PROTERAN: animated terrain evolution for visual analysis of patterns in protein folding trajectory

Ruhong Zhou¹,*, Laxmi Parida¹, Kush Kapila² and Sudhir Mudur³

¹Computational Biology Center, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA, ²Matrox Imaging Inc., Dorval, Quebec, Canada and ³Department of Computer Science, Concordia University, Montreal, Canada

Received on August 18, 2006; revised on October 4, 2006; accepted on October 14, 2006
Advance Access publication October 24, 2006
Associate Editor: Alfonso Valencia

ABSTRACT

Summary: The mechanism of protein folding remains largely a mystery in molecular biology, despite the enormous effort from many groups in the past decades. Currently, the protein folding mechanism is often characterized by calculating the free energy landscape versus various reaction coordinates such as the fraction of native contacts, the radius of gyration and so on. In this paper, we present an integrated approach towards understanding the folding process via visual analysis of patterns of these reaction coordinates. The three disparate processes (1) protein folding simulation, (2) pattern elicitation and (3) visualization of patterns, work in tandem. Thus as the protein folds, the changing landscape in the pattern space can be viewed via the visualization tool, PROTERAN, a program we developed for this purpose. We first present an incremental (on-line) trie-based pattern discovery algorithm to elicit the patterns and then describe the terrain metaphor based visualization tool. Using two example small proteins, a β-hairpin and a designed protein Trp-cage, we next demonstrate that this combined pattern discovery and visualization approach extracts crucial information about protein folding intermediates and mechanism.

Contact: ruhongz@us.ibm.com
Supplementary information: http://www.research.ibm.com/people/r/ruhong/SI.htm

1 INTRODUCTION

Understanding how protein folds continues to be one of the most challenging problems in computational biology. The interest is in both obtaining the final fold—as witnessed by recent advances in protein structure prediction and structural genomics, and understanding the mechanism and kinetics of the folding process. (Snow et al., 2002; Zhou et al., 2004; Zhou and Karplus, 1999)

Many native proteins fold into unique globular structures on a very short time scale. The so-called fast folders can fold into the native structure from random coils in microseconds to milliseconds. Recent advances in experimental techniques that probe proteins at different stages during the folding process have shed light on the nature of the folding kinetics and thermodynamics (Blanco et al., 1994; Munoz et al., 1997). However, due to experimental resolution limits, detailed protein folding pathways remain unknown. Computer simulations performed at various levels of complexity can be used to supplement experiments and fill in some of the gaps in our knowledge about folding mechanisms. Meanwhile, effective analysis and visualization of the trajectory data from the protein folding simulations, either by molecular dynamics (MD) or Monte Carlo (MC), remains yet another challenge due to the large number of degrees of freedom and the huge amount of trajectory data. Currently, the protein folding mechanism is often characterized by calculating the free energy landscape versus the so-called reaction coordinates (Garcia and Sanbonmatsu, 2001; Zhou et al., 2001). We and others have used various reaction coordinates, such as fraction of native contacts and radius of gyration (Garcia and Sanbonmatsu, 2001; Zhou et al., 2001). Searching for better reaction coordinates is still of active interest in protein folding mechanism studies. These analyses have provided important information for a better understanding of protein folding. However, the free energy landscapes usually result in too much information reduction due to their limitation in dimensionality, and worse, the time correlation information about various states is usually not reflected in these free energy contour maps. Therefore, better or complementary analysis tools are in great demand (Parida and Zhou, 2005; Lei et al., 2005).

1.1 Background

We had previously proposed a combinatorial pattern discovery approach to recognize the folding intermediates (Parida and Zhou, 2005). We describe that process briefly here to provide a context to this paper. It is known that the folding process of many proteins takes the amino acid coil through different states before stabilizing on the final folded state. Thus, from a pattern discovery point of view, the task is to recognize these intermediate states that the folding process goes through. The process can be briefly summarized as a four-step process. The first step involves the conversion of folding trajectories from MD simulations to a time series of characteristic features (reaction coordinates in this case), since working in the full structure space of the protein is extremely complex due to the number of degrees of freedom and the very large number of configurations (easily in millions) (Zhou et al., 2001, 2003a). In the second step, we study these data points to extract the characteristic set of features which we call patterned clusters. In the third step, these patterns are filtered to retain the most significant ones. It is non-trivial to model the significant patterns in this domain, so we have combined the second and third steps and used appropriate parameters to filter out possibly insignificant patterns. For instance, if a pattern occurs less than a certain (K) times,
Visualization
generation module, which extracts the patterns and feeds them to the newly generated data is fed into the
scape (animated over time).

