On the Chain Pair Simplification Problem[⋆]

Chenglin Fan[1], Omrit Filtser[2], Matthew J. Katz[2], Tim Wylie[3], and Binhai Zhu[1]

[1] Montana State University
Bozeman, MT, 59717-3880 USA
chenglin.fan@msu.montana.edu, bhz@cs.montana.edu
[2] Ben-Gurion University of the Negev
Beer-Sheva 84105, Israel
{omritna,matya}@cs.bgu.ac.il
[3] The University of Texas-Pan American
Edinburg, TX, 78539 USA
wylietr@utpa.edu

Abstract. The problem of efficiently computing and visualizing the structural resemblance between a pair of protein backbones in 3D has led Bereg et al. [4] to pose the Chain Pair Simplification problem (CPS). In this problem, given two polygonal chains A and B of lengths m and n, respectively, one needs to simplify them simultaneously, such that each of the resulting simplified chains, A' and B', is of length at most k and the discrete Fréchet distance between A' and B' is at most δ, where k and δ are given parameters. In this paper we study the complexity of CPS under the discrete Fréchet distance (CPS-3F), i.e., where the quality of the simplifications is also measured by the discrete Fréchet distance. Since CPS-3F was posed in 2008, its complexity has remained open. In this paper, we prove that CPS-3F is actually polynomially solvable, by presenting an $O(m^2 n^2 \min\{m, n\})$ time algorithm for the corresponding minimization problem. On the other hand, we prove that if the vertices of the chains have integral weights then the problem is weakly NP-complete.

1 Introduction

Polygonal curves play an important role in many applied areas, such as 3D modeling in computer vision, map matching in GIS, and protein backbone structural alignment and comparison in computational biology. Many different methods exist to compare curves in these (and in many other) applications, where one of the more prevalent methods is the Fréchet distance [8].

The *Fréchet distance* is often described by an analogy of a man and a dog connected by a leash, each walking along a curve from its starting point to its end point. Both the man and the dog can control their speed but they are not allowed

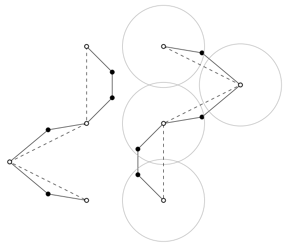
[⋆] A complete version including the one-sided cases and empirical results can be found at http://arxiv.org/abs/1409.2457.
Work by O. Filtser has been partially supported by the Lynn and William Frankel Center for Computer Sciences. Work by M. Katz and O. Filtser has been partially supported by grant 1045/10 from the Israel Science Foundation.

to backtrack. The Fréchet distance between the two curves is the minimum length of a leash that is sufficient for traversing both curves in this manner.

The *discrete Fréchet distance* is a simpler version, where, instead of continuous curves, we are given finite sequences of points, obtained, e.g., by sampling the continuous curves, or corresponding to the vertices of polygonal chains. Now, the man and the dog only hop monotonically along the sequences of points. The discrete Fréchet distance is considered a good approximation of the continuous distance.

One promising application of the discrete Fréchet distance has been protein backbone comparison. Within structural biology, polygonal curve alignment and comparison is a central problem in relation to proteins. Proteins are usually studied using RMSD (Root Mean Square Deviation), but recently the discrete Fréchet distance was used to align and compare protein backbones, which yielded favourable results in many instances [9, 10]. In this application, the discrete version of the Fréchet distance makes more sense, because by using it the alignment is done with respect to the vertices of the chains, which represent α-carbon atoms. Applying the continuous Fréchet distance will result in mapping of arbitrary points, which is not meaningful biologically.

There may be as many as 500$\sim$600 α-carbon atoms along a protein backbone, which are the nodes (i.e., points) of our chain. This makes efficient computation essential, and is one of the reasons for considering simplification. In general, given a chain A of n vertices, a simplification of A is a chain A' such that A' is "close" to A and the number of vertices in A' is significantly less than n. The problem of simplifying a 3D polygonal chains under the discrete Fréchet distance was first addressed by Bereg et al. [4].


(a) Simplifying the chains independently does not necessarily preserve the resemblance between them.

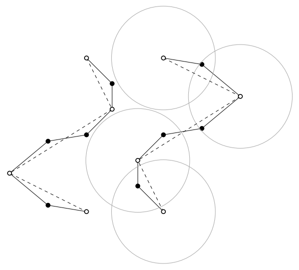

(b) A simplification of both chains that preserves the resemblance between them.

