Vines and Vineyards by Updating Persistence in Linear Time *

David Cohen-Steiner
INRIA, 2004 Route des Lucioles, BP93
Sophia-Antipolis, France
dcohen@sophia.inria.fr

Herbert Edelsbrunner
Dept Computer Science,
Duke University, Durham
Geomagic, RTP
North Carolina, USA
edels@cs.duke.edu

Dmitriy Morozov
Dept Computer Science
Duke University, Durham
North Carolina, USA
morozov@cs.duke.edu

ABSTRACT

Persistent homology is the mathematical core of recent work on shape, including reconstruction, recognition, and matching. Its pertinent information is encapsulated by a pairing of the critical values of a function, visualized by points forming a diagram in the plane. The original algorithm in [10] computes the pairs from an ordering of the simplices in a triangulation and takes worst-case time cubic in the number of simplices. The main result of this paper is an algorithm that maintains the pairing in worst-case linear time per transposition in the ordering. A side-effect of the algorithm's analysis is an elementary proof of the stability of persistence diagrams [7] in the special case of piecewise-linear functions. We use the algorithm to compute 1-parameter families of diagrams which we apply to the study of protein folding trajectories.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations, Computations on discrete structures*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*Counting problems*

General Terms

Algorithms, Theory

1. INTRODUCTION

At first sight, persistent homology may seem like an abstract mathematical notion hopelessly detached from physical reality. We agree with the first assessment but believe that the concept has something fundamental to contribute to scientific understanding in a broad sense.

*The authors were partially supported by NSF under grant CCR-00-86013, by DARPA under grant HR0011-05-1-0007, and by the Lawrence Livermore National Laboratory under grant B543154.

SCG'06, June 5–7, 2006, Sedona, Arizona, USA.

Motivation. The starting point of the work described in this paper is the stability of persistence diagrams established in [7]. Given a continuous function on a topological space, we express topological properties in terms of the homology groups of its sublevel sets and visualize these properties combinatorially, using points in the plane to form the persistence diagram. In a nutshell, the stability result states that small changes in the function imply small changes in the diagram. If we now take a function that changes continuously in time, we can watch the diagram and understand the changes by observing how the points move and rearrange to form fleeting patterns. We may solidify the patterns by stacking up the diagrams, letting each point trace out a curve in space, which we refer to as a vine. We believe that this construction is a powerful metaphor aimed at gaining insight into continuous processes but also to quantify some of their less tangible aspects. However, constructing the vine turned out to be more difficult than expected. Common time-series data is too sparse to compute them by matching the points in contiguous diagrams, and refining the series is expensive and sometimes not sufficiently powerful to remove all ambiguities. This paper describes an alternative approach to constructing the vines by maintaining an ordering of the simplices during a straight-line homotopy.

Results and prior work. This paper builds on prior work on persistent homology and the stability of persistence diagrams. The concept of persistence can be seen embedded within the theory of spectral sequences [13] but has not been treated as a concept in its own right until [10]. The latter paper also describes a fast algorithm for modulo 2 homology and demonstrates that persistence is relevant to applications, including the study of protein structure. The concept and the algorithm have been extended to homology over fields in [6]. The stability of persistence diagrams has been established in [7], opening the concept up to additional applications, including the inference of homology from point clouds, see also [9], the comparison of shapes, see also [5], and the analysis of discrete curvature measures, see [8]. Independently, the same ideas were developed from a somewhat different angle and restricted to zero-homology by a group of researchers in Italy [3]. Using different terminology, they introduced persistence diagrams and proved stability, albeit only for the evolution of components in sublevel sets [2]. The main contributions of this paper to the theory and practice of persistent homology are:

1. an algorithm that maintains the persistence diagram in time $O(n)$ per transposition, where n is the number of simplices used to represent the topological space and the function;
2. a new and elementary proof of the stability of persistence

diagrams in the case of piecewise-linear functions;

3. the definition and computation of vineyards (continuous families of persistence diagrams) for time-series of continuous functions;
4. preliminary steps towards the application of vineyards to the study of protein folding trajectories.

Similar to [17], our aim in the application is to learn about protein folding by viewing the process through a quantifiable combinatorial lens. The preliminary results are encouraging and will hopefully lead to a broader and deeper investigation of the subject.

Outline. Section 2 reviews the technical background needed to describe our work. Section 3 presents the algorithm for updating the persistence diagram. Section 4 gives the new proof of stability, introduces vineyards, and applies them to the study of protein folding trajectories. Section 5 concludes the paper.

2. BACKGROUND

In this section, we introduce the necessary background from algebraic topology. We begin with a brief review of homology groups and refer to Munkres [15] for details. Continuing with the relatively recent concept of persistence, we introduce persistence diagrams and state in which sense they are stable. Finally, we address computations in terms of filtrations and recast the algorithm of [10] as a matrix reduction method.

Homology groups. Recall that a triangulation of a topological space, $\mathbb{X}$, is a simplicial complex, K, whose underlying space is homeomorphic to $\mathbb{X}$. A *p-chain* is a subset of the p-dimensional simplices in K or, equivalently, a formal sum in which each p-simplex appears with coefficient 0 or 1. We add p-chains modulo 2, which is the same as taking the symmetric difference of sets. This gives the (abelian) group of p-chains, C_p. The *boundary* of a p-simplex is its set of $(p-1)$-dimensional faces, and that of a p-chain is the sum of boundaries of its simplices. We thus have a sequence of groups of chains, one for each dimension, connected by boundary homomorphisms,

$$\ldots \stackrel{\partial_{p+2}}{\rightarrow} \mathsf{C}_{p+1} \stackrel{\partial_{p+1}}{\rightarrow} \mathsf{C}_p \stackrel{\partial_p}{\rightarrow} \mathsf{C}_{p-1} \stackrel{\partial_{p-1}}{\rightarrow} \ldots$$