A very and visualization approach is then applied to two small protein standing of the underlying processes. The combined pattern discov-
mental pattern discovery algorithm and the simplicity of the pattern such as that of the protein folding. We present a trie-based incre-
approach has been applied to better understanding of a process under-
intermediates and helps to identify structural motifs that are over-
understanding of the folding process.

The approach extracts crucial information about protein folding -hairpin and a designed 20-residue protein Trp-cage.

Incremental data generation process is the simulation process
that produces the reaction coordinates data with time (Sanbonmatsu, 2001; Zhou et al., 2005). Thus maximality is a combinatorial means
of reducing the number of cluster patterns, without any loss of information. The patterns under discussion in this paper are maxi-
mal. An off-line (i.e. given
algorithm, denoted as
on-line algorithm
on the

In this paper, we present an integrated approach towards under-
standing the folding process via visual analysis of folding patterns in
the form of reaction coordinates. The three disparate processes (i)
protein folding simulation with replica exchange molecular
amics (REMD) (Sugita and Okamoto, 1999; Garcia and Sanbonmatsu, 2001; Zhou et al., 2001), (ii) pattern elicitation
and (iii) visualization of the patterns, work in tandem. Here the Incremental data generation process is the simulation process
that produces the reaction coordinates data with time (x-axis in the
Fig. 1). At each time interval \( t \) (in the figure, \( t_0=1 \)), the newly generated data is fed into the Incremental maximal pattern
generation module, which extracts the patterns and feeds them to the Visualization module which displays an appropriate pattern landscape (animated over time).

To our knowledge, this is the first time such an integrated approach has been applied to better understanding of a process such as that of the protein folding. We present a tri-based incremental pattern discovery algorithm and the simplicity of the pattern descriptions lead to easy interpretation of and thus better understanding of the underlying processes. The combined pattern discovery and visualization approach is then applied to two small protein systems, a \( \beta \)-hairpin and a designed 20-residue protein Trp-cage. The approach extracts crucial information about protein folding intermediates and helps to identify structural motifs that are overlooked by the free energy landscape analysis. This success encourages us to postulate that this combined approach will lead to deeper understanding of the folding process.

2 INCREMENTAL PATTERN DISCOVERY ALGORITHM

Consider a scenario where there are \( n \) data points with each data point having \( m \) attributes that take on real values. Each attribute is denoted by \( J_j \), \( 1 \leq j \leq m \).

Let \( D \) be an \( n \times m \) array where \( D[i,j] \in \mathbb{R} \) that represents the value of property \( J_j \) for the \( i \)-th data point. A cluster pattern, \( p \) is a collection of some \( l \) \( (1 \leq l \leq m) \) attributes or columns denoted as
\[
p = (J_{j_1} = c_{j_1}, J_{j_2} = c_{j_2}, \ldots, J_{j_l} = c_{j_l}).
\]

Given a fixed set of values \( \delta_1 \) for each \( j \), we define the location list \( L_p \) of \( p \) (which is a collection of rows) as follows:
\[
L_p = \{ i | (c_{j_1} - \delta_1) < D[i,j] < (c_{j_1} + \delta_1), j_1 \leq j \leq j_l \}.
\]
This pattern \( p \) is said to satisfy the quorum \( K \) if \( |L_p| \geq K \). In other words, the pattern \( p \) must have support in at least \( K \) data points.

