Protein Chain Pair Simplification Under the Discrete Fréchet Distance

Journal:	*IEEE/ACM Transactions on Computational Biology and Bioinformatics*
Manuscript ID:	TCBBSI-2012-08-0204.R3
Manuscript Type:	Regular Paper
Keywords:	I.2.8.d Dynamic programming < I.2.8 Problem Solving, Control Methods, and Search < I.2 Artificial Intelligence < I Computing Methodologies, F.2 Analysis of Algorithms and Problem Complexity < F Theory of Computation

Protein Chain Pair Simplification Under the Discrete Fréchet Distance

Tim Wylie and Binhai Zhu

Abstract—For protein structure alignment and comparison, a lot of work has been done using RMSD as the distance measure, which has drawbacks under certain circumstances. Thus, the discrete Fréchet distance was recently applied to the problem of protein (backbone) structure alignment and comparison with promising results. For this problem, visualization is also important since protein chain backbones can have as many as 500$\sim$600 α-carbon atoms which constitute the vertices in the comparison. Even with an excellent alignment, the similarity of two polygonal chains can be difficult to visualize unless the chains are nearly identical. Thus, the chain pair simplification problem (CPS-3F) was proposed in 2008 to simultaneously simplify both chains with respect to each other under the discrete Fréchet distance. The complexity of CPS-3F is unknown, so heuristic methods have been developed. Here, we define a variation of CPS-3F, called the constrained CPS-3F problem (CPS-3F$^+$), and prove that it is polynomially solvable by presenting a dynamic programming solution, which we then prove is a factor-2 approximation for CPS-3F. We then compare the CPS-3F$^+$ solutions with previous empirical results, and further demonstrate some of the benefits of the simplified comparisons. Chain pair simplification based on the Hausdorff distance (CPS-2H) is known to be **NP**-complete, and here we prove that the constrained version (CPS-2H$^+$) is also **NP**-complete. Finally, we discuss future work and implications along with a software library implementation, named FPACT (The Fréchet-based Protein Alignment & Comparison Toolkit).

Index Terms—Protein structure alignment, Protein structure simplification and visualization, Discrete Fréchet distance, Approximation algorithms, Dynamic Programming, NP-complete

1 Introduction

THE comparison and simplification of polygonal chains have been well studied in several fields including computer vision, bioinformatics, computational geometry, and parametric curve approximations [1], [4], [28]. Within structural biology, polygonal chain similarity is one of the central problems of protein research. In general, it is believed that a protein's structure might imply its function, and thus to compare the functionality of proteins their structures must be compared [19]. This is known to be true for certain situations, especially with homologous traits between proteins, and the empirical evidence between proteins in general is in agreement [17], [19]. The structure is defined by the α-carbon atoms of the residues (amino acids) along the backbone of each chain. These atoms represent the vertices that constitute our 3D polygonal chains.

Since the structure of the protein is suspected to be related to its function, there have been many software systems designed for protein structure alignment and comparison in the last couple of decades. A few of the more well-known systems are SCOP [9], DALI [13], [14], CATH [20], MAMMOTH [21], CE [23], ProteinDBS [24], SSAP [25], and 3D-BLAST [32]. None of these systems use the discrete Fréchet distance, and the majority of the work previously done on protein global structure alignment and protein local structure alignment uses the RMSD (Root Mean Square Deviation) evaluative measure. Given two m-vectors $V_1 = \langle u_1, u_2, ..., u_m \rangle$ and $V_2 = \langle v_1, v_2, ..., v_m \rangle$, RMSD is defined as:

$$RMSD(V_1, V_2) = \sqrt{\frac{\sum_i (u_i - v_i)^2}{m}}.$$

RMSD gives an average pair-wise distance along the two vectors, which provides some insight into the similarity of the two chains, but the reliance on m shows one of the major drawbacks of using RMSD. The comparison hinges on the necessity that the two vectors be the same length and that the vertices at a given index in each chain be pairwise similar. If we modified the chains, then we could receive substantially different RMSD values. Suppose we have two chains C_1, C_2 with m vertices, and we then add some vertices on C_1 and C_2 by alternatively duplicating/repeating some different vertices in C_1 and C_2 to obtain C_1', C_2', then $RMSD(C_1', C_2')$ could be dramatically different from $RMSD(C_1, C_2)$, even though geometrically C_1' and C_2' are as similar as C_1 and C_2. This suggests that a measure independent of the number of vertices or a pair-wise alignment might be a better indicator of the similarity of the two chains.

As an example, to handle the issue of differing chain lengths, ProteinDBS [24] only computes the RMSD between matched parts of the chains, and disregards the very dissimilar parts in the calculation. This allows for the same number of vertices to be used from each chain, but coverage percentages must be considered to understand similarity. This makes understanding the overall relationship between the two protein backbones difficult.

To achieve a more accurate measure of similarity between two protein structures, Jiang et al. proposed using the discrete Fréchet distance for the protein backbone comparison [16].

• *T. Wylie is with the Department of Computer Science, Montana State University, Bozeman, MT, 59717. E-mail: timothy.wylie@cs.montana.edu*
• *B. Zhu is with the Department of Computer Science, Montana State University, Bozeman, MT, 59717. E-mail: bhz@cs.montana.edu*

The two main problems they addressed were the alignment of the two chains, and then the comparison itself. They showed that the optimal alignment problem, as defined in [16], between two 3D chains under the discrete Fréchet distance takes $O(n^7 m^7 \log(n+m))$ time to solve [16]. Due to the high time complexity they proposed a heuristic method not dependent on the discrete Fréchet distance. We revisited the optimal alignment problem by proposing a possible PTAS (Polynomial-Time Approximation Scheme) algorithm in which all translations and rotations were based on the current discrete Fréchet distance of the two chains [29]. We also showed that this was at worst a 2-approximation algorithm for the optimal alignment problem. The new algorithm provided better alignment results than the previous method for all empirical evaluations.

When comparing polygonal structures, alignment is just one of the issues. Given that protein backbones can have as many as 500$\sim$600 vertices (α-carbon atoms) in each chain, even with an optimal alignment, visualizing the similarity of two chains is difficult unless those chains are nearly identical. To address this issue the chain pair simplification (CPS-3F) problem was proposed by Bereg et al. in 2008 [6]. They were unable to show whether CPS-3F is **NP**-complete, but they proved that the Hausdorff version of the problem (CPS-2H), which simplified the chains via the Hausdorff distance, is **NP**-complete [6]. This led them to postulate that under the discrete Fréchet distance the problem was likely to be **NP**-complete. In our previous work [29] we used a heuristic $O(n)$ time algorithm to simplify pairs of chains. Here, we prove that a variation of CPS-3F, denoted as CPS-3F$^+$, is polynomially solvable, and that it is a factor-2 approximation for CPS-3F. This restricted version is also beneficial because as we simplify the proteins, we usually want to visually compare the two backbone chains without distorting their lengths. However, we also look at cases where uneven simplification may be beneficial for visualization. Further, we prove that the constrained version on CPS-2H, similarly denoted as CPS-2H$^+$, is **NP**-complete.

As previously mentioned, the majority of software systems for aligning and comparing protein backbones use the RMSD measure. Thus, we have created a software library called FPACT (The Fréchet-based Protein Alignment & Comparison Toolkit), which uses the alignment, comparison and simplification algorithms (including CPS-3F$^+$) based on the discrete Fréchet distance [31].

This paper is an extension of our previous work [30]. All sections have been expanded, many sections with better explanations for clarity, including an enriched empirical section looking at a greater variation of protein chain length and similarity. The 2-approximation proof is now included, and there is more explanation for the previous proofs. There is more information about FPACT and about the algorithms. Further, Section 3.3 gives an example demonstrating that CPS-3F does not always have a minimum moving cost. We also answer one of our open questions by proving that CPS-2H$^+$ is **NP**-complete .

The paper is organized as follows. In Section 2 we discuss the discrete Fréchet distance, CPS-3F$^+$, and the related background. In Section 3 we present a polynomial time solution for CPS-3F$^+$ and analyze the algorithm complexity. We then give a pseudocode algorithm to implement the solution in Section 4. In Section 5 we compare the results to our previous heuristic for CPS-3F. Section 6 gives an overview of the libraries in the toolkit. Then in Section 7 we prove that CPS-2H$^+$ is **NP**-complete along with some examples of the reduction. Finally, we outline some implications and future work in Section 8 and then conclude with some open problems.

2 Preliminaries

2.1 The Hausdorff Distance

The Hausdorff distance was first defined by Felix Hausdorff in 1914 [12]. Since its introduction, the Hausdorff distance has become one of the most widely used similarity measures across many disciplines.

Definition 1. *[18] Let X and Y be two non-empty subsets of a metric space (M, d). We define their Hausdorff distance $d_H(X, Y)$ by*

$$d_H(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\},$$

where sup *represents the supremum and* inf *the infimum.*

2.2 The Discrete Fréchet Distance

The Fréchet distance was first defined by Maurice Fréchet in 1906 as a measure of similarity between two parametric curves [11]. Subsequently, it has become a standard measure in parametric analysis. In the early 90s, the Fréchet distance for polygonal curves was first considered by Alt and Godau who gave an $O(mn\log(mn))$ algorithm [2], [3]. Then in 1994 Eiter and Mannila defined the discrete Fréchet distance as an approximation of the Fréchet distance to be used between two polygonal chains using only the nodes along the chains for the measurements [10]. They also referred to this discrete form as the coupling distance, which is used synonymously. Furthermore, they proved the discrete version can be computed in $O(mn)$ time, where m, n are the number of vertices in the polygonal chains.

The discrete Fréchet distance has since been applied in several fields of research, but recently, one of the prominent applications has been in aligning and comparing the similarity of protein backbones [5], [16], [33]. In this comparison between backbones each vertex in the polygonal chains represents an α-carbon atom, which gives the comparison based on the atoms a clear biological meaning. Thus, a comparison between two backbones using the discrete Fréchet distance may be more appropriate than one using the continuous version.

