Electron cryo-microscopy is a fast advancing biophysical technique to derive three-dimensional structures of large protein complexes. Using this technique, many density maps have been generated at intermediate resolution such as 6–10 Å resolution. Although it is challenging to derive the backbone of the protein directly from such density maps, secondary structure elements such as helices and β-sheets can be computationally detected. Our work in this paper provides an approach to enumerate the top-ranked possible topologies instead of enumerating the entire population of the topologies. This approach is particularly practical for large proteins. We developed a directed weighted graph, the topology graph, to represent the secondary structure assignment problem. We prove that the problem of finding the valid topology with the minimum cost is NP hard. We developed an $O(N^2 2^N)$ dynamic programming algorithm to identify the topology with the minimum cost. The test of 15 proteins suggests that our dynamic programming approach is feasible to work with proteins of much larger size than we could before. The largest protein in the test contains 18 helical sticks detected from the density map out of 33 helices in the protein.

Keywords: Constraint graph; topology; secondary structure; shortest path; electron cryo-microscopy.

1. Introduction

Electron cryo-microscopy (cryoEM) is a biophysical technique that has great potential in deriving the three-dimensional structure of large protein complexes. Over the last decade, cryoEM density maps of large protein complexes have been generated to intermediate resolution, such as 6–10 Å, using this technique. At this resolution range, the density is not well resolved, and it is challenging to determine
the atomic structure directly from the density map. Three major approaches have been used to predict the near-atomic structures from the density map. When a near-atomic structure is available for the protein or part of the protein, fitting techniques are frequently applied.\textsuperscript{5–10} A second approach uses the comparative modeling method to build the structures and fit iteratively to the density map.\textsuperscript{11–13} The limitation of the comparative modeling approach is the requirement of a template that share certain level of structural similarity with the target protein. In general, good templates may not be available for many proteins. When the protein is small, it is possible to use \textit{ab initio} prediction method, such as Rosetta, to generate possible structures and then evaluate them using the density map.\textsuperscript{14–16} However, it is quite challenging to extend this method to larger proteins that are often seen in the cryoEM maps. \textit{De novo} folding does not require the templates from known structures. It builds the possible structures directly from the density features that can be identified from the density map.\textsuperscript{17–20} There have been more and more successful cases in the \textit{de novo} folding when the resolution of the density map is better than 6 Å.\textsuperscript{21,22} In this approach, a number of computational steps are involved, such as image segmentation, secondary structure detection, topology modeling, assignment of the backbone and refinement.\textsuperscript{23}

Given a density map at the intermediate resolution, the location of helices can be identified using either the computational tools\textsuperscript{24–26} or through visual examination by the user. Figure 1 shows an example of a density map and the computationally detected helical sticks using Helix Tracer.\textsuperscript{25} In this case, Helix Tracer was able to detect five sticks that represent the electron density of five helices. The shortest helix which has five amino acids was not detectable by Helix Tracer. Each stick can be represented by the coordinates of the central axis of the helix. However, it is not known which segment of the protein sequence corresponds to which stick. Although the short helices may be hard to identify, helices longer than two turns can often be detected.\textsuperscript{25,26} In addition to helices, \(\beta\)-sheets can be roughly located computationally.\textsuperscript{26–28} In general, \(\beta\)-sheets are hard to be identified as accurately as the helices due to their non-uniform shapes. In certain cases, the \(\beta\)-strands can also be computationally detected.\textsuperscript{28} In spite of the multiple computational methods, errors are frequently seen in the detection due to the quality of the density map. For example, the length of helices may not be accurate. False helices are predicted and true helices are missed. A challenge to the topology modeling is how to develop a robust method so that it is not sensitive to such errors in the data.

Many methods have been developed for the secondary structure prediction from the amino acid sequence of a protein.\textsuperscript{29–31} In general, the prediction accuracy of these methods is about 80%.\textsuperscript{32} Although most of the secondary structure elements (SSEs) can be predicted, the inaccuracy exists regarding the length and the positioning of the predicted SSE. It has been observed that the accuracy of the predicted SSE from the sequence plays an important factor in the topology determination of \textit{de novo} folding of the density map. Wrongly predicted SSE can result in wrong topologies.\textsuperscript{15,17}
Fig. 1. Helical sticks and the topologies. (a) The density map (gray) was simulated to 10 Å resolution using protein 1B5L from the Protein Data Bank (PDB). The helical sticks (red S1 to S5) were detected using Helix Tracer and viewed by Chimera. (b) The helix segments of the protein sequence are marked as H1 to H6. The amino acids on the short loops are indicated and those on the long loops are indicated using “...”. Two alternative topologies are shown as diagrams in (c), the correct and in (d) the wrong topology. The N to C direction of the protein sequence (arrow) and for the sticks (cross and dot) is labeled. The true assignment is labeled by the stick with H1 being the first helix segment on the protein sequence.

