Matching Observed Alpha Helix Lengths to Predicted Secondary Structure*

Brian Cloteaux†
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
brian.cloteaux@nist.gov

Nadezhda Serova
University of Maryland, Baltimore County
Baltimore, Maryland, USA
nserova1@umbc.edu

Abstract

Because of the complexity in determining the 3D structure of a protein, the use of partial information determined from experimental techniques can greatly reduce the overall computational expense. We investigate the problem of matching experimentally observed lengths of helices to the predicted secondary structure of a protein. We give a simple and fast algorithm for producing a library of possible solutions. Then we test our algorithm by performing a series of computational experiments of predicting the alpha helix placement of proteins with an already known order. These tests seem to demonstrate that our method, if given a good prediction of the protein's secondary structure, can generate high quality lists of potential placements of the helix length onto the protein sequences.

1 Introduction

Understanding how specific proteins fold, or arrange themselves in three dimensional space (3D) based on environmental and internal chemical constraints, is necessary to determine the how these proteins function. But even with the amino acid sequence of the proteins (1D structure) known, the prediction of their corresponding 3D structure is an extremely challenging problem.

This challenge is both from an experimental and computational viewpoint. Proteins require precise environments to fold properly. Because of these numerous complications in measuring these protein under the correct environment, experimental methods for ascertaining the 3D arrangements are expensive, time consuming, and of limited accuracy. X-ray crystallography, for example, is a powerful technique in the determination of the structures, however it is ineffective in proteins that are not easily crystallized, such as membrane proteins. Many other methods provide only partial information about the protein's 3D structure.

At the same time, the computational problem of determining 3D structure of these proteins is, in general, intractable. In order to reduce the difficulty of this problem, a recent approach has been to use computational methods to match observed secondary structure to the possible placements on the 1D structure. This paper extends an original investigation by He, Lu, and Pontelli [4] of the problem of matching observed lengths of the alpha helices from the electron cryomicroscopy technique to the predicted areas of secondary structure.

Electron cryomicroscopy can be used to produce a density map of some proteins. Although with current technology the resolutions of such maps are relatively low, certain secondary structures (such as alpha helices) can still identified. These alpha helices can be identified as lengths of amino acids, although the location of these helix lengths on the protein sequence is not clear. He, Lu, and Pontelli suggested using placing these observed length on the predicted probability of the individual amino acids in the sequence being in a helix. Prediction servers have been created that are able to use the information about the protein's sequence in order to predict the placement of alpha helices on the 1D sequence. Still, these methods have limited accuracy and so the best result that can be computed is a set of possible arrangements of the observed lengths that a researcher can use as a starting point in determining 3D structure.

This paper offers two contributions to the matching of observed lengths to their placement on the 1D protein structure. The first is to examine the complexity and necessity of computing the optimal length placement. We give evidence that computing optimal solutions may not be worth the computational expense.

A second contribution is to introduce a new approach to computing possible arrangements. He, Lu, and Pontelli introduced a method to produce a library of likely mappings to serve as starting points for a researcher. Our approach is similar to the He, Lu, and Pontelli method in the sense that it does not generate all of the possible mappings nor does it try to find an optimal solution. Instead, we give a simple

†Corresponding author.

heuristic algorithm that gives a good approximation of the placement of the lengths and then using this approximation as a starting point, we randomly modify it to look for other possible solutions. We collect the best arrangements to use as a library of possible mappings.

To test our approach, we compared the predicting length placement produced by our algorithm to actual ordering on several known proteins. These tests show that our method is a fast and simple approach for producing high quality possible placements of observed alpha helix lengths onto a protein's sequence.

2 The Maximal Cover Sum Problem

Before we examine the problem of mapping the observed alpha helices to the predicted secondary structure, we first will consider a closely related problem that we call the *maximal cover sum problem*. This problem consists of a set $\mathscr{C}$ of covers with an associated function $\omega : \mathscr{C} \to \mathbb{N}$ that gives the length of each cover. There is also an n-length string P of positive real numbers, i.e. $P \in \mathbb{R}_+^n$. The expression P_i is used to denote the ith value in the string P.