Fig. 1: Independent simplification vs. simultaneous simplification. Each chain simplification consists of 4 vertices (marked by empty circles) chosen from the corresponding chain. The unit disks illustrate the Fréchet distance between the right chain in each of the figures and its corresponding simplification; their radius in (b) is larger.

Simplifying two aligned chains independently does not necessarily preserve the resemblance between the chains; see Figure 1. Thus, the following question arises: Is it possible to simplify both chains in a way that will retain the resemblance between them? This question has led Bereg et al. [4] to pose the Chain Pair Simplification problem (CPS). In this problem, the goal is to simplify both chains simultaneously, so that the discrete Fréchet distance between the resulting simplifications is bounded. More precisely, given two chains A and B of lengths m and n, respectively, an integer k and three real numbers δ_1,δ_2,δ_3, one needs to find two chains A',B' with vertices from A,B, respectively, each of length at most k, such that $d_1(A,A') \leq \delta_1$, $d_2(B,B') \leq \delta_2$, $d_{dF}(A',B') \leq \delta_3$ (d_1 and d_2 can be any similarity measures and d_{dF} is the discrete Fréchet distance). When the chains are simplified using the Hausdorff distance, i.e., d_1, d_2 is the Hausdorff distance (CPS-2H), the problem becomes **NP**-complete [4]. However, the complexity of the version in which d_1, d_2 is the discrete Fréchet distance (CPS-3F) has been open since 2008.

Related work The Fréchet distance and its variants have been studied extensively in the past two decades. Alt and Godau [2] gave an $O(mn \log mn)$-time algorithm for computing the Fréchet distance between two polygonal curves of lengths m and n. This result in the plane was recently improved by Buchin et al [5]. The discrete Fréchet distance was originally defined by Eiter and Mannila [7], who also presented an $O(mn)$-time algorithm for computing it. A slightly sub-quadratic algorithm was given recently by Agarwal et al. [1].

As mentioned earlier, Bereg et al. [4] were the first to study simplification problems under the discrete Fréchet distance. They considered two such problems. In the first, the goal is to minimize the number of vertices in the simplification, given a bound on the distance between the original chain and its simplification, and, in the second problem, the goal is to minimize this distance, given a bound k on the number of vertices in the simplification. They presented an $O(n^2)$-time algorithm for the former problem and an $O(n^3)$-time algorithm for the latter problem, both using dynamic programming, for the case where the vertices of the simplification are from the original chain. (For the arbitrary vertices case, they solve the problems in $O(n \log n)$ time and in $O(kn \log n \log(n/k))$ time, respectively.) Driemel and Har-Peled [6] showed how to preprocess a polygonal curve in near-linear time and space, such that, given an integer $k > 0$, one can compute a simplification in $O(k)$ time which has $2k-1$ vertices of the original curve and is optimal up to a constant factor (w.r.t. the continuous Fréchet distance), compared to any curve consisting of k arbitrary vertices.

For the chain pair simplification problem (CPS), Bereg et al. [4] proved that CPS-2H is **NP**-complete, and conjectured that so is CPS-3F. Wylie et al. [10] gave a heuristic algorithm for CPS-3F, using a greedy method with backtracking, and based on the assumption that the (Euclidean) distance between adjacent α-carbon atoms in a protein backbone is almost fixed. More recently, Wylie and Zhu [11] presented an approximation algorithm with approximation ratio 2 for the optimization version of CPS-3F. Their algorithm actually solves the optimization version of a related problem called CPS-3F^+, it uses dynamic programming and its running time is between $O(mn)$ and $O(m^2n^2)$ depending on the input simplification parameters.

Some special cases of CPS-3F have recently been studied. Motivated by the need to reduce sensitivity to outliers when comparing curves, Ben Avraham et al. [3] studied the discrete Fréchet distance with shortcuts problem. Both variants of the shortcuts problem can be solved in subquadratic time.

Our results In Section 3, we resolve the question concerning the complexity of CPS-3F by proving that it is polynomially solvable, contrary to what was believed. We do this by presenting a polynomial-time algorithm for the corresponding optimization problem. In Section 4 we devise a sophisticated $O(m^2n^2\min\{m,n\})$-time dynamic programming algorithm for the minimization problem of CPS-3F. Besides being interesting from a theoretical point of view, only after developing (and implementing) this algorithm, were we able to apply the CPS-3F minimization problem to datasets from the Protein Data Bank (PDB), see the full version for the actual empirical results. Finally, in Section 5 we prove that the problem is weakly NP-complete if the vertices of the chains carry integral weights.