Given two paths, we define their discrete Fréchet distance below. (We use the graph-theoretic term “paths” instead of the geometric term “polygonal chains” here because our definition makes no assumption that the underlying space of points is geometric.) We use $d(a, b)$ to represent the Euclidean distance between two 3D points a and b, but certainly it can be replaced with some other distance measure, depending on the application.

Definition 2. *Given a path $P = \langle p_1, \ldots, p_n \rangle$ of n vertices, a t-**walk** along P is a partitioning of P along the path into t disjoint non-empty subpaths $\{P_i\}_{i=1..t}$ such that $P_i = \langle p_{n_{i-1}+1}, \ldots, p_{n_i} \rangle$ and $0 = n_0 < n_1 < \cdots < n_t = n$.*

Definition 3. *Given two paths $A = \langle a_1, \ldots, a_m \rangle$ and $B = \langle b_1, \ldots, b_n \rangle$, a **paired walk** along A and B is a t-walk $\{A_i\}_{i=1..t}$ along A and a t-walk $\{B_i\}_{i=1..t}$ along B for some t, such that, for $1 \le i \le t$, either $|A_i| = 1$ or $|B_i| = 1$ (that is, either A_i or B_i contains exactly one vertex).*

Definition 4. *The **cost** of a paired walk $W = \{(A_i, B_i)\}$ along two paths A and B is*

$$d_{\mathcal{F}}^{W}(A,B) = \max_i \max_{(a,b) \in A_i \times B_i} d(a,b).$$

Definition 5. *The **discrete Fréchet distance** between two paths A and B is*

$$d_{\mathcal{F}}(A,B) = \min_W d_{\mathcal{F}}^{W}(A,B).$$

*A paired walk that achieves the discrete Fréchet distance between two paths A and B is called a **Fréchet alignment** of A and B.*

The continuous Fréchet distance is typically explained as the relationship between a person and a dog connected by a leash walking along the two curves and trying to keep the leash as short as possible. However, for the discrete case, we only consider the nodes of these curves, and thus the man and dog must "hop" along the nodes of the chain. Consider the scenario in which a person walks along A and a dog along B. Intuitively, the definition of the paired walk is based on three cases:

1) $|B_i| > |A_i| = 1$: the person stays and the dog hops forward;
2) $|A_i| > |B_i| = 1$: the person hops forward and the dog stays;
3) $|A_i| = |B_i| = 1$: both the person and the dog hop forward.

Figure 1 shows the relationship between the discrete and continuous Fréchet distances. In Figure 1(a), we have two chains $\langle a_1, a_2, a_3 \rangle$ and $\langle b_1, b_2 \rangle$, the continuous Fréchet distance between the two is the distance from a_2 to segment $\overline{b_1 b_2}$, i.e., $d(a_2, o)$. The discrete Fréchet distance is $d(a_2, b_2)$. The discrete Fréchet distance could be quite larger than the continuous distance. On the other hand, with enough sample points on the two chains, the resulting discrete Fréchet distance, i.e., $d(a_2, b)$ in Figure 1(b), closely approximates $d(a_2, o)$.

With enough evenly sampled nodes the discrete Fréchet distance can closely approximate the continuous version, and with a standard dynamic programming approach, it is straightforward to obtain the following theorem.

Theorem 1. *[10] The discrete Fréchet distance between two paths with m and n vertices respectively can be computed in $O(mn)$ time.*

In 2008 the chain pair simplification problem in three dimensions under the discrete Fréchet distance was defined to improve visualization of the two chains. The problem not only allows one to see the two chains in a simplified form, but it also keeps the characteristic similarities that exist between the chains. Although the problem is not necessarily limited to 3D space, we state the original decision problem as it was defined relating to protein backbone chains.

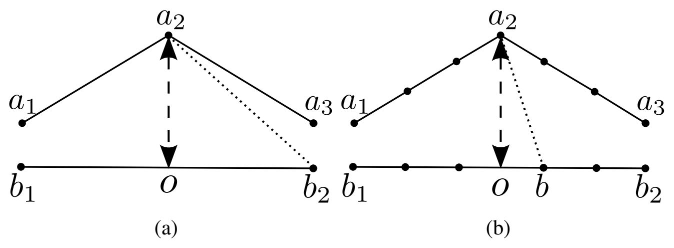

Figure 1: The relationship between the discrete and continuous Fréchet distance where o is the continuous and the dotted line between nodes is the discrete. (a) shows a case where the chains have fewer nodes and a larger discrete Fréchet distance, while (b) is the same basic path with more nodes, and thus provides a better approximation of the Fréchet distance.

The CPS problem is:
Instance: Given a pair of 3D chains A and B, with lengths $O(m), O(n)$ respectively, an integer $K > 0$, and three real numbers $\delta_1, \delta_2, \delta_3 > 0$.
Problem: Does there exist a pair of chains A', B', each of at most K vertices, such that the vertices of A', B' are from A, B, respectively, and $d_1(A, A') \le \delta_1, d_2(B, B') \le \delta_2, d_{\mathcal{F}}(A', B') \le \delta_3$?

When $d_1 = d_2 = d_{\mathcal{F}}$, the problem is called CPS-3F since all three distance measures are the discrete Fréchet distance. When $d_1 = d_2 = d_H$ (the Hausdorff distance), the problem is called CPS-2H since two of the distances are Hausdorff.

2.3 The Moving Cost

We now define a new measure for the discrete Fréchet distance based on the paired walk between two chains.

Definition 6. *The **moving cost** of a paired walk $W = \{(A_i, B_i)\}$ is*

$$m_c^W(A_i, B_i) = \max\{|A_i|, |B_i|\}. \tag{1}$$

The moving cost of a paired walk W between A and B is

$$m_c^W(A,B) = \sum_{i=1}^{t} m_c^W(A_i, B_i). \tag{2}$$

The moving cost for A and B is the sum of the number of "hops" the man or dog make along the two chains. However, when they both move at once, this only counts as a single move. In other words, it is the number of pairs of points, or matched points, between the chains used in calculating the discrete Fréchet distance.

In Figure 2 we show a simple example of two chains that can be simplified in two possible ways. In 2(a) the moving cost is six and the number of nodes for each chain is four. In 2(b) the moving cost is still six, yet the number of nodes is now five in each chain.

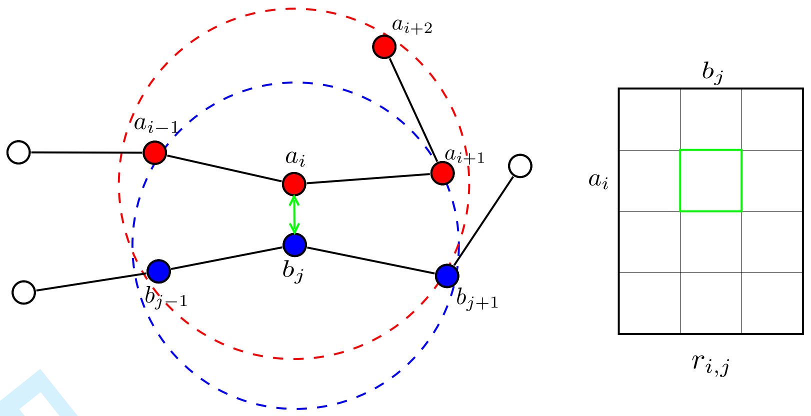

Figure 3: The rectangle $r_{i,j}$ constructed from subchains of A, B where $d(a_i, b_j) \leq \delta_3$. Here $S_A(a_i, \delta_1)$ contains the vertices a_{i-1} to a_{i+2}, and $S_B(b_j, \delta_2)$ contains the vertices b_{j-1} to b_{j+1}. Thus, $r_{i,j}$ is defined by the min and max node indices in each subchain.

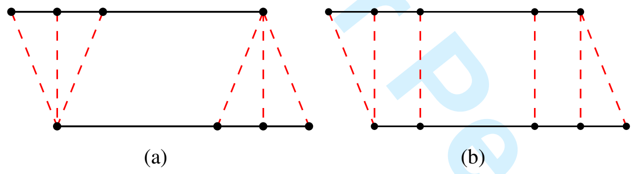

Figure 2: The difference between the number of nodes and the moving cost. Suppose that both (a) and (b) are valid simplifications of two chains. They have the same moving cost, yet (a) only has four nodes in each of the simplified chains, but in (b) both chains have five nodes.

We can prove some nice properties of the moving cost, such as the complexity being polynomial and its ability to approximate the number of vertices. As we make use of later, $\max(|A|, |B|) \leq m_c^W(A, B) \leq |A| + |B| - t$ for a paired t-walk W along A and B. This is because in any paired walk (A_i, B_i), the moving cost is at least $\max\{|A_i|, |B_i|\}$ and at most $|A_i| + |B_i| - 1$. This is the motivation for our variant of CPS-3F and CPS-2H.

The constrained CPS problem is now defined as follows:
Instance: Given a pair of 3D chains A and B, with lengths $O(m), O(n)$ respectively, an integer $K' > 0$, and $\delta_1, \delta_2, \delta_3 \in \mathbb{R}^+$.
Problem: Does there exist a pair of chains A', B' where the vertices are from A, B, respectively, such that for some paired walk W between A', B', $m_c^W(A', B') \leq K'$, and $d_1(A, A') \leq \delta_1$, $d_2(B, B') \leq \delta_2$, $d_{\mathcal{F}}(A', B') \leq \delta_3$?
Now when $d_1 = d_2 = d_{\mathcal{F}}$, we call the problem CPS-3F$^+$, and when $d_1 = d_2 = d_H$, the problem is called CPS-2H$^+$.