We define a placement of the covers on the string using an index function $I : \mathscr{C} \to \{1..|P|\}$. The value $I(c)$ gives the index in P of the first location to place the cover c. Any index function has the following three restrictions.

1. $\forall c_1, c_2 \in \mathscr{C}, I(c_1) = I(c_2)$ if and only if $c_1 = c_2$
2. $\forall c_1, c_2 \in \mathscr{C}$, if $I(c_1) < I(c_2)$ then $I(c_1) + \omega(c_1) < I(c_2)$
3. $\forall c \in \mathscr{C}, I(c) + \omega(c) \leq |P|$

The first two restrictions say that covers are not allowed to overlap as they cover the string P. The last restriction prevents covers from extending beyond the length of the string P. These restrictions trivially imply that if for a problem instance the condition

$$|P| \geq \sum_{c \in \mathscr{C}} \omega(c)$$

does not hold, then no index function can exist for that instance.

The maximal cover sum problem is then to find an index function that maximizes the expression

$$\sum_{c \in \mathscr{C}} \sum_{j=0}^{\omega(c)-1} P_{j+I(c)}$$

In other words, find a arrangement of the covers in $\mathscr{C}$ that covers the largest total of values on P.

We are interested in this problem since we can view matching the observed lengths of helices to the predicted secondary structure of a protein as a maximal cover sum problem. When examining possible arrangements of the helices onto the protein string, we would expect that the arrangements that cover the maximal value for the predicted probabilities of helices would be the most likely to occur in the actual protein. Thus we would be most interested in examining those arrangements first while trying to determine the 3D structure of the protein.

In considering how to obtain an optimal arrangement, we first notice that for any given ordering of the covers, we can compute an optimal covering using that ordering in polynomial time. To show this, we define π as an ordering of the cover set $\mathscr{C} = \{c_1, c_2, ..., c_n\}$, i.e. the value of $\pi(i)$ is the position of the element $c_i \in \mathscr{C}$ in the order π. The inverse function π^{-1} then takes a position i in the ordering and returns the cover in that position. Using a given order π, we can define the following recurrence equation that determines the size of an optimal covering using π .

$$m(a,b) = \begin{cases} 0 & \text{if } a \text{ or } b = 0, \\ \max\Big(m\big(a,b-1\big),\, m\big(a-1, b-\omega(\pi^{-1}(a))\big) + \\ \qquad \sum_{k=b-\omega(\pi^{-1}(a))+1}^{b} P_k\Big) \end{cases}$$

In this definition, the value $m(a,b)$ computes value of the maximal covering of the first a covers in the order π onto the first b positions of the string P. The recursion is based on determining whether or not an optimal covering covers the b position of P with $\pi^{-1}(a)$. By using dynamic programming, this recurrence and its associated index function can be computed in $O(|\mathscr{C}| \cdot |P|)$ time.

Thus the complexity in the maximal cover set problem stems from finding an order that produces an optimal covering. In general, finding an optimal ordering is superpolynomial in the number of covers unless $P = NP$. This is a consequence of the fact that we can reduce the set partition problem, which is NP-complete [3], to the maximal cover sum problem. To see this, consider an instance of a set partition problem with a multiset of values S. Using the multiset S, we can create an instance of the maximal cover problems with by making the set of covers C where $|C| = |S|$ and the length of the covers are the values in S. We then create a string P where

$$P = \left(\underbrace{1,1,1,...1,1}_{\ell}, 0, \underbrace{1,1,1,...1,1}_{\ell} \right)$$

and $\ell = \frac{\sum_{c \in C} \omega(c)}{2}$. The point of this construction is that the maximal cover sum is equal to $\sum_{c \in C} \omega(c) = 2\ell$ if and only if the multiset S can be equally partitioned.

