BIOINFORMATICS

Vol. 20 Suppl. 1 2004, pages i386–i393
DOI: 10.1093/bioinformatics/bth935

Predicting protein folding pathways

Mohammed J. Zaki[1,*] Vinay Nadimpally[1], Deb Bardhan[1] and Chris Bystroff[2]

[1]Department of Computer Science and [2]Department of Biology, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Received on January 15, 2004; accepted on March 1, 2004

ABSTRACT
Summary: A structured folding pathway, which is a time ordered sequence of folding events, plays an important role in the protein folding process and hence, in the conformational search. Pathway prediction, thus gives more insight into the folding process and is a valuable guiding tool to search the conformation space. In this paper, we propose a novel 'unfolding' approach to predict the folding pathway. We apply graph-based methods on a weighted secondary structure graph of a protein to predict the sequence of unfolding events. When viewed in reverse this yields the folding pathway. We demonstrate the success of our approach on several proteins whose pathway is partially known.
Contact: zaki@cs.rpi.edu

1 INTRODUCTION

Given a protein's amino acid sequence and its three-dimensional (3D) structure, the pathway prediction problem is to determine the time ordered sequence of folding events, called the folding pathway, that leads from the primary to the tertiary structure. Whereas the protein structure prediction problem is widely acknowledged as an open problem, the pathway prediction problem has received little attention. It is clear that the ability to predict folding pathways can greatly enhance structure prediction methods. Folding pathway prediction is also interesting in itself, since protein misfolding has been identified as the cause of several diseases such as Creutzfeldt–Jacob disease, cystic fibrosis, hereditary emphysema and some cancers (Dobson, 2003).

Strong experimental evidence for pathway-based models of protein folding has emerged over the years, e.g., experiments revealing the structure of the 'unfolded' state in water (Mok *et al.*, 1999), burst-phase folding intermediates (Colon and Roder, 1996), and the kinetic effects of point mutations ['phi-values' (Nolting *et al.*, 1997)]. These pathway models indicate that certain events always occur early in the folding process and certain others always occur later. Currently, there is no strong evidence that specific non-native contacts[1] are required for the folding of any protein (Chikenji and Kikuchi, 2000). Many simplified models for folding, such as lattice simulations, tacitly assume that non-native contacts are 'off pathway' and are not essential to the folding process (Klimov and Thirumalai, 2001). Therefore, we choose to encode the assumption of a 'native pathway' into our algorithmic approaches. This simplifying assumption allows us to define potential folding pathways based on known 3D structure. We may further assume that native contacts are formed only once in any given pathway.

One approach to enumerate folding pathways is to start with an unfolded protein and consider the various possibilities for the protein to fold. This approach is expensive due to the explosively large number of possibilities to consider for the pathways, although some recent progress has been made by applying a probabilistic road map (motion planning) based approach (Song and Amato, 2002). However, the roadmap approach is still expensive; it takes 2–15 hours even for relatively small proteins (60–110 residues).

Our novel approach is to start with a folded protein in its final state and learn how to 'unfold' the protein in an approximately ordered sequence of steps, to its unfolded state. The reversal of such a sequence then represents a plausible protein folding pathway, a view supported by Fersht and Daggett, 2002. We use a graph representation of a protein, where a vertex denotes a secondary structure element (SSE) and an edge denotes the interactions between two SSEs. The edges are weighted by the strength of the SSE interactions. The basic intuition behind our approach is to break the weakest interactions to obtain a sequence of unfolding events. We present two algorithms: Unfold, that predicts one unfolding sequence, and MultiUnfold, that enumerates many folding pathways and ranks the intermediates based on their expected frequency. The approaches are extremely fast, taking on the order of a few seconds (for proteins as large as 162 residues). Our pathway predictions also show remarkable agreement with experimentally determined pathways.

2 PRELIMINARIES

Weighted graphs An undirected graph $G(V, E)$ is a structure that consists of a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and a set

*To whom correspondence should be addressed.
[1]A native contact is the one retained in the folded protein.

of edges $E \subseteq V \times V$, given as $E = \{e_i = (s,t) \mid s,t \in V\}$, i.e. each edge e_i is an unordered pair of vertices. A weighted graph is a graph with an associated weight function $W : E \rightarrow \Re^+$ for the edge set. For each edge $e \in E$, $W(e)$ is called the *weight* of the edge e.

Minimum cuts A cut C of a weighted graph G, is a partition of the set of vertices into two non-empty subsets C and $\overline{C} = V - C$. We will mostly (unambiguously) refer to a cut by specifying just one of the subsets C. The capacity of the cut C is the sum of the weights of edges that cross (i.e. have exactly one endpoint in C) the cut, given as $W(C) = \sum_{e=(s,t)\in E, s\in C, t\in\overline{C}} W(e)$. A cut C is an *s-t cut* if vertices s and t are in different partitions of the cut. A minimum *s-t* cut is an *s-t* cut of minimum capacity. A (global) minimum cut (mincut) is a minimum *s-t* cut over all pairs of vertices s and t. Note that mincut need not be unique.

Cut-trees A cut tree of weighted graph G, is a weighted tree $T = (V, E')$ on V, which represents the structure of all the *s-t* mincut values of G as follows: for every pair of distinct vertices $s, t \in V$, let $e \in E'$ be a minimum weight edge on the unique path from s to t in T. Deleting e from T separates T into two disjoint vertex sets S (with $s \in S$) and $T = \overline{S}$ (with $t \in T$). Then S and T denote the two partitions in a minimum *s-t* cut. Note that T is not a subgraph of G.