2 Preliminaries

Let $A = (a_1 \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$ be two sequences of m and n points, respectively, in $\mathbb{R}^k$. The discrete Fréchet distance $d_{dF}(A, B)$ between A and B is defined as follows. Fix a distance $\delta > 0$ and consider the Cartesian product $A \times B$ as the vertex set of a directed graph G_δ whose edge set is

$$\begin{aligned} E_\delta = &\big\{\big((a_i, b_j), (a_{i+1}, b_j)\big) \mid d(a_i, b_j), d(a_{i+1}, b_j) \le \delta\big\} \cup \\ &\big\{\big((a_i, b_j), (a_i, b_{j+1})\big) \mid d(a_i, b_j), d(a_i, b_{j+1}) \le \delta\big\} \cup \\ &\big\{\big((a_i, b_j), (a_{i+1}, b_{j+1})\big) \mid d(a_i, b_j), d(a_{i+1}, b_{j+1}) \le \delta\big\}\,. \end{aligned}$$

Then $d_{dF}(A, B)$ is the smallest $\delta > 0$ for which (a_m, b_n) is reachable from (a_1, b_1) in G_δ.

The chain pair simplification problem (CPS) is formally defined as follows.

Problem 1 (Chain Pair Simplification).
Instance: Given a pair of polygonal chains A and B of lengths m and n, respectively, an integer k, 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 $d_1(A, A') \le \delta_1$, $d_2(B, B') \le \delta_2$, and $d_{dF}(A', B') \le \delta_3$?

When $d_1 = d_2 = d_H$, the problem is **NP**-complete and is called CPS-2H, and when $d_1 = d_2 = d_{dF}$, the problem is called CPS-3F.

3 Chain Pair Simplification (CPS-3F)

We now turn our attention to CPS-3F, which we show to be polynomially solvable in this section. We comment that the running time and space for this solution is $O(m^3n^3 \min\{m,n\})$ and $O(m^3n^3)$ respectively, hence this solution is impractical for most of the real protein chains (with m,n as large as 500-600). Nonetheless, this first solution is easier to understand and can be considered as a warm-up. We will present a much better (but more sophisticated) solution in the next section.

We present an algorithm for the minimization version of CPS-3F. That is, we compute the minimum integer k^*, such that there exists a "walk", as above, in which each of the dogs makes at most k^* hops. The answer to the decision problem is "yes" if and only if $k^* < k$.

Returning to the analogy of the man and the dog, we can extend it as follows. Consider a man and his dog connected by a leash of length δ_1, and a woman and her dog connected by a leash of length δ_2. The two dogs are also connected to each other by a leash of length δ_3. The man and his dog are walking on the points of a chain A and the woman and her dog are walking on the points of a chain B. The dogs may skip points. The problem is to determine whether there exists a "walk" of the man and his dog on A and the woman and her dog on B, such that each of the dogs steps on at most k points.

Overview of the algorithm We say that (a_i, a_p, b_j, b_q) is a *possible* configuration of the man, woman and the two dogs on the paths A and B, if $d(a_i, a_p) \leq \delta_1$, $d(b_j, b_q) \leq \delta_2$ and $d(a_p, b_q) \leq \delta_3$. Notice that there are at most m^2n^2 such configurations. Now, let G be the DAG whose vertices are the possible configurations, such that there exists a (directed) edge from vertex $u = (a_i, a_p, b_j, b_q)$ to vertex $v = (a_{i'}, a_{p'}, b_{j'}, b_{q'})$ if and only if our gang can move from configuration u to configuration v. That is, if and only if $i \leq i' \leq i+1$, $p \leq p'$, $j \leq j' \leq j+1$, and $q \leq q'$. Notice that there are no cycles in G because backtracking is forbidden. For simplicity, we assume that the first and last points of A' (resp., of B') are a_1 and a_m (resp., b_1 and b_n), so the initial and final configurations are $s = (a_1, a_1, b_1, b_1)$ and $t = (a_m, a_m, b_n, b_n)$, respectively. (It is easy, however, to adapt the algorithm below to the case where the initial and final points of A' and B' are not specified, see remark below.) Our goal is to find a path from s to t in G. However, we want each of our dogs to step on at most k points, so, instead of searching for any path from s to t, we search for a path that minimizes the value $max\{|A'|, |B'|\}$, and then check if this value is at most k.

For each edge $e = (u, v)$, we assign two weights, $w_A(e), w_B(e) \in \{0, 1\}$, in order to compute the number of hops in A' and in B', respectively. $w_A(u, v) = 1$ if and only if the first dog jumps to a new point between configurations u and v (i.e., $p < p'$), and, similarly, $w_B(u, v) = 1$ if and only if the second dog jumps to a new point between u and v (i.e., $q < q'$). Thus, our goal is to find a path P from s to t in G, such that $max\{\sum_{e \in P} w_A(e), \sum_{e \in P} w_B(e)\}$ is minimized.