Although the set partition problem is NP-complete, our reduction does not necessarily prove that the maximal cover sum problem is NP-hard. This is because the number of bits needed in our create string P can potentially be exponential to the number bits in S and so the size of the instance

Protein id	Number optimal solutions	Minimum Kendall-tau distance	Average Kendall-tau distance	Minimum Hamming distance	Average Hamming distance
1CC5	1	0.667	0.667	0.750	0.750
6TMN_E	2	0.238	0.310	0.286	0.286
3TIM_A	1	0.100	0.100	0.400	0.400
2TSC_A	5	0.095	0.305	0.429	0.600
1ECA	6	0.190	0.294	0.429	0.714
1GD1_O	2	0.067	0.100	0.333	0.500
1L58	8	0.393	0.429	0.625	0.688
2PHH	1	0.476	0.476	1.000	1.000

Table 1. The table shows the distances from the optimal covering orders to the actual arrangement of the alpha helices for the given proteins. All distances have been normalized so that a distance of zero is an identical ordering and a distance of one is maximally dissimilar.

for the maximal cover sum problem can be exponentially larger than that of the set partition problem. But this reduction does tell us two facts about the maximal cover sum problem. The first is that if there are hard instances of this problem where the maximum value in the cover set of these instances is bounded by a polynomial based on the size of the cover sets, then the problem itself would have to be *NP*-hard. But more importantly, even if this problem is in polynomial time, we would still not expect for there to be any algorithm to determine an optimal order that is polynomial in the size of the cover set. In other words, for some instances we probably cannot do much better than brute force checking of the permutations in order to find an optimal one.

3 Examining Optimal Solutions

Since it appears that finding an optimal solution, or especially the top k optimal solutions, to the maximal cover sum problem is computationally difficult, it is necessary to ask whether we need to find these solutions in order to perform the matching. In order to test this assumption, we computed the optimal solutions for a number of proteins and then compared those results with the actual ordering.

The list of proteins we used were originally selected in the He, Lu and Pontelli paper [4]. The covers were generated taking the lengths all alpha helices in the protein of length greater than 7. This was done to match the cover sets in the He, Lu and Pontelli experiments. The predictions of the secondary structure came from the PHD algorithm [6] in the PredictProtein server [7]. This server is able to give probabilities for each amino acid within the sequence, corresponding to the likelihood of its participation in an alpha helix. It assigns for each amino acid in the sequence a value ranging from 0 to 9. A prediction value of 0 means that it is highly unlikely for that amino acid to be a part of an alpha helix, while a value of 9 corresponds to a very high likelihood.

While there are a number of distance measures for list orders (for example, see chapter 6 of Diaconis [1] and Fagin, Kumar and Sivakumar [2]), we focused on two metrics. The first is the Hamming distance between the orders. This is a measure of the number of items that are in the same position between two lists. If π and σ are orderings of the cover set $\mathscr{C} = \{c_1, c_2, ..., c_n\}$, then the Hamming is defined as

$$d_{Ham}(\pi, \sigma) = n - |c_i \in \mathscr{C} : \pi(c_i) = \sigma(c_i)|$$

In order to compare the distance between orders of different lengths, we used a normalized version of the Hamming distance that we obtain by dividing the value d_{Ham} by n. This normalization maps all distances to interval $[0,1]$, where a distance of 0 represents identical orders and a distance of 1 are orders that are maximally disordered to each other. For the Hamming distance, maximally disordered lists share no common item for any position in the list.

A second metric that we will consider is the Kendall-tau distance. This distance is defined as

$$\begin{aligned} d_{K\tau}(\pi, \sigma) \quad = \quad & |(c_i, c_j) \in \mathscr{C}^2 : \pi(c_i) < \pi(c_j) \\ & \wedge\ \sigma(c_i) > \sigma(c_j)| \end{aligned}$$

The Kendall-tau distance is sometimes called the BubbleSort distance since it is equivalent to the number of flips needed in BubbleSort to transform one order to the second. Again, we normalized the Kendall-tau distance to the interval $[0,1]$ by dividing the value $d_{K\tau}$ by $\binom{n}{2}$. For the Kendall-tau distance, two maximally disordered lists are in reverse order of each other.

Our experiment involved computing all the optimal orders for each of the test proteins. In most instances, there

are multiple optimal solutions. We then computed the minimum and mean Kendall-tau and Hamming distances from the set of optimal solutions to the actual arrangement for the protein. These results are shown in Table 1.