A pattern \( p \) with \( l \) columns is maximal if no more columns can be added without decreasing the support of the pattern. More precisely, \( p \) is maximal if there exists no pattern \( p' \), with \( p' \supset p \) and \( L_{p'} \subseteq L_p \) (Parida and Zhou, 2005). Thus maximality is a combinatorial means of reducing the number of cluster patterns, without any loss of information. The patterns under discussion in this paper are maximal. An off-line (i.e. given all the \( n \) data points of the input \( D \)) algorithm to discover these maximal cluster patterns is discussed in (Parida and Zhou, 2005). It has also been shown that it is possible to define discrete alphabet such that \( D[i,j] \in \Sigma \), a finite alphabet set. In the rest of the discussion we will assume that \( D \) is defined on this finite alphabet \( \Sigma \).

Here, we discuss an on-line algorithm to solve the same problem. The assumption is that all data points are not available at the same time and either each data point or chunks of data points are made available with time. More precisely, let each row \( i \) represent the data in time \( i \) and at time \( i \), the rows \( 1,2, \ldots, i \) are available to the algorithm, denoted as \( D[1 \ldots i,m] \) (i.e. \( i \) rows and \( m \) columns). Since maximal pattern \( p \) generated at time \( t_i \) is displayed, it is important to assert that at time \( t_2 > t_1 \), \( p \) remains maximal. The following is easy to verify.

**Lemma 1.** A pattern \( p \) that is maximal in \( D[1 \ldots t_1, m] \) is also a maximal pattern in \( D[1 \ldots t_2,m] \), where \( t_2 > t_1 \).

Since the data arrives one complete row at a time, a previously declared maximal pattern cannot become non-maximal later. Since we have shown that this model is viable, now we present an incremental method to discover the patterns.

2.1 Trie-based algorithm

For simplicity, assume that \( D \) is defined on a set of time records (reaction coordinates) \( \Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_t\} \), with an ordering on the
alphabet as $\sigma_1 < \sigma_2 < \ldots < \sigma_t$. Let a new record (edge node) `-' $\not\in \Sigma$ and let $\Sigma \cup \{\text{`-'}\}$ be denoted by $\Sigma^*$. Further, let $\sigma_i < \text{`-'}$ for all $1 \leq i \leq t$.

We present a method based on a modified trie data structure (Aho et al., 1983) which we call the **augmented trie**. First, as each row of $D$ is read, it is treated as a string of length $m$ and the standard trie data structure is constructed. Recall the following two properties of a standard trie (Aho et al., 1983):

**Property 1.** There is one root node whose depth is assumed to be 0. Each edge is labeled and the label of an internal node is assumed to be the label of the (unique) incoming edge.

**Property 2.** The tree is of height $m$. An internal node at depth $j$ from the root denotes the $j$-th column of $D$.

In this standard trie (whose edges are labeled with characters from $\Sigma$), each unique path from the root node to the leaf node $A$ represents a pattern $p_A$ in $D$ with exactly $m$ columns and has a pointer to $L_{p_A}$ or the set of rows where $p_A$ occurs.

**Augmenting the trie.** Consider a trie or tree with labels from $\Sigma^*$. Now, each unique path from the root node to the leaf node $A$ represents a pattern $p_A$ in $D$; however, if an edge at depth $j$ is labeled with `-', then column $j$ is ignored. Thus in the unique path from the root to the leaf node $A$ if there are $t$ edges with label `-', then the pattern $p_A$ has $m-t$ columns. Further, it also has a pointer to $L_{p_A}$ or the set of rows where $p_A$ occurs.

We next define a **minimal consensus tree** $T$, given $k$ trees as follows: Given the input data $D$ and trees $T_i$, $i = 1, \ldots, k$, labeled on $\Sigma^*$, $T$ is a consensus tree of the $k$ trees, (1) if for each leaf node $A_i$ in $T_i$, there is a leaf node $A$ in $T$ with $p_A = p_{A_i}$ and $L_{p_A} = L_{p_{A_i}}$ in $D$,

(2) for every leaf node $A$ in $T$, there is leaf node $A_i$ in $T_i$, for some $i$ such that $p_{A_i} \subseteq p_A$, and,

(3) no two siblings of $T$ have the same label (i.e. $T$ is a trie). Further, $T$ is a minimal consensus tree if it has no subtree $T'$ that is a consensus of $T_i$, $i = 1, \ldots, k$.

