Computational Topology for Data Analysis: Notes from Book by

Tamal Krishna Dey
Department of Computer Science
Purdue University
West Lafayette, Indiana, USA 46907

Yusu Wang
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, California, USA 92093

Topic 10: Reeb Graphs

Topological persistence provides an avenue to study a function $f : X \to \mathbb{R}$ on a space X. *Reeb graphs* provide another avenue to do the same; although the summarizations produced by the two differ in a fundamental way. Topological persistence produces barcodes as a simplified signature of the function. Reeb graphs instead provides a 1-dimensional (skeleton) structure which represents a simplification of the input domain X while taking the function into account for this simplification. Of course, one loses higher dimensional homological information in the Reeb graphs, but at the same time, it offers a much lighter and computationally inexpensive transformation of the original space which can be used as a signature for tasks such as shape matching and functional similarity. An example from [48] is given in Figure 10.1, where a multiresolutional representation of the Reeb graph is used to match surface models.

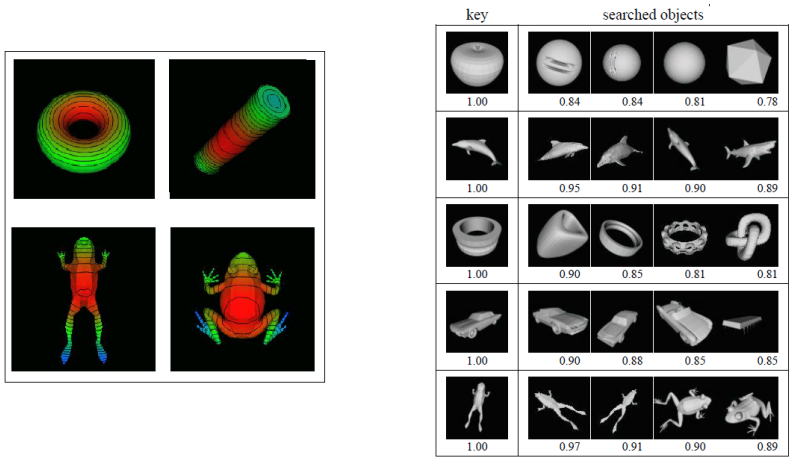


Figure 10.1: (Left). A description function based on averaging geodesic distance is shown on different models, together with some isocontours of this function. This function is robust w.r.t. near-isometric deformation of shapes. (Right) The Reeb graph of the descriptor function (from the left) is used to compare different shapes. Here, given a query shape (called “key"), the most similar shapes from a database are shown on the right. Images taken from [48].

We define the Reeb graph and introduce some properties of it in Section 10.1. We also describe efficient algorithms to compute it for the piecewise-linear setting in Section 10.2. For comparing Reeb graphs, we need to define distances among them. In Section 10.3, we present two equivalent distance measures for the Reeb graphs and give a stability result of these distances w.r.t. changes in the input function that define the Reeb graph. In particular, we note that a Reeb graph can also be viewed as a graph equipped with a “height” function on it which is induced by the original function $f : X \to \mathbb{R}$ on the input domain. This height function provides a natural metric on the Reeb graph, rendering a view of the Reeb graph as a specific metric graph. This further leads to a distance measure for Reeb graphs based on the Gromov-Hausdorff distance idea, which we present in Section 10.3. An alternative way to define distance for Reeb graph is based the interleaving idea, which we will also introduce in Section 10.3. It turns out that these two versions of distances for Reeb graphs are strongly equivalent, meaning that they are within a

constant factor of each other.

10.1 Reeb graph: Definitions and properties

Before we give a formal definition of a Reeb graph, let us recall some relevant definitions that we introduced earlier. A topological space X is *disconnected* if there are two *disjoint* open sets U and V so that $X = U \cup V$. It is called connected otherwise. A *connected component* of X is a maximal subset (subspace) that is connected. Given a continuous function $f : X \to \mathbb{R}$ on a finitely triangulable topological space X, for each $a \in \mathbb{R}$, consider level set $f^{-1}(a) = \{x \in X : f(x) = a\}$ of f. It is a subspace of X and we can talk about its connected components in this subspace topology.

