Online pattern matching in sparse matrices and contact maps

Robert Fraser[a]*

[a]David R. Cheriton School of Computer Science,
University of Waterloo,
Waterloo, ON, Canada, N2L 3G1

ABSTRACT

Contact maps are two dimensional abstract representations of protein structures. Some patterns in contact maps correspond to configurations of protein secondary structures. Searching for such patterns may typically use a naïve sliding window approach, and we study techniques which accelerate the searching operations in the online setting, including a restricted search algorithm which operates only on relevant areas of the matrix.

Keywords: String matching, contact maps, protein structure, adaptive analysis

1 INTRODUCTION

The motivation for this problem is 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. People have attempted to recover the three dimensional structure of a protein from the contact map [1], but success has been limited. We wish to identify local substructures that can be identified and associated with representative patterns in contact maps, by searching the known body of protein structures. The Protein Data Bank (PDB) contains over 30000 known protein structures at present, so an exhaustive search can be very time consuming. Since the PDB is a dynamic entity, we restrict this discussion to the online setting. The essence 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.

2 PROTEIN STRUCTURE & CONTACT MAPS

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. Amino acids interact with other amino acids, resulting in secondary structures such as alpha helices. Finally, these secondary structures interact to form the three-dimensional tertiary structure of the protein. The interaction between pairs of alpha helices is the focus of our research.

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 7Å to 10Å between C_α atoms are commonly used [2], p.26. It has been shown that regions of contact maps can be used to identify physical properties of pairs of alpha helices [3]. 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). 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, since a source map compares the position of every amino acid to every other one, while a 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.

3 SEARCHING IN 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, denoted the '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 is $\theta(N^2 \cdot I \cdot J)$. Note that there is a worst case lower bound of $\Omega(N^2)$ for the matching problem since we desire an exact solution.

*Corresponding author. E-mail: r3fraser@uwaterloo.ca, Telephone: +1(519)885-1211 x35351.

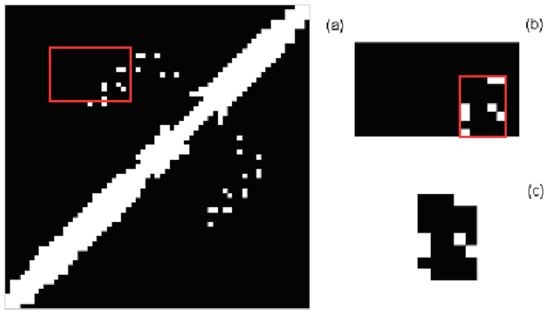


Figure 1. (a) The contact map for protein 1a0a from the Protein Data Bank (PDB). The rectangle indicates the area occupied by two helices, shown in (b). The contact map represents all of the amino acids for one alpha helix along the vertical axis and the other along the horizontal. This has been further 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.

3.1 The Linear Time Algorithm

Bird [4] and Baker [5] (BB) independently discovered linear time algorithms for two-dimensional pattern matching, and here we examine the technique as outlined by Bird. The first step involves searching for the rows of the target in the source pattern. A finite state machine is built that models the transitions representing each row of our target map. We create a trie (or goto function [6]) by moving row by row down the target map, and labelling the states incrementally in the order that we encounter them. The use of a trie rather than a full finite state machine requires the definition of a separate failure function. We find a row of the target by reaching an accepting state, 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 complete match. Another string matching algorithm is used between the columns of the target and source, where each row of the target is treated as a single symbol. By maintaining an array of size N, we can track the value of the last row found in each column. This algorithm runs in $O(N^2 + I \cdot J)$ [4], and since there are $O(N^2)$ elements to search through in our source contact map and $N^2 \gg I \cdot J$, this is a linear time algorithm in the size of the input.

3.2 Expected Sublinear Pattern Matching

Since we need to look at every element in the source, any algorithm will run in $\Omega(N^2)$ time in the worst case. However, sublinear expected time is possible, as introduced by Baeza-Yates and Régnier [7]. 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 nor row $k + I$ of the source, then we know that the pattern will not be found in the intervening rows. This algorithm is superlinear in the worst case, $O(N^2 \cdot I)$, but the average case performance is $O(N^2/I + I \cdot J)$ for randomized data. There are other expected sublinear algorithms, such as that presented by Tarhio [8] which is based on the Boyer-Moore [9] string matching algorithm which searches from right to left. However, the practical performance gains over the others is marginal on the problem sizes we are faced with [8], so we use Baeza-Yates and Régnier's (BYR) approach as the representative for expected sublinear approaches.

4 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 [10]. 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.

We wish to apply this type of analysis to the searching of contact maps for 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. 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. We can now walk along the array and identify regions corresponding to pairs of alpha helices of size greater than or equal to $I \times J$ in time $\theta(N)$. We restrict our search to these regions, and the contact map can be searched as presented in Algorithm 1.

Algorithm 1 is generalized so that any search algorithm can be used as a black box. We implemented each of the algorithms discussed in this paper (naïve, Bird and Baker (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