Also, the following observation is straightforward to verify.

**Lemma 2.** The minimal consensus tree $T$ of $k$ trees is unique.

We augment this standard trie to have labels from $\Sigma^*$ as follows. Note that this is a recursive definition. An internal node, $B$, that has more than one child, denoted as $C_1, C_2, \ldots, C_t$, is augmented to have wild child, say $C_{wild}$, whose edge is labeled by `-'.

The stepwise construction of the trie is shown in Figure 2. The branches labeled with `-' are shown as dashed lines (the rightmost child of a node) for convenience. The quorum is $K = 2$. The maximal pattern $p_1$ is generated when row 2 is read, maximal pattern $p_2$ when row 4 is read; maximal patterns $p_1$ and $p_2$ when row 5 is read. The bold edges in Figure 1b and c denote the new branches generated at that step. Since the labels are ordered and the wild edge is always the rightmost child, a pattern $p_1$ on the left is more specific than $p_2$ on the right, the following observation can be verified.

**Lemma 4.** Let leaf nodes $p_1 < p_2$ in the left-to-right ordering of the augmented trie, (i.e. leaf node $p_1$ is to the left of $p_2$) with $L_{p_1} \subseteq L_{p_2}$. Then $p_2$ must be non-maximal with respect to $p_1$.

This property enables us to display the patterns as they are created (without ever having to backtrack). In the algorithm, as a new location list is generated, it is checked with existing location pointed to by the leaf nodes may be expanded (but not reduced). This is also demonstrated in the simple example below (Fig. 1).

**Example 1.** Consider a scenario with five data points, each with five attributes. For simplicity, let $\Sigma = \{a < b < c < s < p < q < r < s < y < z\}$ and the corresponding $D$ with columns $J_1, J_2, J_3, J_4, J_5$ is shown below:

<table>
<thead>
<tr>
<th></th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
<th>$J_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>z</td>
<td>a</td>
<td>d</td>
<td>p</td>
<td>q</td>
</tr>
<tr>
<td>4</td>
<td>y</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>5</td>
<td>y</td>
<td>r</td>
<td>d</td>
<td>s</td>
<td>c</td>
</tr>
</tbody>
</table>

Let $T$ be the trie under construction. When $D$ is empty, $T$ has only the root node. To add a new row $i$ of $D$ to $T$, this row is treated as a string of length $m$ and added to the trie $T$ in the standard way (Aho et al., 1983). Further, this row is added to the location list pointed to by the leaf node. Let $P$ be a node that has more than one child in $T$ given as $C_i$, $i = 1, \ldots, k$. Using the $\text{MinConsensus()}$ algorithm, a child with label `-' is either created or updated.

**Input:** Two trees $T_1$ and $T_2$

**Output:** $T_1$ is modified as the minimal consensus of $T_1$ & $T_2$

**MinimalConsensus**(T1, T2)

\[Nd_1 = \text{Root}(T_1), Nd_2 = \text{Root}(T_2)\]

If $Nd_1 \text{ OR } Nd_2$ is a leaf node

- Update location list of leaf node of $Nd_1$
- Check if the location list (pattern) exists
- Terminate the traversal

For each child $Ci$ of $Nd_1$

Let $C_{wild}$ of $Nd_2$ be such that $\text{Label}(C_{wild}) = \text{Label}(Ci)$

**MinimalConsensus**(Ci, C_{wild})

For each remaining child $C_{wild}$ of $Nd_2$

Introduce $C_{wild}$ as a child of $Nd_1$

Let $C_{wild}$ be the child of $Nd_1$ with label “-”

**MinimalConsensus**(C_{wild}, Ci)

**Fig. 2.** The minimal consensus tree (recursive) algorithm for two given trees.

Let $T_1$ be the augmented trie corresponding to $D[1 \ldots i]$. Then $T_i$ is a subtree of $T_{i+1}$ for $i_1 < i_2$.

This shows that as each row is read, the augmented trie grows without any need for backtracking. However, the location lists
lists. The location lists are stored in a balanced binary tree to make this checking efficient. If it is a new list, the pattern is output as a maximal pattern.