A *p-cycle* is a p-chain with zero boundary, and a *p-boundary* is the boundary of a $(p+1)$-chain. In other words, the group of p-cycles is the kernel of the p-th boundary homomorphism, $\mathsf{Z}_p = \mathsf{Z}_p(K) = \ker \partial_p$, and the group of p-boundaries is the image of the $(p+1)$-st boundary homomorphism, $\mathsf{B}_p = \mathsf{B}_p(K) = \operatorname{im} \partial_{p+1}$. The boundary of a boundary is always zero, which implies that B_p is a subgroup of Z_p. We can therefore take the quotient of the two, the *p-th homology group*, $\mathsf{H}_p = \mathsf{H}_p(K) = \mathsf{Z}_p/\mathsf{B}_p$. Since we work with modulo 2 arithmetic, the homology groups are vector spaces over $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$. The *$p$-th Betti number* is the dimension or rank of the p-th homology group, $\beta_p = \beta_p(K) = \operatorname{rank} \mathsf{H}_p$. Importantly, homology groups and Betti numbers are properties of $\mathbb{X}$ and do not depend on the particular triangulation we use to compute them.

Assuming an ordering of the $(p-1)$-simplices and of the p-simplices, the boundary of a p-chain can be obtained by multiplication of the corresponding vector with the incidence matrix, $\partial_p(c_p) = D_p c_p$, where $D_p[i,j] = 1$ if the i-th $(p-1)$-simplex is a face of the j-th p-simplex, and $D_p[i,j] = 0$ otherwise. A classic algorithm computes the Betti numbers of K by reducing its incidence matrices to Smith normal form. It uses row and column operations to zero out all entries except along an initial portion of the diagonal, as shown in Figure 1. In the normal form of D_p, the zero columns form a basis of the p-cycles and the non-zero rows form a basis of the $(p-1)$-boundaries. We can thus read the ranks of the cycle and the boundary groups off the normal forms and get $\beta_p = \operatorname{rank} \mathsf{Z}_p - \operatorname{rank} \mathsf{B}_p$, for each p.

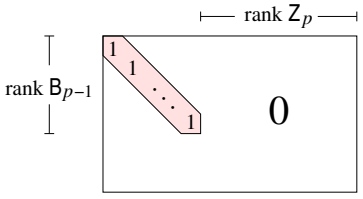

Figure 1: Smith normal form of the incidence matrix between $(p-1)$-simplices and p-simplices.

Persistence. Consider a continuous function $f : \mathbb{X} \to \mathbb{R}$ and its sublevel sets, $\mathbb{X}^r = f^{-1}(-\infty, r]$. The inclusion of $\mathbb{X}^r$ in $\mathbb{X}^s$, for $r \leq s$, implies a homomorphism between the homology groups of same dimension, $h_p^{r,s} : \mathsf{H}_p(\mathbb{X}^r) \to \mathsf{H}_p(\mathbb{X}^s)$. We write $\mathsf{H}_p^{r,s} = \operatorname{im} h_p^{r,s}$ for the image and refer to it as a *dimension p persistent homology group* of f. Indeed, its elements are homology classes that persist in the sense they are born at or before value r and last beyond value s. The corresponding *dimension p Betti numbers* are the ranks of these groups, $\beta_p^{r,s} = \operatorname{rank} \mathsf{H}_p^{r,s}$.

Call $r \in \mathbb{R}$ a *homological critical value* of f if there is a dimension p such that for sufficiently small $\varepsilon > 0$ the map $\mathsf{H}_p(\mathbb{X}^{r-\varepsilon}) \to \mathsf{H}_p(\mathbb{X}^{r+\varepsilon})$ is not an isomorphism. To continue, we assume that f is *tame*, by which we mean it has only finitely many homological critical values and the homology groups of its sublevel sets all have finite rank. Letting $a_1 < a_2 < \ldots < a_\ell$ be the sequence of homological critical values, we add $a_0 = -\infty$ and $a_{\ell+1} = \infty$ and choose interleaving values $a_i < b_i < a_{i+1}$ for $0 \leq i \leq \ell$. Consider the pair (a_i, a_j), for $0 \leq i < j \leq \ell + 1$, and define its *dimension p multiplicity* as

$$\begin{aligned} \mu_p^{i,j} &= \operatorname{rank} \mathsf{H}_p^{b_{i+1},b_j} - \operatorname{rank} \mathsf{H}_p^{b_i,b_j} \\ &\quad + \operatorname{rank} \mathsf{H}_p^{b_i,b_{j+1}} - \operatorname{rank} \mathsf{H}_p^{b_{i+1},b_{j+1}}. \end{aligned}$$

It is not difficult to show that the pairs determine the persistent Betti numbers, namely

$$\beta_p^{r,s} = \sum_{a_i \leq r \leq s < a_j} \mu_p^{i,j}. \tag{1}$$

This relation suggests we draw the pair (a_i, a_j) as $\mu_p^{i,j}$ points in the plane, each at the same location with coordinates a_i and a_j above the diagonal. Some of the coordinates may be infinite so we really talk about the extended plane, $\bar{\mathbb{R}}^2$. The interpretation of Equation (1) in this geometric setting is that $\beta_p^{r,s}$ is the number of points in the north-west quadrant with corner (r, s) whenever r and s are not homological critical values.

Stability. The *dimension p persistence diagram* of f, denoted as $\mathrm{D}_p(f)$, consists of the points (a_i, a_j) introduced above together with infinitely many copies of all points along the diagonal. The reason for adding the diagonal is technical and will be obvious shortly. Given two continuous and tame functions $f, g : \mathbb{X} \to \mathbb{R}$, we measure their distance using the L_∞-norm of their difference: $\|f - g\|_\infty = \sup_{x \in \mathbb{X}} |f(x) - g(x)|$. Similarly, we introduce the *bottleneck distance* between their persistence diagrams as the infimum, over all bijections $\gamma : \mathrm{D}_p(f) \to \mathrm{D}_p(g)$, of the largest

distance between corresponding points,

$$d_B(\mathrm{D}_p(f), \mathrm{D}_p(g)) \quad = \quad \inf_{\gamma} \sup_{u \in \mathrm{D}_p(f)} \|u - \gamma(u)\|_\infty.$$

With this notation, we are ready to state in which sense the persistence diagram is stable.