The topology determination in this paper is to identify the order and the direction of the secondary structure elements (e.g. sticks in Fig. 1) detected from the density map (SSE-D). The computational tools, such as Helix Tracer and SSE hunter, provide the location of the SSEs in the density map, but they do not provide the topology information about the SSEs. For example, the true topology is the one in which the protein sequence starts from stick S3 then goes to S4 [Fig. 1(c) and 1(a)]. However, the order of the sticks is often ambiguous in the density map and has to be determined. The SSEs predicted from the amino acids sequence (SSE-S) provide an estimated location of the SSEs on the protein sequence. Therefore, the topology determination problem is also an assignment problem between the SSE-D and the SSE-S. In the ideal situation when there are N helices and M β-strands in the SSE-D and the SSE-S respectively, the total number of possible topologies is \( N! \times 2^N \times M! \times 2^M \). This number is based on the fact that there are N! different orders for the N helices and two directions to assign for each helix. Due to the large topological space, topology modeling often involves two steps: the generation of a subset of the topologies that are more likely to include the true topology and the evaluation of each such topology by building the corresponding structures. The second
step is computationally intensive and often involves the evaluation of the energy of the resulting structure.\textsuperscript{17,20,33}

This paper focuses on the steps to reduce the large topological space quickly to a small subset of possible topologies without the use of energy evaluation. The goal is to include the true topology in such a subset, so that the conformations can be built for the likely topologies. One approach is to generate all the topologies in the entire solution space and then eliminate the ones that are impossible or less likely to be true. For example, we enumerated the entire \( N!2^N \) topologies for helices up to \( N = 7 \) in our earlier approach.\textsuperscript{20,33} Wu \textit{et al.} enumerated all the topologies and then used geometrical screening to eliminate the less likely ones.\textsuperscript{34} Another approach is to translate the problem into a graph matching problem aiming to find the optimal match of the two attributed related graphs.\textsuperscript{35} One graph was created from the SSEs of the amino acid sequence. The other graph was created from the density map. Each of them describes the connection relationship among the SSEs. However, this method requires that the true link between the SSE-D to be detected in the density map. In reality the true link may be missed due to the quality of the density map. Lindert \textit{et al.} used Monte Carlo method to generate the topologies in which the helical sticks were assembled in different orders and directions.\textsuperscript{17} This approach allows the sampling of the large topological space, particularly with the consideration of the errors in the data. However, the random nature of the Monte Carlo method does not guarantee to find all the top ranked topologies.

In this paper, we propose a general framework to the topology determination problem using a weighted directed graph. The topology determination problem is then represented as a constrained graph problem. We will illustrate our dynamic programming algorithm to find the topology with the minimum cost. This algorithm is, as expected, significantly faster than the naïve method and the depth-first method. It allows us to find the topology with the minimum cost for large proteins. We also developed a method to enumerate the topologies with the top-K cost using the constraint graph. This is the first dynamic programming approach to rank the topologies of the SSE. The dynamic programming method is in \( O(N^22^N) \) run time.

2. Methods

2.1. Topology and the topology graph

Let \( (H_1, H_2, \ldots, H_M) \) be a sequence of the SSE located on the protein sequence. Due to the linear nature of the protein sequence, the order of the sequence segments \( H_i, i = 1, \ldots, M \) is fixed. Let \( \{S_1, S_2, \ldots, S_N\} \) be the set of sticks detected from the density map. The sticks here refer to the helices detected from the density map. In principle, the number of the sequence segments can be different from the number of sticks due to the errors detecting the sticks and the errors estimating the sequence segments. For simplicity, we assume \( M = N \) in the description of our method. The method can be extended in the case of \( M \neq N \), and we did that to obtain the results in this paper. The topology determination problem can be described
as a problem to find a permutation \( \gamma \) of \( \{1, 2, \ldots, N\} \) such that assigning \( H_i \) to \( S_{\gamma(i)}, i = 1, \ldots, N \) minimize the assignment score. In the assignment, each \( H_i \) is assigned to \( S_{\gamma(i)} \) in one of the two opposite directions.