2.4 Dynamic Time Warping

Another parametric distance measure is Dynamic Time Warping (DTW), and was first introduced by Vintsyuk in 1968 in the field of speech processing [27]. During the last 15 years it has been used largely in the data mining community as a method for comparing, indexing, and learning with temporal and spatial data. This technique was recently applied in determining similarity between protein flexibility [26], and could prove useful in alignment and simplification as well.

Dynamic time warping has been developed extensively in many areas for specific applications. The idea is related to the discrete Fréchet distance, but differs in finding the minimum sum between points as opposed to the minimum maximum distance between any two pairs. Although related, no work has been done with DTW and chain pair simplification. Many strategies created for reducing the running time of DTW such as the Itakura Parallelogram [15] or the Sakoe-Chiba Band [22] may be adaptable to our work though.

3 CPS-3F$^+$ IS POLYNOMIALLY SOLVABLE

3.1 CPS-3F$^+$ $\in$ P

In this section we present a polynomial time solution for CPS-3F$^+$, which is an adaptation of CPS-3F. Several versions of the single chain simplification problem were addressed and shown to be polynomially solvable by Bereg et al. [6]. However, CPS-2H (where the Hausdorff distance is used for $d(A, A')$ and $d(B, B')$) was shown to be **NP**-complete, and thus it was postulated that the Fréchet version might also be **NP**-complete. The solution presented here proves that under the discrete Fréchet distance, the constrained chain pair simplification problem (CPS-3F$^+$) is polynomially solvable if the dimension is fixed. The algorithm returns the optimal K' specified in the definition of the decision problem, which is equal to

$$m_c(A', B') = \min_W m_c^W(A', B'), \tag{5}$$

among all feasible W. We now define several necessary terms and data structures.

Given two polygonal chains $A = \langle a_1, a_2, ..., a_m \rangle$, and $B = \langle b_1, b_2, ..., b_n \rangle$, and constraints $\delta_1, \delta_2, \delta_3 \in \mathbb{R}^+$, we design a dynamic programming algorithm to find the optimal moving cost K'. First, let $\mathcal{D} = \{(a_i, b_j)|\ a_i \in A,\ b_j \in B$ and $d(a_i, b_j) \leq \delta_3\}$. This is the set of all pairs of nodes between the two chains that are at a distance of at most δ_3 from each other. We then define a matrix $\mathcal{C}$ of size $m \times n$ that in any cell, $\mathcal{C}_{i,j}$, contains the minimum number, K', of pairs $(a_k, b_l) \in \mathcal{D}$, which given δ_1, δ_2, and δ_3 simplify A and B via CPS-3F$^+$ from (a_1, b_1) up to (a_i, b_j).

In order to maintain $\mathcal{C}$, we need another data structure $\mathcal{R}$ and some other helpful definitions. We define $S_X(x_i, \delta)$

Initial Conditions: $\mathcal{Q}_{1,1} \neq \emptyset$, $\mathcal{R}_{1,1} = \mathcal{Q}_{1,1}$, and $\mathcal{C}_{1,1} = 1$.

$$\mathcal{C}_{i,j} = \min_{(k,l)\in\{(i\text{-}1,j),(i,j\text{-}1),(i\text{-}1,j\text{-}1)\}} \begin{cases} \mathcal{C}_{k,l}, & \text{if } \mathcal{Q}_{i,j} \cap \mathcal{R}_{k,l} \neq \emptyset \\ \mathcal{C}_{k,l}+1, & \text{if } \mathcal{Q}_{i,j} \cap \mathcal{R}_{k,l} = \emptyset,\ \mathcal{Q}_{i,j} \neq \emptyset,\ \mathcal{R}_{k,l} \neq \emptyset \\ NULL, & \text{if } \mathcal{Q}_{i,j} = \emptyset \end{cases} \tag{3}$$

$$\mathcal{R}_{i,j} = \bigcup_{(k,l)\in\{(i\text{-}1,j),(i,j\text{-}1),(i\text{-}1,j\text{-}1)\}} \begin{cases} \mathcal{R}_{k,l} \cap \mathcal{Q}_{i,j}, & \text{if } \mathcal{C}_{i,j} = \mathcal{C}_{k,l},\ \mathcal{R}_{k,l} \cap \mathcal{Q}_{i,j} \neq \emptyset \\ \mathcal{Q}_{i,j}, & \text{if } \mathcal{C}_{i,j} = \mathcal{C}_{k,l}+1,\ \mathcal{R}_{k,l} \neq \emptyset, \\ & \mathcal{R}_{k,l} \cap \mathcal{Q}_{i,j} = \emptyset \end{cases} \tag{4}$$

as the maximal continuous subchain containing x_i on the polygonal chain X such that all the vertices on this subchain are contained in the sphere centered at x_i and with radius δ. Now let $r_{i,j}$ be the rectangle on $\mathcal{C}$ defined as $\langle \min(S_A(a_i,\delta_1)), \max(S_A(a_i,\delta_1)), \min(S_B(b_j,\delta_2)), \max(S_B(b_j,\delta_2))\rangle$ such that $(a_i,b_j) \in \mathcal{D}$. Here, min and max refer to the minimum or maximum indexed element within $S_X(x_i,\delta)$. For every pair in $\mathcal{D}$, we envision the corresponding rectangles as being overlayed on $\mathcal{C}$. A rectangle $r_{i,j}$ covers all the cells of $\mathcal{C}$ that are analogous to the vertices in $S_A(a_i,\delta_1) \cup S_B(b_j,\delta_2)$ as shown in Figure 3.

Our dynamic programming approach must take two different concurrent simplifications into account to find the minimum moving cost. First, the distance between each original and simplified chain is represented by the sphere, $S_X(x_i,\delta)$, and captured as part of a rectangle (Figure 3). Second, $\mathcal{D}$ represents the distance between the two simplified chains. The two are combined by the rectangles where the height and width capture the simplified vertices in A and B, respectively, and the number and placement of those rectangles on $\mathcal{C}$ ties in the relationship between the two chains.

For convenience we also define the set of all rectangles that a cell in $\mathcal{C}$ belongs to: $\mathcal{Q}_{k,l} = \{r_{i,j} | a_k \in A, b_l \in B \text{ and } \min(S_A(a_i,\delta_1)) \leq a_k \leq \max(S_A(a_i,\delta_1)) \text{ and } \min(S_B(b_j,\delta_2)) \leq b_l \leq \max(S_B(b_j,\delta_2))\}$.

Let $\mathcal{R}$ be a matrix of sets where the matrix is of size m by n, and $\mathcal{R}$ provides information needed to fill out $\mathcal{C}$ by storing a list of rectangles for each cell. $\mathcal{R}_{i,j}$ contains a set of rectangles (dynamic array) that pertain to the number of coverings (rectangles) still viable at any (i,j) relating to the number already calculated for $\mathcal{C}_{i,j}$. These are computed by the recurrences in Equations 3 and 4, which are listed along with the initial conditions for the relations.

The idea is to find the minimum covered xy-monotone increasing path from (a_1,b_1) to (a_m,b_n) that corresponds with $\mathcal{C}_{1,1}$ to $\mathcal{C}_{m,n}$. This is the minimum path by dynamic programming with all feasible options explored. If we visited a cell that was not covered, that would mean one of the nodes is not covered by a pair in $\mathcal{D}$. By finding a minimum covered path, one guarantees that every column and every row is covered by at least one rectangle, which means all of the nodes of A and B are covered.

The increasing xy-monotone path is necessary in the recurrence due to the definition of the discrete Fréchet distance. Without the requirement of a monotonically increasing path we would be using the weak discrete Fréchet distance – a version of the Fréchet distance that allows backtracking, or in terms of the analogy would allow the man or the dog to walk backwards.

We first characterize the optimal substructure of CPS-3F$^+$ as an optimization problem given our definitions, and then show this yields the optimal solution for K' and thus decides CPS-3F$^+$.

Theorem 2. *Optimal substructure of CPS-3F$^+$:*
Let $A = \langle a_1,\ldots,a_m\rangle$ and $B = \langle b_1,\ldots,b_n\rangle$ be two polygonal chains, $\delta_1,\delta_2,\delta_3 \in \mathbb{R}^+$, and let $Z_i = \langle z_1,\ldots,z_i\rangle$ be any CPS-3F$^+$ solution such that every z_j is a rectangle.

1) *If (a_k,b_l) is covered by z_i, where $(k,l) \in \{(m\text{-}1,n),(m,n\text{-}1),(m\text{-}1,n\text{-}1)\}$, then Z_i is a CPS-3F$^+$ solution for A_k, B_l.*
2) *If (a_k,b_l) is covered by z_{i-1}, where $(k,l) \in \{(m\text{-}1,n),(m,n\text{-}1),(m\text{-}1,n\text{-}1)\}$, then Z_{i-1} is a CPS-3F$^+$ solution for A_k, B_l.*
3) *If (a_k,b_l) is not covered by z_i or z_{i-1}, where $(k,l) \in \{(m\text{-}1,n),(m,n\text{-}1),(m\text{-}1,n\text{-}1)\}$, then $\nexists$ a CPS-3F$^+$ solution for A_k, B_l.*

Proof: **1)** If z_i covers (a_k,b_l) where $(k,l) \in \{(m\text{-}1,n),(m,n\text{-}1),(m\text{-}1,n\text{-}1)\}$, then $z_i \in \mathcal{Q}_{m,n} \cap \mathcal{R}_{k,l}$, and $C_{k,l} = |Z_i|$. Suppose $C_{k,l} \neq |Z_i|$, then for (a_k,b_l) there are two possibilities: either $C_{k,l} > |Z_i|$ or $C_{k,l} < |Z_i|$. $C_{k,l} > |Z_i|$ implies the solution required another rectangle at the previous step, but since the recurrence is monotonically increasing this is impossible. If $C_{k,l} < |Z_i|$, then given the addition of z_i for (m,n) implies $z_i \notin \mathcal{Q}_{m,n} \cap \mathcal{R}_{k,l}$, but that contradicts our assumption.