STABILITY THEOREM. For a triangulable space $\mathbb{X}$, two continuous and tame functions $f, g : \mathbb{X} \to \mathbb{R}$, and any dimension $p \geq 0$, the bottleneck distance between the two dimension p persistence diagrams is bounded from above by the distance between the functions: $d_B(\mathrm{D}_p(f), \mathrm{D}_p(g)) \leq \|f - g\|_\infty$.

The proof of this result given in [7] is fairly technical and involves commutative diagrams of vector spaces of homology classes. We will give an alternative and comparably elementary proof for piecewise linear functions at the beginning of Section 4. The theorem is interesting because it states that a small change in the function cannot result in a big change in the diagram. In other words, we can tell that two functions are grossly different if they have significantly different persistence diagrams. The reverse of this implication is of course not true.

Computation. Similar to homology, we compute persistence diagrams from a discrete representation of a continuous function $f : \mathbb{X} \to \mathbb{R}$. Assuming a triangulation K of $\mathbb{X}$, we use a *monotone* function $\bar{f} : K \to \mathbb{R}$ (satisfying $\bar{f}(\sigma) \leq \bar{f}(\tau)$ if σ is a face of τ) and a corresponding filter (an ordering of K in which simplices are preceded by their faces and $\bar{f}$ is non-decreasing). For example, if f is given in terms of its values at the vertices of K, we may set $\bar{f}(\sigma)$ equal to the maximum function value at the vertices of σ and form a filter by sorting the simplices in the order of non-decreasing values of $\bar{f}$. If the function values at the vertices are distinct, this is an ordering of the lower stars of the vertices, sorting each lower star in the order of non-decreasing dimension. We note that $\bar{f}$ is a piecewise constant approximation of f. It is not difficult to see that the persistence diagram of $\bar{f}$ is the same as that of the piecewise linear function defined by the values at the vertices.

An algorithm that computes the persistence diagrams by pairing the simplices in a given filter can be found in [10]. We recast this algorithm in terms of the overall incidence matrix, D, defined by

$$D[i,j] \quad = \quad \begin{cases} 1 & \text{if } \sigma_i \in \partial\sigma_j, \\ 0 & \text{otherwise,} \end{cases}$$

where σ_i and σ_j are the i-th and the j-th simplices in the filter. The algorithm uses column operations to reduce D to another 0-1 matrix R. To explain what exactly we mean, let $low_R(j)$ be the row index of the last 1 in column j of R and keep $low_R(j)$ undefined if the column is zero. We call R *reduced* and low_R a *pairing function* if $low_R(j) \neq low_R(j')$ whenever $j \neq j'$ specify two non-zero columns. The algorithm reduces D by adding columns to other columns located to their right.

```
R = D
for j = 1 to n do
  while ∃j' < j with low_R(j') = low_R(j) do
    add column j' to column j
  endwhile
endfor.
```

In matrix notation, the algorithm computes the reduced matrix as $R = DV$, where V is an invertible upper-triangular matrix with $\mathbb{Z}_2$ coefficients. The above algorithm is just one way to compute such a reduced matrix, the complete Smith normal form algorithm being another possibility. The reduced matrix is therefore not unique, but we will prove shortly that the collection of pairs (σ_i, σ_j) with $i = low_R(j)$ is independent of the reduced matrix. Here we call σ_i *positive* and σ_j *negative* because σ_i creates the homology class that σ_j destroys. By construction, the two simplices in a pair have contiguous dimensions, and if $\dim \sigma_i = \dim \sigma_j - 1 = p$ then we add the corresponding point with coordinates $(\bar{f}(\sigma_i), \bar{f}(\sigma_j))$ to the dimension p persistence diagram.

3. UPDATING THE PAIRING

In this section, we present the algorithm that updates the pairing function under a transposition of two simplices in the filter. We begin with a characterization of the persistence pairing in terms of ranks of submatrices of the incidence matrix.

Uniqueness of pairing function. Let R be a reduced 0-1 matrix as defined in the previous section, and write R_i^j for the lower left minor obtained by deleting the first $i-1$ rows and the last $n-j$ columns. Any combination of non-zero columns of R_i^j has its last non-zero entry at the same height as the lowest non-zero entry of any of the involved columns. The combination can therefore not be zero implying that the combined non-zero columns are linearly independent. Recall that the algorithm in [10] can be interpreted as computing the reduced matrix $R = DV$, where V is invertible and upper-triangular. Since invertible upper-triangular matrices form a group, we can write D as the product RU of the reduced matrix R and the invertible upper-triangular matrix $U = V^{-1}$. We call such a decomposition an *RU-decomposition* of D. In this decomposition, positive simplices correspond to zero columns and negative simplices to non-zero columns in R. Define

$$\begin{aligned} r_D(i,j) \quad &= \quad \operatorname{rank} D_i^j - \operatorname{rank} D_{i+1}^j \\ &\quad + \operatorname{rank} D_{i+1}^{j-1} - \operatorname{rank} D_i^{j-1}. \end{aligned}$$

We prove below that the pairing function can be expressed in terms of r_D and is thus independent of the particular RU-decomposition used to define it.

PAIRING UNIQUENESS LEMMA. Letting $D = RU$, we have $low_R(j) = i$ iff $r_D(i,j) = 1$. In particular, the pairing function does not depend on the matrix R in the RU-decomposition.

