University of Waterloo Technical Report
CS-2008-14

Adaptive two-dimensional string matching for protein contact maps [⋆]

Robert Fraser and Alejandro López-Ortiz

University of Waterloo, Waterloo, ON, Canada, N2L 3G1
{r3fraser,alopez-o}@uwaterloo.ca

Abstract. Contact maps are two dimensional abstract representations of protein structures. One of the uses of contact maps is for the identification of patterns which correspond to some known configuration of protein secondary structures. In the past, searching for these patterns has generally used a naïve sliding window approach which is time consuming. We study several approaches that have been used for two dimensional string matching to accelerate the time requirements of these searching operations, and demonstrate experimentally the efficacy of these algorithms in our domain. Finally, we present an adaptive analysis of the problem, by restricting the search to only those regions of secondary structure that we are interested in. The same searches as performed previously are executed again using the adaptive approach, and the improvements make the search tractable.

1 Introduction

The motivation for this problem is protein structure prediction. Our long term research goals are to add to the set of tools available for protein structure prediction. The contact map is an abstract binary representation of the structure of a protein; creating a contact map from a protein with known structure is a lossy procedure. Our desire is to reverse this step and reconstruct the three dimensional structure of the protein from this abstract representation. People have attempted to recover the three dimensional structure of a protein from the contact map in the past [VK97], but the success has been limited. We wish to identify local substructures that can be identified and associated with representative patterns in contact maps. To accomplish this, we wish to search the known body of protein structures to identify such patterns, but the cost of doing so is prohibitive. The Protein Data Bank (PDB) contains over 30000 known protein structures at present, so an exhaustive search becomes very time consuming.

[⋆] This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the University of Waterloo.

Thus, the nature of the problem is simple: we have a small rectangular binary pattern which we wish to search for in a database of many large binary patterns. This problem is well studied, and there are approaches which are optimal with regard to worst case and expected case time performance [Ta96]. The faster algorithms do provide some acceleration in our domain, although the gains are not substantial. A significant improvement is acquired by exploiting additional information available in our application and restricting the search to where the pattern is likely to be found.

2 Protein Structure

In this section, we review some basic structural properties of proteins. The building blocks of proteins are amino acids, which bond together in a chain to form the structure of the protein. The sequence of these structures is often referred to as the primary structure, and it is easy to obtain for unknown proteins. These amino acids also interact with other amino acids besides their immediate neighbours. These interactions result in secondary structures, such as the alpha helices that we are interested in. The prediction of secondary structure is well studied (see Rost [Ro01] for a review). Finally, these secondary structures interact to form the three-dimensional tertiary structure of the protein. This is the focus of our study: the interaction between pairs of alpha helices. The prediction of the tertiary structure of a protein from the amino acid sequence is one of the largest open problems in computational biology.

3 Contact Maps

A contact map can be viewed as an abstract translational and rotational invariant representation of a protein's topology, which captures much of its relevant structural information. A contact map is an $N \times N$ matrix, where N is the number of amino acids in the given protein, and entry C_{ij} in the matrix is a boolean, indicating whether amino acid i is in contact with amino acid j. A threshold distance between atoms is the conventional definition of a contact; values ranging from 7 to 10Å between C_α atoms are commonly used [Fr06], p.26. It has been shown that regions of contact maps can be used to identify physical properties of pairs of alpha helices [FG07]. In this case, we often need to search for a small contact pattern (the target) within a large number of source contact maps, such as the entirety of the Protein Data Bank (PDB). An example of a contact map and a refined region corresponding to a pair of alpha helices is shown in Figure 1. Notice that the source contact maps are always square, while the target map is rarely square. This is because the source maps compares the position of every amino acid to every other one, while the target map compares the positions of amino acids in one alpha helix to those in another, and the two alpha helices will rarely be the same length.

