

ENHANCED PARTIAL ORDER CURVE COMPARISON OVER MULTIPLE PROTEIN FOLDING TRAJECTORIES

Hong Sun^{*}, Hakan Ferhatosmanoglu, Motonori Ota[†], Yusu Wang

Department of Computer Science and Engineering

The Ohio State University, Columbus, OH 43210

[†] *Global Scientific Information and Computing Center*

Tokyo Institute of Technology, O-okayama, Meguro-ku, Tokyo 152-8550, Japan

Email: sunh,hakan,yusu@cse.ohio-state.edu, [†]mota@gsic.titech.ac.jp

Understanding how proteins fold is essential to our quest in discovering how life works at the molecular level. Current computation power enables researchers to produce a huge amount of folding simulation data. Hence there is a pressing need to be able to interpret and identify novel folding features from them. In this paper, we model each folding trajectory as a multi-dimensional curve. We then develop an effective multiple curve comparison (MCC) algorithm, called the *enhanced partial order (EPO)* algorithm, to extract features from a set of *diverse* folding trajectories, including both successful and unsuccessful simulation runs. Our EPO algorithm addresses several new challenges presented by comparing high dimensional curves coming from folding trajectories. A detailed case study of applying our algorithm to a miniprotein Trp-cage²⁴ demonstrates that our algorithm can detect similarities at rather low level, and extract biologically meaningful folding events.

keywords: EPO, trajectory, alignment, protein folding, high dimension.

1. INTRODUCTION

Proteins are the main agents in cells. From a chemical point of view, a protein molecule is a linear sequence of amino acids. This linear sequence, under appropriate physicochemical conditions, folds into a unique native structure rapidly. Understanding folding process is of paramount importance, especially since the outcome of it, namely the three dimensional protein structure, to a large extent decides the functionality of the molecule. Hence a lot of research has been devoted to investigating the kinetics of protein folding. In particular, modern (parallel) computation power makes it possible to perform large-scale folding simulations. As a result, interpreting the huge amount of simulation data obtained becomes a crucial issue.

Given the highly stochastic nature of the protein motion, the study of protein fold usually relies on an ensemble of folding simulations including both *successful* and *unsuccessful* runs, which are trajectories that *do* or *do not* include a sequence of conformations leading to a near native conformation. Given such a diverse data set, scientists wish to answer questions such as what causes the folding process falling into different results, and what common properties are

shared by the successful runs, but not the unsuccessful ones? To this end, it is highly desirable to be able to compare multiple folding trajectories and extract useful information from them.

In this paper, we model each protein folding trajectory as a high dimensional curve, and then present a novel multiple-curve comparison (MCC) algorithm to identify critical information from a set of trajectory curves in an *automatic* manner. In particular, we focus on the geometry of protein chain conformations throughout the folding process, and convert each conformation into a high dimensional point. The goal is to extract lists of *ordered events* common to successful runs but not to unsuccessful ones, such as discovering that a conformation *B* is always formed after *A* and followed by a conformation *C* before reaching a successful folding conformation. (Conformations *A*, *B*, and *C* may not be consecutive.) To this end, we develop an effective new multiple curves comparison algorithm called the *enhanced partial order (EPO)* algorithm, to capture similarities and dis-similarities between a set of input folding trajectories. The EPO algorithm is developed over the concept of POA (partial order alignment) algorithm^{9, 17}, but is greatly improved and extended

^{*}Corresponding author.

in several aspects, especially in its sensitivity in detecting low level of similarity and its capability of handling high dimensional curves. Applying it to the folding trajectories of a miniprotein Trp-cage²⁴ shows that our algorithm is able to *automatically* detect important critical folding events which are observed earlier²⁸ by biological methods. We remark that the EPO algorithm is general, and can be applied to multiple protein structure comparisons as well.

2. RELATED AND NEW WORK

2.1. Related Work

Previously, folding simulations analysis is performed mainly for testing various protein folding models^{2, 18, 33}, such as the folding pathway model and the funnel model; and/or for studying energetic aspects of folding kinetics^{1, 4, 5, 19}. The geometric shapes of the conformations involved in folding trajectories have not been widely explored^{6, 14, 28}, despite their important role in folding. A particularly interesting work in this direction is by Ota *et al.*²⁸, where they provide a quite detailed study of the folding trajectories of a mini-protein Trp-cage using phylogenetic tree combined with expert knowledge. However in general, an *automatic* tool to facilitate the folding simulations analysis at large scales is still missing. This paper provides an important step towards this goal by modeling folding trajectories as curves and using a new multiple curve comparison (MCC) algorithm to detect critical folding events.

The closest relative of our MCC problem in computational biology is the multiple structure alignment (MSTA) problem, which aims at aligning a family of protein structures, each modeled as a three dimensional polygonal curve to represent its backbone.

MSTA is a very hard problem. In fact, even the pairwise comparison problem of aligning two structures A and B is believed to be NP-hard since one has to optimize *simultaneously* both the *correspondence* between A and B and the *relative transformation* of one structure with respect to the other. Numerous heuristic-based algorithms have been developed in practice for this fundamental

problem^{7, 8, 11, 12, 16, 30, 32}. If we have a set of $k > 0$ structures, then even the problem of aligning them optimally *without* considering transformations becomes intractable — it takes $\Omega(n^k)$ time using the standard dynamic programming algorithm, where n is the size of each protein involved.