One point to notice is that for all the given proteins, none of the optimal solutions were the actual ordering of the proteins. In fact, we can see examples, like 1L58, where a number of optimal solutions exist and also where all of these solutions are relatively distant from the actual order. These results call into question whether it is worth the computational expense to compute the optimal coverings of the predicted secondary structure.

4 A Greedy Heuristic for Maximal Cover Sum

Because of the uncertainty inherent in predicting secondary structure strictly from 1D structure, we should not be surprised that the optimal ordering is often a large distance from the actual order. This suggests that finding optimal orders may not be worth the computational expense, and that a simpler approximation method can be used. Towards this goal, we introduce a simple and fast heuristic (Algorithm 1) for producing a covering. The produced cover will not necessarily be optimal, but it will cover a large sum on the string.

The idea behind our algorithm is simple. For a set of covers, we take a cover c having the largest length and place it on the string P where it covers the greatest sum of values. We now create a new string P' which concatenates together the sections of P that are not covers by c. For the remaining covers $\mathscr{C} - \{c\}$, we obtain an index function I' for placing them on the string P' by calling our routine recursively. Now using the index function I', we can construct an index function I for the cover set $\mathscr{C}$ and P. If i is the starting index of c on P, then this construction breaks into three cases. The first is for every cover $c' \in \mathscr{C} - \{c\}$ that is completely placed on P' before i, in other words $I'(c) + \omega(c) \leq i$. For this case, we simply copy the index over, i.e. $I(c') = I'(c')$. The second case is when the cover is completely after i, $(I'(c') > i)$, then we can insert the cover in P by offsetting its index by $\omega(c)$ to make room for the cover c $(I(c') = I'(c') + \omega(c))$. The final case is when a cover overlaps the index i $(I'(c') < i$ and $I'(c') + \omega(c') > i)$. In this case, we notice that we can cover the sames indices in P by keeping c' at the same index and sliding c over by the length of c' (i.e. $I(c') = I'(c')$ and $I(c) = I'(c') + \omega(c'))$. An example of this algorithm is given in Figure 1.

The advantage of this heuristic is that it gives a fast and reasonable covering of the string. As Figure 1 demonstrates, this algorithm does not necessarily produce an optimal placement of the covers, but if there exists optimal solutions where the placement of covers are all separated on the string, this heuristic can often return one of those solutions. Since the recursion depth of the algorithm is the number of covers, and for each cover we need to check the string P for its optimal placement, we see that we can implement this algorithm to run in time $O(|\mathscr{C}| \cdot |P|)$.

Input: a set of covers $\mathscr{C}$ with an associated length function ω and a string P
Output: an array I that maps set of covers $\mathscr{C}$ to indices in P

```
Create empty index array I
if C ≠ ∅ then
    n ← |C|
    c ← a cover in C with maximal length
    i ← smallest index of a maximal cover of c on P
    I[n] ← i
    I' ← cov(C − {c}, P_{1,i} ⊕ P_{i+C_n,|P|})
    for j ← 1 to n − 1 do
        if I'[j] < i then
            I[j] ← I'[j]
            if I'[j] + ω(c) > i then
                c_j ← cover in position j
                I[n] ← I[n] + ω(c_j)
            end
        else
            I[j] ← I'[j] + ω(p)
        end
    end
end
return I
```

Algorithm 1: A heuristic algorithm *cov* for producing a covering of the string with a high sum. In this algorithm, the symbol $\oplus$ denotes string concatenation.