Currently, the prediction of contact maps from amino acid sequences is in its early stages [PB02,PR05], but results are encouraging. For the purpose of our

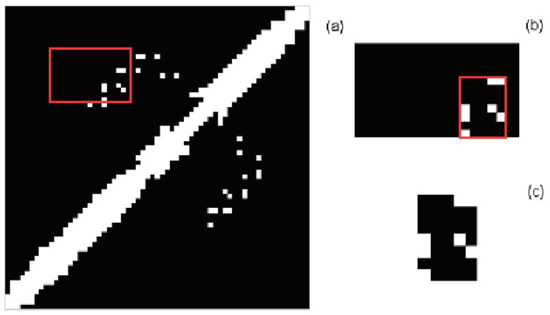


Fig. 1. (a) This is the contact map for protein 1a0a from the Protein Data Bank (PDB). The red rectangle indicates the area occupied by two helices, shown in (b). The contact map represents all of the amino acids for one alpha helix along the vertical axis and the other along the horizontal. This has been further refined to the interface area, shown in (c). The contact map interface is found by isolating the smallest rectangle containing all of the contact points from the contact map for the helix pair.

research, it is assumed that the results of contact map prediction will be accurate at some later stage. Our interest at present is to use an empirically determined contact map to recover the original three-dimensional configuration of a pair of alpha helices. This forms a phase in a hierarchical approach where these pairs could assembled to derive the overall structure of the protein [GKD06].

4 Searching Contact Maps

Consider the pattern shown in Figure 1(b), which corresponds to a pair of alpha helices. To locate pairs of alpha helices with similar properties, the naïve approach is to take this pattern and compare it with each possible position for a match in the source contact map. This would usually be done with a sliding window approach. Given a source contact map of size $N \times N$, and a target map of size $I \times J$, the running time of this approach is $\theta(N^2 \cdot I \cdot J)$. There is a worst case lower bound of $\theta(N^2)$ for this problem, since we desire an exact solution, so the only savings possible are through the removal of the I and J factors. These savings, while seemingly small next to the N^2 term, become significant when searching large numbers of source maps. This is the case in our application, where we search the entirety of the PDB, consisting of over 30 000 files.

There are several well known efficient string matching algorithms for text, but in our case we require two dimensional string matching. Bird [Bi77] and Baker [Ba78] (BB) independently initiated this field of research 30 years ago, and their ideas still form the foundation of recent approaches. We will look at their approach first.

4.1 The Linear Time Algorithm

For our study, we examined the technique as outlined by Bird [Bi77]. The first step involves searching for the rows of the target in the source pattern. He uses an earlier technique created by Aho and Corasick [AC75] for searching text for a several different words simultaneously, which is in turn an extension of the famous KMP string matching algorithm. We begin with the creation of a finite state machine that models the transitions representing each row of our target map. We will illustrate this with a series of examples. Consider the target map shown in Figure 2(a). Now we create a trie that contains all of the rows of the target map. We create the trie (referred to as the goto function in this application [AC75]) by moving row by row down the target map, and the states are labelled incrementally in the order that we encounter them. This trie is shown in Figure 2(b). This is the conventional technique [BYR93]. The use of a trie rather than a full finite state machine requires the definition of a separate failure function. This failure function, along with the accepting states, are shown in Table 1.

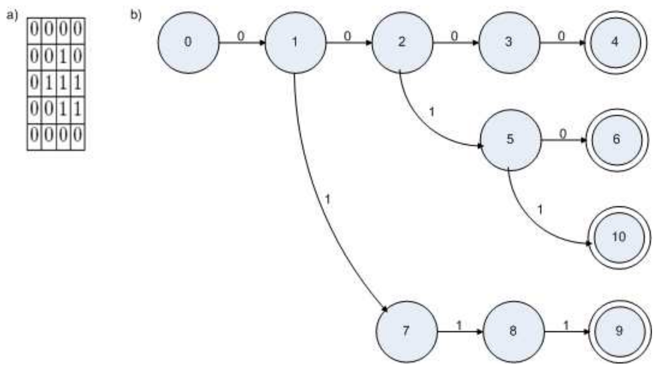


Fig. 2. The trie generated from our contact map in Table 1. State 0 is the starting state, and states 4, 6, 9, and 10 are accepting states.

Table 1. The failure and accepting functions for our contact map.

Node (i)	0	1	2	3	4	5	6	7	8	9	10
$f(i)$	0	0	1	2	2	7	1	0	0	0	8
$Acc(i)$	0	0	0	0	1	0	1	0	0	1	1

Now that we have these functions, we can search the source file for this pattern. However, the problem is only half solved, since we have found only a row in this instance, but we still need to search for the other rows above and below the one that we have identified to determine if we have a match. In order to do this, Bird's [Bi77] approach was another string matching problem, but now we will be matching the columns of the target to the source. This involves an approach again based on KMP string matching, except now we treat each row of our target as a single symbol so that we can maintain linearity. Our contact map would be represented as 12341, since rows 1 and 5 are the same. By maintaining an array of size N, with each entry corresponding to a column in the source contact map, we can track the value of the last row found in each position. If we reach a position where in some column we find a match for row 5 in our second string matching machine, then we know that we have found a match for our target map.