Definition 1 (Reeb graph)**.** We define an equivalence relation $\sim$ on X by asserting $x \sim y$ iff $f(x) = f(y) = \alpha$ and x and y belong to the same connected component of the level set $f^{-1}(\alpha)$. Let $[x]$ denote the equivalent class to which a point $x \in X$ belongs to. The *Reeb graph* R_f of $f : X \to \mathbb{R}$ is the quotient space $X/\sim$, i.e., the set of equivalent classes equipped with the quotient topology and let $\Phi : X \to \mathsf{R}_f$, $x \mapsto [x]$ be the quotient map.

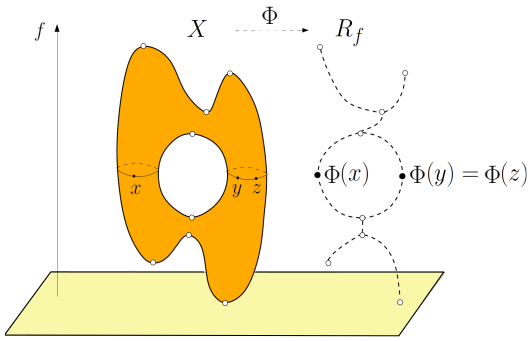


Figure 10.2: The Reeb graph R_f of the function $f : X \to \mathbb{R}$.

If the input is "nice", for example, if f is a Morse function on a compact manifold, or a PL-function on a compact polyhedron, then R_f indeed has the structure of a finite 1-dimensional regular CW complex which is a graph, and this is why it is commonly called a Reeb graph. In particular, from now on, we tacitly assume that the input function $f : X \to \mathbb{R}$ is *level-set tame*, meaning that (i) each level set $f^{-1}(a)$ has finite number of components, and each component is path connected[1]; and (2) f is of Morse type as introduced before. It is known that Morse functions on a compact smooth manifold and PL-functions on finite simplicial complexes are both level-set tame.

A level set may consist of several connected components, each of which is called a *contour*. Intuitively, the Reeb graph R_f is obtained by collapsing contours (connected components) in each level set $f^{-1}(a)$ continuously. In particular, as we vary a, R_f tracks the the changes (e.g, creation, deletion, splitting and merging) of connected components in the level sets $f^{-1}(a)$, and thus is a meaningful topological summary of $f : X \to \mathbb{R}$.

[1] A topological space $\mathbb{T}$ is path connected if any two points $x, y \in \mathbb{T}$ can be joined by a path, i.e. there exists a continuous map $f : [0, 1] \to \mathbb{T}$ of the segment $[0, 1] \subset \mathbb{R}$ onto $\mathbb{T}$ so that $f(0) = x$ and $f(1) = y$.

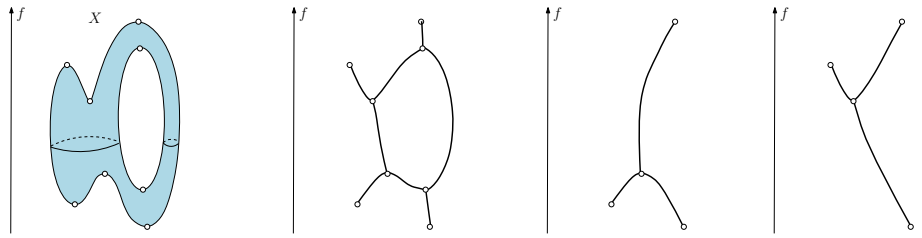
As the function f is constant on each contour in a level set, $f : X \to \mathbb{R}$ also induces a continuous function $\tilde{f} : \mathsf{R}_f \to \mathbb{R}$ defined as $\tilde{f}(z) = f(x)$ for any preimage $x \in \Phi^{-1}(z)$ of z. To simplify notation, we often write $f(z)$ instead of $\tilde{f}(z)$ for $z \in \mathsf{R}_f$ when there is no ambiguity, and use $\tilde{f}$ mostly to emphasize the different domains of the functions. In all illustrations of this chapter, we plot the Reeb graph with the vertical coordinate of a point z to be the function value $f(z)$.