PROOF. Note that adding columns to columns located to their right does not change the rank of lower left minors, so $r_D = r_R$. To prove the claim, it is thus sufficient to show that $low_R(j) = i$ iff $r_R(i,j) = 1$. First assume $low_R(j) = i$. As argued above, the non-zero columns of R_i^j are linearly independent. The last column is non-zero, so $\operatorname{rank} R_i^j - \operatorname{rank} R_i^{j-1} = 1$. Now if we delete the top row from R_i^j then the last column is zero, implying $\operatorname{rank} R_{i+1}^j - \operatorname{rank} R_{i+1}^{j-1} = 0$, as required. Second assume $low_R(j) \neq i$ and consider R_i^j and R_{i+1}^j. If $low_R(j) < i$ the last columns in both matrices are zero and we have $\operatorname{rank} R_i^j = \operatorname{rank} R_i^{j-1}$ as well as $\operatorname{rank} R_{i+1}^j = \operatorname{rank} R_{i+1}^{j-1}$. If $low_R(j) > i$ the last columns in both matrices are non-zero and we have $\operatorname{rank} R_i^j = \operatorname{rank} R_i^{j-1} + 1$ and $\operatorname{rank} R_{i+1}^j = \operatorname{rank} R_{i+1}^{j-1} + 1$. In both cases the claimed result follows. □

Performing a transposition. To swap the simplices in positions i and $i+1$, we exchange rows i and $i+1$ as well as columns i and $i+1$ in D. The new incidence matrix is therefore PDP, where P is the permutation matrix that swaps i and $i+1$. To update the pairing function, we just need to repair the RU-decomposition, which we now show how to do in time O(n). We have $PDP = PRUP =$

$(PRP)(PUP)$, but this is not necessarily an RU-decomposition. As illustrated in Figure 2, PRP is not reduced iff there are columns k and l with $low_R(k) = i$, $low_R(l) = i + 1$, and $R[i, l] = 1$. Furthermore, PUP is not upper-triangular iff $U[i, i + 1] = 1$. We may assume that the algorithm adds only columns that belong

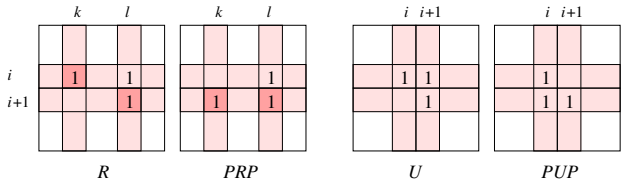

Figure 2: The transposition renders this particular R non-reduced and this particular U non-upper-triangular.

to simplices of the same dimension. The two cases illustrated in Figure 2 therefore arise only if the simplices at positions i and $i+1$ have the same dimension, which we thus assume. In all other cases, PRP is reduced and PUP is upper-triangular.

Case 1 Both i and $i + 1$ are positions of positive simplices. Since column i in R is zero we may set $U[i, i + 1] = 0$, if this is not the case. It follows that PUP is upper-triangular and we only need to consider PRP.

- **Case 1.1** There are columns k and l with $low_R(k) = i$, $low_R(l) = i + 1$, and $R[i, l] = 1$.
  - **Case 1.1.1** $k < l$, as in Figure 2, left. To reduce PRP, we add column k to column l. Letting V be the upper-triangular matrix that performs this operation, we have $PDP = (PRPV)(VPUP)$ since $VV = I$. By construction, $PRPV$ is reduced. Furthermore, adding row l to row k preserves PUP as an upper-triangular matrix. It follows that this is an RU-decomposition of the new incidence matrix.
  - **Case 1.1.2** $l < k$. To reduce PRP, we add column l to column k on its right. Letting V be the upper-triangular matrix that performs this operation, we have $PDP = (PRPV)(VPUP)$ as an RU-decomposition, same as Case 1.1.1.
- **Case 1.2** There are no columns k and l as in Case 1.1. Then $PDP = (PRP)(PUP)$ is an RU-decomposition.

Case 2 Both i and $i+1$ are positions of negative simplices. In this case, rows i and $i + 1$ cannot contain the lowest 1s of any columns. It follows that PRP is reduced and we only need to consider PUP.

- **Case 2.1** $U[i, i + 1] = 1$, as in Figure 2, right. To make the second matrix in the decomposition upper-triangular, we add row $i+1$ of U to row i. Letting W be the upper-triangular matrix that performs this operation, we have $PDP = (PRWP)(PWUP)$. The effect of W on R is it adds column i to column $i + 1$.
  - **Case 2.1.1** $low_R(i) < low_R(i + 1)$. Then RW is reduced and so is $PRWP$ and we have an RU-decomposition.
  - **Case 2.1.2** $low_R(i + 1) < low_R(i)$. Then RW is not reduced, but we can reduce it by adding column $i + 1$ to column i. After the transposition, this is adding column i of RWP to column $i + 1$ and we get $PDP = (PRWPW)(WPWUP)$. The second matrix is upper-triangular and first is reduced, as illustrated in Figure 3, top row.
- **Case 2.2** $U[i, i + 1] = 0$. Then $PDP = (PRP)(PUP)$ is an RU-decomposition.

Case 3 i is the position of a negative simplex and $i + 1$ is the position of a positive simplex.

- **Case 3.1** $U[i, i + 1] = 1$, as in Figure 2, right. Just like in Case 2.1, we add row $i + 1$ of U to row i and get $PDP = (PRWP)(PWUP)$. The second matrix in the decomposition is upper-triangular. However, the first matrix is not reduced, and we reduce it by adding column i of RWP to column $i + 1$, giving $PDP = (PRWPW)(WPWUP)$ as the final decomposition; see Figure 3, bottom row.
- **Case 3.2** $U[i, i + 1] = 0$. Then $PDP = (PRP)(PUP)$ is an RU-decomposition.

Case 4 i is the position of a positive simplex and $i + 1$ is the position of a negative simplex. This is the reverse of Case 3.2. Indeed we set $U[i, i + 1] = 0$ if this is not the case and get the RU-decomposition $PDP = (PRP)(PUP)$.