It has been observed that various constraints related to two adjacent SSEs can be used to reduce the number of possible topologies significantly.\(^{17,20,34,35}\) One such constraint comes from the length of the loop that connects the two SSE-S. This constraint requires that the loop to be long enough to make the connection between the end of the stick and the end of the other stick to be connected by the loop. We can estimate the length of the loop by the number of amino acids between two adjacent SSEs on the sequence. For example, the length of the loop between \( H_1 \) and \( H_4 \) is five amino acids (Fig. 1). Given that there is about 3.8 Å distance between two consecutive amino acids, the maximum distance between the two ends of the loop is 3.8 × 5 = 19 Å. This requires that the end-to-end distance between the sticks to be at most 19 Å. Other constraints such as the geometrical constraints and the connectivity constraints have also been used.\(^4\) The nature of such constraints is that it involves the assignment of two SSEs. Such constraints involving two SSEs can be naturally represented by an edge in a graph. In this paper, we do not focus on the problem how the weights are derived. We assume that the weights can be established from the constraints using one of the existing methods.\(^{17,20,34,35}\)

We use a weighted directed graph \( G = (V, E, w) \) to represent the topology problem; \( V \) has two special nodes \( \text{START} \) and \( \text{END} \) and \( N \times N \times 2 \) “regular” nodes. More precisely,

\[
V = \{(i, j, t)|1 \leq i, j \leq N, t \in \{0, 1\}\} \cup \{\text{START}, \text{END}\}
\]

\[
E = \{(i, j, t), (i + 1, j', t')|1 \leq i \leq N - 1, 1 \leq j, j' \leq N, t, t' \in \{0, 1\}\} \cup
\]

\[
\{(\text{START}, (1, j, t)|1 \leq j \leq N, t \in \{0, 1\}\} \cup
\]

\[
\{(N, j, t), \text{END})|1 \leq j \leq N, t \in \{0, 1\}\}
\]

A node \((i, j, t)\) represents an assignment of \( H_i \) to \( S_j \) in \( t \) direction (Fig. 2). An edge from node \((i, j, t)\) to \((i + 1, j', t')\) represents the assignment of \( H_{i+1} \) to \( S_{j'} \) in direction \( t' \) after the assignment of \( H_i \) to \( S_j \) in direction \( t \). If the two assignments involved in an edge do not satisfy the constraints described in the previous paragraph, the weight of the edge was assigned as \( \infty \), indicating that the edge is not feasible. Otherwise, the weight was given as the cost/penalty of the two assignments. In this paper, our implementation simply assigned the weight for the feasible path as \( w((i, j, t), (i + 1, j', t')) = |\text{LoopLength}_i + 4| \times 3.8 - \text{Dist}((i, j, t), (i + 1, j', t'))| \). \( \text{LoopLength}_i \) is the number of amino acids between \( H_i \) and \( H_{i+1} \) on the protein sequence. We added 4 to simulate the inaccurate positioning of the \( H_i \) on the sequence if it is predicted from a secondary structure prediction method. \( \text{Dist}((i, j, t), (i + 1, j', t')) \) is the distance measured in the three-dimensional space between the two ending points of the two sticks. In particular, the distance was measured from the ending point of the central axis of \( S_j \) in \( t \) direction to the
Fig. 2. An example of the topology graph. (a) The graph built for the N-terminal fragment of NS1 protein from Influenza A virus (PDB ID 1AIL). The weights were restricted to integers to save the space in drawing. (b) The example of $U$ and $f(v, U)$ for the node $(2,2,0)$, $(2,2,1)$, $(3,3,0)$ and $(3,3,1)$. The shortest path is shown in green thick lines and an example of an invalid path is shown in red dashed lines.
beginning point of the central axis of $S_j$ in direction $t'$. The central axis of a helix was detected by Helix Tracer. Ideally, the distance measured in three-dimensional space should be close to the length of the loop. The weight is the difference between the loop length estimated from the linear sequence and that estimated in the three-dimensional space.

Figure 2 illustrates the topology graph. A node $v = (i, j, t)$ is located on the $i$th row, $j$th column and $t$th position inside the box drawn in a dashed line [Fig. 2(a)]. The graph is a directed graph in which each edge points downwards. An edge represents a valid relationship between two adjacent SSE on the sequence. Since each stick in the density map can only be assigned to one SSE on the sequence, there is not an edge between the nodes in the same column, and similarly there is not an edge between the nodes in the same row. When $M = N$ each edge links between two adjacent rows [Fig. 2(a)], although it can link between non-consecutive rows when $M \neq N$ which is not discussed in the method of this paper. Special edges are drawn from node START to each node in the first row and similarly from each node at the last row to node END. The weight on the special edges is zero, and the other weights are non-negative.