In practice, progressive methods are widely used to attack the MSTA problem²¹. For example, given a set of structures, many approaches start with a seed structure and then progressively align the remaining structures onto it one by one^{3, 10, 20, 25, 26, 29, 35}. A *consensus or core structure* is typically built throughout, to maintain the common substructures among the proteins that are already aligned. At each round, only pairwise structure comparison is usually performed to align the current consensus with a new structure.

Obviously, the above progressive MSTA framework is a greedy approach. Its performance depends on the underlying pairwise comparison methods used, the order of structures that are progressively aligned, as well as the consensus structure maintained. Various heuristics have been exploited to find a good order for the progressive alignments. Note that this order can also be guided by a tree instead of a linear sequence, which removes the need of choosing a seed structure. The progressive procedure may also be iterated several times to locally refine the multiple structure alignments.

2.2. Our Results

There are two main differences between the MCC problem we are interested in and the traditional MSTA problem. In the case of protein structures, it is usually explicitly or implicitly assumed that the (majority of the) input proteins belong to one family^a, or at least share some relations. As such, one can expect that some consensus of the family should exist. However in our case, the set of curves are from a set of simulations including both successful and unsuccessful runs, and we wish to classify this *diverse* set of curves, and capture common features *within as well as across* its sub-families. Secondly and *more importantly*, the level of similarity existing in these folding trajectories is usually much lower

^aHow to classify a set of input structures into different families is a related problem, and many such classifications exist^{12, 22, 27}.

way, we map each trajectory of m conformations to a curve in \mathbb{R}^{n^2} with m vertices. In the remaining part of this paper, we use the terms trajectories and curves interchangeably.

3.1. Notations and Algorithm Overview

Before we formally define the MCC problem, we introduce some necessary notations. Given a set of elements $V = \{v_1, \dots, v_l\}$, a relation \prec over V is *transitive* if $v_i \prec v_j$ and $v_j \prec v_k$ imply that $v_i \prec v_k$. In this paper, we also refer to $v_i \prec v_j$ as a *partial order constraint*. A *partial order graph* (POG) $G = (V, E)$ is a directed acyclic graph with $V = \{v_1, \dots, v_l\}$, where $v_i \prec v_j$ if there is an edge (v_i, v_j) . Note that by the transitivity of this relation, two nodes may have a partial order constraint even when there is no edge between them in G . Let R be the set of partial order constraints induced by G . We say that V is a *partial order list w.r.t. G* if for any $v_i \prec v_j \in R$, we have that $i < j$. In other words, the linear order in V is a *total order* satisfying all partial order constraints induced from G . See Figure 2 for an example.

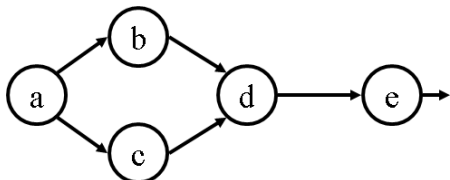


Fig. 2. A POG G of 5 nodes. Note that there is a partial order constraint $a \prec d$ even though there is no edge between them. Both $\{a, b, c, d, e\}$ and $\{a, c, b, d, e\}$ are valid partial order lists w.r.t. G .

Let $\mathcal{T} = \{T_1, \dots, T_N\}$ be a set of N trajectories in \mathbb{R}^d , where each trajectory T_i is an ordered sequence of n points p_1^i, \dots, p_n^i ^c. The goal of the MCC algorithm is to find aligned sub-sequences from \mathcal{T} .

More formally, an *aligned node* o is a collection of vertices from T_i s, with **at most** one point from each T_i . Given a 3-tuple $(\mathcal{T}, \tau, \varepsilon)$, where τ and ε are input thresholds, an *alignment of \mathcal{T}* is a POG G with the corresponding set of partial order constraints R and a partial order list of aligned nodes $\mathcal{O} = \{o_1, \dots, o_L\}$ such that the following three criteria are satisfied:

- C1. $|o_k| \geq \tau$, for any $k \in [1, L]$;

- C2. for any $p_j^i, p_{j'}^{i'} \in o_k$, $\|p_j^i - p_{j'}^{i'}\| \leq \varepsilon$;
 C3. if $p_j^i \in o_{k_1}$ and $p_{j'}^{i'} \in o_{k_2}$ with $o_{k_1} \prec o_{k_2}$, then $j < j'$.

(C1) indicates that the number of vertices of input curves aligned to each aligned node is greater than a *size threshold* τ , and (C2) requires that these aligned points are tightly clustered together (i.e, the diameter is bounded by a *distance threshold* ε). (C3) enforces that points in different aligned nodes still maintain their partial order along their respective trajectory. Our goal is to maximize L , the size of such an alignment \mathcal{O} . See Figure 3 (b) for an example of an alignment graph.

Algorithm overview At a high level, the EPO algorithm has two stages (see Figure 3): (S1) initial POG construction stage and (S2) merging stage. The first stage generates an initial alignment for \mathcal{T} , encoded in a POG G . The procedure has the same framework as the POA algorithm, but its performance, especially when the similarity is low, is significantly improved, via the use of a clustering preprocessing step and a new two-level scoring function. In the second stage, we develop a novel and effective procedure to merge nodes from G to produce aligned nodes with large size, and output a better final alignment G^* . Below, we describe each stage in detail.