3 WEIGHTED SSE GRAPH

A protein can be represented as a weighted secondary structure element graph (WSG), where the vertices are the SSEs comprising the protein and the edges denote proximity relationship between the secondary structures. Furthermore, the edges are weighted by the strength of the interaction between two SSEs. We construct the WSG for a given protein as follows: we determine the list of SSEs and their sequence positions from the known 3D structure taken from the Protein Data Bank (PDB) (http://www.rcsb.org/pdb/). Every SSE is a vertex in the WSG. Let $V = \{v_1, v_2, \ldots, v_n\}$ denote a protein with n SSEs. Each SSE v_i has starting $(v_i \cdot s)$ and ending $(v_i \cdot e)$ sequence positions, where $1 \leq v_i \cdot s < v_i \cdot e \leq N$, and N is the length of the protein. The edge weights are determined as follows: Let v_i and v_j be a pair of SSEs. Let the indicator variable $b(v_i, v_j) = 1$ if v_i and v_j are consecutive on the protein backbone chain, else $b(v_i, v_j) = 0$. Let $\kappa(v_i, v_j)$ denote the interaction strength between the two SSEs. An edge exists between two SSEs if their interaction strength exceeds some threshold, i.e. if $\kappa(v_i, v_j) > \kappa^{\min}$, or if the two SSEs are consecutive on the backbone chain. The weight assigned to the edge (v_i, v_j) is given as follows: $W(v_i, v_j) = \Delta \times b(v_i, v_j) + \kappa(v_i, v_j)$, where Δ is some constant. In our study, we set Δ as the average interaction strength between SSEs, i.e. $\Delta = \left[\sum_{\kappa(v_i,v_j)\in \mathbf{S}} \kappa(v_i, v_j)\right]/|\mathbf{S}|$, where $\mathbf{S} = \{\kappa(v_i, v_j) > \kappa^{\min} \mid v_i, v_j \in V\}$. This weighting scheme gives higher weights to backbone edges and also to SSEs with greater interaction strength between them. The backbone edges are given higher weight since they represent strong covalent bonds, while the other interactions represent weaker non-covalent bonds.

There are several schemes to compute the strength of interaction between two SSEs: contact, distance and solvent accessible surface (SAS) based.

Contact-based Given two amino acids a_i and a_j along with the 3D coordinates of their α-carbon atoms (or alternately β-carbon), (x_i, y_i, z_i) and (x_j, y_j, z_j), respectively, define the Euclidean distance between them as follows: $\delta(a_i, a_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$. We say that a_i and a_j are in contact, if $\delta(a_k, a_l) \leq \delta^{\max}$, where $\delta^{\max}$ is some maximum allowed distance threshold (a common value is $\delta^{\max} = 7$ Å). A contact map for a protein with N residues is an $N \times N$ binary matrix C whose element $C(a_i, a_j) = 1$ if residues a_i and a_j are in contact, and $C(a_i, a_j) = 0$ otherwise. The contact-based interaction strength is given as the number of contacts between the two SSEs in the contact map, given as $\kappa(v_i, v_j) = \sum_{a_k=v_i\cdot s}^{v_i\cdot e} \sum_{a_l=v_j\cdot s}^{v_j\cdot e} C(a_k, a_l)$.

Distance based Let $D(a_k, a_l) = \delta^{\max}/\delta(a_k, a_l)$, if $C(a_k, a_l) = 1$, and $D(a_k, a_l) = 0$, otherwise. The distance-based interaction strength is defined as $\kappa(v_i, v_j) = \sum_{a_k=v_i\cdot s}^{v_i\cdot e} \sum_{a_l=v_j\cdot s}^{v_j\cdot e} D(a_k, a_l)$. Intuitively, we consider only interaction between residues within the $\delta^{\max}$ threshold [for $C(a_k, a_l) = 1$], and the closer the residues are the greater the strength of interaction $(\delta^{\max}/\delta(a_k, a_l))$. The distance-based method is thus an extension of the contact-based method, where we scale each residue pair interaction by how close they are.

Solvent accessible surface based SAS is the area of the molecular surface that is in contact with a spherical solvent molecule of a defined size (1.4 Å). Let S_i and S_j denote the SAS for residues a_i and a_j, and S_{ij} the SAS for the combined atoms. We compute SAS using our MASKER (Bystroff, 2002) program. The buried surface (in Å^2) between the two amino acids is given as follows: $\lambda(a_i, a_j) = S_i + S_j - S_{ij}$. The buried surface area between two residues is a good measure of the amount of water displaced by the residue–residue contact. Desolvation of hydrophobic groups and hydrogen bonding groups is the primary driving force for protein folding (Dill, 1990). The solvation free energy is proportional to the buried surface area[2]. The SAS-based interaction strength between two SSEs is defined as $\kappa(v_i, v_j) = \sum_{a_k=v_i\cdot s}^{v_i\cdot e} \sum_{a_l=v_j\cdot s}^{v_j\cdot e} \lambda(a_k, a_l)$.

Consider the 3D structure of IgG-binding protein G (PDB code 2IGD; 61 residues) as shown in Figure 1, which also shows the WSG for 2IGD, using contact-based interaction strength. Following the convention used in protein topology

[2]Ignoring differences in solvation energy between polar and non-polar atoms; we can safely neglect these differences because we consider only native, or generally favorable, contacts.

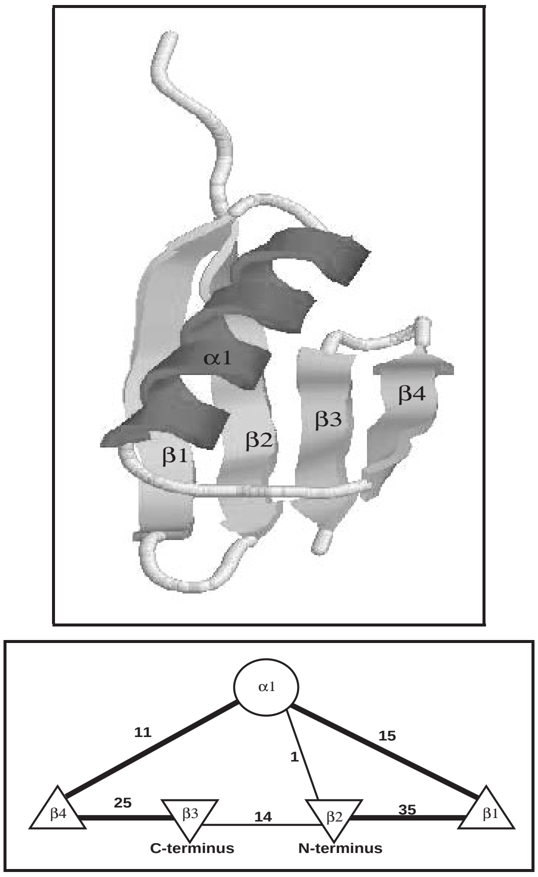


Fig. 1. 3D structure and WSG for protein G (PDB 2IGD).

or TOPS diagrams (Westhead *et al.*, 1999; Gilbert *et al.*, 1999), we use triangles to represent β-strands, and circles to represent α-helices. The thick lines denote backbone edges. SSEs are arranged from the N-terminus (start) to the C-terminus (end), and numbered as given in the PDB file; 2IGD has five SSEs, $\beta_2\beta_1\alpha_1\beta_4\beta_3$ arranged from the N- to C-terminus.