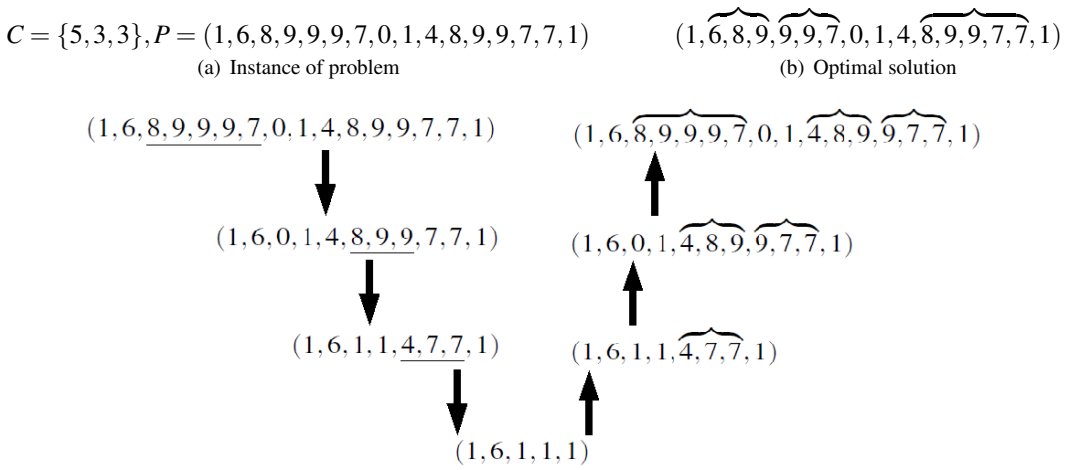
In Table 2, we compare the distance of the cover order produced by this simple heuristic to the actual ordering of the helices on the proteins. Surprisingly, it seems that this heuristic often produces a result that it closer to the protein helix arrangement than using the set of optimal solutions.

5 Randomized Orderings Based on Bubble-Search

Since the complexity in the maximal cover sum problem is in determining the correct order, our approach is, instead of trying to compute every possible placement, to use a randomized process to find the high value orderings and then use only the optimal placement of those orderings. Our method is based on the randomized BubbleSearch algorithm of Lesh and Mitzenmacher [5]. Their approach is a heuristic used to search for optimal orderings, especially in *NP*-hard applications. The basic algorithm starts with a simple approximation of the optimal order, called