2) If z_{i-1} covers (a_k,b_l), where $(k,l) \in \{(m\text{-}1,n),(m,n\text{-}1),(m\text{-}1,n\text{-}1)\}$, but not (a_m,b_n) then $z_i \notin \mathcal{Q}_{m,n} \cap \mathcal{R}_{k,l}$, and $C_{k,l} = |Z_{i-1}| = |Z_i|$-1. If we suppose $C_{k,l} \neq |Z_{i-1}|$, then either $C_{k,l} > |Z_{i-1}|$ or $C_{k,l} < |Z_{i-1}|$. If $C_{k,l} > |Z_{i-1}|$, then $C_{k,l} = |Z_{i-1}|$+1 $= |Z_i|$, and since Z_i is an optimal solution we have added another rectangle that must cover (a_k,b_l) and (a_m,b_n) in order to be a solution. However, this means $\exists$ $z_i \in \mathcal{Q}_{m,n} \cap \mathcal{R}_{k,l}$, which contradicts our assumption. Suppose $C_{k,l} < |Z_{i-1}|$, then to cover (a_m,b_n) we must add another rectangle, but that contradicts Z_i being an optimal solution since we have covered A_m, B_n with $|Z_i|$-1 rectangles.

3) Since $\nexists$ any rectangles z_i and z_{i-1} that cover both (a_m,b_n) and (a_k,b_l), where $(k,l) \in \{(m\text{-}1,n),(m,n\text{-}1),(m\text{-}1,n\text{-}1)\}$, then $\mathcal{Q}_{k,l} = \emptyset$ or $\mathcal{Q}_{m,n} = \emptyset$. By definition, if no rectangles cover the cells, then there is no solution for A_k, B_l or A_m, B_n. □

Theorem 3. *Constrained chain pair simplification, under the* *discrete Fréchet distance, is polynomially solvable, i.e. CPS-3F$^+ \in$ **P**.*

Proof: Since we have shown that CPS-3F$^+$ has an optimal substructure, given A, B, K', δ_1, δ_2, and δ_3, we can find an optimal K'' from our dynamic programming algorithm (Algorithm 1). Then we decide CPS-3F$^+$ by comparing whether $K' \leq K''$. □

Corollary 1. *Constrained chain pair simplification gives a factor 2-approximation to the chain pair simplification problem under the discrete Fréchet distance, i.e., CPS-3F$^+$ provides a 2-approximation of CPS-3F.*

Proof: Given two polygonal chains A, B, let K be an optimal solution from CPS-3F yielding the simplified chains A', B', and assume K' is an optimal solution for CPS-3F$^+$ yielding the simplified chains A'', B'', i.e. $K' = m_c(A'', B'')$.

Here, $K \leq K'$ because K is the minimum number of vertices possible and the moving cost for any pair A'', B'' is at least equal to $\max(|A''|, |B''|)$, where $K \leq |A''|$ or $K \leq |B''|$.

Now, we know that the moving cost for A', B' satisfies $m_c(A', B') \geq K'$ since K' is an optimal moving cost for A, B. Thus, a monotonically increasing moving cost in A', B' is at most $|A'| + |B'| - t$ for a t-walk. Now, $K' = m_c(A'', B'') \leq m_c(A', B') \leq |A'| + |B'| - t \leq 2\max(|A'|, |B'|) = 2K$. Thus, $K \leq K' \leq 2K$. □

3.2 Complexity

The time complexity of the algorithm is largely dependent on δ_1, δ_2, and δ_3 because they define the size and number of rectangles. We allow δ_1, δ_2, and δ_3 to be absorbed in the complexity because their values do not guarantee a specific number of rectangles to be considered, nor how large a given rectangle is.

We can easily bound the complexity between $O(mn)$ and $O(m^2n^2)$. If the values of δ_1, δ_2, and δ_3 are small then any cell will only have a small constant number of rectangles to consider and the algorithm runs in $O(mn)$ time, which is the case for most protein related data.

However, in the worst case, if δ_1, δ_2, and δ_3 are set larger than the lengths of the chains causing every $\mathcal{Q}_{i,j}$ to contain all possible rectangles between the two chains, then the complexity is $O(m^2n^2)$ given the largest $|\mathcal{D}|$ possible is mn. The optimal solution for CPS-3F$^+$ in this case is $K' = 1$, but would require $O(m^2n^2)$ time. δ_3 is the largest contributing variable to the running time because it determines the number of possible rectangles (size of $\mathcal{D}$).

Filtering steps could be added to watch for large δ values. With filtering, the time at any step could be kept to a constant (or logarithmic) value and the time complexity would remain close to $O(mn)$ or have an extra logarithmic factor dependent on δ_1, δ_2, and δ_3.

The space complexity also has similar bounds, requiring a minimum of $O(mn)$ space and a maximum of $O(m^2n^2)$ space, if $\mathcal{Q}$ is used naïvely and built beforehand. The recurrences themselves only require two rows of data for either $|A|$ or $|B|$, so the space complexity is linear to the size of the smaller chain (WLOG $O(n)$). However, this would require calculating $\mathcal{Q}_{i,j}$ at every step for the cell, which as discussed, could be expensive if the δ values are large.

3.3 Counter Example

We now briefly give an example proving that finding a solution to CPS-3F does not imply a solution to CPS-3F$^+$, and that having all solutions of CPS-3F$^+$ does not guarantee that one of the solutions will have a minimum K for CPS-3F.

The nodes of A and B are listed in Table 1, and Figure 4 shows the resulting polygonal chains in $\mathbb{R}^3$. The dotted lines show the nodes between A and B that are within δ_3 of each other. The settings for both CPS-3F and CPS-3F$^+$, given A, B, are $\delta_1 = \delta_2 = \delta_3 = 1$.

A	B
$a_1 = (0, 0.85, 2.4)$	$b_1 = (0, 2, 2.5)$
(0, 1, 1.5)	(0, 2, 1.5)
(0.4, 1, 0.6)	(0, 2.5, 1)
(0.9, 1.1, 0)	(0, 1.95, 0.5)
(0.6, 0.6, −0.3)	(0, 1, 0.5)
(0.5, 0, 0)	(0, 0, 0.5)
(1.5, 0, 0)	(0.75, 0.15, 1.1)
(2.45, 0, 0)	(1, 1, 1.5)
(3, 0.5, 0)	(1.3, 0.5, 0.75)
(2.5, 1, 0)	(1.6, 0.9, 0)
$a_{11} = (2.5, 2, 0)$	$b_{11} = (1.5, 1.8, -0.4)$

Table 1: The nodes for chains A and B in the counter-example.

This example has only nine pairs of nodes that correlate to rectangles in the dynamic programming solution (dotted lines). There are two paths that can be taken which are covered by rectangles. The minimum moving cost is equal to five and uses five nodes in the simplified chains, i.e. $K = 5$, $K' = 5$. The other path requires six pairs, or rectangles, to be used but only needs four vertices in each simplified chain, $K = 4$, $K' = 6$. Thus, a worse CPS-3F$^+$ solution (higher moving cost) may yield a better CPS-3F solution (lower number of vertices).

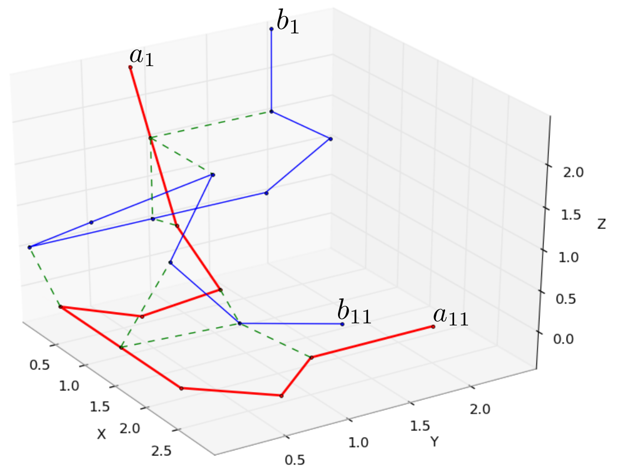

Figure 4: Two polygonal chains A and B. The dotted lines represent the nodes between A and B that are within δ_3 of each other.

In the next section we outline our algorithm and then walk through it using these example chains to demonstrate some of the intuition behind the algorithm. This explicitly shows that two paths are possible, and why we must choose which one is optimal based on the moving cost.

4 Dynamic Programming Algorithm

In this section we use the recurrences as a basis to find the optimal solution for a given set of inputs. Once the optimal solution is calculated we can easily decide CPS-3F$^+$. The algorithms presented assume many global or class variables outside of the functions. These variables are explained momentarily.

Since the recurrences only require the previous row and column for any decision, the function can be implemented as a simple iterative algorithm as opposed to a recursive one. Similar to the edit distance problem between two strings, only two rows at a time are needed, so the amount of memory can be reduced by only storing the two rows of $\mathcal{C}$ and $\mathcal{R}$ that the algorithm is currently using. This approach gives the optimal K' value, but in order to retrieve the actual K' pairs (via Algorithm 2), all rows of $\mathcal{C}$ and $\mathcal{R}$ must be stored.

Algorithm 1 assumes that $\mathcal{C}$ is an $m \times n$ matrix, where m, n are the sizes of A, B, respectively, and every cell is initialized to NULL. $\mathcal{R}$ is an $m \times n$ dynamic 3D jagged array where every element, $\mathcal{R}_{i,j}$, is its own array. We assume that $\mathcal{Q}$ has been calculated before this function is called. $\mathcal{Q}$ requires all rectangles to be enumerated, and for lower and upper bounds for each one to be set. Since this may vary between cells, $\mathcal{Q}$ is also a 3D jagged array like $\mathcal{R}$, and this makes assignments between the two easier.