4 PREDICTING FOLDING PATHWAYS

In this section, we outline our approach to predict the folding pathway of a protein using the idea of unfolding. We first describe Unfold, an algorithm that predicts one plausible folding pathway. We next present MultiUnfold that finds out many possible folding pathways and then ranks the intermediate configurations.

4.1 The Unfold algorithm

Given a weighted SSE graph for a protein, a mincut represents the set of vertices that partition the WSG into two components that have the weakest interactions between them, and hence, a mincut indicates the point in the protein where unfolding is likely to occur. Through a series of mincuts on the WSG, we predict the most likely sequence of unfolding events. Reversing the unfolding steps yields plausible pathways for protein folding.

```
//$G$ is a graph with weight function $W$
Unfold ($G = (V, E), W : E \rightarrow \Re^+$):
    $C$ = NOI-MinCut(G,W);
    $G_C = (C, E_C); G_{\overline{C}} = (\overline{C}, E_{\overline{C}})$;
    if ($|C| > 1$) Unfold($G_C, W$);
    if ($|\overline{C}| > 1$) Unfold($G_{\overline{C}}, W$);
```

Fig. 2. The Unfold algorithm.

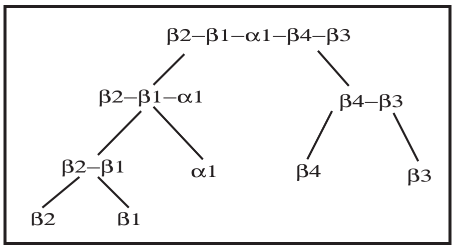


Fig. 3. Unfold 2IGD.

Figure 2 shows the pseudo-code for the Unfold algorithm to determine the folding pathway for a given protein. An unfolding event according to our model is a set of vertices that form a mincut in the WSG $G = (V, E)$ for a protein. In Unfold, first, a mincut C for the initial WSG is determined by the Nagamochi–Ono–Ibaraki (NOI-MinCut) (Nagamochi *et al.*, 1994) deterministic polynomial-time mincut algorithm[3], which is one of the fastest current methods, running in time $O(|V||E| + |V|^2 \log |V|)$. This gives the first event in the unfolding process. The input graph is then partitioned into two disjoint subgraphs, $G_C = (C, E_C)$ and $G_{\overline{C}} = (\overline{C}, E_{\overline{C}})$, where $E_C = \{(u, v) \in E \mid u, v \in C\}$ ($E_{\overline{C}}$ is defined similarly). We recursively process each subgraph to yield a sequence of mincuts, corresponding to the unfolding events. This sequence when reversed produces our prediction for the folding pathway for the given protein.

As an example of how Unfold works, consider again protein 2IGD. Given the WSG for 2IGD in Figure 1, NOI-MinCut determines $C = \{\beta_1\alpha_1\beta_2\}$ and $\overline{C} = \{\beta_4, \beta_3\}$ to be the mincut with capacity $W(C) = 11 + 14 = 25$. After recursive processing Unfold produces a sequence of mincuts which can be easily visualized as a tree shown in Figure 3. Here each node represents a set of vertices comprising a graph obtained in the recursive application of Unfold, and the children of a node are the partitions resulting from the mincut. For example, the node $\beta_2\beta_1\alpha_1$ is partitioned into $\beta_2\beta_1$ and α_1. If we proceed

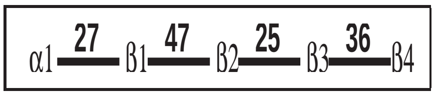
[3]NOI-MinCut breaks ties among mincuts arbitrarily.

from the leaf nodes of the tree to the root, we obtain the predicted folding pathway of 2IGD. We find that SSEs β_2 and β_1 fold to form anti-parallel β-sheet. Simultaneously SSEs β_3 and β_4 may also form an anti-parallel β-sheet. SSE α_1 then forms a $\beta_2\alpha_1\beta_1$ arrangement, and then the whole protein comes together by forming a parallel β-sheet between β_2 and β_3. We should be careful not to impose a strict linear timeline on the unfolding events predicted by Unfold; rather allowance should be made for several folding events to take place simultaneously. However, there may be intermediate stages that must happen before higher order folding can take place. We show that our approach is particularly suited to provide insights into such intermediate folding states.

4.2 The MultiUnfold algorithm

The Unfold algorithm finds only one plausible folding pathway, by always picking, at each recursive stage, only one (arbitrary) mincut, out of possibly several mincuts of the same capacity. We would like to enumerate several possible unfolding sequences of the same capacity. Moreover, we may also want to consider cuts with capacity close to the global mincut. More formally, let $W(C_{\min})$ denote a mincut of graph G; a near-mincut is a cut with capacity $W(C) \geq (1+\epsilon)W(C_{\min})$, where $\epsilon \geq 0$ is small non-negative margin of tolerance.

Our approach to find near-mincuts uses, as a substep, the efficient Gomory–Hu (GH) (Gomory and Hu, 1961) algorithm to determine the cut-tree **T** for a weighted graph G, which takes time $O(|V|^2|E|)$. The cut-tree for 2IGD WSG obtained by GH algorithm is shown below:



Note that by definition of cut-tree, $C = \{\alpha_1\beta_1\beta_2\}$ is a mincut for G with capacity $W(C) = 25$. If we set $\epsilon = 0.1$, then $C' = \{\alpha_1\}$ is a near-mincut, since its capacity $W(C') = 27 \leq (1+\epsilon)W(C) = (1+0.1)25 = 27.5$.

