is a weak signal. Fine-tuned and efficient mapping of the complete peptides against the proteome of interest and an appropriate scoring function are thus the central points of our solution. In this manuscript we mainly focus on the mapping step.

The underlying assumption of our work is that small peptides mimic molecular interactions of larger proteins. This assumption is based on a number of observations :

— Proteins containing similar sub-sequences tend to themselves be homologous ;
— These sub-sequences align at the same locations where homologous proteins align among themselves ;
— These locations are strongly preserved by the evolutionary process (and are mostly located within functional domains) ;

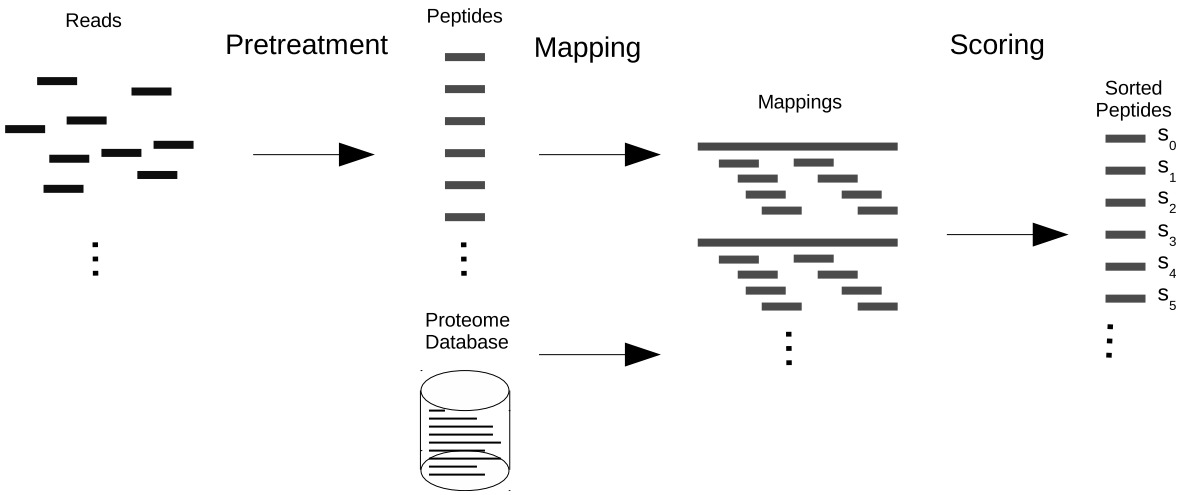
Moreover, a recent study [7] shows that most of the energy in proteins' interactions is located in short linear segments, and most of these segments bind independently of their context, which further supports our working hypothesis.

Given a peptide repertoire $R = \{r\}$ where all peptides r have the same size $s \in [7, 12]$ and proteome $P = \{p\}$, each peptide r is mapped against P, which produces a set of mapping locations L_r. This mapping is restricted to an ungapped alignment up to a certain similarity score threshold t. Once the peptides are mapped on the proteins, each peptide r is scored by integrating its mapping scores for all locations L_r combined with the scores of other peptides over the same L_r. Intuitively, it singles out those peptides that participate in those locations that are mapped by many other peptides. Consequently, the scoring function is highly sensitive to the detection of the complete set of peptides that map at a certain location L_r. This is why it is essential for a given peptide to uncover all the mappings over the given threshold t.

2 Pipeline

Our solution is implemented as a pipeline with three major steps (see figure Fig. 1 here below).

Figure 1. Peptides scoring pipeline



1. Sequenced reads are preprocessed in order to obtain the repertoire of phage displayed peptides.
2. Given this repertoire and a subject database, each peptide is mapped on each protein of the database.
3. Given the list of mappings, a scoring function sorts the peptides.

In this manuscript we concentrate on the computationally challenging step : mapping of the peptide repertoire against the proteome. We formulate this problem as simultaneous matching of multiple patterns against multiple strings and propose an algorithmically efficient solution.

3 State of the art

The central computational problem concerns mapping of R over P, which is known as the string matching problem. There exists a number of well-known algorithms for string matching [1]. They can be classified in two categories : exact and fuzzy matching.

1. In this section we only quote seminal works and citations are in no way exhaustive.

Exact matching can itself be subdivided in two major algorithmic classes.

— Matching a simple pattern against a single string.
These algorithms are of limited interest for our problem since we need to match multiple patterns against multiple strings. However, they provide a complexity reference point where we would match each peptide one by one against each protein. Classical examples are Boyer–Moore [8] and the text-partitioning matching [9].