2.2. Constraints in finding valid paths

A path of $G$ begins at node START and ends at node END. The problem of enumerating all valid topologies becomes the problem of enumerating all valid paths. Not every path is a valid path. For example, those paths that visited the same column more than once are not valid paths, since each stick cannot be assigned to multiple SSEs on the sequence. A valid path needs to satisfy the following constraints:

1. The path begins at START and stops at END.
2. A valid path visits exactly one node in each row.
3. A valid path visits exactly one node in each column.

More precisely, a valid path is a sequence of nodes $(\langle \text{START} \rangle, \langle 1, j_1, t_1 \rangle, \langle 2, j_2, t_2 \rangle, \ldots, \langle N, j_N, t_N \rangle, \langle \text{END} \rangle)$ where $\{j_1, j_2, \ldots, j_N\} = \{1, 2, \ldots, N\}$ and $\{t_1, t_2, \ldots, t_n\} = \{0, 1\}$. An example of a valid path is shown in green thick lines and a non-valid path is shown in red dashed lines [Fig. 2(a)]. With the formulation of the topology graph and the valid path, a valid topology corresponds to a valid path from START to END. The optimal path is the path that has the minimum cost. In this paper, the cost of a path is simply the sum of the weights along the path.

2.3. The complexity of the problem

The solution space for the problem of assigning $N$ sticks to $N$ sequence segments is $N!2^N$. It is often desired to reduce the solution space to a set of small number of highly ranked possible topologies, and the correct topology is contained in
the set. We investigated the complexity of the reduction problem that involves the constraint from a pair of nodes. We will show that finding the set of the top-K ranked topologies is NP hard by showing that finding the top-ranked topology in the topology graph is already NP hard. The dynamic programming method (Sec. 2.4) will improve the shortest path search from $N!2^N$ time, as in the naïve search, to $O(N^22^N)$ time.

Note that the problem of finding the shortest valid path in the topology graph is equivalent to finding the shortest valid path from a node in row 1 to a node in row $N$. We know that the following variant of travelling salesman problem is NP hard: Given a complete weighted graph $G = (V,E,c)$ find the minimum cost Hamiltonian path in $G$. We call this variant as MCHP. We provide a polynomial-time reduction from MCHP to our problem. Intuitively, this reduction is possible because the constraint of not revisiting a column in the shortest path in $G_{TOP}$ is similar to the constraint of not revisiting a vertex in a Hamiltonian path.

Claim. The problem of finding the shortest valid path from a node in row 1 to a node in row $N$ in the topology graph $G_{TOP}$ is NP hard.

Proof. Consider an instance of MCHP, $G_{MCHP} = (V,E,c)$ of $N$ nodes, with $c(i,j)$ being the cost of travelling from node $i$ to node $j$. We construct an instance of $G_{TOP} = (V',E',w)$ as follows.

$$V' = \{(i,j,t) : i \in V \text{ and } 1 \leq j \leq N, 0 \leq t \leq 1\},$$

$$E' = \{(k,i,t),(k+1,j,t') : (i,j) \in E \text{ and } 1 \leq i,j \leq N, 1 \leq k \leq N-1, 0 \leq t,t' \leq 1\},$$

$$w((k,i,t),(k+1,j,t')) = w((k,j,t),(k+1,i,t')) = c(i,j) \text{ for } 1 \leq i,j \leq N, 1 \leq k \leq N-1, 0 \leq t,t' \leq 1.$$

Intuitively, for each vertex $i$ of MCHP, we create a row of nodes in $G_{TOP}$. For each edge $(i,j)$ of MCHP, we create the links, using the weight $c(i,j)$, between the nodes at the $i$th column and the $j$th column in two consecutive rows. Now consider the path $(1,j_1,t_1),(2,j_2,t_2)\cdots(N,j_N,t_N)$ in $G_{TOP} = (V',E',w)$ and the path $j_1,j_2,\ldots,j_N$ in $G_{MCHP}$. Noticing that both paths have the same cost, if $(1,j_1,t_1),(2,j_2,t_2)\cdots(N,j_N,t_N)$ is the shortest valid path in $G_{TOP}$, then $j_1,j_2,\ldots,j_N$ is a minimum-cost Hamiltonian path in $G_{MCHP}$.