Once the cut-tree **T** is obtained for a graph G, it is fairly easy to enumerate possible near-mincuts. In a naive method, we first determine the mincut by finding the lowest weight (w) edge in the cut-tree **T** . Unlike Unfold which picks only one mincut, we now pick each edge $e \in \mathbf{T}$ such that $W(e) \leq w(1+\epsilon)$, and partition the graph into two subgraphs based on each resulting cut C. By recursively processing each new subgraph we can enumerate all possible mincuts, and we can also obtain the frequency of each cut, i.e. how often does a given set of vertices (SSEs) lie on some pathway. However, this naive approach is rather expensive, since we may have to repeatedly process the same cut over and over again, each time it appears on a pathway, leading to combinatorial explosion.

```
//$G$ is a graph with weight function $W$
MultiUnfold $(G, W)$:
1.  $Q = \{G\}$;
2.  while $Q \neq \emptyset$
3.      $G'(V, E)$ = remove from front of $Q$;
4.      if $\mathbf{M}(G')$ exists, skip lines 5-12
5.      $\mathbf{T}(V', E')$ = GH-CutTree(G',W);
6.      $w = \min\{W(e) | e \in E'\}$;
7.      for all $e \in E'$ with $W(e) \leq w(1+\epsilon)$
8.          $S$ = connected component
              of $\mathbf{T} - \{e\}$ containing $s$;
9.          $\overline{S} = V - S$;
10.           $G_S = (S, E_S); G_{\overline{S}} = (\overline{S}, E_{\overline{S}})$;
11.           $\mathbf{M}(G') = \mathbf{M}(G') \cup \{(G_S, G_{\overline{S}})\}$;
12.           $Q = Q \cup \{G_S, G_{\overline{S}}\}$;
13.   $\mathbf{F}(G)$ = ComputePathways($\mathbf{M}$);
```

Fig. 4. The MultiUnfold algorithm.

MultiUnfold adopts a more efficient approach. The basic idea is to first determine the structure of the recursive near-mincuts, processing each subgraph only once, since a given (sub)graph always produces the same near-mincuts for a given ϵ. The number of pathways and the frequency for each graph can then be determined using the near-mincut graph. Figure 4 shows the pseudo-code for MultiUnfold.

We first initialize a queue Q of graphs to be processed with the original graph G (line 1). We then process Q until it is empty. We remove each graph G' from Q (line 3), and check if we have already determined its near-mincuts [stored in $\mathbf{M}(G')$]. If so, we move on to the next graph in Q. If not, we compute the cut-tree **T** for G' and process all the near-mincuts (line 7). For each near-mincut S (lines 8–9), we partition G' into G_S and $G_{\overline{S}}$ (line 10) and store each of such pairs in $\mathbf{M}(G')$ (line 11) and also add them to Q (line 12). Once the near-mincut graph **M** has been determined by processing every graph in Q, we call ComputePathways (line 13) to find the frequency of each cut among all the pathways. Note that the for loop on line 7 is linear in the number of distinct near-mincuts.

As an example, consider the WSG for 2IGD (Fig. 1), with $\epsilon = 0.5$, allowing a near-mincut to have capacity at most $1.5 \times 25 = 37.5$. Thus in addition to $C = \{\alpha_1\beta_1\beta_2\}$ with $W(C) = 25$ and $C' = \{\alpha_1\}$ with $W(C') = 27$, we also get $C'' = \{\beta_4\}$ with $W(C'') = 36$ as a possible near-mincut. Thus $\mathbf{M}(G) = \{(\alpha_1\beta_1\beta_2, \beta_3\beta_4), (\alpha_1, \beta_1\beta_2\beta_3\beta_4), (\alpha_1\beta_1\beta_2\beta_3, \beta_4)\}$. Likewise, it is easy to see that for $G'' = G_{\overline{C''}}$, $\mathbf{M}(G'') = \{(\alpha_1\beta_1\beta_2, \beta_3), (\alpha_1, \beta_1\beta_2\beta_3)\}$, and so on. The complete near-mincut graph **M** is shown in Figure 5 (a pair of edges with same type denotes a pair of near-mincuts).

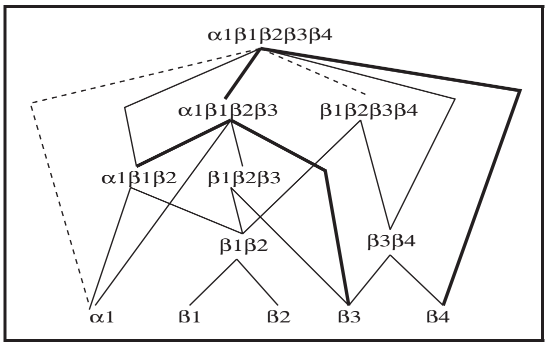


Fig. 5. Near-mincut graph for 2IGD.

```
ComputePathways ($G, \mathbf{M}$):
  //Compute Number of Pathways
1.  for all $G'(V', E') \in \mathbf{M}$ from leaves to root
2.    if $|V'| = 1$ then $N(G') = 1$;
3.    else for all $(G'_1, G'_2) \in \mathbf{M}(G')$
4.      $N(G') = N(G') + N(G'_1) \times N(G'_2)$;
  //Compute Frequency of Each Graph
5.  for all $G'(V', E') \in \mathbf{M}$ from leaves to root
6.    $\mathbf{F}(G') = \{(G', N(G'))\}$;
7.    for all $(G'_1, G'_2) \in \mathbf{M}(G')$
8.      $\mathbf{F}(G') = \mathbf{F}(G') \cup \mathbf{F}(G'_1) \times N(G'_2)$;
9.      $\mathbf{F}(G') = \mathbf{F}(G') \cup \mathbf{F}(G'_2) \times N(G'_1)$;
10. return $\mathbf{F}(G)$;
```

Fig. 6. Computing the pathways.