Critical points. As we describe above, the Reeb graph can be viewed as the underlying space of a 1-dimensional cell complex, where there is also a function $\tilde{f} : \mathsf{R}_f \to \mathbb{R}$ defined on R_f. We can further assume that the function $\tilde{f}$ is monotone along each 1-cell of R_f – if not, we simply insert a new node where this condition fails, and the tameness of $f : X \to \mathbb{R}$ guarantees that we only need to add finite number of nodes. Hence we can view the Reeb graph as the underlying space of a 1-dimensional simplicial complex (graph) (V, E) associated with a function $\tilde{f}$ that is monotone along each edge $e \in E$. Note that we can further insert more nodes into an edge in E, breaking it into multiple edges; see, e.g., the augmented Reeb graph in Figure 10.4 (c). We now continue with this general view of the Reeb graph, whose underlying space is a graph equipped with a function $\tilde{f}$ that is monotone along each edge. We can then talk about the induced critical points (see the book). An alternative (and simpler) way to describe such critical points are as follows: Given a node $x \in V$ in the vertex set $V := V(\mathsf{R}_f)$ of the Reeb graph R_f, we use the term *up-degree* (resp. *down-degree*) of x to denote the number of edges incident to x that have higher (resp. lower) values of $\tilde{f}$ than x. A node is *regular* if both of its up-degree and down-degree equal to 1, and *critical* otherwise. A critical point is a minimum (maximum) if it has down-degree 0 (up-degree 0), and a down-fork (up-fork) if it has down-degree (up-degree) larger than 1. A critical point can be degenerate, having more than one types of criticality: e.g., a point with down-degree 0 and up-degree 2 is both a minimum and an up-fork.

Note that because of the monotonicity of $\tilde{f}$ at regular points, the Reeb graph together with its associated function is completely described, up to homeomorphisms preserving the function, by the function values on the critical points.

Now imagine that one sweeps the domain X in increasing order of f values, and tracks the changes in the connected components during this process. New components appear (down-degree 0 nodes), existing components vanish (up-degree 0), or components merge or split (down/up-forks). The Reeb graph R_f encodes such changes thereby making it a simple but meaningful topological summary of the function $f : X \to \mathbb{R}$. However, it only tracks the connected components in the level set, thus cannot capture complete information about f. Nevertheless, it reflects certain aspects about both the domain X itself and the function f defined on it, which we describe in Section 10.2.3.

Variants of Reeb graphs. Treating a Reeb graph as a simplicial 1-complex, we can talk about 1-cycles (loops) in it. A loop-free Reeb graph is also called a *contour tree*, which itself has found many applications in computer graphics and visualization. Instead of tracking the connected components within a *level set*, one can also track them within the *sublevel set* while sweeping X in increasing f values, or track them within the *superlevel set* while sweeping X in decreasing f vales. The resulting topological summaries are called the *merge tree* and the *split tree*, respectively. See the precise definition below and examples in Figure 10.3.



(a) Input scalar field (b) Reeb graph (c) Merge tree Split tree

Figure 10.3: Examples of the Reeb graph, the merge tree and the split tree of an input scalar field.

Definition 2. Define $x \sim_M y$ if and only if $f(x) = f(y) = a$ and x is connected to y within the sublevel set $f^{-1}((-\infty, a])$. Then the quotient space $T_M = X/\sim_M$ is the *merge tree* w.r.t. f.

Alternatively, if we define $x \sim_S y$ if and only if $f(x) = f(y) = a$ and x is connected to y within the superlevel set $f^{-1}([a, +\infty))$, then the quotient space $T_S = X/\sim_S$ is the *split tree* w.r.t. f.

Indeed, for level-set tame functions we consider, T_M and T_S are both finite trees. If R_f is loop-free (thus a tree), then this contour tree is uniquely decided by, and can be computed from, the merge and split trees of f.

Finally, instead of real-valued functions, one can define a similar quotient space $X/\sim$ for a continuous map $f : X \to Z$ to a general metric space (e.g, $Z = \mathbb{R}^d$), where $\sim$ is the equivalence relation $x \sim y$ iff $f(x) = f(y) = a$ and x is connected to y within the levelset $f^{-1}(a)$. The resulting structure is called *Reeb space*. See the book where we consider this generalization in the context of another structure called *mapper*.