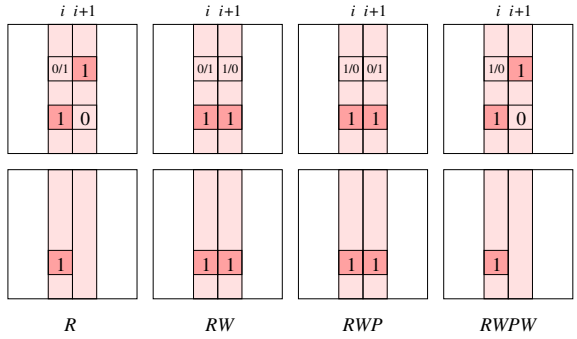

Figure 3: Evolutions of R to $RWPW$, in Case 2.1.2 on top and in Case 3.1 on bottom.

Changes in pairing. There are three cases in which the pairing function changes, all illustrated in Figure 4. The first is Case 1.1.2,

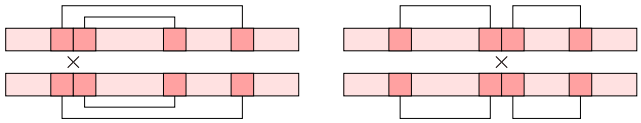

Figure 4: On the left we see Case 1.1.2 and, if read backwards, Case 2.1.2. On the right we see Case 3.1.

characterized by $i = low_R(k) < i + 1 = low_R(l) < l < k$, in which two nested intervals swap their left endpoints to remain nested. The second is Case 2.1.2, characterized by $low_R(i + 1) < low_R(i) < i < i+1$, in which two nested intervals swap their right endpoints to remain nested. The third is Case 3.1, characterized by $low_R(i) < i < i + 1 = low_R(l) < l$, in which two disjoint intervals swap their near endpoints to remain disjoint.

Running time. In every case, a transposition triggers at most one row exchange, one column exchange, and two column additions

in the matrix R. Symmetrically, there are at most one column exchange, one row exchange, and two row additions in U. This takes time at most linear in the number of simplices.

In all applications we have encountered, R and U are both sparse and we can save time and storage using a sparse matrix implementation. We explain such a data structure for R consisting of two linear arrays, one for the set of columns and one for the set of rows, and a singly linked list for each column, as sketched in Figure 5. The j-th element of the column array points to the linked list of 1s in the column. The i-th element in the row array represents the i-th row in the original row sequence and stores its index in the current row sequence. We also store the reverse link, from the current row back to its corresponding original row. Each node in a linked list stores its index in the original row sequence, which we interpret as a pointer into the row array. To exchange two columns, we swap

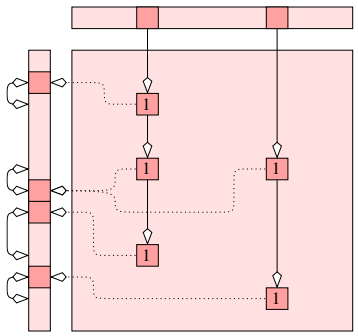

Figure 5: The sparse matrix representation of R sketched by showing the row and column arrays and the linked lists of two columns.

their pointers (lists), which takes only constant time. To add a column to another, we merge the two lists, deleting nodes that come in duplicates, and substitute the result for the second column. This takes time proportional to the number of 1s in the two columns as long as the lists are consistently sorted. We achieve this by protecting the lists from row exchanges, keeping them sorted with respect to the original row indices. A row exchange thus only updates the correspondence between the original and the current orderings of the rows, which takes only constant time.

We use a symmetric sparse matrix data structure preferring rows over columns for U. The result is an implementation that takes storage proportional to n plus the number of 1s in R and in U. The amortized time for each operation is at most proportional to the number of 1s in the affected rows and columns. The worst-case time is $O(n)$ per update, as before, but in our experiments the average update time is about constant.

4. VINES AND VINEYARDS

We begin this section with an elementary proof of the Stability Theorem for piecewise linear functions. Motivated by the result, we consider stacks of diagrams to get curves traced out by the points. We use this construction to distill information about a parametrized family of distance functions generated from the folding trajectory of a protein.

Stability of persistence diagrams. Let $f, g : \mathbb{X} \to \mathbb{R}$ be continuous functions, K a simplicial complex, and Φ a homeomorphism from the underlying space of K to $\mathbb{X}$. The function $\bar{f} : K \to \mathbb{R}$ that maps each simplex $\sigma \in K$ to $\bar{f}(\sigma) = \max_{x \in \sigma} f(\Phi(x))$ is a piecewise constant approximation of f that is monotone in the sense of Section 2. Similarly, $\bar{g}(\sigma) = \max_{x \in \sigma} g(\Phi(x))$ is a piecewise constant approximation of g and monotone.

COMBINATORIAL STABILITY THEOREM. For monotone functions $\bar{f}, \bar{g} : K \to \mathbb{R}$ and any dimension p, the bottleneck distance between the two dimension p persistence diagrams satisfies $d_B(\mathrm{D}_p(\bar{f}), \mathrm{D}_p(\bar{g})) \leq \|\bar{f} - \bar{g}\|_\infty$.

PROOF. Consider the straight-line homotopy $\bar{f}_t(\sigma) = (1-t)\bar{f}(\sigma) + t\bar{g}(\sigma)$, and note that $\bar{f}_t$ is monotone for each $0 \leq t \leq 1$. Let t_1 to t_k be the values at which the ordering of the simplices changes by one or several transpositions, and set $t_0 = 0$ and $t_{k+1} = 1$. Let $t_i \leq r < s < t_{i+1}$, for any $0 \leq i \leq k$, and consider a pair of simplices, (σ, τ), defined for the ordering that exists during the open time interval. Then $u_r = (\bar{f}_r(\sigma), \bar{f}_r(\tau))$ is a point in $\mathrm{D}_p(\bar{f}_r)$ and $u_s = (\bar{f}_s(\sigma), \bar{f}_s(\tau))$ is a point in $\mathrm{D}_p(\bar{f}_s)$. The Manhattan distance between the two points is the larger of the two coordinate differences, which implies

$$\begin{aligned} d_B(\mathrm{D}_p(\bar{f}_r), \mathrm{D}_p(\bar{f}_s)) &\leq \|\bar{f}_r - \bar{f}_s\|_\infty \\ &= (s-r) \cdot \|\bar{f} - \bar{g}\|_\infty. \end{aligned}$$