3.2. Initial POG Construction

Standard dynamic programming (DP)^{23, 31} is an ideal method for pairwise comparison between sequences. It produces the optimal alignment between two sequences with respect to a given scoring function. One can perform multiple sequences alignment progressively based on this DP pairwise comparison method. Roughly speaking, in the i th round of the algorithm, the alignment of the first $i-1$ sequences is represented in a consensus sequence. The algorithm then update this consensus by aligning it with the i th sequence S_i using the standard DP algorithm. Information from S_i that is not aligned to the consensus sequence is essentially lost. See Figure 1(a).

The partial order alignment (POA) algorithm¹⁷ greatly alleviates this problem by encoding the consensus in a POG instead of a linear sequence

^cFor simplicity, we assume without loss of generality that all T_i s have the same length n .

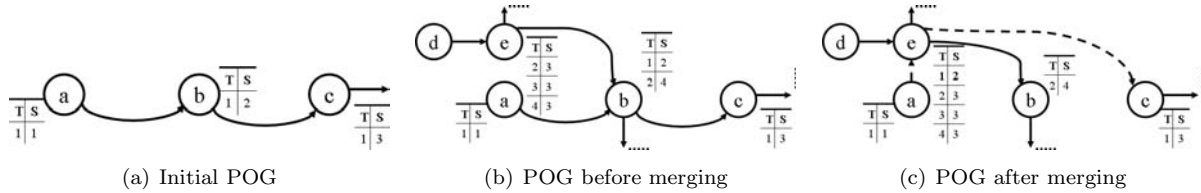


Fig. 3. Symbols inside the circles are the node IDs. The table associated with each node encodes the set of points aligned to it. In particular, each row represents a point with its trajectory ID (T column) and its index along the trajectory (S column). In (a), a POG is initialized by the trajectory T_1 . An example of a POG after aligning a few trajectories is shown in (b). Note that a new node/branch is created when a point cannot be aligned to any existing nodes. For example, node e is created when p_3^2 (i.e., the 3rd point of T_2) is inserted. (c) shows the POG after merging point p_2^2 from the node b to the node e constrained by the distance threshold ε .

(see Figure 1(b)). In particular, the alignment of S_1, \dots, S_{i-1} is encoded in a partial order graph G_i , which is then updated by aligning it with S_i . The alignment between G_i and S_i can still be achieved by a DP algorithm. The main difference is that in this DP procedure, to find the optimal score of aligning a node $v \in G_i$ and an element $s \in S_i$, one has to inspect the alignment between all parents of v in G_i and the parent of s in S_i . The POA algorithm reduces the influence of the order of the sequences aligned, and is able to capture alignments between a subset of sequences. More details of the POA algorithm can be found in ^{17, 9}.

In our case, each trajectory is mapped to an ordered sequence of points (i.e., a polygonal curve), and a similar algorithm can be applied to our trajectory data, where instead of the usual 1D sequences, we now have dD sequences^d. Below we explain two main differences between our EPO algorithm and the POA algorithm.

3.2.1. Size of POG

The first problem with current POA algorithm is that the size of the POG graph maintained expands quickly when the level of similarity is low. For example, suppose we are updating the current POG G_i to G_{i+1} by aligning it with a new curve T_i . If a point $p \in T_i$ cannot be aligned to any node in G_i , then it will create a new node in G_{i+1} , as this node may potentially be aligned later with the remaining curves. Consequently, if the similarity is sparse, many new nodes are created without producing significantly aligned nodes later and the size of the POG graph

increases rapidly. This induces high computational complexity.

To address this problem, our algorithm first preprocesses all points from the input curves \mathcal{T} by clustering them into groups¹³, the diameter of which is smaller than a user defined threshold, which is fixed as the distance threshold ε in our experiments. We keep only those clusters whose size is greater than a certain threshold ($\tau/2$ in our experiments), and collect their centers in $C = \{c_1, \dots, c_r\}$, which we refer to as the *set of canonical cluster centers*. Intuitively, C provides a synopsis of the input curves and represents potentially aligned nodes.

If, in the process of aligning T_i with G_i , a point $p \in T_i$ is not aligned to any node in G_i , then we insert a new node in G_{i+1} **only if** p is within ε away from some canonical center from C . If p is far from all the canonical cluster centers, then there is little chance that p can form significant alignment with points from later curves, as that would have implied that p should belong to a dense cluster. The set of canonical cluster centers will also contribute to the scoring function described below.

3.2.2. Scoring Function

The choice of the scoring function when aligning $G_i = (V_i, E_i)$ with T_i , is in general a crucial aspect of an alignment algorithm. Given a point $p \in T_i$ and a node $o \in G_i$, let $\delta(o, p)$ be the *similarity* between p and o , the definition of which will be described shortly. The *score* of aligning p with o is usually

^dSince each point corresponds to the distance map of a conformation, no transformation is needed when comparing such curves.

defined as:

$$\begin{aligned} \text{Score}(o, p) = & \\ \max \{ & \max_{(o', o) \in E_i} (\text{Score}(o', q) + \delta(o, p)), \\ & \max_{(o', o) \in E_i} \text{Score}(o', p), \text{Score}(o, q) \}, \end{aligned}$$

where q is the predecessor (i.e. parent) of the point p along T_i , and o' ranges over all predecessors of o in the POG G_i . It is easy to verify that such scores can be computed by a dynamic programming procedure due to the inherent order existing in both the trajectory and the POG.