10.2 Algorithms in the PL-setting

Piecewise-linear setting. Consider a simplicial complex K and a PL-function $f : |K| \to \mathbb{R}$ on it. Since R_f depends only on the connectivity of each level set, for a generic function f (where no two vertices have the same function value), the Reeb graph of f depends only on the 2-skeleton of K. From now on, we assume that f is generic and $K = (V, E, T)$ is a simplicial 2-complex with vertex set V, edge set E and triangle set T. Let n_v, n_e and n_t denote the size of V, E, and T, respectively, and let $m = n_v + n_e + n_t$. We sketch algorithms to compute the Reeb graph for the PL-function f. Sometimes, they output the so-called *augmented Reeb graph*, which is essentially a refinement of the Reeb graph R_f with certain additional degree-2 vertices inserted in arcs of R_f.

Definition 3 (Augmented Reeb)**.** Given a PL-function $f : |K| \to \mathbb{R}$ defined on a simplicial complex $K = (V, E, T)$, let R_f be its Reeb graph and $\Phi_f : |K| \to \mathsf{R}_f(K)$ be the associated quotient map. The *augmented Reeb graph* of $f : |K| \to \mathbb{R}$, denoted by $\widehat{\mathsf{R}}_f$, is obtained by inserting each point in $\Phi_f(V) := \{\Phi_f(v) \mid v \in V\}$ as graph nodes to R_f (if it is not already in).

For a PL-function, each critical point of the Reeb graph R_f (w.r.t. $\tilde{f} : \mathsf{R}_f \to \mathbb{R}$ induced by f) are necessarily the image of some vertex in K, and thus the critical points form a subset of points in $\Phi_f(V)$. The augmented Reeb graph $\widehat{\mathsf{R}}_f$ then includes all remaining points in $\Phi_f(V)$ as (degree-2) graph nodes. See Figure 10.4 for an example, where as a convention, we plot a node $\Phi_f(v)$ at the same height (function value) as v.

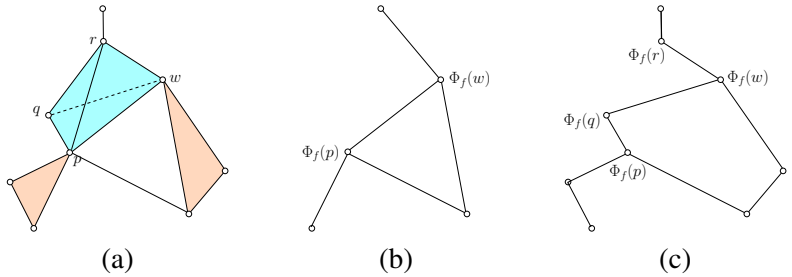


Figure 10.4: (a) A simplicial complex K. The set of 2-simpices of K include $\triangle rpq$, $\triangle rpw$, $\triangle rqw$, as well as the two pink-colored triangles incident to p and to w, respectively. (b) Reeb graph of the height function on $|K|$. (c) Its augmented Reeb graph.

We now sketch the main ideas behind two algorithms that compute the Reeb graph for a PL-function with the best time complexity, one deterministic and the other randomized.

10.2.1 A $O(m \log m)$ time algorithm via dynamic graph connectivity

Here we describe an $O(m \log m)$-time algorithm [51] for computing the Reeb graph of a PL-function $f : |K| \to \mathbb{R}$, whose time complexity is the best among all existing algorithms for Reeb graph computation. We assume for simplicity that no two vertices in V share the same f-value.

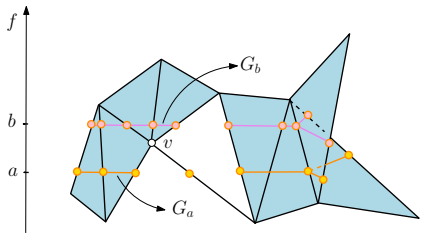


Figure 10.5: As one sweeps past v, the combinatorial structure of the pre-image graph changes. G_a has 3 connected components (one of which contains a single point only), while G_b has only 2 components.

As $K = (V, E, T)$ is a 2-dimensional simplicial complex, the level set $f^{-1}(a)$ for any function value a consists of nodes (intersection of the level set $f^{-1}(a)$ with edges in E) and edges (intersection of the level set $f^{-1}(a)$ with some triangles in T). This can be viewed as yet another graph, which we denote by $G_a = (W_a, F_a)$ and refer to as the *pre-image graph*, where each vertex in W_a corresponds to some edge in E. Each edge in F_a connects two vertices in W_a and thus can be associated to a pair of edges in E adjoining a certain triangle in T. See Figure 10.5 for an example. Obviously, connected components in G_a correspond to connected components in $f^{-1}(a)$, and under the quotient map Φ, each component is mapped to a single point in the Reeb graph R_f.