Once the structure of the near-mincuts has been stored in graph **M**, ComputePathways processes this graph bottom-up from leaves to the root, as shown in Figure 6. We first compute the number of pathways each node contributes. For any leaf, there is only one pathway it can contribute (line 2). For a non-leaf node, the number of pathways is given as $N(G') = \sum_{(C,\overline{C})\in \mathbf{M}(G')} N(G_C) \times N(G_{\overline{C}})$ (lines 3–4). For example, in Figure 5, node $C = \alpha_1\beta_1\beta_2\beta_3$ has $N(G_C) = 1\times1+1\times1 = 2$, while for the original graph G, we have $N(G) = 1 \times 1 + 2 \times 1 + 1 \times 1 = 4$. The complexity of this step is $O(|\mathbf{M}| \times f)$, where f is the maximum cardinality of $\mathbf{M}(G')$ over all G'.

Next, we compute the frequency of each node $F(G')$ by summing up the frequencies from all of its near-mincut pairs, scaled by the number of pathways the pair contributes (lines 8 and 9), starting with the leaves[4]. The root node then contains the final frequency of all the cuts (line 10). Consider the near-mincut graph for 2IGD shown in Figure 5. The intermediate $F(G')$ at each node is given in Table 1; e.g. $F(G) = \{\alpha_1\beta_1\beta_2\beta_3\beta_4(4)\} \cup \{F(\alpha_1) \cup F(\beta_1\beta_2\beta_3\beta_4)\} \cup \{F(\alpha_1\beta_1\beta_2) \cup F(\beta_3\beta_4)\} \cup \{F(\alpha_1\beta_1\beta_2\beta_3) \cup 2 \times F(\beta_4)\}$. The complexity of this step is $O(|\mathbf{M}|^2 \times f)$ in the worst case.

[4] Note that $N \times F(G')$ means we multiply the counts of each cut in $F(G)$ by N.

Table 1. Frequencies of near-minCuts.

G'	$F(G')$
$\beta_1\beta_2$	$\beta_1\beta_2, \beta_1, \beta_2$
$\beta_3\beta_4$	$\beta_3\beta_4, \beta_3, \beta_4$
$\alpha_1\beta_1\beta_2$	$\alpha_1\beta_1\beta_2, \alpha_1, \beta_1\beta_2, \beta_1, \beta_2$
$\beta_1\beta_2\beta_3$	$\beta_1\beta_2\beta_3, \beta_1\beta_2, \beta_1, \beta_2, \beta_3$
$\beta_1\beta_2\beta_3\beta_4$	$\beta_1\beta_2\beta_3\beta_4, \beta_1\beta_2, \beta_1, \beta_2, \beta_3\beta_4, \beta_3, \beta_4$
$\alpha_1\beta_1\beta_2\beta_3$	$\alpha_1\beta_1\beta_2\beta_3(2), \alpha_1\beta_1\beta_2, \beta_1\beta_2\beta_3, \alpha_1(2), \beta_1\beta_2(2), \beta_1(2), \beta_2(2), \beta_3(2)$
$\alpha_1\beta_1\beta_2\beta_3\beta_4$	$\alpha_1\beta_1\beta_2\beta_3\beta_4(4), \alpha_1\beta_1\beta_2\beta_3(2), \alpha_1\beta_1\beta_2(2), \beta_1\beta_2\beta_3\beta_4, \beta_1\beta_2\beta_3, \alpha_1(4), \beta_1\beta_2(4), \beta_1(4), \beta_2(4), \beta_3\beta_4(2), \beta_3(4), \beta_4(4)$

4.2.1 Ranking near-minCuts Once we obtain the frequency with which a graph appears over all the possible pathways, we need a way to rank the cuts. Note that a ranking by frequency is not satisfactory; for instance, the original graph G must appear on each pathway (since it is always the starting point of MultiUnfold), and each individual SSE must also always be on each pathway (since a sequence of mincuts ends only when each graph contains a single vertex). Thus frequency alone is not sufficient to rank the graphs. A better method is to rank inversely proportional to the expected probability of occurrence of a graph G'. Since the full graph and each vertex always occur on every pathway their probability of occurrence is 1.0, and they will be ranked low. However, it is not easy to analytically compute the probability of each graph. We adopt the method of comparing G with a new graph, $U_G = (V, U_E)$ with the same edge connectivity as G, but with uniform edge weights [i.e. $W(e) = 1, \forall e \in U_E$]. Let $f(G')$ and $f(U_{G'})$ be the relative frequency (defined as ratio of the frequency of G' divided by the total number of pathways) of graph G' obtained by running MultiUnfold on G and U_G, respectively. Then we rank each graph in the decreasing order of $f(G')/f(U_{G'})$.

4.3 Detailed example: Dihydrofolate Reductase

Although no one has determined the precise order of appearance of secondary structures for any protein, there is evidence that supports intermediate stages in the pathway for several well-studied proteins, including specifically for the protein Dihydrofolate Reductase (PDB 4DFR; 159 residues), a two-domain α/β enzyme that maintains pools of tetrahydrofolate used in nucleotide metabolism (Heidary *et al.*, 2000; Clementi *et al.*, 2000).

Experimental data indicate that the adenine-binding domain, which encompasses the two tryptophans Trp-47 and Trp-74, is folded, and is an intermediate essential in the folding of 4DFR, and happens early in the folding

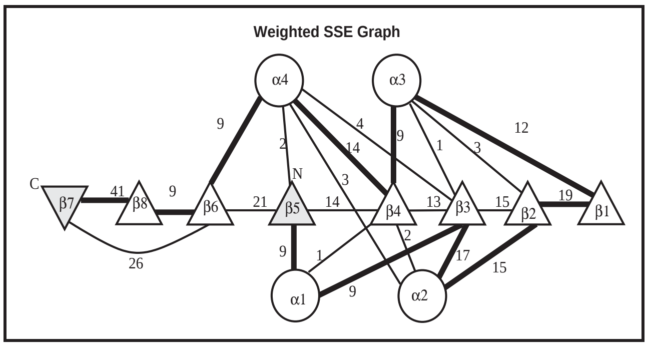


Fig. 7. WSG Dihydrofolate Reductase (4DFR).