**Time complexity.** In the application, D is defined on real values. The implications of converting these real values to discrete characters is discussed in (Parida and Zhou, 2005). We use the same approach here in the on-line algorithm.

Recall in our application, $m \ll n$. The augmented trie can be compacted to give a Patricia or a radix tree (Aho et al., 1983; Morrison, 1968) for efficiency in space. Figure 3 shows a compact Patricia tree for the running example.

In the worst case there can be $O(2^m)$ distinct patterns where each column has at most $d$ different values. Also, the location lists are stored in a balanced binary tree and the time to check if a list already exists takes $O(n \log n)$ time. At step $i$, let $N$ be the number of patterns including the non-maximal patterns. The algorithm then takes $O(N n \log n)$ time for each step $i$.

Reducing the pattern space. It is quite clear that using maximality and quorum $K$ is not adequate to control the number of patterns to be studied. We are currently experimenting with the following as a way of reducing this space without losing important information. Two patterns $p_1$ and $p_2$ are $\varepsilon$-equal if $\frac{|p_1 \setminus p_2|}{|p_1|} > \varepsilon$, for some fixed $0 < \varepsilon \leq 1$. When two patterns $p_1$ and $p_2$ are $\varepsilon$-equal, the two are replaced with $p = p_1 \cap p_2$, and $L_p = L_{p_1} \sqcap L_{p_2}$. Thus in the reduced pattern space, no two distinct patterns $p_1$ and $p_2$ are $\beta$-equal.

3 ON-LINE PATTERN VISUALIZATION

It is clear that trying to manually analyze large amounts of data will result in important information being overlooked. Memorizing a few patterns is quite difficult and with the possibility of thousands of patterns getting a global view of the pattern space will be even more difficult. The user is now left with trying to memorize both the patterned clusters and their relative order in time. Hence the need arises for a good visualization tool that can give both a global view of the pattern space and the interaction of the patterned clusters as they evolve over time. When the pattern clusters are fed in a continuous fashion into our visualization tool, PROTERAN, the time correlation of the patterns and therefore the order of the folding events can be easily visualized and recorded (see supplementary material for additional details on the PROTERAN design).

**PROTERAN design.** We note that the patterns in this application have two important properties. One important aspect is the number of columns in the pattern. The second important aspect is the extent of prevalence of the pattern in the data in terms of its frequency or the number of times the pattern appears in the data. We design the system keeping these two major aspects in mind. The functional requirements of the visualization program were identified as follows:

1. **Global View:** The user should be able to at any point see a global view of the total current data.
2. **Zoom-In:** The user should be able to identify important clusters and focus on any individual patterned cluster.
3. **Animation:** The patterned cluster should be animated over time.
4. **Query:** The column and its corresponding values of the patterned cluster should be obtainable at any time.

The terrain metaphor. Here we use the metaphor of a terrain (Chalmers, 1993; Wise et al., 1995) to represent our pattern space. We identify the number of columns defining a pattern cluster as a way to type the patterns. The first step to visualizing as a terrain is to break down the plane into sub-regions corresponding to different types of patterns. Thus if there are $m$ columns, the terrain is broken down into $(m - 1)$ regions: each region corresponds to a pattern with $1 < l \leq (m - 1)$ columns. Next, a region corresponding to $l$ columns can further be broken down into $(\frac{m!}{l!})$ distinct patterns. For example, for $m = 7$ and $l = 2$ there can be at most 21 different patterns (see Supplementary Material for the layout of patterns).