A common way to define $\delta(o, p)$, the similarity between o and p , is as follows. Assume that each node o is associated with a *node center* $\omega(o)$ to represent all the points aligned to this node. Then

$$\delta(o, p) = \begin{cases} \varepsilon - \|p - \omega(o)\| & \text{if } \|p - \omega(o)\| < \varepsilon \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

An alternative way to view this is that each node o has an influence region of radius ε around its center. A point p can be aligned to a node o only if it lies within the influence region of o . In order to be able to align as many points as possible, intuitively, it is more desirable that the influence regions of nodes in current POG cover as much space as possible.

Natural choices for the node center $\omega(o)$ of o include using a canonical cluster center computed earlier, or the center of the minimum enclosing ball of points already aligned to this node (or some weighted variants of it). The advantage of the former is that canonical cluster centers tend to spread apart, which helps to increase coverage. Furthermore, the canonical cluster centers serve as good candidates for node centers as we already know that there are many points around them. The disadvantage is that it does not consider the distribution of points aligned to this node. See Figure 4, where without considering the distribution of points aligned to o_a and o_b , the new point p will be aligned to o_b even though o_a is a better choice. Using the center of the minimum enclosing ball alleviates this problem. However, such centers depend heavily on the order of curves aligned, and the influences regions of nodes produced this way tend to overlap much more than using the canonical cluster centers. We combine the advantages of both approaches into the following two-level

scoring function for measuring similarities.

More specifically, for a node o , let q be the first point aligned to this node. This means that at the time we were examining q , q cannot be aligned to any existing node in the POG. Let $c_k \in C$ be the nearest canonical cluster center of q — recall that the node o was created because $\|q - c_k\| \leq \varepsilon$. We add c_k as a *point* aligned to this node, and at any time, the center of the minimum enclosing ball of currently aligned points, *including* c_k , will be used as the node center $\omega(o)$. Now let

$$D(o) = \max_{q, q' \in o} \|q - q'\|$$

be the diameter of points currently aligned to o . We define that:

$$\delta(o, p) = \begin{cases} 2\varepsilon & \text{if } \|p - \omega(o)\| < D(o) \\ \varepsilon & \text{else if } \|p - \omega(o)\| < \varepsilon \\ 0 & \text{else} \end{cases} \quad (2)$$

In other words, the new scoring function encourages centering points around previously computed cluster centers, thus tending to reduce overlaps between the influence regions of different nodes. Furthermore, it gives higher similarity score for points that are more tightly grouped together with those already aligned at current node, addressing the problem shown in Figure 4. Our experimental tests have shown that this two level scoring function significantly outperforms the ones using either only the canonical centers or only the centers of minimal enclosing balls. We remark that it is possible to use variants of the above two-level scoring function, such as making it continuous (instead of being a step function). We choose the current form for its simplicity. Furthermore, experiments show that there is only marginal difference if we use the continuous version.

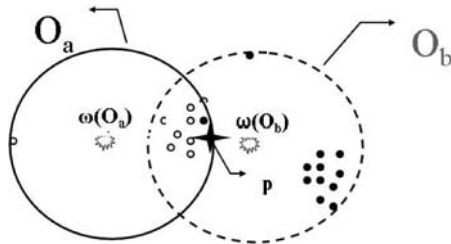


Fig. 4. Empty and solid points are aligned to the nodes o_a and o_b , respectively. For a new point p (the star), although it is closer to $\omega(o_b)$, it is better grouped with points aligned to o_a . Hence ideally, it should be aligned to o_a instead of to o_b .

3.3. Merging Stage

In the first stage, we have applied a progressive method to align each trajectory onto an alignment graph one by one. In the i th iteration, a point from T_i is either aligned to the best matched node in the current POG G_i , or a new node is created containing this point and the corresponding canonical cluster center. After processing all of the N trajectories in order, we return the final POG $G = G_N$. In the second stage of our EPO algorithm, we further improve the quality of the alignment in G using a novel merging process.

Given the greedy nature of the POA algorithm, the alignment obtained in G is not optimal and depends on the alignment order. Furthermore, given that the influence regions of different nodes may overlap, no matter how we improve the scoring function, sometimes it is simply ambiguous to decide *locally* where to align a new point, and a wrong decision may have grave consequence later.

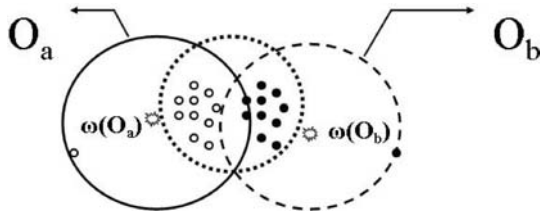


Fig. 5. Empty and solid points are aligned to the nodes o_a and o_b , respectively, while points in the dotted region should be grouped together.