2.4. Finding the shortest path satisfying the constraints

The valid paths represent the valid topologies, and ideally, the valid path with the minimum cost represents the true topology of the protein. Finding the shortest path in a weighted graph is a classical problem and many methods are available. One of them is the Dijkstra algorithm, in which it finds the shortest path between a single source and destination when the weights are non-negative. Bellman–Ford
algorithm finds the shortest path when the weights are allowed to be negative.\textsuperscript{39,40} However, these generic shortest path algorithms cannot be applied directly in this problem due to the constraints that need to be satisfied by a valid path. In this section, we give a dynamic programming algorithm to find the shortest valid path.

To find the shortest valid path, our method keeps track of, at each node, the columns visited along the path. At each node $v = (i,j,t)$, let $U \subseteq \{1,2,\ldots,N\}, |U| = i,j \in U$ with $U$ representing the set of columns visited in a path. For example, there are maximum $3! = 6$ different paths that visit the three columns when $U = \{1,3,5\}$, regardless of the order of the visits. Let $f(v,U)$ be the minimum cost of the path for reaching $v$ by using the elements of $U$ as columns.

$$f(v,U) = f((i,j,t),U)$$

$$= \begin{cases} 
0 & v = (\text{START}) \\
 w((\text{START},v) & i = 1,j \in U \\
 \min_{j' \in U \setminus \{j\}, t' \in \{0,1\}} [f((i-1,j',t'),U \setminus \{j\}) + w(((i-1,j',t'),(i,j,t)))] & i \in [2,N], j \in U \\
\infty & \text{otherwise}
\end{cases}$$

Figure 2(b) shows an example of the $f(v,U)$. At node $(2,2,0)$, there are two instances of $U$ with $U_1 = \{1,2\}$ and $U_2 = \{3,2\}$. The minimum cost of reaching $(2,2,0)$ using the column 3 and column 2 is 4. The pseudo code for finding the shortest valid path is given in Fig. 3. If we assume that the access of any entry in the table $f(v,U)$ is in constant time, the time to find the shortest valid path is $O(N^22^N)$. The dynamic programming approach reduced the $N!$ component in the naïve approach ($N!2^N$) to $N^2$, although the nature of the problem is still NP hard.

### 2.5. Top-K shortest paths satisfying the constraints

The K shortest path is a generalization of the shortest path problem where not only the shortest but the first K paths $(p_1,p_2,\ldots,p_k), K \geq 1$ is determined in non-decreasing order of the cost. The top-K shortest path is more practical than the shortest path since the shortest path may not be the true topology, although the true topology often has near minimum cost. One of the most popular deviation algorithms was initially proposed by Yen,\textsuperscript{41} and later improved by Lawler.\textsuperscript{42} Our implementation of top-K method is based on the MPS algorithm,\textsuperscript{43} an alternative with improved expected running time. We replaced the generic shortest path algorithm with our constrained shortest path algorithm that was described in the previous section.

### 3. Results

To investigate the performance of the dynamic programming approach, we first tested if the shortest valid path identified by our method is indeed the valid path
Fig. 3. Pseudo code for finding the shortest valid path in the topology graph.

Notation:
- For a given set \( U = \{1, 2, 3, ..., N\} \), we define \( U_k^{(i)} \) as the \( k^{th} \) subset of \( U \) of size \( i \) for \( 1 \leq i \leq N, 1 \leq k \leq \binom{N}{i} \).
- \( v_{(j,t)}^{(i)} \) is the \( j^{th} \) column with \( t \) direction in the \( i^{th} \) row.
- \( f((i, j, t), U_k^{(i)}) \rightarrow f(v_{(j,t)}^{(i)}, U_k^{(i)}) \).