Assume w.l.o.g. that $m \leq n$. Since $|A'| \leq m$ and $|B'| \leq n$, we maintain, for each vertex v of G, an array $X(v)$ of size m, where $X(v)[r]$ is the minimum number z such that v can be reached from s with (at most) r hops of the first dog and z hops of the second dog. We can construct these arrays by processing the vertices of G in topological order (i.e., a vertex is processed only after all its predecessors have been processed). This yields an algorithm of running time $O(m^3n^3\min\{m,n\})$, as described in Algorithm 1.

Algorithm 1 CPS-3F

```
1. Create a directed graph $G=(V,E)$ with two weight functions $w_A$, $w_B$, such that:
   – $V$ is the set of all configurations $(a_i,a_p,b_j,b_q)$ with $d(a_i,a_p)\le\delta_1$, $d(b_j,b_q)\le\delta_2$, and $d(a_p,b_q)\le\delta_3$.
   – $E=\{((a_i,a_p,b_j,b_q),(a_{i'},a_{p'},b_{j'},b_{q'}))\mid i\le i'\le i{+}1,\ p\le p',\ j\le j'\le j{+}1,\ q\le q'\}$.
   – For each $((a_i,a_p,b_j,b_q),(a_{i'},a_{p'},b_{j'},b_{q'}))\in E$, set
      • $w_A((a_i,a_p,b_j,b_q),(a_{i'},a_{p'},b_{j'},b_{q'}))=\begin{cases}1, & p<p'\\ 0, & otherwise\end{cases}$
      • $w_B((a_i,a_p,b_j,b_q),(a_{i'},a_{p'},b_{j'},b_{q'}))=\begin{cases}1, & q<q'\\ 0, & otherwise\end{cases}$
2. Sort $V$ topologically.
3. Initialize the array $X(s)$ (i.e., set $X(s)[r]=0$, for $r=0,\dots,m-1$).
4. For each $v\in V\setminus\{s\}$ (advancing from left to right in the sorted sequence) do:
   (a) Initialize the array $X(v)$ (i.e., set $X(v)[r]=\infty$, for $r=0,\dots,m-1$).
   (b) For each $r$ between 0 and $m-1$, compute $X(v)[r]$:
       $X(v)[r]=\min_{(u,v)\in E}\begin{cases}X(u)[r]+w_B(u,v), & w_A(u,v)=0\\ X(u)[r-1]+w_B(u,v), & w_A(u,v)=1\end{cases}$
5. Return $k^*=\min_r\max\{r,\ X(t)[r]\}$.
```

Running time The number of vertices in G is $|V|=O(m^2n^2)$. By the construction of the graph, for any vertex (a_i,a_p,b_j,b_q) the maximum number of outgoing edges is $O(mn)$. So we have $|E|=O(|V|mn)=O(m^3n^3)$. Thus, constructing the graph G in Step 1 takes $O(n^3m^3)$ time. Step 2 takes $O(|E|)$ time, while Step 3 takes $O(m)$ time. In Step 4, for each vertex v and for each index r, we consider all configurations that can directly precede v. So each edge of G participates in exactly m minimum computations, implying that Step 4 takes $O(|E|m)$ time. Step 5 takes $O(m)$ time. Thus, the total running time of the algorithm is $O(m^4n^3)$.

Theorem 1. *The chain pair simplification problem under the discrete Fréchet distance (CPS-3F) is polynomial, i.e., CPS-3F $\in$ **P**.*

Remark 1. As mentioned, we have assumed that the first and last points of A' (resp., B') are a_1 and a_m (resp., b_1 and b_n), so we have a single initial configuration (i.e., $s=(a_1,a_1,b_1,b_1)$) and a single final configuration (i.e., $t=$

$(a_m, a_m, b_n, b_n))$. However, it is easy to adapt our algorithm to the case where the first and last points of the chains A' and B' are not specified. In this case, any possible configuration of the form (a_1, a_p, b_1, b_q) is considered a potential initial configuration, and any possible configuration of the form (a_m, a_p, b_n, b_q) is considered a potential final configuration, where $1 \le p \le m$ and $1 \le q \le n$. Let S and T be the sets of potential initial and final configurations, respectively. (Then, $|S| = O(mn)$ and $|T| = O(mn)$.) We thus remove from G all edges entering a potential initial configuration, so that each such configuration becomes a "root" in the (topologically) sorted sequence. Now, in Step 3 we initialize the arrays of each $s \in S$ in total time $O(m^2n)$, and in Step 4 we only process the vertices that are not in S. The value $X(v)[r]$ for such a vertex v is now the minimum number z such that v can be reached from s with r hops of the first dog and z hops of the second dog, over *all* potential initial configurations $s \in S$. In the final step of the algorithm, we calculate the value k^* in $O(m)$ time, for each potential final configuration $t \in T$. The smallest value obtained is then the desired value. Since the number of potential final configurations is only $O(mn)$, the total running time of the final step of the algorithm is only $O(m^2n)$, and the running time of the entire algorithm remains $O(m^4n^3)$.