A natural idea to construct the Reeb graph R_f of $f : |K| \to \mathbb{R}$ is to sweep the domain K with increasing value of a, track the connected components in G_a during the course, and record the changes (merging or splitting of components, or creation and removal of components) in the resulting Reeb graph.

Furthermore, as f is a PL-function, the combinatorial structure of G_a can only change when we sweep past a vertex $v \in V$. When that happens, only edges / triangles from K incident to v can incur changes in G_a. See Figure 10.5. Let s_v denote the total number of simplicies incident on v. It is easy to see that as one sweeps through the vertex v, only $O(s_v)$ number of insertions and deletions are needed to update the pre-image graph G_a. To be able to build the Reeb graph R_f, we simply need to maintain the connectivity of G_a as we sweep. Assuming we have a data structure to achieve this, the high level framework of the sweep algorithm is then summarized in Algorithm 1:Reeb-SweepAlg.

Algorithm 1 Reeb-SweepAlg(K, f)

1: Sort vertices in $V = \{v_1, \ldots, v_{n_v}\}$ in increasing f-values;
2: Initialize the Reeb graph R and the pre-image grpah G_a to be the empty;
3: **for** $i = 1$ to n_v **do**
4: Lc = LowerComps(v_i)
5: UpdatePreimage(v_i); //*Update the pre-image graph* G_a;
6: Uc = UpperComps(v_i))
7: UpdateReebgraph(R, Lc, Uc, v_i);
8: **end for**
9: Output R as the Reeb graph

In particular, suppose we have a data structure, denoted by DynSF, that maintains a spanning forest of the pre-image graph at any moment. Each connected component in the pre-image graph will be associated with a vertex from V, called *representative vertex* of this component, which indicates that this component is created when passing through v. We assume that the data structure DynSF provides the following operations: First, recall that a graph node e_a in the pre-image graph $G_a = (W_a, F_a)$ in fact corresponds to an edge $e \in K$, and e_a is the intersection of e with the levelset $f^{-1}(a)$.

- Find(e): which given an edge $e \in E$, returns the representative vertex of the component in the current pre-image graph G_a containing the node e_a in W_a generated by e.
- Insert(e, e'), Delete(e, e'): which insert an edge (e_a, e'_a) into G_a, or delete (e_a, e'_a) from G_a, while still maintaining a spanning forest for G_a under these operations.

Using these operations, the pseudo-codes for the subroutines involved in Reeb-SweepAlg() are given below. (The routine UpperComps(v) is symmetric to LowerComps(v) and thus omitted.) These codes assume that edges of K not intersecting the levelsets are still in the pre-image graphs as isolated nodes; hence there is no need to add or remove isolated nodes.

Algorithm 2 LowerComps(v)

```
1: Lc = empty list;
2: for all edges e in the lower-star of v do
3:     c = DynSF.Find(e);
4:     if c is not marked 'listed' then
5:         Lc.add(c);   abd mark c as 'listed'
6:     end if
7: end for
```

Algorithm 3 UpdateReebGraph(R, Lc, Rc, v)

```
1: Create a new node v̂ in R corresponding to v;
2: Assign node v̂ to each component in Uc;
3: Create an arc in R between v̂ to the Reeb
   graph node corresponding to the representa-
   tive vertex of each c in Lc;
4: Return updated Reeb graph R
```

Algorithm 4 UpdatePreImage(v)

```
1: for all triangles uvw incident on v do
2:         // w.l.o.g, assume f(u) < f(w)
3:     if f(v) < f(u) then
4:         DynSF.Insert(vu, vw);
5:     else
6:         if f(v) > f(w) then
7:             DynSF.Delete(vu, vw);
8:         else
9:             DynSF.Delete(uv, uw);
10:            DynSF.Insert(vw, uw);
11:        end if
12:    end if
13: end for
```

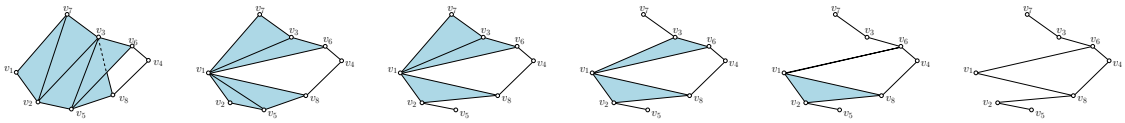
Time complexity analysis. Suppose the input simplicial 2-complex $K = (V, E, T)$ has n number of vertices and m total number of simplices. Sorting the vertices takes $O(n \log n)$ time. Then steps 4 to 7 of Algorithm Reeb-SweepAlg performs $O(m)$ number of Find, Insert and Delete operations using the data structure DynSF.