— Matching multiple patterns against a single string.
These algorithms make it possible to map R (our multiples patterns) against the proteins of the proteome one by one. A classical example of this class of algorithms is the Aho–Corasick [10].

Fuzzy matching approaches can also be classified in two major classes, both relying on the notion of metric space.

— Approximate matching
These algorithms use the metric space to find the set of similar words. A well-known example of the underlying distance measure is the Damerau–Levenshtein distance [11] used in many spell-checkers.

— Sequence alignment
These algorithms are particularly widespread in bioinformatics. Indeed, in this context, character mismatch or insertion/deletion are often seen as mutation points in the evolutionary process. Metric space usually relies on a substitution score matrix rather than on a fixed score in the case of mismatch. Classical examples are Smith–Waterman [12] and BLAST [13] for local alignment and Needleman–Wunch [14] for global alignment.

On one hand, our application case requires fuzzy matching, since we are looking for similarity between peptides and protein sequences. Moreover, we rely on a substitution matrix for sequence similarity. On the other hand, this is not a classical fuzzy matching problem since we are interested in the ungapped alignments. It is also important to obtain all the positions where peptides match on the proteins. Consequently, neither probabilistic (e.g, BLAST) nor best-match (e.g, Needleman–Wunsch) approaches can be used.

4 Mapping

First we preprocess the proteome $P = \{p\}$. Since we are interested in the ungapped alignments, this can be seen as fragmenting each p by a sliding window of size s, which gives us $F = \{f\}$, where f are proteic sub-sequences, that we call fragments, of size s. The problem can be now seen as matching R against F. However, this would be computationally inefficient.

To overcome this problem, we encode P in a compact way by constructing the keyword tree [15] S of F. Information on the positions L_f of each fragment and the proteins where it belongs is stored in the leaves of the tree. Another keyword tree Q is constructed for the repertoire R. To be efficient, the construction of S involves the intermediate construction of a suffix tree (over a finite alphabet, see [16]) where only the prefixes of size s of each suffix are kept.

Given a keyword tree Q (S, respectively), we denote by Q_n (S_n, respectively) any node of this tree. Each node contains a ⟨letter, node⟩ map that represents a tree edge. Given a node Q_n and a letter l, $Q_n[l]$ denotes following the corresponding edge and reaching the child of Q_n ; $S_n.data$ denotes the ⟨protein, position set⟩ map.

The problem can then be formulated as mapping the two keyword trees one against another. The general matching scheme is presented in the algorithm 1 here below.

This algorithm can be trivially used to perform exact matching by replacing the scoring function by an equality comparison between l_1 and l_2. The thresholding function then becomes a placeholder present only to check the validity of the boolean score.

4.1 Similarity score

Since our mapping is ungapped, in order to compute the *similarity* between a peptide r and a fragment f, it is sufficient to compute the letter-wise substitution cost based on a substitution matrix M, namely :

Algorithm 1 Recursive mapping of Q onto S

function MAPFUNCTION(Q_n, S_n, $word$, $score$, $depth = 0$)
  $W \leftarrow$ empty set of $\langle$word, data$\rangle$ pairs
  for each letter $l_1 \in Q_n$ **do**
    for each letter $l_2 \in S_n$ **do**
      $new\_score \leftarrow$ SIMILARITYFUNCTION(l_1, l_2, $score$)
      if THRESHOLDING(new_score, $depth$) **then**
        $new\_word \leftarrow$ CONCAT($word$, l_1)
        if Q_n is a leaf node **then** ▷ Sn is necessarily a leaf too
          add (new_word, $Sn.data$) pair to W
        else
          $W \leftarrow W \cup$ MAPFUNCTION($Q_n[l_1]$, $S_n[l_2]$, new_word, new_score, $depth + 1$)
        end if
      end if
    end for
  end for
  return W
end function

$$score_{r/f} = \frac{2 \sum_{l_r \in r, l_f \in f} M(l_r, l_f)}{\sum_{l_r \in r} M(l_r, l_r) \sum_{l_f \in f} M(l_f, l_f)}.$$

Using this formula, similarity of two exact same words is 1.0 and the similarity score drops with each substitution involved.

Algorithm 1 is recursive, consequently, both similarity score computation and the thresholding have to respect the recursivity.