4 An Efficient Implementation

The time and space complexity of Algorithm 1 (which is $O(m^3n^3 \min\{m, n\})$ and $O(m^3n^3)$, respectively) makes it impractical for our motivating biological application (as m, n could be 500∼600). In fact, when m, n are around 200 we already had memory overflows in the implemented Algorithm 1. In this section, we show how to reduce the time and space bounds by a factor of mn, using dynamic programming.

We generate all configurations of the form (a_i, a_p, b_j, b_q), where the outermost for-loop is governed by i, the next level loop by j, then p, and finally q. When a new configuration $v = (a_i, a_p, b_j, b_q)$ is generated, we first check whether it is *possible*. If it is not possible, we set $X(v)[r] = \infty$, for $1 \le r \le m$, and if it is, we compute $X(v)[r]$, for $1 \le r \le m$.

We also maintain for each pair of indices i and j, three tables $C_{i,j}$, $R_{i,j}$, $T_{i,j}$ that assist us in the computation of the values $X(v)[r]$:

$$\begin{aligned}
C_{i,j}[p,q,r] &= \min_{1 \le p' \le p} X(a_i, a_{p'}, b_j, b_q)[r] \\
R_{i,j}[p,q,r] &= \min_{1 \le q' \le q} X(a_i, a_p, b_j, b_{q'})[r] \\
T_{i,j}[p,q,r] &= \min_{\substack{1 \le p' \le p \\ 1 \le q' \le q}} X(a_i, a_{p'}, b_j, b_{q'})[r]
\end{aligned}$$

Notice that the value of cell $[p, q, r]$ is determined by the value of one or two previously-determined cells and $X(a_i, a_p, b_j, b_q)[r]$ as follows:

$$C_{i,j}[p, q, r] = \min\{C_{i,j}[p-1, q, r], X(a_i, a_p, b_j, b_q)[r]\}$$
$$R_{i,j}[p, q, r] = \min\{R_{i,j}[p, q-1, r], X(a_i, a_p, b_j, b_q)[r]\}$$
$$T_{i,j}[p, q, r] = \min\{T_{i,j}[p-1, q, r], T_{i,j}[p, q-1, r], X(a_i, a_p, b_j, b_q)[r]\}$$

Observe that in any configuration that can immediately precede the current configuration (a_i, a_p, b_j, b_q), the man is either at a_{i-1} or at a_i and the woman is either at b_{j-1} or at b_j (and the dogs are at $a_{p'}$, $p' \leq p$, and $b_{q'}$, $q' \leq q$, respectively). The "saving" is achieved, since now we only need to access a constant number of table entries in order to compute the value $X(a_i, a_p, b_j, b_q)[r]$.

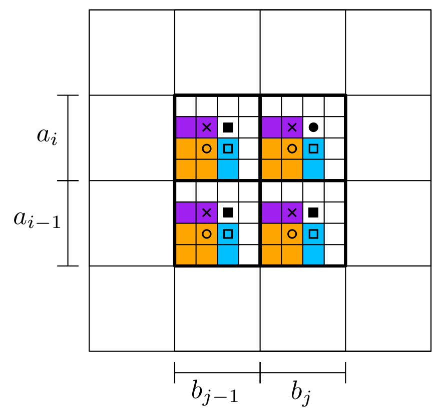


Fig. 2: Illustration of Algorithm 2.