For simplicity, to dynamically append to the end of an array or create a new row we use a generic function ‘ADD’ that takes the data structure as the first argument, and the value to append as the second. The method by which it works is context-sensitive to the type of data structure.

Algorithm 2 defines a recursive function that finds the rectangles through which the optimal path exists. The ‘Cost’ variable represents the pairs of nodes used in the minimum moving cost, and thus begins as the optimal K' value ($\mathcal{C}_{m,n}$) when the function is originally called. ‘i’ and ‘j’ are originally set to $|A|$ and $|B|$ respectively, and ‘CurrRect’ is set to NULL. The method assumes a jagged array ‘$Path$’ of size K' that must exist to store the rectangles of the path. The idea is that we know our optimal path is of size K', and thus we store all possible paths of that length in our $Path$ variable. ‘CurrRect’ simply refers to the current rectangle that the function is traversing.

This method does not tell us which rectangles connect with the rectangles at the next level. Therefore, in an actual implementation these stored rectangles should also have child or parent pointers to make it easier to follow the path. The pointers have been omitted here for algorithm clarity. An alternative to having pointers is to add an ‘if’ statement to check if $Path[1]$ has a value and then exit if it does. This means at least one optimal path has been found (there could be multiple).

Example Walkthrough

Here, we briefly walk through parts of Algorithm 1 for the example chains listed in Subsection 3.3. We let $\delta_1 = \delta_2 = \delta_3 = 1$ for our given chains A and B. Subsequently, we get nine pairs in $\mathcal{D} = \{(a_2,b_2),(a_2,b_5),(a_2,b_8),(a_3,b_5),(a_4,b_{10}),(a_6,b_6),(a_7,b_9),(a_7,b_{10}),(a_{10},b_{10})\}$, which are each represented by a rectangle. This example was designed so that each rectangle only covers its nearest neighbors in the chain. Thus, for example, (a_3,b_5) covers all nodes in our $A \times B$ grid from (a_2,b_4) to (a_4,b_6).

Algorithm 1 FIND-CPS-3F$^+$ $\rightarrow$ Computes the optimal moving cost, K', iteratively.

```
1:  procedure FIND-CPS-3F$^+$
2:    for $i \leftarrow 1, |A|$ do
3:      for $j \leftarrow 1, |B|$ do
4:        if $i = 1$ and $j = 1$ then
5:          $\mathcal{C}_{1,1} \leftarrow 1$
6:          $\mathcal{R}_{1,1} \leftarrow \mathcal{Q}_{1,1}$
7:        else
8:          Values, Rectangles $\leftarrow$ Array
9:          for each $(k,l) \in \{(i\text{-}1,j),(i,j\text{-}1),(i\text{-}1,j\text{-}1)\}$ do
10:           if $k > 0$ and $l > 0$ then
11:             for each $rect \in \mathcal{R}_{k,l}$ do
12:               if $(i,j) \in rect$ then
13:                 ADD(Values, $\mathcal{C}_{k,l}$ )
14:                 ADD(Rectangles, $rect$ )
15:               else
16:                 ADD(Values, $\mathcal{C}_{k,l} + 1$ )
17:                 ADD(Rectangles, NULL )
18:
19:         $\mathcal{C}_{i,j} \leftarrow \min($ Values $)$
20:         for $r \leftarrow 1, |$Values$|$ do
21:           if Values[ $r$ ] = $\mathcal{C}_{i,j}$ then
22:             if Rectangles[ $r$ ] = NULL then
23:               ADD($\mathcal{R}_{i,j}$, $\mathcal{Q}_{i,j} \setminus \mathcal{R}_{i,j}$ )
24:             else
25:               ADD($\mathcal{R}_{i,j}$, Rectangles[ r ] )
26:   return $\mathcal{C}_{m,n}$
```

Algorithm 2 GET-PATH-CPS-3F$^+$ $\rightarrow$ Return all simplified paths between the chains of optimal moving cost length.

```
1:  procedure GET-PATH-CPS-3F$^+$($i$, $j$, Cost, CurrRect)
2:    if CurrRect = NULL then
3:      for each $rect \in \mathcal{R}_{i,j}$ do
4:        ADD($Path$[ Cost ], $rect$ )
5:        GET-PATH-CPS-3F$^+$( $i$, $j$, Cost, $rect$ )
6:    for each $(k,l) \in \{(i\text{-}1,j),(i,j\text{-}1),(i\text{-}1,j\text{-}1)\}$ do
7:      if $k > 0$ and $l > 0$ and $\mathcal{C}_{k,l}$ != NULL then
8:        if $\mathcal{C}_{k,l}$ = $\mathcal{C}_{i,j}$ then
9:          GET-PATH-CPS-3F$^+$( $k$, $l$, Cost, CurrRect )
10:       else if $\mathcal{C}_{k,l}$ = $\mathcal{C}_{i,j} - 1$ then
11:         for each $rect \in \mathcal{R}_{k,l}$ do
12:           if $rect \notin Path$[ Cost ] then
13:             if $rect \notin Path$[ Cost - 1 ] then
14:               ADD($Path$[ Cost - 1 ], $rect$ )
15:             GET-PATH-CPS-3F$^+$( $k$, $l$, Cost - 1, $rect$ )
```

Protein Chain (B)	$\|B\|$	RMSD [16]	$d_{\mathcal{F}}(A,B)$	δ_1	δ_2	δ_3 [29]	$\|A'\|$ [29]	$\|B'\|$ [29]	K' [29]	$d_{\mathcal{F}}(A',B')$ [29]	δ_3	$\|A'\|$	$\|B'\|$	K'	$d_{\mathcal{F}}(A',B')$
1hfj.c	325	0.27	0.95	4	4	1	109	109	109	0.95	1	109	109	109	0.95
1qd1.b	325	2.81	22.65	4	4	50	109	109	109	24.96	21	117	126	150	20.70
1toh	325	2.91	22.06	4	4	60	109	110	110	23.39	21	149	130	178	20.54
4eca.c	325	1.10	5.55	4	4	17	109	109	109	7.96	6	110	111	111	5.97
1d9q.d	297	2.88	20.87	4	4	43	109	108	109	23.68	20	130	127	166	19.86
4eca.b	325	1.09	5.64	4	4	17	109	109	109	7.51	5	110	111	111	4.89
4eca.d	325	1.45	5.71	4	4	18	109	109	109	7.82	5	111	113	113	4.94

Table 2: Comparison of Algorithm SIMPLIFY [29] and FIND-CPS-3F$^+$ with 107j.a (Chain A) of length 325. $\delta_1 = \delta_2 = 4$, and δ_3 set to the minimal value.

There are only two paths through the rectangles. The path $p_1 = \langle r_{2,2}, r_{3,5}, r_{6,6}, r_{7,9}, r_{10,10}\rangle$ and the path $p_2 = \langle r_{2,2}, r_{2,5}, r_{2,8}, r_{4,10}, r_{7,10}, r_{10,10}\rangle$. When we begin $\mathcal{C}_{1,1} = 1$ and $\mathcal{R}_{1,1} = r_{2,2}$. When the path leaves the rectangle $r_{2,2}$ is at $\mathcal{C}_{k,3}$, where $k \in \{1,2,3\}$. For $k = 1$ the only rectangle to choose is $r_{2,5}$, but with $k = 2$ or $k = 3$ the corresponding $\mathcal{R}_{k,3} = \{r_{2,5}, r_{3,5}\}$.

The two paths then diverge and are straightforward until we evaluate $\mathcal{C}_{9,9}$ when we face the choice of which path is "better". Looking at the values of the previous three squares $\mathcal{C}_{8,9}, \mathcal{C}_{9,8}$, and $\mathcal{C}_{8,8}$, the path p_1 has fewer rectangles (four) compared to p_2, which has five, and thus we set $\mathcal{C}_{9,9} = 4$ and $\mathcal{R}_{9,9} = r_{7,9}$. The subsequent steps will all be equal, $\mathcal{C}_{9,10} = \mathcal{C}_{10,9} = \mathcal{C}_{10,10} = 5$, and $\mathcal{R}_{9,10} = \mathcal{R}_{10,9} = \mathcal{R}_{10,10} = r_{10,10}$ because the algorithm leaves rectangle $r_{7,9}$ and is now in the only rectangle left.

When we get to $\mathcal{C}_{11,11}$ the value remains at five because we are still covered by $r_{10,10}$, and thus we return five as the moving cost. Running Algorithm 2 returns the pairs of vertices that comprise the path p_1, which is $\langle(a_2,b_2),(a_3,b_5),(a_6,b_6),(a_7,b_9),(a_{10},b_{10})\rangle$.

5 Comparison of Results

We now present some results comparing our previous heuristic method SIMPLIFY [29] and the 2-approximation solution (Algorithm 1, 2) of CPS-3F$^+$. We present the results for chains with a similar length and then consider dissimilar chains of various lengths in order to vary the amount of simplification per chain. The algorithms were implemented in both Python and C# and more information about the FPACT software is available in Section 6. On an older 32 bit quad-core machine the algorithm took anywhere from a few seconds to several minutes depending on the parameters. The long runs were for simplifications in which δ_1, δ_2, and δ_3 were all set to large values (Tables 3 and 4).

We note that in both result sections, the RMSD values were taken from ProteinDBS, and thus the alignment length, or coverage, is not the full length of each chain [24]. This is especially true when discussing chains of different lengths in 5.2. This makes a straightforward comparison between the chains using both RMSD and the discrete Fréchet distance difficult. However, the results are mainly to compare CPS-3F$^+$ to our previous algorithm SIMPLIFY [29], and thus the coverage is not listed.