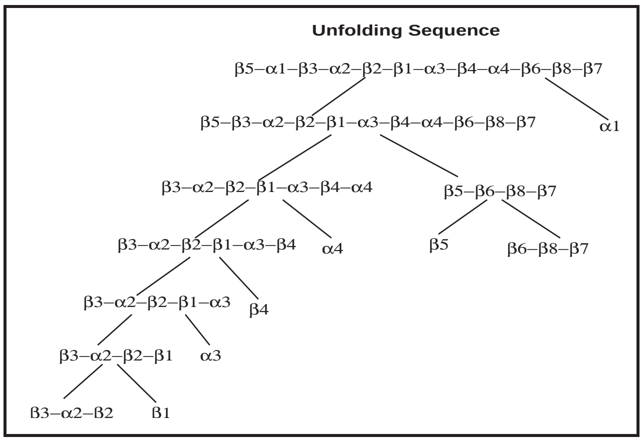


Fig. 8. 4DFR: unfolding sequence.

(Heidary *et al.*, 2000). Figures 7–11, shows the WSG, unfolding sequence, and a series of intermediate stages in the folding pathway of protein 4DFR. Trp-47 and Trp-74 lie in SSEs α_2 and β_1, respectively. According to our mincut-based Unfold algorithm, the vertex set $\{\beta_2, \alpha_2, \beta_3, \beta_1\}$ lies on the folding pathway, in agreement with the experimental results!

We can see from Figure 7, that 4DFR has four α-helices and eight β-strands. The WSG shows the interactions weights among the different SSEs (the bold lines indicate the backbone). Applying Unfold to 4DFR yields the sequence of cuts shown (Fig. 8). For clarity the unfolding sequence tree has been stopped when there are no more than three SSEs in any given node. The remaining illustrations show some selected intermediate stages on the folding pathway by reversing the unfolding sequence.

We find that SSE group $\beta_2\alpha_2\beta_3$ and $\beta_6, \beta_8, \beta_7$ are among the first to fold (Fig. 9), suggesting that they might be the folding initiation sites. Next β_1 joins $\beta_2\alpha_2\beta_3$, in agreement with the experimental results (Heidary *et al.*, 2000), as shown in Figure 10; the Trp-47 and Trp-74 interaction is also shown, and the other group now becomes $\beta_5, \beta_6, \beta_8, \beta_7$. The final native structure including $\alpha_3\beta_4\alpha_4$ and α_1 is shown in Figure 11.

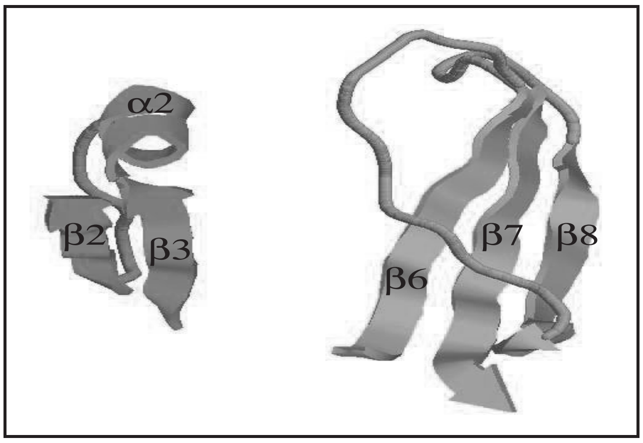


Fig. 9. 4DFR: early stages in the folding pathway.

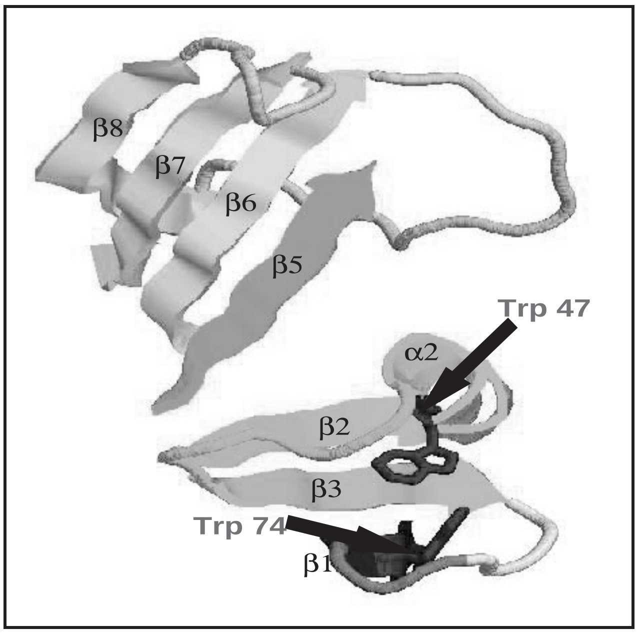


Fig. 10. 4DFR: intermediate stages in the folding pathway.

We again underscore that the results should not be taken to imply a strict folding timeline, but rather as a way to understand major events that are mandatory in the folding pathway. One such experimentally verified case is the $\{\beta_2, \alpha_2, \beta_3, \beta_1\}$ group that is known to fold early, and our approach was able to predict that.

5 PATHWAYS FOR OTHER PROTEINS

To establish the utility of our methodology we predict the folding pathway for several proteins for which there are known intermediate stages in the folding pathway.