One can illustrate the algorithm using the matrix in Figure 2. There are mn large cells, each of them containing a matrix of size mn. The large cells correspond to the positions of the man and the woman. The inner matrices correspond to the positions of the two dogs (for given positions of the man and woman). Consider an optimal "walk" of the gang that ends at cell (a_i, a_p, b_j, b_q) (marked by a full circle), such that the first dog has visited r points. The previous cell in this "walk" must be in one of the 4 large cells (a_i, b_j),(a_{i-1}, b_j),(a_i, b_{j-1}),(a_{i-1}, b_{j-1}). Assume, for example, that it is in (a_{i-1}, b_j). Then, if it is in the blue area, then $X(a_i, a_p, b_j, b_q)[r] = C_{i-1,j}[p-1, q, r-1]$ (marked by an empty square), since only the position of the first dog has changed when the gang moved to (a_i, a_p, b_j, b_q). If it is in the purple area, then $X(a_i, a_p, b_j, b_q)[r] = R_{i-1,j}[p, q-1, r]+1$ (marked by a x), since only the position of the second dog has changed. If it is in the orange area, then $X(a_i, a_p, b_j, b_q)[r] = T_{i-1,j}[p-1, q-1, r-1]+1$ (marked by an empty circle), since the positions of both dogs have changed. Finally, if it is the cell marked by the full square, then simply $X(a_i, a_p, b_j, b_q)[r] = X(a_{i-1}, a_p, b_j, b_q)[r]$, since both dogs have not moved. The other three cases, in which the previous cell is in one of the 3 large cells (a_i, b_j),(a_i, b_{j-1}),(a_{i-1}, b_{j-1}), are handled similarly.

Algorithm 2 CPS-3F using dynamic programming

```
for $i = 1$ to $m$
    for $j = 1$ to $n$
        for $p = 1$ to $m$
            for $q = 1$ to $n$
                for $r = 1$ to $m$
                    $X_{(-1,0)} = \min \begin{cases} C_{i-1,j}[p-1, q, r-1] \\ R_{i-1,j}[p, q-1, r] + 1 \\ T_{i-1,j}[p-1, q-1, r-1] + 1 \\ X(a_{i-1}, a_p, b_j, b_q)[r] \end{cases}$
                    $X_{(0,-1)} = \min \begin{cases} C_{i,j-1}[p-1, q, r-1] \\ R_{i,j-1}[p, q-1, r] + 1 \\ T_{i,j-1}[p-1, q-1, r-1] + 1 \\ X(a_i, a_p, b_{j-1}, b_q)[r] \end{cases}$
                    $X_{(-1,-1)} = \min \begin{cases} C_{i-1,j-1}[p-1, q, r-1] \\ R_{i-1,j-1}[p, q-1, r] + 1 \\ T_{i-1,j-1}[p-1, q-1, r-1] + 1 \\ X(a_{i-1}, a_p, b_{j-1}, b_q)[r] \end{cases}$
                    $X_{(0,0)} = \min \begin{cases} C_{i,j}[p-1, q, r-1] \\ R_{i,j}[p, q-1, r] + 1 \\ T_{i,j}[p-1, q-1, r-1] + 1 \end{cases}$
                    $X(a_i, a_p, b_j, b_q)[r] = \min\{X_{(-1,0)}, X_{(0,-1)}, X_{(-1,-1)}, X_{(0,0)}\}$

                    $C_{i,j}[p, q, r] = \min\{C_{i,j}[p-1, q, r], X(a_i, a_p, b_j, b_q)[r]\}$
                    $R_{i,j}[p, q, r] = \min\{R_{i,j}[p, q-1, r], X(a_i, a_p, b_j, b_q)[r]\}$
                    $T_{i,j}[p, q, r] = \min\{T_{i,j}[p-1, q, r], T_{i,j}[p, q-1, r], X(a_i, a_p, b_j, b_q)[r]\}$
return $\min_{r,p,q} \max\{r, X(a_m, a_p, b_n, b_q)[r]\}$
```

We are ready to present the dynamic programming algorithm. The initial configurations correspond to cells in the large cell (a_1, b_1). For each initial configuration (a_1, a_p, b_1, b_q), we set $X(a_1, a_p, b_1, b_q)[1] = 1$.

Theorem 2. *The minimization version of the chain pair simplification problem under the discrete Fréchet distance (CPS-3F) can be solved in $O(m^2 n^2 \min\{m, n\})$ time.*

We comment that this algorithm has been implemented with C++ and tested with real datasets from the PDB. Compared with Algorithm FIND-CPS3F$^+$, i.e., the algorithm (mentioned in the introduction) for the optimization version of CPS-3F$^+$, proposed by Wylie and Zhu [11], the improvement is huge. Due to space constraints, we leave the empirical results out and interested readers are referred to the full version for the details.

5 Weighted Chain Pair Simplification

In this section, we consider a more general version of CPS-3F, namely, **Weighted CPS-3F**. In the weighted version of the chain pair simplification problem, the vertices of the chains A and B are assigned arbitrary weights, and, instead of limiting the length of the simplifications, one limits their weights. That is, the total weight of each simplification must not exceed a given value. The problem is formally defined as follows.