5.1 Similar Chain Length Comparisons

Using the same format as our previous results, we set $\delta_1 = \delta_2$ for simplicity and to ensure chains A', B' will have similar reduced lengths since nearly all are the same length initially. δ_3 is set to the minimum integer value that reduces the chains via CPS-3F$^+$ given δ_1, δ_2. The comparison tables in both cases are using the protein backbone 107j.a (protein A) and comparing it with seven other chains from the Protein DataBank: 1hfj.c, 1qd1.b, 1toh, 4eca.c, 1d9q.d, 4eca.b, 4eca.d. These seven chains were reported to be similar to 107j.a by the ProteinDBS software [24] (this took a few seconds searching the whole PDB, which contained over 30,000 protein backbones at that time). Previously, [16] used a heuristic algorithm based on the discrete Fréchet distance and showed that three of the seven chains were not actually similar to 107j.a, and ProteinDBS has subsequently updated their page to reflect this. 107j.a has 325 nodes along the backbone and all but one of the seven other chains do as well.

For the CPS-3F$^+$ algorithm, all chains are assumed to be aligned, and we use the alignments from our previous algorithm ALIGN [29]. In Table 2 we fixed $\delta_1 = \delta_2 = 4$ since the distance between two α-carbon atoms in the backbone is approximately ≈ 3.7 to 3.8 (angstroms). This value ensures that we will be simplifying the chains by a minimal amount. We can see that we get an approximate reduced length of $1/3$, which is what we would expect (since this distance will only use the neighboring nodes). The optimal algorithm allows for δ_3 to be much smaller than the heuristic because it can simplify the chains with a value often less than $d_{\mathcal{F}}(A,B)$, and hence $d_{\mathcal{F}}(A',B')$ is a lower value.

In Table 3 we vary δ_1 and δ_2 for different amounts of simplification and again set δ_3 to the minimum integer value that allows for simplification via CPS-3F$^+$. We keep $\delta_1 = \delta_2$ for simplicity and to ensure a similar reduced size for both chains. Here we have a more dramatic difference in δ_3 and in $d_{\mathcal{F}}(A',B')$ because of the greater simplification possibilities between A, A' and B, B' since δ_1, δ_2 are much larger. This demonstrates how CPS-3F$^+$ is able to simplify the two chains simultaneously while highlighting the similarities between the two chains. This is especially noticeable in that the discrete Fréchet distance between the simplified chains, $d_{\mathcal{F}}(A',B')$, is drastically less than that of the original chains, $d_{\mathcal{F}}(A,B)$.

We can see that the optimal results far exceed the heuristic approximation. If we look at 4eca.c in Table 3, the difference between the heuristic (11.73) and the optimal (2.90) is dra-

Protein Chain (B)	$\|B\|$	RMSD [16]	$d_{\mathcal{F}}(A,B)$	δ_1	δ_2	δ_3 [29]	$\|A'\|$ [29]	$\|B'\|$ [29]	K' [29]	$d_{\mathcal{F}}(A',B')$ [29]	δ_3	$\|A'\|$	$\|B'\|$	K'	$d_{\mathcal{F}}(A',B')$
1hfj.c	325	0.27	0.95	12	12	4	28	28	28	3.77	1	26	26	26	0.95
1qd1.b	325	2.81	22.65	15	15	33	16	17	17	22.64	12	21	23	24	11.94
1toh	325	2.91	22.06	16	16	44	17	16	17	27.24	13	22	19	22	12.80
4eca.c	325	1.10	5.55	12	12	12	28	28	28	11.73	3	27	27	27	2.90
1d9q.d	297	2.88	20.87	15	15	34	16	21	21	23.65	13	22	24	26	12.99
4eca.b	325	1.09	5.64	12	12	13	28	28	28	12.57	3	26	26	26	2.94
4eca.d	325	1.45	5.71	12	12	14	28	28	28	13.65	3	32	32	32	2.99

Table 3: Comparison of Algorithm SIMPLIFY [29] and FIND-CPS-3F$^+$ with 107j.a (Chain A) of length 325. $\delta_1 = \delta_2$, and δ_3 set to the minimal value.

Protein Chain (B)	$\|B\|$	RMSD	$d_{\mathcal{F}}(A,B)$	δ_1	δ_2	δ_3 [29]	$\|A'\|$ [29]	$\|B'\|$ [29]	K' [29]	$d_{\mathcal{F}}(A',B')$ [29]	δ_3	$\|A'\|$	$\|B'\|$	K'	$d_{\mathcal{F}}(A',B')$
3ntx.a	322	2.14	10.04	10	10	22	35	40	40	16.21	5	39	39	39	4.91
1wls.a	316	2.18	11.97	15	13	32	16	25	25	20.50	6	22	22	22	5.99
2eq5.a	215	2.72	22.35	8	6	39	53	43	53	23.47	19	58	53	66	18.91
2zsk.a	219	2.85	21.92	12	8	30	27	31	31	24.60	17	38	34	43	16.90
1zq1.a	363	3.01	23.38	10	12	40	36	37	37	28.30	19	51	53	56	18.47
3jq0.a	457	11.52	27.36	6	9	52	71	54	71	30.75	26	65	70	80	25.67
2fep.a	273	3.33	24.55	20	17	27	13	13	13	25.00	10	10	11	11	9.94

Table 4: Comparison of Algorithm SIMPLIFY [29] and FIND-CPS-3F$^+$with 107j.a (Chain A) of length 325, and various δ_1, δ_2, and δ_3 set to simplify both chains to a similar length.

matic. The optimal δ_3 for CPS-3F$^+$ is 3 to 4 times smaller than the heuristic in general, and the discrete Fréchet distance between A' and B' is smaller than the original distance between A, B.

The heuristic algorithm only allowed for a constant number of backtracking steps, which resulted in both chains being simplified to a similar number of vertices. With CPS-3F$^+$, we can see that the chains can vary greatly in the amount they simplify in order to have a minimum moving cost.

5.2 Varying Chain Length Comparisons

One aspect of chain pair simplification we have not exploited is simplifying the chains differently. Here, we look at chains that vary in length, are not aligned as well with the base chain, and that subsequently have a large discrete Fréchet distance. Table 4 shows these results. The values for δ_1 and δ_2 were chosen in an attempt to simplify both chains to a similar size via CPS-3F$^+$. This allows us to pull out the similarities of two chains that may be vastly different without simplification, yet still have some subset of nodes that align and compare well. For visualization purposes, it lets us see the overall subset similarity structure of the two chains. This method could prove useful for finding nodes in each chain that match well, i.e. they have a low moving cost and small discrete Fréchet distance.

The heuristic method SIMPLIFY [29] does not find similar optimal simplifications and results in a much higher moving cost and values of δ_3. The discrete Fréchet distance, consequently, is also much higher. As in our previous results, for both SIMPLIFY and CPS-3F$^+$, we pick δ_1 and δ_2, and then report the smallest integer value of δ_3 that worked for the respective algorithm.

The disparity between the number of vertices and the moving cost (K') is lessened if the chains are simplified in a similar fashion. When δ_1 and δ_2 are large, but δ_3 is small, it allows for these larger "hops" to be made. Thus, the simplified chains are similar in length, and the moving cost is a closer approximation to K. Using larger than minimum values for δ_3, we allow for greater flexibility in the simplification and yield a lower moving cost.

6 The Fréchet-based Protein Alignment & Comparison Toolkit

The FPACT libraries were designed for easy access to the algorithms by being modular and protein file format independent. The toolkit includes methods and classes such as the discrete Fréchet distance, ALIGN [29], SIMPLIFY [29], versions of CPS-3F$^+$ optimized for space or time efficiency (Algorithm 1), the CPS-3F$^+$ backtracking algorithm (Algorithm 2), and some other utility functions. The libraries will be updated with any future algorithms or results as well. All libraries are written and available in both C# or Python with Numpy.

We have also implemented a simple web-based application that uses these libraries. The web-based application runs within the Silverlight framework and can be used in any browser supporting the Silverlight or Moonlight runtime. The software is available to the public, thus providing the ability to align, compare, and simplify protein backbones with the discrete Fréchet distance without directly using the libraries [31].

FPACT, the web application, and relevant documentation about the research, can be found at the website http://www.cs.montana.edu/∼timothy.wylie/frechet/.

7 CPS-2H$^+$ is NP-complete

We now prove the complexity of CPS-2H$^+$ by a reduction from 3-SAT. Although they were not dealing with a geometric problem, the basic idea of this proof comes from [7].

Theorem 4. *CPS-2H$^+$ is* ***NP****-complete.*

Proof: First, CPS-2H$^+$ is in **NP** since the three distance conditions and the moving cost can all be verified in polynomial time.

For 3-SAT we are given a set of boolean variables $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and a set of clauses $\mathcal{C} = \{c_1, c_2, \ldots, c_M\}$ where each $c = (v_i \vee v_j \vee v_k)$ such that $v_i, v_j, v_k \in \mathcal{X} \vee \neg\mathcal{X}$. We assert that any clause may not contain both x_i and $\neg x_i$ since that clause would be true and can be discarded from our construction, and that either x_i or $\neg x_i$ must be in at least one clause. The problem is whether $\varphi = c_1 \wedge c_2 \wedge \cdots \wedge c_M$ in conjunctive normal form is satisfiable.

Since each clause, c_i, has three variables, WLOG we can enumerate them as c_i^k where $k \in \{0,1,2\}$. We can now create a point for each numbered variable k in clause c_i as $p_{i,k} = (i, i^2, k \cdot \varepsilon)$ where $k \in \{0,1,2\}$ and $0 < \varepsilon < 0.5$. Now we construct two polygonal chains A and B each with $(3M+N)$ vertices, such that φ is satisfiable $\iff$ A, B can be simplified into A', B', respectively, and $d_H(A, A') \leq 2\varepsilon$, $d_H(B, B') \leq 2\varepsilon$, $d_{\mathcal{F}}(A', B') = 0$, $|A'| = |B'| = M + N$ and the moving cost $K' = M + N$. Given our construction, K' is equivalent to the number of vertices in each chain and the number of walks in the Fréchet alignment.