One could use state-of-the-art data structure for dynamic graph connectivity as DynSF – indeed, this is the approach of [31]. However, note that this is an offline version of the dynamic graph connectivity problem, as all insertion / deletion operations are known (or can be pre-computed). To this end, we assign each edge in the pre-image graph a weight, which is the time (f-function value) it will be deleted from the pre-image graph G_a. We then maintain a maximum spanning forest of G_a during the sweeping to maintain connectivity. In general, a deletion of a maximum-spanning tree edge (u, v) can incur expensive search in the pre-image graph for a replacement edge (as u and v may still be connected). However, given the specific way edge weights are assigned, in this case, if a maximum spanning tree edge is to be deleted, it will simply break the tree in the maximum spanning forest containing this edge, and no replacement edge needs to be identified. One can use a standard dynamic tree data structure, such as the Link-Cut trees [57], to maintain the maximum spanning forest efficient in $O(\log m)$ (amortized) time for each find / insertion / deletion operation. Putting everything together, it takes $O(m \log m)$ time to maintain the Reeb graph through the sweep. Note that we can easily modify this algorithm to update the augmented Reeb graph in the same time complexity.

Theorem 1. *Given a PL-function $f : |K| \to \mathbb{R}$, let m denote the total size of the 2-skeleton of K. One can compute the (augmented) Reeb graph R_f of f in $O(m \log m)$ time.*

10.2.2 A randomized algorithm with $O(m \log m)$ expected time

In this section we describe a randomized algorithm [46] whose expected time complexity matches the previous algorithm. However, it uses a strategy different from sweeping: Intuitively, it directly models the effect of the quotient map Φ, but does so in a randomized manner so as to obtain a good (expected) running time.



(a) (b) (c) (d) (e) (f)

Figure 10.6: The vertices are randomly ordered. Starting from the initial simplicial complex in (a), the algorithm performs vertex-collapse for vertices in this random order, as shown in (b) – (f).

In general, given $f : X \to \mathbb{R}$ and associated quotient map $\Phi : X \to \mathsf{R}_f$, each connected component (contour) C within a level set $f^{-1}(a)$ is mapped (collapsed) to a single point $\Phi(C)$ in R_f. For the case where $X = |K|$ and f is piecewise-linear over simplices in K, the image of the collection of contours passing through every vertex in V decides the nodes in the augmented Reeb graph $\widehat{\mathsf{R}}$, and intuitively contains sufficient information for constructing $\widehat{\mathsf{R}}$. The high-level algorithm to compute the augmented Reeb graph $\widehat{\mathsf{R}}$ is given in Algorithm 5:Reeb-RandomAlg. See Figure 10.6 for an illustration of the algorithm.

Algorithm 5 Reeb-RandomAlg(K, f)

Input:
A simplicial 2-complex K and a vertex function $f : V(K) \to \mathbb{R}$
Output:
The augmented Reeb graph of the PL-function induced by f

1: Let $V = \{v_1, \ldots, v_{n_v}\}$ be a random permutation of vertices in V
2: Set $K_0 = K$ and $f_0 = f$
3: **for** $i = 1$ to n_v **do**
4: Collapse the contour of $f_{i-1} : |K_{i-1}| \to \mathbb{R}$ passing through (incident to) v_i and obtain complex K_i
5: $f_i : |K_i| \to \mathbb{R}$ is the PL-function on K_i induced from f_{i-1}
6: **end for**
7: Output the final complex K_{n_v} as the augmented Reeb graph

In particular, algorithm Reeb-RandomAlg starts with function $f_0 = f$ defined on the original simplicial complex $K_0 = K$. Take a random permutation of all vertices in $V = V(K)$. At the beginning of the i-th iteration, it maintains a PL-function $f_{i-1} : |K_{i-1}| \to \mathbb{R}$ over a partially collapsed simplicial complex K_{i-1} whose augmented Reeb graph is the same as that of f. It then "collapses" the contour of f_{i-1} passing through vertex v_i and obtains a new PL-function $f_i : |K_i| \to \mathbb{R}$ over a further collapsed simplicial complex K_i that maintains the augmented Reeb graph.