This algorithm runs in $O(N^2 + I \cdot J)$ [Bi77], given a source contact map of size $N \times N$, and a target map of size $I \times J$. The construction of the machinery takes $O(I \cdot J)$ time, and then the actual search takes $O(N^2)$ time. Since there are $O(N^2)$ elements to search in our source contact map, and N is greater than I and J, this is a linear time algorithm in the size of the input. This improved time does come at a slight extra space requirement for the extra array and the machinery needed, $O(N)$ and $O(I \cdot J)$ respectively, but again this is less than the $O(N^2)$ space needed for the source data itself. This solution is worst case optimal, but Baeza-Yates and Régnier [BYR93], among others, have provided an algorithm that is sub-linear in the expected case.

4.2 Expected Sublinear Pattern Matching

As we stated earlier, since we need to look at every element in the source, any optimal algorithm will run in $\theta(N^2)$ time in the worst case. There are several algorithms the run in sublinear expected time, however. The first was introduced by Baeza-Yates and Régnier [BYR93]. The key insight in their approach is that since we have a pattern with I rows, we really only need to search every I^{th} row of the source for matches. If no target rows are found in row k of the source, nor in row $k + I$ of the source, then we know that the pattern will not be found in the intervening rows, and they can be skipped. This algorithm is superlinear in the worst case, $O(N^2 \cdot I)$, but the average case performance is $O(N^2/I + I^2)$ for randomized data. There are other expected sublinear algorithms, such as that presented by Tarhio [Ta96], which is based on the ideas of the Boyer-Moore [BM77] string matching algorithm which searches from right to left. However, the practical performance gains of Tarhio's algorithm over the others is marginal on the problem sizes we are faced with [Ta96], so we use Baeza-Yates and Régnier's (BYR) approach as the representative for expected sublinear approaches in our study. Furthermore, as we shall see, the advantages presented by these sublinear approaches are virtually eliminated in our adaptive analysis due to the nature of our data.

5 Adaptive Analysis of Contact Map Search

Adaptive analysis is a study of a problem where some properties of the source data can be exploited to achieve both practical and theoretical gains in the complexity of a problem. Perhaps the best known application of adaptive analysis is sorting. Given some sequence of numbers that are to be sorted, it is clear that some sequences are easier to sort than others. This concept leads to the idea of measures of presortedness, which are metrics for quantifying how far from being sorted a sequence is. There are many such measures for sorting, such as the number of inversions, or the maximum distance that an element is from the position that it will occupy in the sorted array. Estivill-Castro and Wood [ECW92] provide a review of these measures as applied to sorting.

We wish to apply this type of analysis to the searching of contact maps, and more specifically to the searching of contact maps related to pairs of alpha helices. The insight here is clear: there is no point in searching an area of a source contact map if the contacts do not correspond to alpha helices. For an empirically defined contact map, we can obtain the secondary structure information using DSSP [KS83]. Studies using predicted contact maps would use the predicted secondary structure information [Ro01]. Thus, our source data for a protein is an array of length N containing the secondary structure for each amino acid, and the $N \times N$ contact map. Given our target pattern of size $I \times J$, we can now walk along the array and identify regions corresponding to pairs of alpha helices of size greater than $I \times J$ in time $\theta(N)$. Finally, we restrict our search to these regions, and the contact map can be searched in sublinear time. This adaptive algorithm is presented in Algorithm 1.