A transposition changes the pairing but it does not affect the persistence diagram. Hence,

$$\begin{aligned} d_B(\mathrm{D}_p(\bar{f}), \mathrm{D}_p(\bar{g})) &\leq \sum_{i=0}^{k} d_B(\mathrm{D}_p(\bar{f}_{t_i}), \mathrm{D}_p(\bar{f}_{t_{i+1}})) \\ &\leq \|\bar{f} - \bar{g}\|_\infty \sum_{i=0}^{k} (t_{i+1} - t_i). \end{aligned}$$

The latter sum is 1, which implies the claimed inequality. □

As mentioned in Section 2, the persistence diagrams of the piecewise constant maps $\bar{f}_t$ are the same as those of the piecewise linear maps defined by the same values at the vertices. The theorem thus implies the stability of persistence diagrams for the class of piecewise linear functions.

Stacking up persistence diagrams. Consider a homotopy f_t interpolating between $f_0 = f$ and $f_1 = g$. Assuming every f_t is tame, we have a dimension p persistence diagram for every t and p, and the Stability Theorem relating the various diagrams. We draw $\mathrm{D}_p(f_t)$ in the (extended) plane $x_3 = t$ in $\bar{\mathbb{R}}^3$ thus getting a 1-parameter family of diagrams which we call the *dimension p vineyard*. Each off-diagonal point in $\mathrm{D}_p(f_t)$ moves in time, tracing out a curve we refer to as a *vine*. Each vine is either open (starting and ending on the diagonal plane, $x_1 = x_2$), half-open, or closed (starting at an off-diagonal point in $x_3 = 0$ and ending at an off-diagonal point in $x_3 = 1$). If the homotopy is smooth then so are the vines, except when the pairing of critical values changes. We call such points *knees* and observe that they come in pairs. In Cases 1 and 2, the two knees of a pair belong to two vines in the same vineyard, while in Case 3, they belong to vines in vineyards of contiguous dimensions. In practice, homotopies of functions arise from time-series data, given as a sequence of frames which are snapshots of the data at successive moments in time. Naturally, an assumption needs to be made about how the function changes in between the available frames. Ideally, such an assumption reflects the change in the underlying phenomenon described by the function, but in the absence of any such an assumption it is convenient to use the straight-line homotopy between the frames.

We illustrate these concepts by computing a sequence of vineyards for the folding trajectory of a protein. They are generated by

a piecewise linear function on a fixed triangulation, which is perhaps the most common situation we will encounter in applications.

Folding trajectories. The question of how proteins fold is a grand challenge in molecular biology and only modest progress has been reported in the last decades. It appears that the scientific community has not yet succeeded in simulating the folding process computationally. Exceptions are very short sequences or simulations over very short time intervals. We feel that vineyards can be useful in understanding the few folding trajectories that have been computed. One such trajectory describes the simulated folding motion of BBA5, a short peptide of $N = 23$ amino acids [16]. The trajectory is given as $m + 1 = 201$ frames covering a total of 40 picoseconds at regular intervals of 200 femtoseconds. For each $0 \leq i \leq m$, the i-th snapshot is a configuration of this backbone represented by a sequence S_i of N points in $\mathbb{R}^3$, each the center of an alpha carbon along the backbone; see Figure 6. We turn the folding trajectory into vineyards using a 1-parameter family of functions described below.

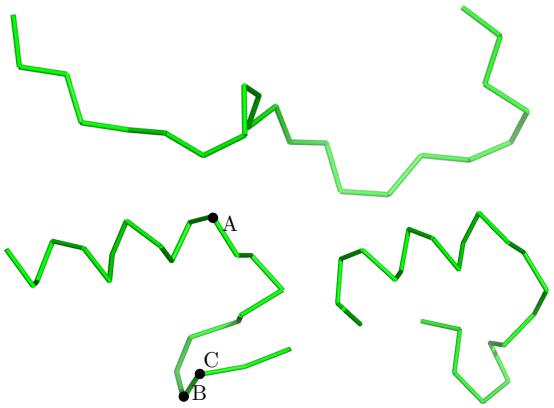

Figure 6: Top: snapshot 0, the initial backbone. Lower left: snapshot 68, the alpha helix is complete. Lower right: snapshot 200, the final backbone.

Pairwise distance. Given a curve $b : [0, 1] \to \mathbb{R}^3$ in space, the pairwise distance function $[0, 1]^2 \to \mathbb{R}$ is defined by mapping (r, s) to $\|b(r) - b(s)\|$. Each function we consider is a piecewise linear approximation of such a pairwise distance function defined by the corresponding backbone configuration. We need some notation. Recall that S_i is the sequence of points describing the i-th backbone and let $c_{i,j}$ be the j-th point in S_i, for $1 \leq j \leq N$. Let K be the triangulation of $[1, N]^2$ obtained by connecting contiguous integer points along common horizontal, vertical, and 45-degree lines. It consists of N^2 vertices, $(3N - 1)(N - 1)$ edges, $2(N - 1)^2$ triangles, and therefore of $n < 6N^2$ simplices in total. For each $0 \leq i \leq m$, we construct $f_i : [1, N]^2 \to \mathbb{R}$ by defining $f_i(j, k) = \|c_{i,j} - c_{i,k}\|$ and extending the values at the vertices by linear interpolation over the edges and triangles. To form a homotopy from f_0 to f_m that passes through all intermediate functions, we finally define $f_{i+\lambda} = (1 - \lambda)f_i + \lambda f_{i+1}$, for all $0 \leq i < m$ and all $0 \leq \lambda \leq 1$.