Problem 2 (Weighted Chain Pair Simplification).
Instance: Given a pair of 3D chains A and B, with lengths m and n, respectively, an integer k, three real numbers $\delta_1, \delta_2, \delta_3 > 0$, and a weight function $C : \{a_1, \ldots, a_m, b_1, \ldots, b_n\} \rightarrow \mathbb{R}^+$.
Problem: Does there exist a pair of chains A',B' with $C(A'), C(B') \leq k$, such that the vertices of A',B' are from A, B respectively, $d_1(A, A') \leq \delta_1$, $d_2(B, B') \leq \delta_2$, and $d_{dF}(A', B') \leq \delta_3$?

When $d_1 = d_2 = d_{dF}$, the problem is called WCPS-3F. When $d_1 = d_2 = d_H$, the problem is **NP**-complete, since the non-weighted version (i.e., CPS-2H) is already **NP**-complete [4].

We prove that WCPS-3F is weakly **NP**-complete via a reduction from the *set partition* problem: Given a set of positive integers $S = \{s_1, \ldots, s_n\}$, find two sets $P_1, P_2 \subset S$ such that $P_1 \cap P_2 = \emptyset$, $P_1 \cup P_2 = S$, and the sum of the numbers in P_1 equals the sum of the numbers in P_2. This is a weakly **NP**-complete special case of the classic subset-sum problem.

Our reduction builds two curves with weights reflecting the values in S. We think of the two curves as the subsets of the partition of S. Although our problem requires positive weights, we also allow zero weights in our reduction for clarity. Later, we show how to remove these weights by slightly modifying the construction.

Theorem 3. *The weighted chain pair simplification problem under the discrete Fréchet distance is weakly **NP**-complete.*

Proof. Given the set of positive integers $S = \{s_1, \ldots, s_n\}$, we construct two curves A and B in the plane, each of length $2n$. We denote the weight of a vertex x_i by $w(x_i)$. A is constructed as follows. The i'th odd vertex of A has weight s_i, i.e. $w(a_{2i-1}) = s_i$, and coordinates $a_{2i-1} = (i, 1)$. The i'th even vertex of A has coordinates $a_{2i} = (i + 0.2, 1)$ and weight zero. Similarly, the i'th odd vertex of B has weight zero and coordinates $b_{2i-1} = (i, 0)$, and the i'th even vertex of B has coordinates $b_{2i} = (i + 0.2, 0)$ and weight s_i, i.e. $w(b_{2i}) = s_i$. Figure 3 depicts

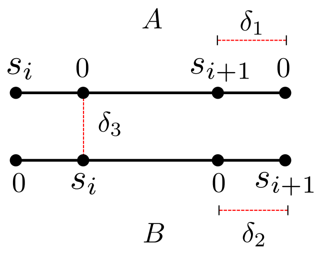


Fig. 3: The reduction for the weighted chain pair simplification problem under the discrete Fréchet distance.

the vertices $a_{2i-1}, a_{2i}, a_{2(i+1)-1}, a_{2(i+1)}$ of A and $b_{2i-1}, b_{2i}, b_{2(i+1)-1}, b_{2(i+1)}$ of B. Finally, we set $\delta_1 = \delta_2 = 0.2$, $\delta_3 = 1$, and $k = \mathfrak{S}$, where $\mathfrak{S}$ denotes the sum of the elements of S (i.e., $\mathfrak{S} = \sum_{j=1}^{n} s_j$).

We claim that S can be partitioned into two subsets, each of sum $\mathfrak{S}/2$, if and only if A and B can be simplified with the constraints $\delta_1 = \delta_2 = 0.2$, $\delta_3 = 1$ and $k = \mathfrak{S}/2$, i.e., $C(A'), C(B') \leq \mathfrak{S}/2$.

First, assume that S can be partitioned into sets S_A and S_B, such that $\sum_{s \in S_A} s = \sum_{s \in S_B} s = \mathfrak{S}/2$. We construct simplifications of A and of B as follows.

$$A' = \{a_{2i-1} \mid s_i \in S_A\} \cup \{a_{2i} \mid s_i \notin S_A\} \text{ and } B' = \{b_{2i} \mid s_i \in S_B\} \cup \{b_{2i-1} \mid s_i \notin S_B\} .$$

It is easy to see that $C(A'), C(B') \leq \mathfrak{S}/2$. Also, since $\{S_A, S_B\}$ is a partition of S, exactly one of the following holds, for any $1 \leq i \leq n$:

1. $a_{2i-1} \in A', b_{2i-1} \in B'$ and $a_{2i} \notin A', b_{2i} \notin B'$.
2. $a_{2i-1} \notin A', b_{2i-1} \notin B'$ and $a_{2i} \in A', b_{2i} \in B'$.