Algorithm 1:

Input: A \( AN(N-1)^2 \) weight array \( w \) (no edge \( \Rightarrow \infty \) weight)
Output: Minimum path cost \( \min_{\text{cost}} \)
\[ U \leftarrow \{1, 2, 3, ..., N\} \]
\[ f(\cdot, U_k^{(i)}) \leftarrow 0, \quad 1 \leq k \leq N \]
\[ f(\cdot, U_k^{(i)}) \leftarrow \infty, \quad 2 \leq i \leq N, 1 \leq k \leq \binom{N}{i} \]
for \( i \leftarrow 2 \) to \( N \) do
  for \( k \leftarrow 1 \) to \( \binom{N}{i} \) do
    for each \( p \in U_k^{(i)} \), \( t \leftarrow 0 \) to \( 1 \) do
      \[ U' \leftarrow U_k^{(i)} \backslash p \]
      for each \( q \in U' \), \( t' \leftarrow 0 \) to \( 1 \) do
        \[ f(v_{(p,t)}^{(i)}, U_k^{(i)}) = \min[f(v_{(q,t')}^{(i)}, U') + w(v_{(q,t')}^{(i)}, v_{(p,t)}^{(i)}), f(v_{(p,t)}^{(i)}, U_k^{(i)})] \]
      endfor
    endfor
  endfor
  \[ \min_{\text{cost}} = \min\{f(v_{(j,t)}^{(i)}, U) : 1 \leq j \leq N, 0 \leq t \leq 1\} \]

with the minimum cost. We sorted the cost of all the valid paths and found that the shortest path identified by the dynamic programming approach is indeed the top 1 ranked (column 5, Table 1) among all the valid paths. Note that the shortest valid path may not be the path of the true topology, due to the potential error in the weight. However, we notice that the valid path of the true topology often has near minimum cost (to be discussed with Table 3). We then compared our method with two other approaches: the naïve and the depth-first search. In the naïve approach, each of the entire \( N!2^N \) topologies were evaluated to search for the one with the minimum cost. As expected, the naïve method and the depth-first search methods took significantly longer time than the dynamic programming approach in the large proteins (Table 1). For example, the time to find the shortest valid path is 1410.49s (row 8 of Table 1) for the naïve approach, 8.753s for the depth-first, and 0.014s for our dynamic programming approach. In this case, the protein has nine actual helices on the sequence and seven helical sticks detected using Helix Tracer.
Table 1. The shortest valid path.

<table>
<thead>
<tr>
<th>No.</th>
<th>Protein ID</th>
<th>#True helices</th>
<th>#Sticks</th>
<th>Rank shortest</th>
<th>Dynamic</th>
<th>Depth first</th>
<th>Naive</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1SU0</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0.002</td>
<td>0.014</td>
<td>0.125</td>
</tr>
<tr>
<td>2</td>
<td>1BO9</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>0.001</td>
<td>0.013</td>
<td>0.498</td>
</tr>
<tr>
<td>3</td>
<td>1JW2</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>0.001</td>
<td>0.008</td>
<td>0.209</td>
</tr>
<tr>
<td>4</td>
<td>1ATD</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>0.002</td>
<td>0.018</td>
<td>0.944</td>
</tr>
<tr>
<td>5</td>
<td>1AA2</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>0.009</td>
<td>0.068</td>
<td>1.268</td>
</tr>
<tr>
<td>6</td>
<td>1DUS</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>0.004</td>
<td>0.186</td>
<td>8.038</td>
</tr>
<tr>
<td>7</td>
<td>1FLP</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>0.010</td>
<td>1.224</td>
<td>9.452</td>
</tr>
<tr>
<td>8</td>
<td>1NG6</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>0.014</td>
<td>8.753</td>
<td>1410.49</td>
</tr>
</tbody>
</table>

*a* the number of helices in the native protein.

*b* the number of helices detected by HelixTracer.

*c* the rank of the shortest valid path using the dynamic programming approach.

*d* the time (in seconds) to find the shortest valid path using the dynamic programming algorithm. It includes the time to build all subsets at each node.

*e* the time (in seconds) to find the shortest valid path using the depth first method.

*f* the time (in seconds) to find the shortest valid path using the naive method.

This protein, our dynamic programming approach is 100,000 faster than the naive method. It is expected that the difference in performance is even more for larger proteins. The time in Table 1 includes the time to build the graph and the search for the shortest path. All the tests in this paper were run on a generic PC — Dell Optiplex 980 machine at 2.8 GHz and 8 GB of memory.

Table 2 shows the performance and the memory usage for large proteins. We were not able to work with proteins with more than seven helices in our earlier work due to the large number of topologies to be evaluated. Now we are able to work with proteins with 33 actual helices on the protein sequence and 18 detected sticks (Table 2, row 15). For this protein, it took 34610.7 s to build the graph, 1.57 s to find the shortest valid path and it used 1.11 GB memory. We also listed the time it takes to get the top 100 shortest paths (Table 2, column 7). Notice that the search time is generally much shorter than the time to build the graph. However, the graph is only needed to build once for the search of top-K paths. This makes our approach particularly effective to obtain the top-ranked valid topologies. Although most proteins do not have as many as 33 helices, the total number of helices and β-strands can be over 20 in a medium-sized protein.