Algorithm 1 is generalized so that any search algorithm can be used as a black box at the search step. For example, this could be the naïve algorithm, or one of the faster algorithms discussed earlier. We implemented each of the algorithms discussed in this paper (naïve, BB, and BYR) to determine which was best in this adaptive approach. For our formal analysis, we will first consider the naïve approach. The first loop to identify all of the regions of the protein that are in alpha helices takes linear time, as stated earlier. The second for loop depends on the number of alpha helices that are found. We define two variables, ξ_I and ξ_J, which represent the fraction of amino acids that are in alpha helices that will be searched in each direction. For example, suppose that our protein had 100 amino acids, and there are four helices a, b, c, d of length 5, 10, 15, and 20. If our target map is 8×12, then we only need to search the regions $b \times c$, $b \times d$, $c \times d$, and $d \times c$. So in this example, $\xi_I = (10 + 15 + 20)/100 = 0.45$, and $\xi_J = (15 + 20)/100 = 0.35$. We define $\xi = \xi_1 \cdot \xi_2$ to simplify notation (recall that each value is from a different axis, so they are independent). Therefore, the searchable region in the source map is $O(N^2 \cdot \xi \cdot I \cdot J + N)$. If we were searching a contact map that contained zero or one alpha helices, the cost of this search would thus be $\theta(N)$. To be more precise, we would subtract the regions included in theses ranges that we do not search, which is each alpha helix with itself ($c \times c$ and $d \times d$ in our example). Also, we should subtract I and J from the cost for each pair of helices. This cost appears much higher than it really is because the

Algorithm 1 The algorithm for Adaptive Contact Map Search. The function takes three arguments: C is the $N \times N$ source contact map, SS is the array of length N giving the secondary structure values for each element, and T is the target contact map of size $I \times J$. We build a set AH, which is a list of the alpha helices in C.

```
SearchMaps($C, SS, T$)
$AH = \{\}$
for $i = 1, ..., N$ do
  if $SS(i)$ =alpha helix then
    start= $i$
    while $SS(i)$ =alpha helix do
      $i = i + 1$
    end while
    end= $i - 1$
    $AH = AH \bigcup \{$(start,end)$\}$
  end if
end for
for $i = 1, ..., |AH|$ do
  for $j = 1, ..., |AH|$ do
    if $i \neq j$ &&
    $(AH(i).end - AH(i).start) >= I$ &&
    $(AH(j).end - AH(j).start) >= J$ then
      run search algorithm on $AH(i) \times AH(j)$ region of $C$
    end if
  end for
end for
```

analysis presented the regions as being contiguous. We can thus redefine ξ'_I and ξ'_J as follows:

$$\xi'_I = \sum_{i=1}^{|AH|} AH(i).end - AH(i).start - I + 1,$$

where AH is as defined in Algorithm 1, and we set the term $AH(i).end - AH(i).start - I = 0$ if it is a negative value for any given i. ξ'_J is defined analogously. This represents the fact that searching a 3×3 map with a 2×2 map requires 4 shifts of the sliding window, not 9. Our new values for our example are now $\xi'_I = (0 + 3 + 8 + 13)/100 = 0.24$, $\xi'_J = (0 + 0 + 4 + 9)/100 = 0.13$, and $\xi' = 0.0312$, which represents significant savings. The ξ' argument does not apply to the other algorithms used however, since they shift through each column of the source maps, so they use the original definition of ξ.

For the Bird [Bi77] algorithm, we carry over the savings of the linear time algorithm. Their bound was $O(N^2 + I \cdot J)$, and the same arguments above apply to their approach since their algorithm will work just as well on these smaller subregions of the map. Thus, their algorithm takes time $O(N^2 \cdot \xi + I \cdot J + N)$. As indicated by this bound, we expect that the advantages of their approach will be less pronounced in this adaptive scenario. The algorithm of Baeza-Yates and Régnier [BYR93] will have similar worst case performance, and the expected case performance is expressed as $O(N^2 \cdot \xi/I + I^2 + N)$.

6 Results

We first chose a pattern at random, which corresponds to the third and fourth helices in the 1a0a protein, shown in Figure 1. We searched for this pattern using each of the three algorithms discussed in this paper, and both using the original method and the new adaptive approach. The results are shown below in Table 2.

Table 2. The time in minutes required by each algorithm to find the selected pattern, both at the 7 and 10Å resolution maps.

	Original (min)			Adaptive (min)		
	Naive	BB	BYR	Naive	BB	BYR
7Å	541.1	208.4	448.8	65.4	68.4	67.3
10Å	517.1	205.6	295.8	65.2	65.6	64.3

We note some interesting observations in these results. First, the linear time approach of Bird and Baker is faster than the naïve approach and the expected sublinear BYR algorithm; substantially so in the case of the 7Å maps. Second, all of the algorithms have fairly equal performance in the adaptive implementations. Most significant for our research is that the adaptive approach is much faster than the original implementations, regardless of which algorithm is used. The reason that the BYR algorithm is not doing as well as expected in the original implementation is because of the sparsity of our data. The patterns that we are searching for contain multiple rows comprised entirely of zeros, and there are many occurrences of sequences of zeros in the source data sets. Therefore, this algorithm is running close to the worst case time complexity, $O(N^2 \cdot I)$.