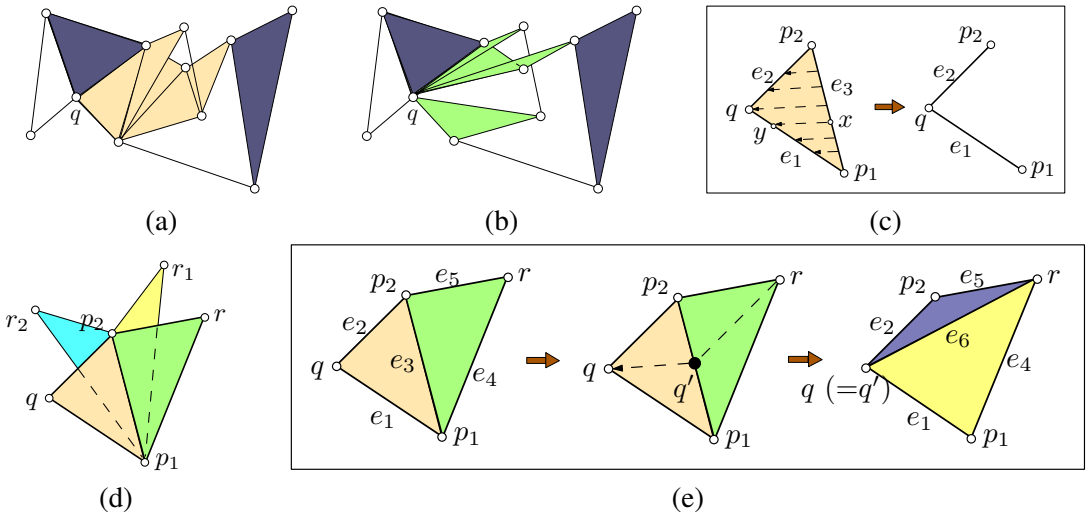


Figure 10.7: The function f is the height function. The contour incident to point q for the complex in (a) is collapsed, resulting a new complex in (b); (c) the collapse of the contour within a single triangle incident to q, and (e) an example where this triangle is bordering another triangle.

The key is to implement this “collapse" step (lines 4-5). To see the effect of collapsing the contour incident to a vertex, see Figure 10.7 (a) and (b). To see how the collapse is achieved, first consider the triangle qp_1p_2 incident to vertex q as in Figure 10.7 (c), and assume that q is the *mid-vertex* of this triangle, that is, its height value ranks second among the three vertices of the triangle. Intuitively, we need to map each horizontal segment (part of a contour at different height) to the corresponding point along the edges qp_1 and qp_2. If there are multiple triangles incident to q with q being the mid-vertex of them, then we perform this operation to each of them. If this triangle incident to q that we are collapsing has one or more triangles sharing the edge p_1p_2 as shown in (d), then for each such incident triangle, we need to process it appropriately. In particular, see one such triangle (p_1, p_2, r) in (e), then, as q' is sent to q, the dotted edge rq' now becomes edge rq in the last picture. Thus the triangle rp_1p_2 is now split into two new triangles qrp_1 and qrp_2. In this case, it is easy to see that at most one of the new triangles will have q as the mid-vertex. We collapse this triangle and continue the process until no more triangle incident to q having q as the mid-vertex is left (Figure 10.7 (b)). At this point, the entire contour passing through q is collapsed into a single point, and lines 4-5 of the algorithm are finished.

After processing each vertex in the way described above, the algorithm Reeb-RandomAlg in the end computes the final complex K_{n_v} in line 7. It is necessarily a simplicial 1-complex because no vertex can be the mid-vertex of any triangle, implying that there is no triangle left. It is easy to see that by construction, K_{n_v} is the augmented Reeb graph w.r.t. $f : |K| \to \mathbb{R}$.