To construct the vineyards, we first compute the persistence diagrams of f_0, which we then update through a sequence of transpositions, as explained in Section 3. We generate this sequence by sweeping the arrangement of polylines $P_{jk} : [0, m] \to \mathbb{R}$ defined by $P_{jk}(t) = f_t(j, k)$, as illustrated in Figure 7. We have N^2 polylines with at most m crossings between each pair. Each crossing corresponds to a transposition of two vertices. Using a standard

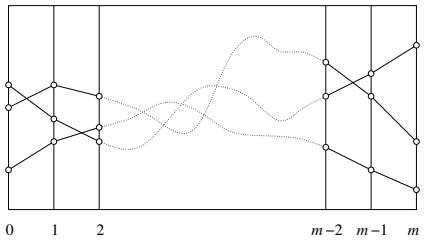

Figure 7: Sketch of the arrangement formed by the N^2 polylines representing the variation of function value at the vertices of K.

plane-sweep algorithm, we can compute the ordered sequence in time $O(\log n)$ per crossing. The resulting algorithm takes worst-case time $O(mn^3)$ to construct the vineyards. In practice, the algorithm runs significantly faster, first because mn^2 is a gross overestimate of the usual number of crossings, and second because our sparse-matrix implementation takes only about constant time per update.

Discussion of the vineyards. The results are illustrated in Figure 8, which shows the dimension 0 and 1 vineyards of the pairwise distance function. Each vine is drawn twice, as viewed from the front (normal to the diagonal direction) and from the side (along the diagonal direction). To interpret Figure 8, we fix a value $t \in [0, m]$

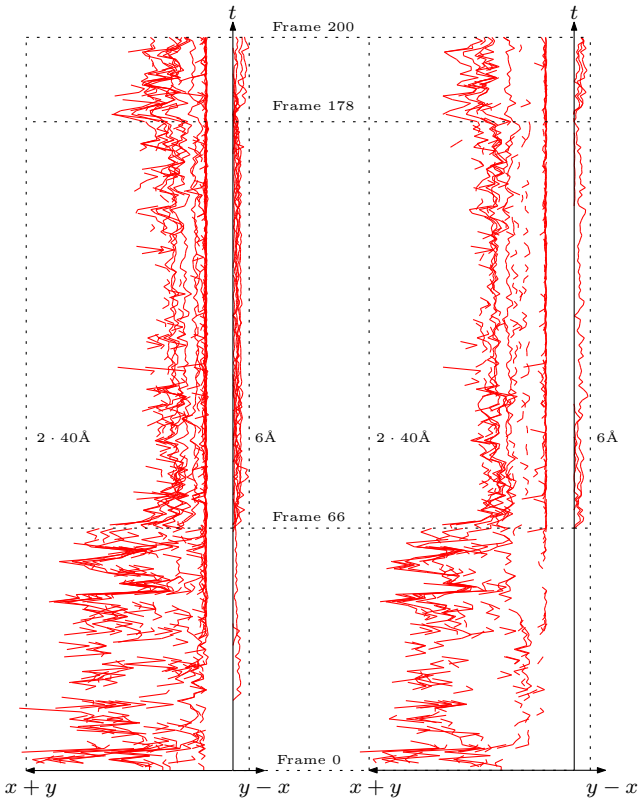

Figure 8: The front view $(x+y, t)$ and the side view $(y-x, t)$ of the dimension 0 vineyard on the left and the dimension 1 vineyard on the right. The side views are simplified by removing vines with lifetime less than 20 frames.

and consider a horizontal cross-section at height t. We note that two points (r, s) and (v, w) belong to a common component of the sublevel set $f_t^{-1}[0, \alpha]$ iff the component contains a path from the first point to the second. In other words, we can continuously move point $b(r)$ to $b(v)$ and simultaneously $b(s)$ to $b(w)$, both along the backbone b, such that the distance is less than or equal to α at all times. For $\alpha = 0$, the sublevel set consists of a single component, the diagonal of the domain. As we increase α, we see new components start at off-diagonal minima and components merge at saddles of f_t. The first critical points with non-zero value appear at α between 5 and 6 Å, causing the characteristic gap of about twice 5.5 Å to the time axis in the front views of the vineyards. The gap becomes particularly well defined when the alpha-helix is formed, suggesting the gap measures the distance between two alpha carbons separated by a single turn of the helix.

The dimension 1 vineyard is somewhat more difficult to interpret. It helps to break the folding process into three stages, the first from Frame 0 to 68, the second from 68 to 170, and the third from 170 to 200. The first stage is characterized by large and seemingly chaotic motions of the backbone that precipitate in vines across a relatively wide range of scales visible in the front view, both for the dimension 0 and 1 vineyards. At the end of the first stage, an alpha helix forms and the backbone assumes a rough S-shape, which remains until the end of the second stage. Covering almost the same time interval, we see a dimension 1 vine emerging from the diagonal at Frame 66 and surviving until the Frame 178 when it disappears into the diagonal. There are 59 knees on this vine, and its maximum persistence (distance from the diagonal which is visible in the side view) is less than 6Å and at times drops well below 1Å. Nevertheless, the vine is very long-lived which suggests that even subtle configurations can stay around for a while. Let us take a closer look at this long vine representing a cycle created at a saddle and destroyed at a maximum. While the atom pairs responsible for the saddle and the maximum (AC and AB in Figure 6 for Frame 68) change as the vine evolves, they always span the tail of the backbone which remains intact during the second stage. Figure 9 shows the graph of the function together with a cycle in the homology class of the feature. During the third stage, we see the tail of the S-shape turn around and point back to the alpha helix. In the dimension 1 vineyard, we see three vines of persistence up to 5Å emerge from the diagonal shortly after Frame 178 and survive until the end, at Frame 200.

In conclusion, we note that the folding process is very complex and it seems difficult to agree on when exactly events begin and end. This is in sharp contrast to a vine, which is unambiguously associated to a feature and has a precisely defined beginning and end. Furthermore, at any moment in time, the scale and the persistence of that feature are quantitatively expressed by the coordinates of the corresponding point on the vine. We thus believe that vines can be used to objectively and quantitatively encapsulate events in the process described by a homotopy.