Since we used a simple criterion to assign the weight for an edge, error is expected in the weights. We wanted to see if the true topology is near the top of the solution space using the current weighting strategy. We applied our top-K deviation algorithm to identify the top-K shortest path and see where the true topology is ranked. We tested six proteins with less than 8 sticks detected in the density map. The rank of the true topology is between 1 and 97 for these proteins. For the largest protein (1NG6, row 6, Table 3), the true topology is ranked the 97th out of \( \binom{7}{2} \approx 23 \) million possible topologies in the entire solution space. This suggests that the simple weight could be fairly effective in eliminating most of the possible
Table 2. Run time and memory usage.

<table>
<thead>
<tr>
<th>No.</th>
<th>Protein ID</th>
<th>#Helices</th>
<th>#Sticks</th>
<th>BUILD time</th>
<th>1st time</th>
<th>Top 100</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1B5L</td>
<td>6</td>
<td>5</td>
<td>0.008</td>
<td>0.000</td>
<td>0.005</td>
<td>0.16</td>
</tr>
<tr>
<td>2</td>
<td>1FLP</td>
<td>7</td>
<td>6</td>
<td>0.010</td>
<td>0.000</td>
<td>0.007</td>
<td>0.18</td>
</tr>
<tr>
<td>3</td>
<td>1NG6</td>
<td>9</td>
<td>7</td>
<td>0.014</td>
<td>0.000</td>
<td>0.011</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>1ZA0</td>
<td>13</td>
<td>8</td>
<td>0.291</td>
<td>0.001</td>
<td>0.039</td>
<td>0.72</td>
</tr>
<tr>
<td>5</td>
<td>2H7O</td>
<td>14</td>
<td>9</td>
<td>0.270</td>
<td>0.003</td>
<td>0.075</td>
<td>0.83</td>
</tr>
<tr>
<td>6</td>
<td>3ACW</td>
<td>17</td>
<td>10</td>
<td>5.200</td>
<td>0.001</td>
<td>0.481</td>
<td>2.68</td>
</tr>
<tr>
<td>7</td>
<td>3L9T</td>
<td>14</td>
<td>11</td>
<td>3.000</td>
<td>0.000</td>
<td>0.337</td>
<td>2.44</td>
</tr>
<tr>
<td>8</td>
<td>2XB5</td>
<td>13</td>
<td>8</td>
<td>0.276</td>
<td>0.000</td>
<td>0.065</td>
<td>0.64</td>
</tr>
<tr>
<td>9</td>
<td>3ODS</td>
<td>21</td>
<td>12</td>
<td>17.80</td>
<td>0.003</td>
<td>1.000</td>
<td>7.44</td>
</tr>
<tr>
<td>10</td>
<td>1A4S</td>
<td>20</td>
<td>12</td>
<td>56.00</td>
<td>0.003</td>
<td>3.200</td>
<td>8.61</td>
</tr>
<tr>
<td>11</td>
<td>2PFT</td>
<td>27</td>
<td>14</td>
<td>242.9</td>
<td>0.020</td>
<td>14.20</td>
<td>39.99</td>
</tr>
<tr>
<td>12</td>
<td>2X79</td>
<td>24</td>
<td>17</td>
<td>1678.4</td>
<td>0.024</td>
<td>120.4</td>
<td>211.1</td>
</tr>
<tr>
<td>13</td>
<td>2OEV</td>
<td>26</td>
<td>18</td>
<td>5650.2</td>
<td>0.028</td>
<td>322.5</td>
<td>531.0</td>
</tr>
<tr>
<td>14</td>
<td>2XVY</td>
<td>33</td>
<td>17</td>
<td>9270.4</td>
<td>0.030</td>
<td>60.1</td>
<td>515.64</td>
</tr>
<tr>
<td>15</td>
<td>2XSI</td>
<td>33</td>
<td>18</td>
<td>34610.7</td>
<td>1.570</td>
<td>1570.6</td>
<td>1110.26</td>
</tr>
</tbody>
</table>

a the number of helices in the native protein.
b the number of helices detected by HelixTracer.
c the time (in seconds) to build all subsets in the graph.
d the time (in seconds) to find the shortest valid path.
e the time (in seconds) to find the shortest 100 paths.
f the memory (in MB) to store all subsets and paths.

topologies for these proteins. Note that the weight in our method employs minimal constraints. In fact, it does not involve sophisticated analysis of the density and only reflects the fact that the end-to-end distance of the sticks is comparable to the length of the loop connecting them. It is expected that more accurate weights of the edge can improve the ranking of the true topology even more.