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 pairs of endpoints 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** $i \neq j$ && $(AH(i).end - AH(i).start + 1) >= I$ && $(AH(j).end - AH(j).start + 1) >= J$ **then**
      run search algorithm on $AH(i) \times AH(j)$ region of $C$
    **end if**
  **end for**
**end for**
```

of the regions of the protein that are in alpha helices takes linear time. The second for loop depends on the number of alpha helices that are found. We define two variables $\xi_I, \xi_J \in [0,1]$, which represent the fraction of amino acids relevant to the search. 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 $d \times c$. Further, we subtract I and J from the cost for each pair of helices. We can thus define ξ_I and ξ_J as follows:

$$\xi_\Xi = \sum_{i=1}^{|AH|} AH(i).end - AH(i).start - \Xi + 1,$$

where $\Xi \in \{I, J\}$ and AH is a list of the alpha helices, such that $AH(i)$ corresponds to the i^{th} alpha helix in the protein. $AH(i)$ stores a pair $\{$start,end$\}$, which are the indices of the first and last amino acids in the i^{th} helix. $|AH|$ is the size of AH (which is the total number of alpha helices in the protein). We set the value $AH(i).end - AH(i).start - \Xi = 0$ if it is a negative value for any given i when computing the sum for ξ_Ξ. The values for our example are $\xi_I = (0+3+8+13)/100 = 0.24$, $\xi_J = (0+0+4+9)/100 = 0.13$, and $\xi = 0.0312$, where we define $\xi = \xi_I \cdot \xi_J$ to simplify notation (recall that each value is from a different axis, so they are independent). Therefore, the running time of adaptive naïve search 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 be $\theta(N)$.

For the Bird [4] 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 [7] will have similar worst case performance, and the expected case performance is expressed as $O(N^2 \cdot \xi / I + I \cdot J + N)$.

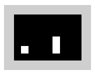
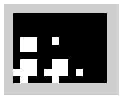
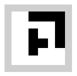
5 EXPERIMENTAL RESULTS

We searched for matches to the pattern shown in Figure 1 in the PDB using each of the three algorithms discussed in this paper, both using the standard method and the adaptive approach. The results are shown in Table 1. 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. All of the algorithms have fairly equal performance in the adaptive implementations, but the adaptive approach is overall much faster than the original implementations, regardless of which algorithm is used. The reason that the BYR algorithm is not doing as well as might be expected in the original implementation is because of the sparsity of the data. The patterns that we are searching for contain multiple rows

Table 1. The time required by each algorithm to find the chosen pattern, both at the 7Å and 10Å resolution maps.

	Original (min)			Adaptive (min)		
	Naive	BB	BYR	Naive	BB	BYR
7Å	483.8	151.1	391.5	8.1	11.1	10.0
10Å	459.8	148.3	238.5	7.9	8.3	7.0

Table 2. A comparison of the time required by each adaptive algorithm.

	Small 7Å	Small 10Å	No Zeros
Naive (min)	76.2	42.6	17.7
BB (min)	16.6	16.6	18.8
BYR (min)	24.5	16.4	10.3
map			

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 superlinear worst case time complexity, $O(N^2 \cdot I)$.

The algorithms have similar performance in the adaptive approaches; each is running at close to linear time. In addition, since the naïve implementation has the sliding window shift as soon as there is a mismatch, the expected running time improves further. To further distinguish the adaptive algorithms, we searched for a small map where the naïve approach should not do as well. Also, we searched for a map with no rows of zeros so that the BYR approach should excel. The results are shown in Table 2, along with images of the target contact maps used in this study. Typically, the maps 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 original 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, but impair the performance of the BYR algorithm.

5.0.1 Experimental setup

We used Matlab for implementations with the Bioinformatics toolkit for parsing PDB files. This study could be done using another language to obtain faster search times for each approach, but the relative results should be similar to those obtained here. Also, times reported in this study express the total time used while operating on files, but the time used by Matlab for opening and closing individual PDB files was not included.

6 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 in an online setting. Each of them provides better performance than the naïve algorithm, increasing the tractability of these search problems, as was observed through their implementation and application. Contact maps are typically sparse, as are the patterns that we are searching for, which results in poor performance for the naïve algorithm. The speed-up expected from the BYR algorithm is often negated due to sparsity. Based on these observations, the Bird approach is usually the best choice.

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 domain-specific knowledge analogous to our secondary structure information. We presented experiments suggesting that the BB algorithm is best when searching for patterns with rows of zeros in sparse matrices. In the adaptive setting, the best algorithm was dependent on the number of rows containing zeros in the target pattern. When performing searches where one is padding the interface with zeros, the BB approach is best, while the BYR algorithm is better for searches for interfaces with no blank rows.

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 seem ideal for run-length encoding schemes, and such techniques allow better performance than online approaches. Further, it is possible that the database could be indexed, allowing instantaneous searches.

ACKNOWLEDGMENTS

The author wishes to thank Janice Glasgow for the inspiration for this project and Alejando Salinger and Arash Farzan for their insights and discussions. Finally, the detailed and high quality feedback received from the anonymous reviewers is greatly appreciated.

References

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

[2] R. Fraser, "A tale of two helices: A study of alpha helix pair conformations in three-dimensional space," 2006.

[3] R. Fraser and J. Glasgow, "A demonstration of clustering in protein contact maps for alpha helix pairs," in *Proc. Int'l Conf. on Adapt. and Nat. Comp. Algs (ICANNGA)*, *LNCS* **4431**, pp. 758–766, 2007.

[4] R. Bird, "Two dimensional pattern matching," *Inf. Proc. Lett.* **6**(5), pp. 168–170, 1977.

[5] T. Baker, "A technique for extending rapid exact string matching to arrays of more than one dimension," *SIAM J. on Comp.* **7**, pp. 533–541, 1978.

[6] A. Aho and M. Corasick, "Efficient string matching: an aid to bibliographic search," *CACM* **18**(6), pp. 333–340, 1975.

[7] R. Baeza-Yates and M. Régnier, "Fast two dimensional pattern matching," *Inf. Proc. Lett.* **45**, pp. 51–57, 1993.

[8] J. Tarhio, "A sublinear algorithm for two-dimensional string matching," *Patt. Rec. Lett.* **17**, pp. 833–838, 1996.

[9] R. Boyer and S. Moore, "A fast string matching algorithm," *CACM* **20**, pp. 762–772, 1977.

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