Protein id	Val	p values 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1CC5	Min. Kτ	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.167
	Mean Kτ	0.379	0.332	0.338	0.334	0.307	0.300	0.261	0.243	0.209
	Std. Kτ	0.191	0.186	0.183	0.165	0.153	0.155	0.128	0.101	0.075
	Min. Ham.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.500
	Mean Ham.	0.640	0.645	0.636	0.675	0.660	0.659	0.650	0.649	0.601
	Std. Ham.	0.248	0.264	0.260	0.254	0.239	0.243	0.236	0.200	0.186
6TMN_E	Min. Kτ	0.000	0.143	0.095	0.143	0.143	0.143	0.143	0.190	0.286
	Mean Kτ	0.474	0.468	0.466	0.501	0.497	0.511	0.518	0.535	0.553
	Std. Kτ	0.152	0.139	0.138	0.147	0.145	0.127	0.114	0.100	0.074
	Min. Ham.	0.000	0.286	0.429	0.286	0.571	0.429	0.429	0.429	0.714
	Mean Ham.	0.824	0.826	0.843	0.856	0.857	0.852	0.858	0.897	0.949
	Std. Ham.	0.156	0.153	0.139	0.140	0.120	0.140	0.146	0.116	0.092
3TIM_A	Min. Kτ	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.100
	Mean Kτ	0.439	0.412	0.380	0.339	0.328	0.295	0.284	0.220	0.164
	Std. Kτ	0.206	0.180	0.186	0.172	0.163	0.174	0.161	0.144	0.100
	Min. Ham.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.400
	Mean Ham.	0.705	0.692	0.697	0.664	0.671	0.634	0.657	0.591	0.526
	Std. Ham.	0.230	0.212	0.224	0.215	0.188	0.212	0.206	0.199	0.176
2TSC_A	Min. Kτ	0.048	0.048	0.000	0.000	0.000	0.000	0.048	0.048	0.048
	Mean Kτ	0.348	0.325	0.307	0.260	0.252	0.228	0.215	0.174	0.142
	Std. Kτ	0.131	0.131	0.154	0.127	0.131	0.138	0.132	0.116	0.106
	Min. Ham.	0.286	0.286	0.000	0.000	0.000	0.000	0.286	0.286	0.286
	Mean Ham.	0.678	0.647	0.629	0.596	0.576	0.551	0.528	0.496	0.519
	Std. Ham.	0.178	0.157	0.186	0.185	0.185	0.196	0.160	0.150	0.124
1ECA	Min. Kτ	0.095	0.048	0.095	0.143	0.143	0.143	0.143	0.143	0.190
	Mean Kτ	0.436	0.442	0.435	0.424	0.416	0.419	0.418	0.400	0.391
	Std. Kτ	0.151	0.148	0.144	0.151	0.124	0.110	0.112	0.092	0.072
	Min. Ham.	0.286	0.286	0.429	0.429	0.286	0.286	0.286	0.286	0.571
	Mean Ham.	0.824	0.794	0.824	0.834	0.841	0.854	0.864	0.816	0.801
	Std. Ham.	0.160	0.172	0.148	0.146	0.153	0.138	0.141	0.145	0.132
1GD1_O	Min. Kτ	0.067	0.000	0.000	0.067	0.067	0.067	0.067	0.067	0.067
	Mean Kτ	0.438	0.423	0.361	0.365	0.338	0.294	0.250	0.238	0.186
	Std. Kτ	0.170	0.170	0.167	0.165	0.153	0.148	0.129	0.128	0.097
	Min. Ham.	0.333	0.000	0.000	0.333	0.333	0.333	0.333	0.333	0.333
	Mean Ham.	0.772	0.769	0.713	0.712	0.717	0.688	0.673	0.663	0.663
	Std. Ham.	0.191	0.174	0.189	0.175	0.167	0.164	0.140	0.136	0.093
1L58	Min. Kτ	0.071	0.036	0.071	0.071	0.036	0.036	0.071	0.107	0.143
	Mean Kτ	0.322	0.298	0.301	0.306	0.308	0.296	0.280	0.272	0.262
	Std. Kτ	0.113	0.117	0.104	0.092	0.102	0.089	0.083	0.063	0.044
	Min. Ham.	0.250	0.250	0.250	0.375	0.250	0.250	0.250	0.375	0.375
	Mean Ham.	0.676	0.674	0.672	0.686	0.666	0.664	0.649	0.651	0.676
	Std. Ham.	0.144	0.134	0.130	0.121	0.124	0.123	0.125	0.105	0.095
2PHH	Min. Kτ	0.095	0.048	0.048	0.000	0.048	0.000	0.048	0.048	0.048
	Mean Kτ	0.437	0.407	0.394	0.381	0.350	0.309	0.277	0.254	0.189
	Std. Kτ	0.159	0.153	0.156	0.153	0.146	0.158	0.134	0.134	0.095
	Min. Ham.	0.286	0.286	0.286	0.000	0.286	0.000	0.286	0.286	0.286
	Mean Ham.	0.759	0.739	0.745	0.725	0.715	0.692	0.633	0.628	0.625
	Std. Ham.	0.166	0.175	0.178	0.188	0.174	0.192	0.194	0.184	0.163

Table 3. The table gives the normalized distances from the randomized BubbleSearch orders to the actual arrangement of the alpha helices on the given proteins. For each protein and p-value, 200 random orders were produced, using the greedy heuristic as the base order. This table shows the minimum, mean, and standard deviation of the Kendall-tau and Hamming distances for these random orders.



(c) Example run of algorithm

Figure 1. This figure gives an example of how the heuristic algorithm constructs an ordering. Starting with the set C of cover lengths and a string P in 1(a), the algorithm (shown in 1(c)) greedily places the largest cover onto P and then removes that covered section from the string. When it has selected all cover lengths, it then inserts the covers while sliding over any covers that overlap. Figure 1(b) gives the optimal solution for this covering showing that the algorithm is a heuristic.