For example, see Figure 5, where the set of points P (enclosed in the dotted circle) should have been aligned to one node. However, suppose the nodes o_a and o_b already exist before any point in P is inserted. Then as points from P come in, it is rather likely that they are distributed evenly into both o_a and o_b . This problem becomes much more severe in higher dimensions, where P can be distributed to several nodes whose centers are well-separated around P , but whose influence regions still covers some points from P (the number of such regions grows exponentially w.r.t. the dimension d). Hence instead of being captured in one heavily aligned node, P is broken into several nodes of small size. Our experimental tests confirm that this is happening rather commonly in the POA algorithm.

To address this problem, we propose a novel post-processing on G . The goal is to merge qualified points from neighboring less-aligned nodes to augment more heavily loaded nodes. In particular, the following two invariants are maintained during the merging process:

- (I1) At any time, the diameter of the target node is still bounded by the distance threshold ε ;
- (I2) The partial order constraints induced by the POG graph are always consistent with the order of points along each trajectory.

The second criterion means that at any time in the POG graph G' , if $p \in o_1$, $q \in o_2$, $p, q \in T_i$ and p precedes q along the trajectory T_i , then either $o_1 \prec o_2$, or there is no partial order relation between them. In other words, the resulting POG still corresponds to a valid alignment of \mathcal{T} with respect to the same thresholds.

As an example, see Figure 3, where the point p_2^1 (i.e, the second point of T_1) in the node b in (b) is moved to the node e in (c). Note that the graph is also updated to reflect the change (the dashed edge in (c)), in order to maintain the invariants (I1) and (I2). When all points aligned a node o are merged to other nodes (i.e, o becomes empty), we delete o , and its successors in the POG will then become the successors of its parent.

Algorithm 3.1: MERGING PROCESSING(

$G = \{o_1, \dots, o_m, \dots\}, |o_m| \geq |o_{m+1}|$)

```

while significant progress
  for each  $o_m \in G$  in increasing order of  $m$ 
    for each neighbor  $o_n, |o_n| < |o_m|$ 
      for each  $t^n$ 
        if  $mergeOK() == \mathbf{true}$ 
          then  $merge\ t^n \rightarrow o_m$ 
  \ \  $mergeOK()$  checks if the two invariants
  \ \ can be maintained if performing
  \ \ the candidate merging operation

```

Fig. 6. The merging algorithm.

A high level pseudocode of the merging process is shown in Figure 6. It augments better aligned nodes from the current POG G by processing first the nodes with larger size. We perform this procedure a few times till there is no significant increases in the quality of the resulting alignment. In practice, to speed up the algorithm, we merge neighbors to a node o only if its size is greater than some threshold, as otherwise, there is low probability that o will become a heavy node later.

4. EXPERIMENTAL RESULTS

In this section, we report a systematic performance study on a biological dataset that contains 200 molecular dynamics simulations. The experiments achieve the following goals: First, we show that the quality of the alignments produced by our EPO algorithm is significantly better than that of the original POA algorithm. Second, we demonstrate the effectiveness of our algorithm by applying it to real protein simulation data and obtaining biologically meaningful results that are consistent with previous discoveries²⁸. The algorithm is implemented using Java.

4.1. Background of Dataset

Our input dataset includes 200 simulated folding trajectories for a particular protein called Trp-cage. The dataset is provided by the Ota’s Lab²⁸. The folding simulations were performed at 325 K by using the AMBER99 force field with a small modification and the generalized Born implicit solvent model. Trp-cage (see Figure 7) is a mini-protein consisting of 20 amino acids. It has been widely used for folding study because of its short, simple sequence and its quick folding kinetics. Following the definition from¹⁵, a successful folding event has to satisfy the following two criteria:

- The RMSD for a conformation from the native NMR structure²⁴ is less than 2Å.
- A subsequence of such near-native conformations holds for at least 200ps.

In²⁸, 58 successful folding trajectories reaching successful folding events are identified, and each trajectory includes 101 successive conformations sampled at 20ps interval. Furthermore, there are two

crucial observations in²⁸ that we will exam in the our experiment. First, before moving to the native conformation, a “ring” sub-structure (see Figure 7) has to be formed. Second, the distinction of native and pseudonative confirmations heavily relies on side-chain position of “ring” sub-structure.²⁸ obtained the above results by aligning each pair of trajectories first and then applying a neighbor joining method to group similar trajectories together. However this semi-automatic approach requires dedicated expert knowledge. The following experiments applied on the same dataset show that our EPO algorithm can automatically detect the above folding events with little prior knowledge.

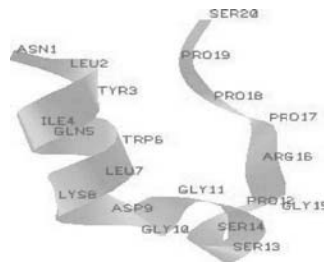


Fig. 7. NMR structure of trp-cage protein 1l2y. Labels on graph mark amino acids(AAs). AA2 to AA7 roughly form an alpha-helix. AA2 to AA19 form a ring-type structure. In particular, AA2 to AA5 and AA16 to AA19 form the “neck” of this ring.