To begin we define a sequence of N points $q_j = (0, j, 10)$, where $1 \leq j \leq N$, as separators. Now for each $x_i \in \mathcal{X}$, we construct two sequences S_i and S_i^*. Let $c_{i_1}, \ldots, c_{i_u}$ be the sequence of clauses that contain x_i, and $c_{j_1}, \ldots, c_{j_v}$ the clauses containing $\neg x_i$. $S_i = \langle c_{i_1}, \ldots, c_{i_u}, c_{j_1}, \ldots, c_{j_v} \rangle$ and $S_i^* = \langle c_{j_1}, \ldots, c_{j_v}, c_{i_1}, \ldots, c_{i_u} \rangle$. Note that a clause c_i appears exactly three times in $\cup_{j=1}^N S_j$ and $\cup_{j=1}^N S_j^*$ since there are three literals in each clause, and we enumerate those literals as c_i^k where $k \in \{0,1,2\}$. Now, for each literal $x_i \in \mathcal{X}$, we convert S_i, S_i^* into the sequences T_i, T_i^* where every clause label is replaced by the point $p_{j,k}$ where $k \in \{0,1,2\}$ and k corresponds to the enumerated c_j^k in the clause of variable x_i. Since the order of the enumeration of the $p_{j,k}$ points is arbitrary in a clause, we will assume that they always occur in order $(p_{j,0}, p_{j,1}, p_{j,2})$.

Let $A = \langle T_1, q_1, T_2, q_2, \ldots, T_N, q_N \rangle$ and $B = \langle T_1^*, q_1, T_2^*, q_2, \ldots, T_N^*, q_N \rangle$.

We first show the forward direction. Suppose there exists a sequence of boolean assignments $\mathcal{Z} = \langle z_1, z_2, \ldots, z_N \rangle$ such that $x_i = z_i$ $(1 \leq i \leq N)$ satisfies φ. If $z_i = 1$, then T_i and T_i^* simplify to $T_i' = p_{i_1}, \ldots, p_{i_u}$ and $T_i^{*'} = p_{i_1}, \ldots, p_{i_u}$ respectively. However, if $z_i = 0$ we simplify both to the other sequence, $T_i' = p_{j_1}, \ldots, p_{j_v}$ and $T_i^{*'} = p_{j_1}, \ldots, p_{j_v}$. To make $d_H(A, A') \leq 2\varepsilon$, $d_H(B, B') \leq 2\varepsilon$, $d_{\mathcal{F}}(A', B') = 0$, and $|A'| = |B'| = M + N$, $K' = M + N$, we keep exactly one point among each triple of points $p_{j,k}$ $(k \in \{0,1,2\})$, and remove the remaining ones. Then, after the deletion, $A' = \langle T_1', q_1, T_2', q_2, \ldots, T_N', q_N \rangle$ is equivalent to $B' = \langle T_1^{*'}, q_1, T_2^{*'}, q_2, \ldots, T_N^{*'}, q_N \rangle$. The reason that $d_H(A, A') \leq 2\varepsilon$ and $d_H(B, B') \leq 2\varepsilon$ is because each point selected in $p_{j,k}$, $k \in \{0,1,2\}$, is at most a distance of 2ε from the removed points in the same triple (clause).

We now prove the opposite direction. Suppose that A, B are simplified into A'', B'', respectively, by removing some points in $\{p_{i,j} | 1 \leq i \leq M, 0 \leq j \leq 2\}$ such that $d_H(A, A'') \leq 2\varepsilon$, $d_H(B, B'') \leq 2\varepsilon$, $d_{\mathcal{F}}(A'', B'') = 0$, and $|A''| = |B''| = K' = M + N$. We know $d(p_{i,k}, p_{j,l}) > 1$ and $d(q_i, q_j) \geq 1$ where $i \neq j$. The conditions $d_H(A, A'') \leq 2\varepsilon$ and $d_H(B, B'') \leq 2\varepsilon$ imply that we can only remove points in $\{p_{i,j} | 1 \leq i \leq M, 0 \leq j \leq 2\}$ while leaving at least one point in each triple for A'', B'' for each clause, i.e. one $p_{i,k}$, $0 \leq k \leq 2$. Since c_i can not contain both x_j and $\neg x_j$, there will be only one point p_a, for some a, on the subchain between q_r and q_{r+1} on A or B. The condition that $d_{\mathcal{F}}(A', B') = 0$ implies that in A'' and B'' we must keep the same point among the triple $\{p_{i,j} | 1 \leq i \leq M, 0 \leq j \leq 2\}$. Finally, since $|A''| = |B''| = K' = M+N$, to make $d_{\mathcal{F}}(A', B') = 0$ we must use all separator points (q's).

Let T_i'' and $T_i^{*''}$ be the subchains in A'' and B'' obtained from simplifying T_i and T_i^* in A and B, respectively. If T_i'' is empty, we can arbitrarily assign a value to x_i. However, if T_i'' is not empty, this implies that T_i'' and $T_i^{*''}$ have the same size and $d_{\mathcal{F}}(T_i'', T_i^{*''}) = 0$. If T_i'' is not empty and it is a subsequence of $p_{i_1}, \ldots, p_{i_u}$, then we assign $x_i = 1$. If T_i'' is not empty and it is a subsequence of $p_{j_1}, \ldots, p_{j_v}$, then we assign $x_i = 0$. Thus, we can see that φ is satisfied by the assignments to the variables $x_i \in \mathcal{X}$.

Finally, we note that this reduction is polynomial and takes linear time based on the length of φ. □

7.1 Examples

Let $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$, $\mathcal{C} = \{c_1, c_2, c_3, c_4\}$, and $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$ where the clauses are assumed to be labeled in order. Thus, $N = 4$ and $M = 4$.

Now we construct the sequences. $S_1 = \langle c_1, c_2 \rangle$, $S_1^* = \langle c_2, c_1 \rangle$, $S_2 = \langle c_4, c_1, c_3 \rangle$, $S_2^* = \langle c_1, c_3, c_4 \rangle$, $S_3 = \langle c_1, c_2, c_3, c_4 \rangle$, $S_3^* = \langle c_2, c_3, c_4, c_1 \rangle$, $S_4 = \langle c_3, c_2, c_4 \rangle$, and $S_4^* = \langle c_2, c_4, c_3 \rangle$. The conversion to the T_i, T_i^* sequences merely replaces the label of the clause c_j with the point $p_{j,k}$ such that $k \in \{0,1,2\}$ is the place of variable x_i. For readability, we omit the comma in the subscript. $T_1 = \langle p_{10}, p_{20} \rangle$, $T_1^* = \langle p_{20}, p_{10} \rangle$, $T_2 = \langle p_{40}, p_{11}, p_{30} \rangle$, $T_2^* = \langle p_{11}, p_{30}, p_{40} \rangle$, $T_3 = \langle p_{12}, p_{21}, p_{31}, p_{41} \rangle$, $T_3^* = \langle p_{21}, p_{31}, p_{41}, p_{12} \rangle$, $T_4 = \langle p_{32}, p_{22}, p_{42} \rangle$, and $T_4^* = \langle p_{22}, p_{42}, p_{32} \rangle$.

Then we construct $A = \langle p_{10}, p_{20}, q_1, p_{40}, p_{11}, p_{30}, q_2, p_{12}, p_{21}, p_{31}, p_{41}, q_3, p_{32}, p_{22}, p_{42}, q_4 \rangle$ and $B = \langle p_{20}, p_{10}, q_1, p_{11}, p_{30}, p_{40}, q_2, p_{21}, p_{31}, p_{41}, p_{12}, q_3, p_{22}, p_{42}, p_{32}, q_4 \rangle$ where all points $q_j = (0, j, 10)$ for $j = 1, 2, 3, 4$. Note that $|A| = |B| = K' = 3M + N = 16$.

Suppose we set $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$. Then $A' = B' = \langle p_{10}, q_1, p_{40}, q_2, p_{31}, q_3, p_{22}, q_4 \rangle$ after we remove points where we already had a point from the triple. Thus, there are several possible simplified chains, depending on which points are removed. This gives $|A| = |B| = K' = M + N = 8$. Also notice that it is not necessary that the separating points and clauses (p_{ik}'s) are visited in order.

Figure 5 shows the points in the construction as well as one possible simplification. Each p_{ik} represents three points that have z values 0, ε, and 2ε for the clause c_i. Notice this does not visit the nodes in the same order as our example, but is easier to see the general idea.

Now we show a simple example that is not satisfiable. Let $\varphi = (x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1)$.

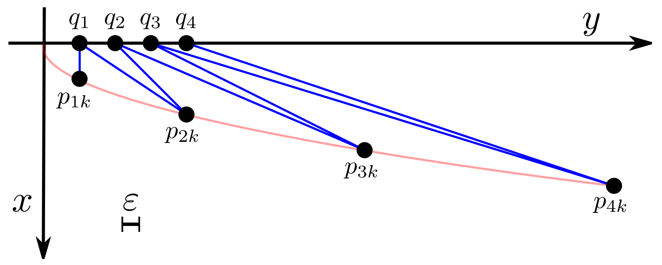

Figure 5: CPS-2H$^+$ reduction example viewed from the positive z axis. Each p_{ik} represents the three points for clause c_i which vary only in z. The line through only the p_{ik} points is $y = x^2$ to show where the clause points are placed.

$S_1 = \langle c_1, c_1, c_1, c_2, c_2, c_2\rangle$ and $S_1^* = \langle c_2, c_2, c_2, c_1, c_1, c_1\rangle$. Then $T_1 = \langle p_{10}, p_{11}, p_{12}, p_{20}, p_{21}, p_{22}\rangle$ and $T_1^* = \langle p_{20}, p_{21}, p_{22}, p_{10}, p_{11}, p_{12}\rangle$.