Bovine Pancreatic Trypsin Inhibitor (PDB 6PTI; 58 residues) is a small protein containing 2 α-helices and 2

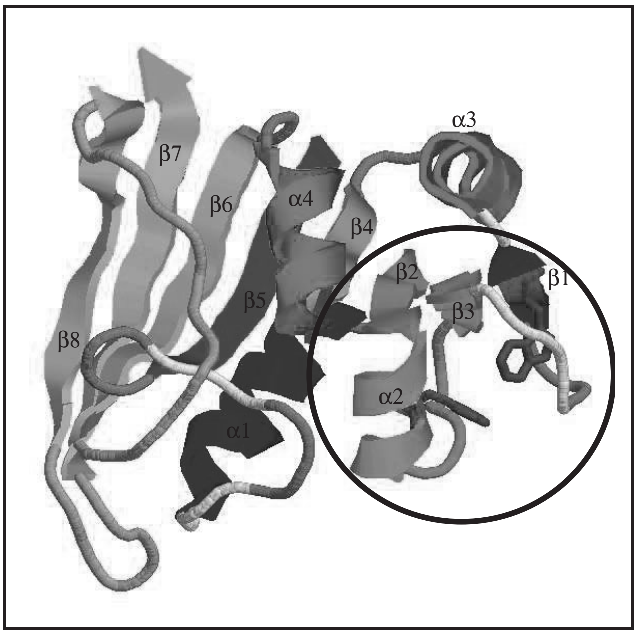


Fig. 11. 4DFR: final stages and native structure of the folding pathway.

β-strands (Kazmirski and Daggett, 1998). It is known that the unfolding pathway of this protein involves the loss of the helix structure followed by the beta structure. We found that indeed $\beta_2\beta_3$ remain together until the end.

Chymotrypsin Inhibitor 2 (PDB 2CI2; 83 residues) is also a small protein with 1 helix and 4 strands, arranged in sequence as follows $\beta_1\alpha_1\beta_4\beta_3\beta_2$. Previous experimental and simulation studies have suggested an early displacement of β_1, and a key event in the disruption of the hydrophobic core formed primarily by α_1 and the strands β_3 and β_4 (Lazardis and Karplus, 1997). Our approach predicts that β_1 is the first to go, while $\beta_3\beta_4$ remain intact until the end.

The activation domain of Human Procarboxypeptidase A2 (PDB 1O6X) has 81 residues, with 2 α-helice and 3 β-strands arranged as follows $\beta_2\alpha_1\beta_1\alpha_2\beta_3$. The folding nucleus of 1O6X is made by packing of α_2 with $\beta_2\beta_1$ (Villegas *et al.*, 1998). We found that the unfolding sequence indeed retains $\beta_2\beta_1\alpha_2$ and then finally $\beta_2\beta_1$.

The pathway of cell-cycle protein p13suc1 (PDB 1SCE; 112 residues) shows the stability of $\beta_2\beta_4$ interaction even though β_4 is the strand involved in domain swapping (Alonso *et al.*, 2000). 1SCE has four domains, with seven SSEs (three α and four β). β_{4C} of domain C interacts with β_2 of domain A, and vice versa (the same is true for domains B and D). We found that $\beta_1\beta_2\beta_{4C}$ is the last to unfold.

β-Lactoglobulin (PDB 1CJ5; 162 residues) contains 10 strands and 3 helices. Beta strands F, G and H are formed immediately once the refolding starts (Kuwata *et al.*, 2001), which was thus identified as the folding core of 1CJ5. In the predicted unfolding sequence obtained for 1DV9, we found that the SSEs $\beta_8, \beta_9, \beta_{10}$ corresponding to the F,G and H beta strands remain together till the last stages of unfolding.

Interleukin-1β (PDB 1I1B; 153 residues) is an all-β protein with 12 β-strands. Experiments indicate that strands $\beta_6\beta_7\beta_8$ are well folded in the intermediate state and $\beta_4\beta_5$ are partially formed (Clementi *et al.*, 2000). We found $\beta_4\beta_5$ and $\beta_8\beta_9$ to be among the last unfolding units, including other pairs.

Myoglobin (PDB 1MBC; from sperm whale; 153 residues) and Leghemoglobin (PDB 1BIN; from Soybean; 143 residues), both belonging to the globin family of heme binding proteins, share a rather low sequence similarity, but share highly similar structure. Both are all-α proteins with eight helices, denoted $\alpha_1(A)\alpha_2(B)\alpha_3(C)\alpha_4(D)\alpha_5(E)\alpha_6(F)\alpha_7(G)$-$\alpha_8(H)$. Nishimura *et al.* (2000) observed that the main similarity of their folding pathways is in the stabilization of the G and H helices in the burst phase folding intermediates. However, the details of the folding pathways are different. In 1MBC intermediate additional stabilizing interactions come from helices A and B, while in 1BIN they come form part of E helix. Running Unfold on 1MBC indeed finds that $\alpha_2(B)\alpha_3(C)\alpha_7(G)\alpha_8(H)$ remain together until the very last. For 1BIN, we found a pathway passing through $\alpha_2(B)\alpha_5(E)\alpha_7(G)\alpha_8(H)$. Unfold was thus able to detect the similarity in the folding pathways, as well as the details. We also ran MultiUnfold with $\epsilon = 0.1$, and the results confirmed that in 1MBC $\alpha_2(B)\alpha_7(G)\alpha_8(H)$ never occurs in the uniform graph, and for 1BIN, $\alpha_5(E)\alpha_7(G)\alpha_8(H)$ never occurs in the uniform graph. We also found that $\alpha_5(E)$ showed interactions with $\alpha_7(G)\alpha_8(H)$ in 1BIN, but never for 1MBC.

Protein Acylphosphatase (PDB 2ACY; 98 residues), with two α and five β SSEs ($\beta_2\alpha_1\beta_4\beta_3\alpha_2\beta_1\beta_5$), displays a transition state ensemble with a marked tendency for the β-sheets to be present, particularly β_3 and β_4, and while α_2 is present, it is highly disordered relative to rest of the structure (Vendruscolo *et al.*, 2001). We found that $\beta_2\beta_4\beta_3\alpha_2\beta_1$ form an intermediate in the unfolding sequence.

Twitchin Immunoglobulin superfamily domain protein (PDB 1WIT; 93 residues) has a β-sandwich consisting of nine β-strands, and one very small helix. The folding nucleus consists of residues in the structural core $\beta_3\beta_4\beta_7\beta_9\beta_{10}$ centered around β_3 and β_9 on opposite sheets (Clarke *et al.*, 1999). We found that the unfolding sequence passes through the intermediate $\beta_3\beta_4\beta_7\beta_9\beta_{10}$.