4.2. Experimental Setting

In order to be consistent with the results from²⁸, we select all 58 successful folding events, and call it *SuccData*. We also randomly select 58 unsuccessful folding trajectories, each containing 101 conformations, and collect them in a set called *FailData*. The union of successful and unsuccessful data is referred to as the *MixData*. We set the distance threshold $\varepsilon = 1\text{\AA}$, and $\tau = 40$ in the following experiments, unless specified otherwise.

4.3. Investigation on Entire Protein Structure

In the first set of experiments, we convert each conformation to a high dimensional point based on the distance matrix between all of the alpha-carbon atoms. Figure 8 compares the quality of the alignments of the *SuccData* by performing the POA algorithm, our EPO algorithm without the merging procedure (EPO-NoMerge), and the EPO algorithm. It

shows the number of aligned nodes (y-axis) versus the size of aligned nodes (x-axis). Note that EPO-NoMerge is essentially POA with a clustering preprocessing and the new two-level scoring function.

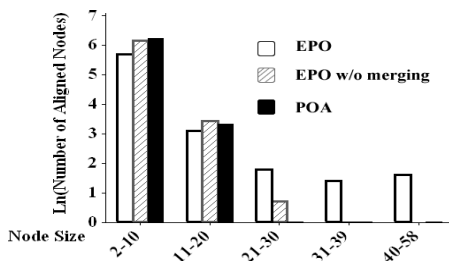


Fig. 8. Distribution of aligned nodes produced by the EPO algorithm, EPO-NoMerge (i.e, first stage of the EPO algorithm), and the traditional POA algorithm. The histogram is the number of aligned nodes (y-axis) versus the size of aligned nodes (x-axis).

The similarity level between these trajectories is low (i.e, the number of aligned nodes with large size is small). It is clear from this histogram that our EPO algorithm significantly outperforms the other two by producing more aligned nodes with large sizes. The comparison between EPO and EPO-NoMerge demonstrates the effectiveness of our merging procedure, and that EPO-NoMerge is better than POA shows that the *two-level scoring function* as well as the *clustering preprocessing* greatly enhances the performance. We have also performed experiments which show that compared to the POA algorithm, EPO-NoMerge is much less sensitive to the order of curves aligned, and we omit the results due to lack of space. Comparisons of the three algorithms over the *MixData* produces a similar result, and majority of points aligned to heavy nodes (i.e, $|o| \geq 40$) are from successful runs.

We also observe that most of the heavily aligned nodes are close to the end of the trajectories for the *SuccData*. In fact, many aligned points have conformation IDs around and greater than 90, which is indeed the time that the folding starts to get stabilized. More specifically, consider the set of aligned nodes of size greater than 40 for the *SuccData*. Among all points aligned to these nodes, 67.2% has a con-

formation ID greater than 90, and 24.4% has an ID between 80 and 90. This implies that our algorithm has the potential to detect the stabilization of successful folding events in an automatic manner.

This also implies that using the entire protein structure may be too coarse to detect critical folding events, as they are usually induced by small key motifs. In what follows, we map only a substructure of the input protein into a high dimensional point and provide more detailed analysis of this folding data.

4.4. Investigation on Substructures

It is usually believed that certain critical motifs play important roles which stabilize the whole structure in the folding process^{18, 33}. We wish to have a tool that can identify such critical motifs (substructures) automatically. We define a candidate motif to be two subchains of Trp-cage, each of length 4. These two pieces induce a sub-window in the distance map of each conformation of the protein. We further require that the number of contacts in this subwindow w.r.t. the distance map of the native structure is at least 4, where a contact corresponds to two alpha-carbon atoms within distance 6\AA .

For each conformation of a candidate motif, we consider the distance matrix between its alpha-carbon atoms as before, and convert the folding trajectory of this motif into a curve in the $4 \times 4 = 16$ dimensional space. In order to be more discriminative, we also introduce a *side-chain weighting factor* α , ranging from 0 to 1, to include the side chain information when comparing two high dimensional points^e: $\alpha = 0$ means that side-chain information is completely ignored. We perform our EPO algorithm on both the *SuccData* and the *MixData*, and there are two motifs that especially stand out.

4.4.1. Alpha-Helix Substructure

The first one corresponds to an alpha-helix substructure. In Figure 7, five successive amino acids (No.2-7) form an alpha-helix structure which is a simple, self-contained secondary structure (SSE)²⁴. From the results returned by our EPO algorithm, we note

^eRoughly speaking, for every conformation, we record for each residue also the relative position of the centroid of its side-chain with respect to its alpha-carbon atom. This provides another high dimensional point that we call a *side-chain point*. The distance between two conformations will combine the distance between their side-chain points by the side-chain weighting factor.

that this alpha-helix is formed rather early consistently in both successful and unsuccessful runs. Once formed, it remains stable. This is consistent with the common conception that due to its chemical property, alpha-helix is a stable secondary structure, and can be formed quickly. Hence the formation of alpha-helix *cannot* be used to differentiate successful runs from unsuccessful ones.

4.4.2. Ring-Substructure

The second motif corresponds to the neck of a ring structure. In particular, it consists of the sub-chains of No. 2 - 5 and No. 16 - 19 amino acids. The following results demonstrate that EPO can automatically not only find but also track the formation of such fingerprint sub-structure(critical motif).