The frequency of the pattern is represented by the height of the mountain whose base is the assigned territory in the plane. Thus by a quick visual inspection of the heights of the mountain, one gets a fair idea of the major states at play. In addition, because the mountains grow over time it is easy to see a time-line of the states or the patterns. The ability to ‘fly’ through the terrain allows the user to get different perspectives on the data. The zoom in and out buttons give the user the flexibility to either get a global view of the terrain or focus in on a specific pattern. Finally by pointing and clicking on a mountain the values of the respective patterns and all their properties (values of each reaction coordinate and their occurrence times) can be extracted. A sample screen shot is shown in Figure 4 and PROTERAN is available for use at http://www.kapila.ca/Proteran.htm

4 RESULTS

4.1 Folding of a $\beta$-hairpin

The first example protein is a 16-residue $\beta$-hairpin from the C-terminus of protein G (GEWYDADTKITFTVTE, 2gb1.pdb). Table 1 lists some representative patterns of size two from the previous study (Parida and Zhou, 2005), which are shown as red patterns in PROTERAN in Figure 4 (See Supplementary Material for the details of simulation and reaction coordinates). The time
More importantly, the new approach reveals important structures overlooked previously, which might help to understand the folding mechanism better. Eaton and co-workers (Munoz et al., 1997) proposed a ‘hydrogen bond zipping’ mechanism for this β-hairpin in which folding initiates at the turn and propagates toward the tails by making beta-strand hydrogen bonds one by one, so that the hydrophobic core, from which most of the stabilization derives, forms relatively late during the folding. In our previous study, we proposed a different folding mechanism that this β-hairpin undergoes a hydrophobic core collapse first, then forms native β-strand hydrogen bonds. Figure 5c shows a representative structure for the first pattern in Table 2, which lists multi-column patterns. The structure shows that all the five native β-strand H-bonds have been formed, but the hydrophobic core is not completely aligned yet (RMSDs up to 4 Å). The loop region also bends towards the hydrophobic core to somewhat offset the non-perfect hydrophobic core. This implies that the hairpin can also have a path to form β-strand hydrogen bonds before the core is finalized. The current findings indicate that the final hydrophobic core and β-strand hydrogen bonds might be formed almost simultaneously. This can also be seen from the low free energy barrier in the previous free energy landscape analysis (Zhou et al., 2001). Interestingly, Thirumalai et al. also found that the lag time between collapse and hydrogen bond formation is very short and the two processes can occur nearly simultaneously (Klimov and Thirumalai, 2000).

### 4.2 Folding of the Trp-cage

As mentioned in Introduction section, it is also important to study the time correlation between various folding patterns or states. For example, it is extremely useful to know which pattern or state precedes the other and by how much time. Of course, this requires continuous trajectory data, ideally the true folding kinetics data. This subsection will use another protein Trp-cage, a designed 20-residue mini-protein (Neidigh et al., 2002; Simmerling et al., 2002; Snow et al., 2002), as an example to demonstrate PROTERAN on this time correlation feature. The current data were also obtained from the REMD simulations, but the trajectory data were organized according to each replica which can climb up or down in temperature ladder (in the previous β-hairpin case, the trajectory was obtained at the fixed biological temperature 310K); thus, these trajectories are continuous in the structural space. Even though the time sequences are not truly kinetic, they can still provide useful information on the time correlated or time sequential folding events. The goal here is not trying to compare with experimental kinetic data, but rather to demonstrate the current pattern discovery algorithm and visualization tool. Another point to mention is that the continuum solvent model, Generalized Born (GB) model (Still et al., 1990) is used in the Trp-cage simulation, which might have some artifacts. The GB model has been found to have a tendency to form erroneous salt-bridges between charged residues. (Zhou et al., 2001) The reason why we used the Trp-cage folding data from the GB model is largely because we have access to the REMD data, (Pitera and Swope, 2003) and also because this Trp-cage seems to be a robust folder and others (Simmerling et al., 2002) have also successfully folded this protein from fully stretched initial configuration using the GB model.