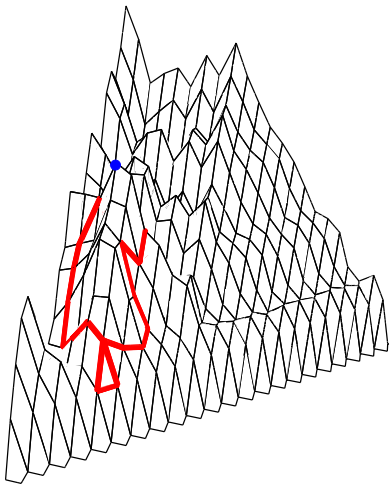

Figure 9: Pairwise distance function for Frame 68 of the BBA5 folding trajectory. The highlighted cycle is destroyed by the marked maximum and belongs to the homology class responsible for the long-lived vine.

5. DISCUSSION

We conclude with a small number of questions aimed at improving and extending the results presented in this paper.

- Are there variants of the update algorithm that are more efficient than the one described in Section 3 or that are simpler and just as efficient? For example, can we update the pairing by only maintaining the reduced matrix, R, and not the matrix U that does the reducing? Is there an advantage in treating rows and columns differently or is the symmetric version more efficient?
- In many applications, the points in the persistence diagram further away from the diagonal are more important than the points close to the diagonal. Can we use or adopt the update algorithm to compute the points with persistence beyond some threshold without spending time on the others? We desire an algorithm whose running time depends only on the size of the output it produces and not on the size of the entire diagram.
- Vineyards trace critical values and do not require any notion of critical points. However, when critical points are available, such as for smooth and for piecewise linear functions on manifolds [4, 14], we can use the update algorithm to maintain their association with the points in the persistence diagram. Can we exploit this ability to gain a better understanding of the stability or instability of critical points? In particular, can this ability be developed into a global alignment algorithm for shapes that is more general and more reliable than what is currently available [11, 12]?

Finally, we would like to suggest that vineyards should not be limited to homotopies but rather considered an analysis and visualization tool for parametrized families of functions. A point in case is the elevation function [1] whose maxima have been useful in coarse protein docking [18]. For a surface in space, this function is based on the sphere of height functions whose vines are 2-manifolds in $\bar{\mathbb{R}}^2 \times \mathbb{S}^2$.

Acknowledgments

The authors thank Vijay Pande, Young Min Rhee, and Vishal Vaidyanathan for providing the BBA5 data set.

6. REFERENCES

[1] P. K. AGARWAL, H. EDELSBRUNNER, J. HARER AND Y. WANG. Extreme elevation on a 2-manifold. *In* "Proc. 20th Ann. Sympos. Comput. Geom., 2004", 357–365.

[2] M. D'AMICO, P. FROSINI AND C. LANDI. Optimal matching between reduced size functions. Tech. Rept. 35, DISMI, Univ. degli Studi di Modena e Reggio Emilia, Italy, 2003.

[3] M. D'AMICO, P. FROSINI AND C. LANDI. Natural pseudo-distance and optimal matching between reduced size functions. Tech. Rept. 66, DISMI, Univ. degli Studi di Modena e Reggio Emilia, Italy, 2005.

[4] T. BANCHOFF. Critical points and curvature for embedded polyhedra. *J. Diff. Geom.* **1** (1967), 245–256.

[5] G. CARLSSON, A. COLLINS, L. GUIBAS AND A. ZOMORODIAN. Persistence barcodes for shapes. *In* "Proc. 2nd Sympos. Geometry Process., 2004", 127–138.

[6] G. CARLSSON AND A. ZOMORODIAN. Computing persistent homology. *In* "Proc. 20th Ann. Sympos. Comput. Geom., 2004", 347–356.

[7] D. COHEN-STEINER, H. EDELSBRUNNER AND J. HARER. Stability of persistence diagrams. *In* "Proc. 21st Ann. Sympos. Comput. Geom., 2005", 263–271.

[8] D. COHEN-STEINER AND H. EDELSBRUNNER. Inequalities for the curvature of curves and surfaces. *In* "Proc. 21st Ann. Sympos. Comput. Geom., 2005", 272–277.

[9] V. DE SILVA AND G. CARLSSON. Topological estimation using witness complexes. *In* "Proc. Sympos. Point-Based Graphics, 2004", 157–166.

[10] H. EDELSBRUNNER, D. LETSCHER AND A. ZOMORODIAN. Topological persistence and simplification. *Discrete Comput. Geom.* **28** (2002), 511–533.

[11] N. GELFAND, N. J. MITRA, L. J. GUIBAS AND H. POTTMANN. Robust global alignment. *In* "Proc. 3rd Ann. Eurographics Sympos. Geom. Process., 2005", 197–206.

[12] D. HUBER AND M. HEBERT. Fully automatic registration of multiple 3D data sets. *Image Vision Comput.* **21** (2003), 637–650.

[13] J. MCCLEARY. *User's Guide to Spectral Sequences*. Publish or Perish, Wilmington, Delaware, 1985.

[14] J. MILNOR. *Morse Theory.* Princeton Univ. Press, New Jersey, 1963.

[15] J. R. MUNKRES. *Elements of Algebraic Topology*. Addison-Wesley, Redwood City, California, 1984.

[16] Y. M. RHEE, E. J. SORIN, G. JAYACHANDRAN, E. LINDAHL AND V. PANDE. Simulations of the role of water in the protein-folding mechanism. *Proc. Natl. Acad. Sci.* **101** (2004), 6456–6461.

[17] D. RUSSELL AND L. J. GUIBAS. Exploring protein folding trajectories using geometric spanners. *In* "Proc. Pacific Sympos. Biocomput., 2005", 40–51.

[18] Y. WANG, P. K. AGARWAL, P. BROWN, H. EDELSBRUNNER AND J. RUDOLPH. Coarse and reliable geometric alignment for protein docking. *In* "Proc. Pacific Sympos. Biocomput., 2005", 65–75.