Table 1. EPO on ring Structure(*MixData*). Column 1 - 3 shows the size of an aligned node (i.e, the number of points aligned to this node) from *MixData*, *SuccData*, and *FailData*, respectively. Column 4 shows the diameter of this node (note that the distance threshold $\varepsilon = 1\text{\AA}$ means that the diameter of a node can be upto 2\AA).

$ o_i \geq \tau$	Classification		D(o) \AA
	<i>SuccData</i>	<i>FailData</i>	
49	27	22	1.852
45	28	17	1.798
41	29	12	2.189
40	31	9	1.447
48	31	16	1.761
40	32	8	1.322
47	34	13	1.133
42	35	7	1.923
44	36	8	0.873
49	42	7	1.428
54	48	6	1.020
59	50	9	1.294
60	51	9	0.932
56	52	4	1.255
62	56	6	1.782
62	58	4	1.503

First, we observe from Table 1 that when applying the EPO algorithm to the *MixData* (with the sidechain weight factor $\alpha = 0.9$), significant alignments involve mainly trajectories from *SuccData*. For example, the last row of Table 1 shows that among the 62 points (from 62 trajectories) aligned to a particular node, 58 are from *SuccData*, with the remaining 4 from *FailData*. Hence this motif is potentially critical to the success of the folding of Trp-cage. It also suggests that we can automatically

classify the *MixData* into *SuccData* and *FailData* with few false positives based on this ring-neck motif, while previously, the classification in the input data was obtained by a few expert defined rules.

Second, when the side-chain weighting factor $\alpha = 0.9$, we note that 49.6% of significant aligned nodes are formed before the conformation ID 85 (compared to results from Section 4.3). For example, there are two aligned nodes from the successful runs, where 80% of points (i.e. trajectories) aligned to them has a conformation ID between 75 - 85. This implies that the *complete* formation of this ring-neck usually *immediately* precedes the stabilization of the folding structure (which is roughly at conformation ID 90 for successful trajectories).

If reducing the side-chain weighting factor α to 0.5, naturally, we found more aligned nodes. In particular, other than the cluster with conformations of IDs around 80, we observe more significant clusters involving conformations with IDs from 50 - 70. By comparing the conformations of the ring-neck motif in these clusters with those in the aligned nodes around 80, we found that the backbone structures are rather similar, but the side-chains are of different orientations. In other words, the shape of the ring-neck motif is first stabilized by the backbone structure, and then the side-chains gradually move into right position. There are a few trajectories where the side-chains eventually move to the mirror image of their correct positions, and lead to pseudo-native conformations which can only be detected when considering the side-chains.

The above results are consistent with the results from ²⁸, where such a ring-shaped substructure was discovered semi-automatically by pairwise structure comparisons together with expert knowledge.

4.5. Timing of EPO

The above experiments are implemented on a Windows XP machine with 1.5GHz CPU and 512 MB Memory. The running time of performing the experiments on the entire protein structure is about 30 minutes, and that of the small motifs is about 20 minutes. We believe that the time complexity of the current algorithm can be significantly improving by optimizing our code. The merging stage is the most

time consuming component and takes about three quarters of the total time.

5. CONCLUSIONS

In this paper we proposed and developed EPO, an effective multiple curve comparison method, to analyze protein folding trajectories. Our new method greatly improved the performance of the POA algorithm by using a clustering preprocessing, a more discriminative two-level scoring function, as well as a novel merging post-processing procedure. It can detect low level of similarity among input curves. We demonstrated the effectiveness of our method by applying it to a set of simulated folding trajectories of the miniprotein Trp-cage.

Currently, we have only experimented the EPO algorithm with a mini-protein (Trp-cage). One immediate question is to understand the scalability of the EPO algorithm for larger proteins or longer trajectories. In particular, a *larger* protein means a curve of *higher dimensions*. Our EPO algorithm seems to scale linearly with the dimension, from current experiments. Furthermore, in practice, it is likely that we only perform the algorithm on small motifs. For longer trajectories, it seems that our algorithm scales in a quadratic manner. However, further experiments are necessary to investigate the scalability issue.

There are some previous work that analyze protein folding trajectories by collecting various statistics on measures such as the contact number (i.e, the number of native contacts) of each conformation along a trajectory and the URMS distance between a conformation and the native structure¹⁴. One way to view this is that a trajectory is mapped into a time-series data representing the evolution of, say, the number of native contacts, which can be considered as a one-dimensional curve. In this regard, we can use our EPO algorithm to analyze a collection of such curves induced by one measure. In general, there may be multiple measures, geometric or physio-chemical, that a user may wish to inspect. Hence it is highly desirable to build a framework for analyzing folding trajectories that can incorporate these *multiple* measures, and that also enables the addition of new properties easily. This is one important future direction for us.

Finally, we remark that compared to other multiple curve alignment algorithms, our algorithm is specifically effective at capturing low level of similarities. As such, another important future direction is to apply similar techniques to classifying protein structures, as well as extracting structural motifs from a family of proteins that may not share high global similarity.

ACKNOWLEDGMENT

The work was supported in part by the Department of Energy (DOE) under grant DE-FG02-06ER25735 and in part by National Science Foundation under grant IIS-0546713.