Figure 6 shows a few snapshots of all the patterns at \( t = 0, 1.25, 2.5, \) and 5 ns for one replica. Some patterns, such as \( R_4 = 7.914 ± 0.5, \) \( N_{helix} = 5.5 ± 0.5, \) show up earlier than other patterns.
indicating some structural patterns or intermediate structures develop in early stage of the folding (see Supplementary Material for definitions of all reaction coordinates). For example, this particular pattern, \( R_g = 7.914 \pm 0.5, N_{\text{helix}} = 5.5 \pm 0.5 \), is found to be related to the alpha helix formation near residues 2–9 (more details below). The time sequences when one particular pattern appears can be collected for each pattern, and consequently, collective patterns can be obtained at each time sequence window (window size 200–400 frames). For those time sequences with many patterns appearing at the same time, some structural signatures or motifs might be expected. Table 3 lists the top time sequences with most patterns identified in that time sequence window. As found previously, many patterns are redundant, for example, patterns \( (R_g = 7.914 \pm 0.5, \rho = 0.729 \pm 0.15), (\rho = 0.729 \pm 0.15, \text{RMSD} = 3.151 \pm 0.5, N_{\text{helix}} = 8.5 \pm 0.5,) \) and \( (R_g = 7.914 \pm 0.5, \rho = 0.729 \pm 0.15, \text{RMSD} = 3.151 \pm 0.5, N_{\text{helix}} = 8.5 \pm 0.5) \) all represent the folded state in this particular case.

A closer look at the structures in these time sequences might reveal important intermediate folding events. The representative structures for each top time sequence window can be obtained by clustering (Zhou et al., 2001, 2003). These representative structures for the first five time sequences as well as the initial extended structure are shown in Figure 7. The key hydrophobic residues forming the Trp-cage core, Tyr3, Trp6, Leu7, Pro12, Pro17, Pro18 and Pro19 are represented by sticks, whereas the rest of the proteins are represented by the ribbon view. The first major time sequence (or event) shows that an \( \alpha \)-helix between residues 2–9 starts to develop after \( /C_24 \) 0.50 ns (sequence 1). Interestingly, the \( 3_{10} \)-helix near residues 11–14 is also partially formed in the early stage, but it comes and goes with time. The \( \alpha \)-helix keeps developing, as shown in time sequences 2 (\( t = /C_24 \) 0.76 ns) and 3 (\( t = /C_24 \) 1.2 ns). During this process, the C-terminal poly-proline II helix has not packed against the \( \alpha \)-helix or central tryptophan residue yet. The \( 3_{10} \)-helix is reformed at \( /C_24 \) 1.5 ns (sequence 4). Meanwhile, another important folding event occurs—the Cα-terminal poly-proline II helix forms and packs against the \( \alpha \)-helix. At approximately 2.0 ns (sequence 5), the sidechain of Trp6 has optimized its position inside the hydrophobic cage formed by the \( \alpha \)-helix and poly-proline II helix. Thus, the Trp-cage protein has been folded, with a Cα-RMSD of only 2.4 Å from the native

---

**Table 2. Complex patterns with multi-columns (6 and 7 in this case)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Cluster pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( J_1 = 4.950 \pm 0.2 )</td>
</tr>
<tr>
<td>2</td>
<td>( J_2 = 5.748 \pm 0.6 )</td>
</tr>
<tr>
<td>3</td>
<td>( J_1 = 4.903 \pm 0.2 )</td>
</tr>
</tbody>
</table>

**Table 3. Top time sequences with most patterns collected**

<table>
<thead>
<tr>
<th>Sequ. ID #</th>
<th>Start frame</th>
<th>End frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>900</td>
<td>1100</td>
</tr>
<tr>
<td>2</td>
<td>1300</td>
<td>1500</td>
</tr>
<tr>
<td>3</td>
<td>2200</td>
<td>2400</td>
</tr>
<tr>
<td>3</td>
<td>2900</td>
<td>3100</td>
</tr>
<tr>
<td>4</td>
<td>3900</td>
<td>4100</td>
</tr>
<tr>
<td>5</td>
<td>4400</td>
<td>4600</td>
</tr>
<tr>
<td>6</td>
<td>5400</td>
<td>5700</td>
</tr>
</tbody>
</table>

**Fig. 6.** Animation of patterns in the Trp-cage folding versus time for one replica which can climb up and down in the temperature ladder. (a) 0 ns; (b) 1.25 ns; (c) 2.50 ns; and (d) 5.0 ns.