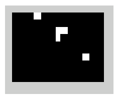
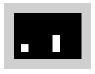
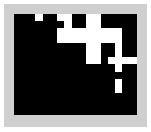
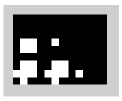
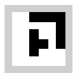
The adaptive approach to the algorithm results in doing many more individual searches than previously, but the size of the source data set for each problem is much smaller. In fact, the source matrices are now on the scale of the target maps, rather than being an order of magnitude or two larger. As a result, the algorithms have similar performance in the adaptive approaches; each is running at close to linear time. The naïve algorithm is running at close to linear because the sliding window does not have to shift much. Assuming that we are using square patterns for the moment for simplicity, where the source pattern is of size N^2 and the target is of size M^2, we know that the worst case running time of this algorithm is $O(N^2 \cdot M^2)$. In reality, this is $(N - M + 1)^2 + M^2$, so as N approaches M, the running time of the algorithm approaches true linearity. In addition, the naïve implementation has the sliding window shift as soon as there is a mismatch rather than continuing to check the whole target pattern at a given position, so the expected running time improves as well. Between these two factors and the observations in Table 2, the adaptive naïve implementation appears to be a reasonable option.

To test these results further, we searched for a few more maps using the adaptive approaches only. The first is a smaller map, so that the naïve approach should not do as well, which we will run at both resolutions. The second is a map with no zeros so that the BYR approach should do better. The results are shown below in Table 3. All of the target contact maps used in this study are shown graphically in Table 4. All of the maps but the last consist of the interface region of the contact map for the pair plus up to three rows and columns of zeros around the interface if they are present in the map for the pair (recall that the map for the pair corresponds to Figure 1(b), and the interface is Figure 1(c)). These three extra rows or columns ensure that there is a turn of the helix where there are no further contacts, which ensures that there is a true match for the patterns. Alternatively, the last search which uses just the interface would be desirable if you were not concerned about the presence of contacts in the next turn of one of the helices. We do not perform the latter search at 7Å resolution because we couldn't find a large target with no zero rows (the diameter of an alpha helix is 5.4Å).

Table 3. The time in minutes required by each adaptive algorithm to find the small pattern, both at the 7 and 10Å resolution maps, and a map with no zeros at 10Å resolution.

	Naive (min)	BB (min)	BYR (min)
Small 7Å	133.5	73.9	81.8
Small 10Å	99.9	73.9	73.7
No Zeros	75	76.1	67.6

Table 4. The graphical representations of each target map. All targets were obtained from the 1a0a source contact map.

Resolution	Run 1	Run 2	Zeros
7Å			Not performed
10Å			

For completeness, we describe our experimental setup. First, we used Matlab because the Bioinformatics toolkit facilitates parsing PDB files. However, this study could also be done using C++ or some other language since these algorithms are just operating on the binary matrices and the Bioinformatics toolkit is not used for the string matching operations. This would likely result in faster search times for each approach, but we expect that the relative results should be similar to those obtained here. Also, we were curious to know how much time was being used performing file I/O, since this could be mitigated by using another language or advanced caching techniques or something of the sort. To do this, we performed a run where we just opened and read the contents of each file without doing any analysis, and this took 57.3 minutes, or roughly three quarters of the adaptive execution time. The contacts maps are stored as binary *.mat* files, they take up about 800MB of disk space.

7 Conclusions and Future Work

We have presented several algorithms for searching for a small contact map pattern in numerous large source contact maps for exact matches. Each of them promises to provide better performance than the naïve algorithm, which would increase the tractability of these search problems, and this was observed through their implementation and application in our domain. Contact maps are typically sparse, as are the patterns that we are searching for. This results in poor performance for the naïve algorithm. In randomized data, the naïve algorithm typically does well [BYR93] since it is simple to stop checking the pattern once a mismatch has occurred rather than checking all possible positions. However, because our data is sparse, we often have entire rows of zeros that will match and will cost this approach. This can be seen in Figure 1 (b), where the contact map is composed of strictly zeroes in the top rows. This is precisely the situation where the Bird approach will excel, since these points will not be searched again. The speed-up given by the BYR algorithm will be virtually negated due to sparsity, since we will be finding many matches in each row searched for the rows of zeroes in our target. Based on these observations, the Bird approach is the best choice.