This implies that $d_{dF}(A, A') \leq 0.2 = \delta_1$, $d_{dF}(B, B') \leq 0.2 = \delta_2$ and $d_{dF}(A', B') \leq 1 = \delta_3$.

Now, assume there exist simplifications A', B' of A, B, such that $d_{dF}(A, A') \leq \delta_1 = 0.2$, $d_{dF}(B, B') \leq \delta_2 = 0.2$, $d_{dF}(A', B') \leq \delta_3 = 1$, and $C(A'), C(B') \leq k = \mathfrak{S}/2$. Since $\delta_1 = \delta_2 = 0.2$, for any $1 \leq i \leq n$, the simplification A' must contain one of a_{2i-1}, a_{2i}, and the simplification B' must contain one of b_{2i-1}, b_{2i}. Since $\delta_3 = 1$, for any i, at least one of the following two conditions holds: $a_{2i-1} \in A'$ and $b_{2i-1} \in B'$ or $a_{2i} \in A'$ and $b_{2i} \in B'$. Therefore, for any i, either $a_{2i-1} \in A$ or $b_{2i} \in B$, implying that s_i participates in either $C(A')$ or $C(B')$. However, since $C(A'), C(B') \leq \mathfrak{S}/2$, s_i cannot participate in both $C(A')$ and $C(B')$. It follows that $C(A') = C(B') = \mathfrak{S}/2$, and we get a partition of S into two sets, each of sum $\mathfrak{S}/2$.

Finally, we note that WCPS-3F is in **NP**. For an instance I with chains A, B, given simplifications A', B', we can verify in polynomial time that $d_{dF}(A, A') \leq \delta_1$, $d_{dF}(B, B') \leq \delta_2$, $d_{dF}(A', B') \leq \delta_3$, and $C(A'), C(B') \leq k$. $\square$

Although our construction of A' and B' uses zero weights, a simple modification enables us to prove that the problem is weakly **NP**-complete also when only positive integral weights are allowed. Increase all the weights by 1, that is, $w(a_{2i-1}) = w(b_{2i}) = s_i + 1$ and $w(a_{2i}) = w(b_{2i-1}) = 1$, for $1 \leq i \leq n$, and set $k = \mathfrak{S}/2 + n$. It is easy to verify that our reduction still works. Finally, notice that we could overlay the two curves choosing $\delta_3 = 0$ and prove that the problem is still weakly **NP**-complete in one dimension.

6 Concluding Remarks

In this paper we showed that CPS-3F, which has been an open problem since 2008, is polynomially solvable. We also proved that the weighted version of the problem is weakly NP-complete. In the full version, we include a summary of empirical results that show that Algorithm 2 can handle real datasets, while the $O(m^3n^3)$ space requirement of Algorithm 1 causes memory overflow for most pairs of protein backbones. Still, it would be interesting and desirable to further reduce the running time of CPS-3F, as some cases take 20 hours to compute.

References

1. Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2):429–449, 2014.
2. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geometry Appl.*, 5:75–91, 1995.
3. Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. The discrete Fréchet distance with shortcuts via approximate distance counting and selection. In *Proc. 30th Annual ACM Sympos. on Computational Geometry, SOCG'14*, page 377, 2014.
4. Sergey Bereg, Minghui Jiang, Wencheng Wang, Boting Yang, and Binhai Zhu. Simplifying 3D polygonal chains under the discrete Fréchet distance. In *Proc. 8th Latin American Theoretical Informatics Sympos., LATIN'08*, pages 630–641, 2008.
5. Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog — with an application to alt's conjecture. In *Proc. 25th Annual ACM-SIAM Sympos. on Discrete Algorithms, SODA'14*, pages 1399–1413, 2014.
6. Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM J. Comput.*, 42(5):1830–1866, 2013.
7. Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Information Systems Dept., Technical University of Vienna, 1994.
8. Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo*, 22(1):1–72, 1906.
9. Minghui Jiang, Ying Xu, and Binhai Zhu. Protein structure-structure alignment with discrete Fréchet distance. *J. Bioinformatics and Computational Biology*, 6(1):51–64, 2008.
10. Tim Wylie, Jun Luo, and Binhai Zhu. A practical solution for aligning and simplifying pairs of protein backbones under the discrete Fréchet distance. In *Proc. Internat. Conf. Computational Science and Its Applications, ICCSA'11, Part III*, pages 74–83, 2011.
11. Tim Wylie and Binhai Zhu. Protein chain pair simplification under the discrete Fréchet distance. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 10(6):1372–1383, 2013.