6 CONCLUSIONS

In this paper, we developed automated techniques to predict protein folding pathways. We construct a weighted SSE graph for a protein, where each vertex is an SSE, and each edge represents the strength of interaction between two SSEs. We use a repeated mincut approach (via the Unfold algorithm) on the WSG graph to discover strongly inter-related groups of SSEs and we then predict an (approximate) order of appearance of SSEs along the folding pathway. We also proposed MultiUnfold algorithm to find many possible folding pathways, which also ranks each intermediate inversely proportional to its expected frequency.

We proposed three methods for calculating the interaction strength between SSEs: contact, distance and SAS based. The SAS-based approach incorporates the solvation free energy of the molecules. We found that while the three approaches give similar results, the SAS-based approach seems to agree more with the known pathways.

Currently, we consider interactions only among the α-helices and β-strands. In the future we also plan to incorporate the loop regions in the WSG, and see what effect it has on the folding pathway. Furthermore, we plan to test our folding pathways on the entire collection of proteins in the PDB. We would like to study different proteins from the same family and see if our methods predict consistent pathways; both similarities and dissimilarities may be of interest. We also plan to make our software available online so other researchers may first try the pathways predictions before embarking on time-consuming experiments and simulations.

ACKNOWLEDGEMENTS

This work was supported in part by NSF Career Award IIS-0092978, DOE Career Award DE-FG02-02ER25538 and NSF grant EIA-0103708.

REFERENCES

Alonso,D., Alm,E. and Daggett,V. (2000) The unfolding pathway of the cell cycle protein p13suc1: implications for domain swapping. *Structure*, **8**, 101–110.

Bystroff,C. (2002) Masker: improved solvent excluded molecular surface area estimations using boolean masks. *Protein Eng.*, **15**, 959–965.

Chikenji,G. and Kikuchi,M. (2000) What is the role of non-native intermediates of beta-lactoglobulin in protein folding? *Proc. Natl Acad. Sci., USA*, **97**, 14273–14277.

Clarke,J., Cota,E., Fowler,S.B. and Hamill,S.J. (1999) Folding studies of immunoglobulin-like β-sandwich proteins suggest that they share a common folding pathway. *Structure*, **7**, 1145–1153.

Clementi,C., Jennings,P.A. and Onuchic,J.N. (2000) How native-state topology affects the folding of dihydrofolate reductase and interleukin-1beta. *Proc. Natl Acad. Sci., USA*, **97**, 5871–5876.

Colon,W. and Roder,H. (1996) Kinetic intermediates in the formation of the cytochrome *c* molten globule. *Nat. Struct. Biol.*, **3**, 1019–1025.

Dill,K.A. (1990) Dominant forces in protein folding. *Biochemistry*, **29**, 7133–7155.

Dobson,C.M. (2003) Protein folding and misfolding. *Nature*, **426**, 884–890.

Fersht,A.R. and Daggett,V. (2002) Protein folding and unfolding at atomic resolution. *Cell*, **108**, 573–582.

Gilbert,D.R., Westhead,D.R., Nagano,N. and Thornton,J.M. (1999) Motif-based searching in tops protein topology databases. *Bioinformatics*, **5**, 317–326.

Gomory,R.E. and Hu,T.C. (1961) Multi-terminal network flows. *SIAM J. Appl. Math.*, **9**, 551–570.

Heidary,D.K., O'Neill,J.C., Jr, Roy,M. and Jennings,P.A. (2000) An essential intermediate in the folding of dihydrofolate reductase. *Proc. Natl Acad. Sci., USA*, **97**, 5866–5870.

Kazmirski,S.L. and Daggett,V. (1998) Simulations of the structural and dynamical properties of denatured proteins: the molten coil state of bovine pancreatic trypsin inhibitor. *J. Mol. Biol.*, **277**, 487–506.

Klimov,D.K. and Thirumalai,D. (2001) Multiple protein folding nuclei and the transition state ensemble in two-state proteins. *Proteins*, **43**, 465–475.

Kuwata,K., Shastry,R., Cheng,H., Hoshino,M., Bhatt,C.A., Goto,Y. and Roder,H. (2001) Structural and kinetic characterization of early folding events of lactoglobulin. *Nature*, **8**, 151–155.

Lazardis,T. and Karplus,M. (1997) New view of protein folding reconciled with the old through multiple unfolding simulations. *Science*, **278**, 1928–1931.

Mok,Y., Kay,C., Kay,L. and Forman-Kay,J. (1999) NOE data demonstrating a compact unfolded state for an sh3 domain under non-denaturing conditions. *J. Mol. Biol.*, **289**, 619–638.

Nagamochi,H., Ono,T. and Ibaraki,T. (1994) Implementing an efficient minimum capacity cut algorithm. *Math. Programm.*, **67**, 325–341.

Nishimura,C., Prytulla,S., Dyson,H.J. and Wright,P.E. (2000) Conservation of folding pathways in evolutionary distant globin sequences. *Nat. Struct. Biol.*, **7**, 679–686.

Nolting,B., Golbik,R., Neira,J., Soler-Gonzalez,A., Schreiber,G. and Fersht,A. (1997) The folding pathway of a protein at high resolution from microseconds to seconds. *Proc. Natl Acad. Sci., USA*, **94**, 826-830.

Song,G. and Amato,N.M. (2002) Using motion planning to study protein folding pathways. *J. Comput. Biol.*, **9**, 149–168.

Vendruscolo,M., Paci,E., Dobson,C.M. and Karplus,M. (2001) Three key residues form a critical contact network in a protein folding transition state. *Nature*, **409**, 641–645.

Villegas,V., Martinez,J.C., Aviles,F.X. and Serrano,L. (1998) Structure of the transition state in the folding process of human procarboxypeptidase A2 activation domain. *J. Mol. Biol.*, **283**, 1027–1036.

Westhead,D.R., Slidel,T.W.F., Flores,T.P.J. and Thornton,J.M. (1999) Protein structural topology: automated analysis, diagrammatic representation and database searching. *Protein Sci.*, **8**, 897–904.