This observation gives us an insight that provides for improved searching regardless of which algorithm is used. These algorithms will run close to their worst case complexity because of all the zeroes present. This can be avoided by searching for the interface region (Figure 1 (c)), followed by a check for all the surrounding zeroes for positive matches, such our last run in Table 3.

This paper presents several new ideas. This is the first known adaptive analysis to be performed on the two dimensional string matching problem. Although it is application specific, the general technique may be applied in other areas given some extra domain-specific knowledge such as our secondary structure information. Also, we have shown experimentally that the BB algorithm for string searching is best when searching for patterns with rows of zeros in sparse matrices. In the adaptive application, the best algorithm was dependent on the number of rows containing zeros in the target pattern. When performing searches where one is adding rows and columns of zeros around the interface, then the BB approach is best, while the BYR algorithm is better for searches for the interface region only.

There are several other refinements that may be carried out to further improve the speed of searching. One is to perform the matching on compressed data. Due to the sparsity of the data, contact maps are prime candidates for run-length encoding schemes, and searches that use this technique should perform better than the on-line approaches discussed here. Further, it is possible that the database could be indexed so the indexed searches may be performed. Another question pertains to the threshold that is chosen for the contact map. We have used 7 and 10Åas examples in our study, but it is unclear which is the best choice for any given application, or how to determine which is best. Also, since the selection of the threshold is arbitrary and there will be values close to the threshold that could go one way or the other, it would be interesting to test the effectiveness of approximate pattern matching techniques for our application. Finally, it would be interesting to explore algorithms for 2-dimensional pattern matching which are tailored to sparse data.

8 Acknowledgements

The authors wish to thank Janice Glasgow for the inspiration for this project and Alejando Salinger and Arash Farzan for their insights and discussions.

References

[AC75] A.V. Aho and M.J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6), 333-340, 1975.

[Ba78] T. Baker. A technique for extending rapid exact string matching to arrays of more than one dimension. *SIAM Journal on Computing*, 7, 533-541, 1978.

[Bi77] R.S. Bird. Two dimensional pattern matching. *Information Processing Letters*, 6(5), 168-170, 1977.

[BM77] R. Boyer and S. Moore. A fast string matching algorithm. *Communications of the ACM*, 20, 762-772, 1977.

[BYR93] R. Baeza-Yates M. and Régnier. Fast two dimensional pattern matching. *Information Processing Letters*, 45, 51-57, 1993.

[ECW92] V. Estivill-Castro and D. Wood. A survey of adaptive sorting algorithms. *ACM Computing Surveys*, 24(4), 441-476, 1992.

[FG07] R. Fraser and J. Glasgow. A Demonstration of Clustering in Protein Contact Maps for alpha Helix Pairs. In *Proceedings of the 2007 International Conference on Adaptive and Natural Computing Algorithms (ICANNGA)*, LNCS 4431, 758-766.

[Fr06] R. Fraser. A Tale of Two Helices: A study of alpha helix pair conformations in three-dimensional space. Master's thesis, Queen's University, 2006.

[GKD06] J. Glasgow, T. Kuo and J. Davies. Protein Structure from Contact Maps: A Case-Based Reasoning Approach *Information Systems Frontiers*, 8(1), 29-36, 2006.

[KS83] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12), 2577-2637, 1983.

[PB02] G. Pollastri and P. Baldi. Prediction of contact maps by GIOHMMs and recurrent neural networks using lateral propagation from all four cardinal corners. *Bioinformatics*, 18(Supplement 1):S62–S70, 2002.

[PR05] M. Punta and B. Rost. PROFcon: novel prediction of long-range contacts. *Bioinformatics*, 21(13):2960–2968, 2005.

[Ro01] B. Rost. Review: Protein Secondary Structure Prediction Continues to Rise. *Journal of Structural Biology*, 134(2-3), 204-218, 2001.

[Ta96] J. Tarhio. A sublinear algorithm for two-dimensional string matching. *Pattern Recognition Letters*, 17, 833-838, 1996.

[VK97] M. Vendruscolo, E. Kussell, and E. Domany. Recovery of protein structure from contact maps. *Folding and Design*, 2(5):295–306, 1997.