We developed a layered graph to represent the secondary structure assignment problem. The current weight mainly represents the difference between the estimated length of the loop connecting the two sticks and the estimated distance between them in three-dimensional space. In order for the true topology to be ranked near the top of the list, the predicted helices based on the protein sequence have to be accurate enough. Two situations will affect the ranking the most. One is when
a long helix is wrongly predicted as two short helices that are connected by a short loop. The other is when two short helices are predicted as one long helix. In order to identify the true topology, the search needs to consider the possible errors from both the secondary structure prediction and those from the detection in three-dimensional space.

The current implementation is limited in the ranking of the helices, although our dynamic programming approach is general for either helices or \( \beta \)-strands. In order to extend the current approach to \( \beta \)-strands, the location of the \( \beta \)-strands needs to be estimated. The \( \beta \)-sheets are generally not as accurately detected as the helices in the intermediate resolution density maps. It is expected that the \( \beta \)-strands will be estimated with many possible alternatives. It is still a challenging problem to rank the topology with both \( \alpha \)-helices and \( \beta \)-sheets without the knowledge of a template.

4. Conclusion

The topology determination for the secondary structure elements detected from the density map is a critical question in deriving the backbone from the cryoEM map at the intermediate resolution. The major challenge in this problem is the large solution space due to the combinatorial nature of the problem. Our work in this paper provided an approach to enumerate the top-ranked possible topologies instead of enumerating the entire population of the topologies. We formulated the problem of secondary structure assignment into a constraint graph and proved that it is an NP-hard problem. We gave an \( O(N^2 2^N) \) dynamic programming algorithm to find the valid topology with the minimum cost. We also showed the concept of finding the top-K ranked valid topologies. The tests suggest that our dynamic programming method is feasible for the proteins of much larger size than we could handle before. They also suggest that it is possible to derive a small subset of the entire topological space and contain the true topology.

References


**Kamal Al Nasr** is a Ph.D. candidate at the Department of Computer Science at Old Dominion University, USA. He has received his B.S. and M.S. degrees from Yarmouk University, Jordan, in 2003 and 2005, respectively. He has worked as a part-time instructor in Yarmouk University, Jordan. Between August 2007 and
August 2009, he was a graduate student conducting research at the CREST Center (Center for Research Excellence in Bioinformatics and Computational Biology) at the Department of Computer Science at New Mexico State University before joining Old Dominion University.

Desh Ranjan received his undergraduate degree in computer science from Indian Institute of Technology, Kanpur, India, in 1987 and a Masters’ and Ph.D. in Computer Science, with a minor in Mathematics, from Cornell University in Ithaca, New York, in 1990 and 1992, respectively. After finishing his Ph.D., he spent one year (1992–1993) as a post-doctoral fellow at the Max Planck Institute for Computer Science in Saarbrucken, Germany. He was a member of the faculty of the computer science department at New Mexico State University for 16 years and served as the chair of the department from 2004 to 2009. He was the founding director of the NSF funded Center for Bioinformatics and Computational Biology at NMSU. His research interests include algorithms, bioinformatics and computational complexity.

Mohammad Zubair received the B.Sc. from Electrical & Electronics Engineering at Delhi University, India, in 1981. He received his Ph.D. degree from Indian Institute of Technology, India, in 1987. He worked in IBM T. J. Watson Research Center for three years. Currently he is a professor at the Department of Computer Science at Old Dominion University. Dr. Zubair has more than 20 years of research experience in the area of experimental computer science and engineering, both in academia as well as industry. He has conducted research in the following areas: digital library, advanced web applications including e-commerce, high performance computing and bioinformatics.

Jing He obtained her B.S. degree in Mathematics from Jilin University, China, and M.S. degree in Mathematics from New Mexico State University, USA. She worked in the area of three-dimensional reconstruction and analysis of virus structures at Baylor College of Medicine from which she obtained her Ph.D. in Structural and Computational Biology and Molecular Biophysics. Currently she is an Associate Professor at the Department of Computer Science at Old Dominion University.