— For the similarity this implies to maintain through the recursive call both the numerator and the denominator independently. And it is within the thresholding function that the fraction is computed.
— For the thresholding this implies that the threshold value has to be defined as function of the number of letters that have been processed, which means that it is different for each tree depth. Were this computation done naively, it would require to check for the threshold only when all of the letters have been processed. Indeed, the first processed letters may drop the score below the threshold even if the similarity score can increase after that. In the recursive solution we can compute at each tree depth, given the word size s and the maximal value in the substitution matrix $\max(M)$, the maximal similarity score with respect to the letters that have already been processed.

Resulting functions can be seen in the algorithm 2.

Algorithm 2 Recursive scoring and thresholding

function SIMILARITYFUNCTION(l_1, l_2, $score : (num, den)$)
  $n \leftarrow 2M(l_1, l_2)$
  $d \leftarrow M(l_1, l_1) + M(l_2, l_2)$
  return pair $(num + n, den + d)$
end function

function THRESHOLDING($score : (num, den)$, $depth$)
  $k \leftarrow 2(s - depth - 1) \max(M)$
  return $(num + k)/(den + k) > t$
end function

Références

[1] G. P. Smith, Filamentous fusion phage : novel expression vectors that display cloned antigens on the virion surface. *Science*, 228(4705) :1315-1317, 1985.

[2] D. Wacker, C. Wang, V. Katritch, G. W. Han, X. Huang, E. Vardy, J. D. McCorvy, Y. Jiang, M. Chu, F. Y. Siu, W. Liu and H. E. Xu, V. Cherezov, B. L. Roth and R. C. Stevens, Structural Features for Functional Selectivity at Serotonin Receptors. *Science*, doi : 10.1126/science.1232808, 2013.

[3] R. Pasqualini and E. Ruoslahti, Organ targeting in vivo using phage display peptide libraries. *Nature*, 380(6572) :364-366, 1996.

[4] W. Arap, M. G. Kolonin, M. Trepel, J. Lahdenranta, M. Cardó-Vila, R. J. Giordano, P. J. Mintz, P. U. Ardelt, V. J. Yao, C. I. Vidal, L. Chen, A. Flamm, H. Valtanen, L. M. Weavind, M. E. Hicks, R. E. Pollock, G. H. Botz, C. D. Bucana, E. Koivunen, D. Cahill, P. Troncoso, K. A. Baggerly, R. D. Pentz, K. A. Do, C. J. Logothetis and R. Pasqualini, Steps toward mapping the human vasculature by phage display. *Nature Medicine*, 8(2) :121-127, 2002.

[5] M. G. Kolonin, J. Sun, K. A. Do, C. I. Vidal, Y. Ji, K. A. Baggerly, R. Pasqualini and W. Arap, Synchronous selection of homing peptides for multiple tissues by in vivo phage display. *The FASEB Journal*, 20(7) :979-981, 2006.

[6] L. G. León-Novelo, P. Müller, W. Arap, M. Kolonin, J. Sun, R. Pasqualini and K. A. Do, Semiparametric Bayesian Inference for Phage Display Data. *Biometrics*, doi : 10.1111/j.1541-0420.2012.01817.x, 2013.

[7] N. London, B. Raveh, D. Movshovitz-Attias and O. Schueler-Furman, Can Self-Inhibitory Peptides be Derived from the Interfaces of Globular Protein-Protein Interactions ?. *Proteins*, 78(15) :3140-3149, 2010.

[8] R. S. Boyer and J. S. Moore, A fast string searching algorithm. *Communications of the ACM*, 20(10) :762-772, 1977.

[9] S. Kim, A new string-pattern matching algorithm using partitioning and hashing efficiently. *Journal of experimental algorithmics*, 4, 1999.

[10] A. V. Aho and M. J. Corasick, Efficient string matching : an aid to bibliographic search. *Communications of the ACM*, 18(6) :333-340, 1975.

[11] F. J. Damerau, A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3) :171-176, 1964.

[12] T. F. Smith and M. S. Waterman, Identification of common molecular subsequences. *Journal of menucular biology*, 147(1) :195-197, 1981.

[13] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, Eugene W. and D. J. Lipman, Basic local alignment search tool. *Journal of molecular biology*, 215(3) :403-410, 1990.

[14] 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*, 48(3) :443-453, 1970.

[15] D. Gusfield, *Algorithms on Strings, Trees and Sequences : Computer Science and Computational Biology*, Cambridge University Press, 1997.

[16] E. Ukkonen, On-line construction of suffix trees. *Algorithmica*, 14(3) :249-260, 1995.