References

1. V. I. Abkevich, A. M. Gutin, and E. I. Shakhnovich. Specific nucleus as the transition state for protein folding: evidence from the lattice model. *Biochemistry*, 33:10026–10036, 1994.
2. J. M. Borreguero, F. Ding, S. V. Buldyrev, H. E. Stanley, and N. V. Dokholyan. Multiple Folding Pathways of the SH3 Domain. *ArXiv Physics e-prints*, 87, May 2003.
3. L. P. Chew and K. Kedem. Finding the Consensus Shape for a Protein Family (Extended Abstract).
4. F. Chiti, N. Taddei, P. M. White, M. Bucciantini, F. Magherini, M. Stefani, and C. M. Dobson. Mutational analysis of acylphosphatase suggests the importance of topology and contact order in protein folding. *Nature Struct. Biol.*, 6:1005–1009, 1999.
5. N. V. Dokholyan, S. V. Buldyrev, H. E. Stanley, and E. I. Shakhnovich. Molecular dynamics studies of folding of a protein-like model. *Fold. Design*, 3:577–587, 1998.
6. R. Du, V. S. Pande, A. Y. Grosberg, T. Tanaka, and E. Shakhnovich. On the role of conformational geometry in protein folding. *Journal of Chemical Physics*, 111:10375–10380, Dec. 1999.
7. M. Gerstein and M. Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Protein Science*, 7:445–456, 1998.
8. J. F. Gibrat, T. Madej, and S. H. Bryant. Surprising similarities in structure comparison. *Curr. Opin. Struct. Biol.*, 6(3):377–385, 1996.
9. C. Grasso and C. Lee. Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. *Bioinformatics*, 20(10):1546–1556, June.
10. C. Guda, S. Lu, E. D. Scheeff, P. E. Bourne, and L. N. Shindyalov. CE-MC: a multiple protein

- structure alignment server. *Nucleic Acids Research*, 32:"W100–3", 2004.
11. L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233:123–138, September.
 12. L. Holm and C. Sander. Dali/FSSP classification of three-dimensional protein folds. *Nucleic Acids Res.*, 25(1):231–234, 1997.
 13. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.
 14. K. Kedem, L. Chew, and R. Elber. Unit-Vector RMS(URMS) as a Tool to Analyze Molecular Dynamics Trajectories. *Proteins: Structure, Function and Genetics*, 37:554–564, 1999.
 15. R. Koike, K. Kinoshita, and A. Kidera. Ring and Zipper formation is the key to understanding the structural variety in all- β proteins. *FEBS Letters*, 533:9–13, 2003.
 16. E. Krissinel and K. Henrick. Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions. *Acta Cryst.*, D60:2256–2268, 2004.
 17. C. Lee, C. Grasso, and M. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.
 18. C. Levinthal. Are there pathways for protein folding? *J. Chim. Phys.*, 65:44–45, 1968.
 19. S. W. Lockless and R. Ranganathan. Evolutionarily Conserved Pathways of Energetic Connectivity in Protein Families. *Science*, 286(5438):295–299, October 1999.
 20. D. Lupyan, A. Leo-Macias, and A. R. Ortiz. A new progressive-iterative algorithm for multiple structure alignment. *Bioinformatics*, 21(15):3255–3263, 2005.
 21. M. J. Sutcliffe, I. Haneef, D. Carney, and T. L. Blundell. Knowledge based modelling of homologous proteins, part I: three-dimensional frameworks derived from the simultaneous superposition of multiple structures. *Protein Engineering*, 1(5):377–384, 1987.
 22. A. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.
 23. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.
 24. J. Neidigh, R. Fesinmeyer, and N. Andersen. PDB ID:1L2Y Mini-proteins Trp the light fantastic. *Nat. Struct. Biol.*, 9(6):425–430, June 2002.
 25. M. E. Ochagavia and S. Wodak. Progressive combinatorial algorithm for multiple structural alignments: application to distantly related proteins. *Proteins*, 55:436–454, 2004.
 26. C. A. Orengo. CORA—Topological fingerprints for protein structural families. *Protein Science*, 8:699–715, 1999.
 27. C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton. CATH—A hierarchical classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.
 28. M. Ota, M. Ikeguchi, and A. Kidera. Phylogeny of protein-folding trajectories reveals a unique pathway to native structure. *PNAS*, 101(51):17658–17663, December 2004.
 29. E. Sandelin. Extracting multiple structural alignments from pairwise alignments: a comparison of a rigorous and heuristic approach. *Bioinformatics*, 21(7):1002–1009, 2005.
 30. I. N. Shindyalov and P. E. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of optimal path. *Protein Engineering*, 11(9):739–747, 1998.
 31. T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.
 32. W. Taylor and C. Orengo. Ssap: sequential structure alignment program for protein structure comparison. *Methods Enzymol.*, 266:617–35, 1996.
 33. P. Wolynes, J. Onuchic, and D. Thirumalai. Navigating the folding routes. *Science*, 267:1619–1920, 1995.
 34. Y. Caspi and M. Irani. Spatio-temporal alignment. *Proc. IEEE Transactions On Pattern Analysis and Machine Intelligence.*, pages 1409–1424, 2002.
 35. Y. Ye and A. Godzik. Multiple flexible structure alignment using partial order graphs. *Bioinformatics*, 21(10):2362–2369, 2005.