Protein id	Kendall-tau distance	Hamming distance
1CC5	0.167	0.500
6TMN_E	0.571	1.000
3TIM_A	0.100	0.400
2TSC_A	0.095	0.571
1ECA	0.381	0.714
1GD1_O	0.133	0.667
1L58	0.250	0.750
2PHH	0.143	0.714

Table 2. Distances to the actual ordering from the ordering produced by the greedy heuristic algorithm *cov*

the *base order*, which is usually derived by using some type of greedy method. Using the base order, a series of random orders are then created. Starting from first cover in the base order and then progressing through the order sequentially, each cover is added to a new order with some probability p. If the end of the order is reached without placing all the covers, then it starts from the first remaining one. If the base order is π and the new order is σ, then the probability of producing σ is proportional to $(1-p)^{d_{K\tau}(\pi,\sigma)}$. In other words, the closer the two orders are in Kendall-tau distance, the more likely it is to produce σ from π. The choice of p controls how large the radius of probable orders will be around the base order. Thus, as p approaches 1, then the Kendall-tau distance between the orders approaches 0.

For the observed helix matching problem, we conducted a series of computational experiments where we generated a base order using our heuristic and then generated 200 random orders each for a series of values for p. We then computed the Kendall-tau and Hamming distances between the randomly generated orders and the actual order of the protein. These results are shown in Table 3. We notice from these test results, that in every instance, the BubbleSearch method was able to find an ordering that was either the correct ordering or very close.

Since the value of p tends to be domain specific, it is not specified for the randomized BubbleSearch algorithm. Instead, the user must tune that parameter for their individual problems. In examining the results of our experiments, it seems that $p = 0.6$ provides a good initial value for examining protein orders. This value seems to be a compromise between keeping most orders reasonable close to the base ordering, but still allowing the structure to be modified into a fundamentally different orderings to avoid being trapped in a local minima. Using this idea, we purpose the following work flow to produce k potential matchings when investigating how to relate the observed helix lengths to the 3D structure.

1. Generate a base order using our greedy heuristic.
2. Run the randomized BubbleSearch algorithm with $p =$

0.6 for $c \cdot k$ times for some $c \geq 1$.

3. For each order generated, find its optimal placement and compute the sum covered, saving the k top valued placements.

In the Lesh and Mitzenmicher paper, a second form of randomized BubbleSearch is suggested where the base ordering is modified whenever an ordering is found that produces an better value. Since we are interested in generating a list of values and not just an optimal value, we have not implemented this extension in our tests. It is open whether this extension can have produce better results for our problem. Also, we point out that this method can be trivially parallelized if needed.

6 Conclusions

In this paper, we examined the problem of matching observed lengths of alpha helices to their proper location on a protein's amino acid sequence. This is a first step towards determining the 3D structure of the protein. Our first conclusion is that finding an optimal solution does not seem to be worth the computational effort. In particular, we showed evidence that, because of the uncertainty of the helix prediction, the optimal coverings can be relatively distance from the actual ordering on the protein.

Instead, we introduced a simple greedy heuristic for estimating the order. Using this heuristic as a starting point, we choose random order around it using the BubbleSearch method of Lesh and Mitzenmacher. When compared to the actual orderings, this method always found the correct ordering or produced an ordering that was very close. Thus, we believe that our method is a fast and efficient algorithm for determining a set of potential placements of helix lengths onto a protein sequence.

Acknowledgments

This research was performed while the second author was at NIST supported by the Summer Undergraduate Research Fellowship (SURF) program.

References

[1] P. Diaconis. *Group Representations in Probability and Statistics*, volume 11 of *Lecture Notes - Monograph Series*. Institute of Mathematical Statistics, 1988.

[2] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM Journal on Discrete Mathematics*, 17(1):134–160, 2003.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[4] J. He, Y. Lu, and E. Pontelli. A parallel algorithm for helix mapping between 3d and 1d protein structure using the length constraints. In *ISPA '04: Proceedings of Second International Symposium on Parallel and Distibuted Processing and Applications*, pages 746–756, 2004.

[5] N. Lesh and M. Mitzenmacher. Bubblesearch: A simple heuristic for improving priority-based greedy algorithms. *Information Processing Letters*, 97(4):161–169, 2006.

[6] B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology*, 232(2):584–599, 1993.

[7] B. Rost, G. Yachdav, and J. Liu. The PredictProtein server. *Nucleic Acids Research*, 32:W321–326, 2004.