**Fig. 7.** Representative structures of the Trp-cage protein from the major time sequences in Table 3.
structure. These results indicate that the folding process starts with the formation of the α-helix near residues 2–9; then the poly-proline II helix (residues 15–20) forms and packs against the α-helix; and finally the sidechain of Trp6 optimizes its position within the cage formed by the α-helix and the poly-proline II helix. The 3αH near residues 11–14, on the other hand, comes and goes during the folding process. Once the protein is folded, it can stay in the folded state for quite some time before it unfolds again due to its climbing to high temperatures in the REMD simulation.

It should be pointed out that the folding time ~2.0 ns here is much faster than the experimental value of ~4 μs (Qiu et al., 2002). This seemingly much faster folding speed is due to two important factors: one is that in REMD the energy barrier crossings can be tens or hundreds of times faster than the regular MD (Garcia and Sanbonmatsu, 2001; Zhou et al., 2001); and the other is that the folding kinetics in the continuum solvent model GB can be much faster than the experiment as found by others as well (Summerling et al., 2002; Pitera and Swope, 2003; Chowdhury et al., 2003). Nevertheless, we are more interested in the time sequential orders of the folding events here.

As mentioned above, the time correlation function can also be calculated for patterns:

\[ C_{AB}(\tau) = \langle A(t)B(t + \tau) \rangle, \tag{1} \]

where \( A(t) \) and \( B(t) \) represent two different patterns as a function of time and \( \langle \cdot \rangle \) indicates the ensemble average. When \( A(t) \) is the same as \( B(t) \), the correlation function reduces to the normal autocorrelation function:

\[ C_{AA}(\tau) = \langle A(t)A(t + \tau) \rangle. \tag{2} \]

Here we use two example correlation functions to illustrate the concept (see Supplementary Material for more details and graphs). One is the autocorrelation function of the helix-pattern (representing the helix formation—defined as 1 if eight or more residues are alpha-helical, and 0 otherwise), and the other is the time correlation function between the helix-pattern and the folded-pattern (\( R_\text{N} = 7.914 \pm 0.5, \rho = 0.729 \pm 0.15, \text{RMSD} = 3.151 \pm 0.5, N_{\text{helix}} = 8.5 \pm 0.5, \text{see above, pattern for folded conformation}. The helix-pattern autocorrelation function shows a \( C_{AA}(t) \propto \exp(-1.384t) \) (in ns), indicating a helix-pattern lifetime of ~0.72 ns; while the helix-folded-patterns correlation function shows a \( C_{AB}(t) \propto \exp(-0.268t) \), indicating a correlation characteristic time of 3.73 ns (the helix formation on average leads the folded conformation by 3.73 ns). This is consistent with the above time sequence analysis and the patterns we observe in PROTERAN, even though the exact time differs slightly due to the insufficient data in the correlation function calculations. Finally, it should be noted that the current method is equally applicable to real-time kinetic MD trajectories, once these long-scale simulations at biological temperature become routinely accessible.

5 CONCLUSION

In this paper, we have presented an integrated approach towards understanding the protein folding via visual analysis of folding patterns, which consists of three processes (1) protein folding simulation, (2) pattern elicitation and (3) pattern visualization. The online pattern discovery algorithm uses a trie-based approach and the visualization tool uses the metaphor of a terrain to represent a pattern, which can also animate the changing pattern landscape with time. Two small but important protein systems, a β-hairpin from C-terminus of protein G and a designed fast folding protein Trp-cage, are then used to demonstrate the effectiveness of this combined approach. It is shown that the method not only reproduces the previously found free energy states (most populated states) in free energy contour maps, but also reveals new information overlooked previously about the intermediate structures and folding mechanism. In addition, the combined approach can reveal the time correlated or time-dependent information about various folding events. Furthermore, it is shown that the method can identify possible force field artifacts by discovering previously overlooked structural motifs. The success with the β-hairpin and Trp-cage proteins is very encouraging and we are currently exploring the application of this method to other larger proteins, such as lysozyme. For these larger and more complicated protein systems, more structural features and intermediates might be expected in the molten globule and unfolded states (Klein-Seetharaman et al., 2002).

ACKNOWLEDGEMENTS

We would like to thank Jed Pitera and William Swope for sharing with us the protein Trp-cage folding data for analysis.

Conflict of Interest: none declared.

REFERENCES