Now $A = \langle p_{10}, p_{11}, p_{12}, p_{20}, p_{21}, p_{22}, q_1\rangle$ and $B = \langle p_{20}, p_{21}, p_{22}, p_{10}, p_{11}, p_{12}, q_1\rangle$. We end up with almost identical chains whether x_1 is 0 or 1. Let $x_1 = 1$, then $A' = \langle p_{10}, q_1\rangle$ and $B' = \langle p_{10}, q_1\rangle$ (any of the p_{1k}'s could have been chosen). Since none of the points from clause c_2 are in the simplified chains, $d_H(A, A') > 2\varepsilon$ and $d_H(B, B') > 2\varepsilon$ and thus it is not a valid CPS-2H$^+$ solution. The same result happens should $x_1 = 0$. Any solution will require $|A| = |B| = K' \geq M + N + 1$ and thus violate our reduction condition, meaning φ was not satisfiable.

8 Concluding Remarks

In this paper we have shown that the restricted version of the chain pair simplification problem under the discrete Fréchet distance (CPS-3F$^+$) is polynomially solvable. We then presented algorithms to find K', the minimum moving cost between chains A' and B', and a backtracking method to return the vertices of the simplified chains. Further, we proved that CPS-3F$^+$ is a 2-approximation for CPS-3F and looked empirically at the benefits and possible use-cases of CPS-3F$^+$. Along these lines the FPACT libraries are now available to use these algorithms as well as some of the past methods based on the discrete Fréchet distance [31]. We then showed that the restricted version of the chain pair simplification problem under the Hausdorff distance (CPS-2H$^+$) is **NP**-complete.

There are still several issues that need to be addressed to fully realize the benefits of the discrete Fréchet distance in comparing polygonal curves, and specifically protein backbones:

(1) Is the chain pair simplification problem under the discrete Fréchet distance (CPS-3F) **NP**-complete?

(2) The ALIGN [29] running time still needs to be improved since all comparisons, including CPS-3F$^+$, rely on the two polygonal chains being aligned. Can the alignment be further simplified?

(3) More generally, the question of whether it is theoretically possible to design a practical PTAS (global structure-structure) alignment algorithm based on the discrete Fréchet distance needs to be answered.

(4) For protein backbone structures, can we exploit the physical properties of the chains in order to hasten alignment, comparison, or simplification, e.g. the fixed distance between each node (α-carbon atom), and the minimum distance two atoms can be in relation to each other (they can not touch)?

(5) Dynamic Time Warping (DTW) shares many similarities with the Fréchet distance and may prove useful in conjunction with the discrete Fréchet distance. More research needs to be done exploring this relationship and the applications to protein alignment and simplification.

(6) There are several possible strategies to keep CPS-3F$^+$ running in $O(mn)$ time, including filtering redundant rectangles or applying similar strategies used in DTW such as the Itakura Parallelogram [15] or the Sakoe-Chiba Band [22]. Can these be identified and integrated into our algorithm?

Acknowledgments

This research is partially supported by NSF under grant DMS-0901034 and by NSF of China under project 60928006.

References

[1] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes (extended abstract). In *Proceedings of the 7th Annual Symposium on Computational Geometry (SoCG'91)*, pages 186–193, 1991.
[2] H. Alt and M. Godau. Measuring the resemblance of polygonal curves. In *Proceedings of the 8th Annual Symposium on Computational Geometry (SoCG'92)*, pages 102–109, 1992.
[3] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995.
[4] H. Alt, C. Knauer, and C. Wenk. Matching polygonal curves with respect to the Fréchet distance. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS'01)*, pages 63–74, 2001.
[5] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, C. Wenk. Fréchet Distance for Curves, Revisited. In *Proceedings of the 14th conference on Annual European Symposium - Volume 14 (ESA'06)*, pages 52–63, 2006.
[6] S. Bereg, M. Jiang, W. Wang, B. Yang and B. Zhu. Simplifying 3D polygonal chains under the discrete Fréchet distance. In *Proceedings of the 8th Latin American Theoretical Informatics Symposium (LATIN'08)*, LNCS 4957, pages 630–641, 2008.
[7] Z. Chen, B. Fu, and B. Zhu. The Approximability of the Exemplar Breakpoint Distance Problem. In *Proceedings of the 2nd international conference on Algorithmic Aspects in Information and Management (AAIM'06)*, LNCS 4041, pages 291–302, 2006.
[8] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34:200-208, 1987.
[9] L. Conte, B. Ailey, T. Hubbard, S. Brenner, A. Murzin and C. Chothia. SCOP: a structural classification of protein database. *Nucleic Acids Research*, **28**:257-259, 2000.
[10] T. Eiter and H. Mannila. Computing discrete Fréchet distance. *Technical Report CD-TR 94/64, Information Systems Department, Technical University of Vienna*, 1994.
[11] M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Mathematico di Palermo*, **22**:1-74, 1906.
[12] F. Hausdorff. *Grundzge der mengenlehre*. Von Veit, Leipzig, 1914.
[13] L. Holm and J. Park. DaliLite workbench for protein structure comparison. *Bioinformatics*, **16**:566-567, 2000.
[14] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, **233**:123-138, 1993.
[15] F. Itakura. Minimum Prediction Residual Principle Applied to Speech Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **23**:67-72, 1975.
[16] M. Jiang, Y. Xu and B. Zhu. Protein structure-structure alignment with discrete Fréchet distance. *Journal of Bioinformatics and Computational Biology*, **6**:51-64, 2008.
[17] C. Mauzy and M. Hermodson. Structural homology between rbs repressor and ribose binding protein implies functional similarity, *Protein Science*, **1**:843-849, 1992.
[18] J. R. Munkres. *Topology*. Prentice Hall, Incorporated, 2000.

[19] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, **48**:443-453, 1970.

[20] C. Orengo, A. Michie, S. Jones, D. Jones, M. Swindles and J. Thornton. CATH—a hierarchic classification of protein domain structures. *Structure*, **5**:1093-1108, 1997.

[21] A. Oritz, C. Strauss and O. Olmea. MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison. *Protein Science*, **11**:2606-2621, 2002.

[22] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **26**:43-49, 1978.

[23] I. Shindyalov and P. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, **11**:739-747, 1998.

[24] C.-R. Shyu, P.-H. Chi, G. Scott, and D. Xu. ProteinDBS: a real-time retrieval system for protein structure comparison. *Nucleic Acids Research*, 32:W572–575, 2004.

[25] W. Taylor and C. Orengo. Protein structure alignment. *Journal of Molecular Biology*, **208**:1-22, 1989.

[26] S. Vasilache, N. Mirshahi, S. Ji, J. Mottonen, D. J. Jacobs, and K. Najarian. A Signal Processing Method to Explore Similarity in Protein Flexibility. *Advances in Bioinformatics*, Volume 2010 (2010).

[27] T. K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, **4(1)**:52-57

[28] C. Wenk. Shape Matching in Higher Dimensions. *PhD thesis, Freie Universitaet Berlin*, 2002.

[29] T. Wylie, J. Luo and B. Zhu. A Practical Solution for Aligning and Simplifying Pairs of Protein Backbones Under the Discrete Fréchet Distance. In *Proceedings of the 11th International Conference on Computational Science and Its Applications (ICCSA'11)*, LNCS 6784, pages 74–83, 2011.

[30] T. Wylie and B. Zhu. A Polynomial Time Solution for Protein Chain Pair Simplification Under the Discrete Fréchet Distance. In *Proceedings of the 2012 International Symposium on Bioinformatics Research and Applications (ISBRA'12)*, LNBI 7292, pages 287–298, 2012.

[31] T. Wylie. FPACT: The Fréchet-based Protein Alignment & Comparison Toolkit. *http://www.cs.montana.edu/∼timothy.wylie/frechet*, 2012.

[32] J.-M. Yang and C.-H. Tung. Protein structure database search and evolutionary classification. *Nucleic Acids Research*, **34**:3646-3659, 2006.

[33] B. Zhu. Protein local structure alignment under the discrete Fréchet distance. *Journal of Computational Biology*, **14(10)**:1343-1351, 2007.

Tim Wylie is a Computer Science doctoral candidate at Montana State University. He obtained B.S. degrees in Mathematics and Computer Science in 2004 from Harding University. He earned an M.S. in Computer Science from the University of Montana in 2010. He currently works with Dr. Binhai Zhu in the areas of computational geometry, algorithms, and bioinformatics.

Binhai Zhu Binhai Zhu is currently a professor in computer science at Montana State University, USA. He obtained his Ph.D. in Computer Science from McGill University, Canada, in 1994. He was a post-doctoral research associate at Los Alamos National Laboratory, USA from 1994 to 1996. From 1996 to 2000, he was an assistant professor at City University of Hong Kong. He has been at Montana State University since 2000 (associate professor until 2006, professor since 2006). Professor Zhu's research interests are geometric computing, biological/geometric modeling, bioinformatics, and combinatorial optimization. He has published over 130 papers in these areas.

Summary of Changes for TCBBSI-2012-08-0204

We thank the reviewers again for their detailed comments. We have taken all reviewer feedback into consideration and made all appropriate changes. Below we list the minor revisions requested for the paper.

- We have changed the sentence that was said to be overly strong. It now reads that the discrete measure may be more appropriate than the continuous for this application.
- For the concern about the number of points being used to approximate the continuous version, we have changed the sentence to state that enough "evenly sampled" points will allow an arbitrarily close approximation.
- We have added a couple of sentences to address our inequalities on page 4 in the moving cost section. This explains why the inequalities hold given the definitions in the background section.
- For the concerns about the webpage location, we have ensured that a mirror of this site will remain for the next couple of years. Further, Dr. Zhu currently has, and will maintain a link to the website should it be moved to another URL. Thus, any user will always be able to find the current site through his webpage.