Time complexity. For each vertex v, the time complexity of the collapse is proportional to the number of triangles T_v intersected by the contour C_v passing through v. In the worst case, $T_v = |n_t|$, giving rise to $O(n_v n_t)$ worst case running time of Algorithm 5. This worst case time complexity turns out to be tight. However, if one processes the vertices in a random order, then the worst case

behavior is unlikely to happen, and the expected running time can be proven to be $O(m \log n_v) = O(m \log m)$. Essentially, one argues that an original triangle from the input simplicial complex is split only $O(\log n_v) = O(\log m)$ expected number of times thus creating $O(\log m)$ expected number of intermediate triangles which takes $O(\log m)$ expected time to collapse. The argument is in spirit similar to the analysis of path length in a randomly built binary search tree [20].

Theorem 2. *Given a PL-function $f : |K| \to \mathbb{R}$ defined on a simpicial complex K whose 2-skeleton has size m, Algorithm 5 computes the (augmented) Reeb graph in $O(m \log m)$ expected time.*

10.2.3 Homology groups of Reeb graphs

Homology groups for a graph may have non-trivial ranks only in dimension zero and one. Therefore, for a Reeb graph R_f , we only need to consider $\mathsf{H}_0(\mathsf{R}_f)$ and $\mathsf{H}_1(\mathsf{R}_f)$. In particular, their rank $\beta_0(\mathsf{R}_f)$ and $\beta_1(\mathsf{R}_f)$ are simply the number of connected components and the number of independent loops in R_f respectively.

Fact 1. *For a tame function $f : X \to \mathbb{R}$, $\beta_0(X) = \beta_0(\mathsf{R}_f)$ and $\beta_1(X) \geq \beta_1(\mathsf{R}_f)$.*

That $\beta_0(X) = \beta_0(\mathsf{R}_f)$ in the above statement comes from the fact that R_f is the quotient space $X/\sim$ and each equivalent class itself is connected (it is a connected component in some level set). The relation on β_1 can be proven directly, and it is also a by-product of Theorem 4 below (combined with Fact 2). The above statement also implies that if X is simply connected, then R_f is loop-free.

For the case where X is a 2-manifold, more information about X can be recovered from the Reeb graph of a Morse function defined on it.

Theorem 3 ([19])**.** *Let $f : X \to \mathbb{R}$ be a Morse function defined on a connected and compact 2-manifold of genus g. Then:*

(i) if X is orientable, then $\beta_1(\mathsf{R}_f) = g$; and

(ii) if X is non-orientable, then $\beta_1(\mathsf{R}_f) \leq g/2$.

We now present a result that characterizes $\mathsf{H}_1(\mathsf{R}_f)$ w.r.t. $\mathsf{H}_1(X)$ in a more precise manner, which also generalizes Theorem 3.

Horizontal and vertical homology. Given a continuous function f, let $X_{=a} := f^{-1}(a)$ and $X_I := f^{-1}(I)$ denote its level set and interval set as before for $a \in \mathbb{R}$ and for an open or closed interval $I \subseteq \mathbb{R}$ respectively. We first define the so-called horizontal and vertical homology groups with respect to f.

A p-th homology class $h \in \mathsf{H}_p(X)$ is *horizontal* if there exists a finite set of values $\{a_i \in \mathbb{R}\}_{i \in A}$, where A is a finite index set, such that h has a pre-image under the map $\mathsf{H}_p(\bigcup_{i \in A} X_{=a_i}) \to \mathsf{H}_p(X)$ induced by inclusion. The set of horizontal homology classes form a subgroup $\overline{\mathsf{H}}_p(X)$ of $\mathsf{H}_p(X)$ since the trivial homology class is horizontal, and the addition of any two horizontal homology class is still horizontal. We call this subgroup $\overline{\mathsf{H}}_p(X)$ the *horizontal homology group of X with respect to f*. The *vertical homology group of X with respect to f* is then defined as:

$$\check{\mathsf{H}}_p(X) := \mathsf{H}_p(X)/\overline{\mathsf{H}}_p(X), \text{ the quotient of } \mathsf{H}_p(X) \text{ with } \overline{\mathsf{H}}_p(X).$$

The coset $\omega + \overline{\mathsf{H}}_p(X)$ for every class $\omega \in \mathsf{H}_p(X)$ provides an equivalence class in $\check{\mathsf{H}}_p(X)$. We call h a *vertical homology class* if $h + \overline{\mathsf{H}}_p(X)$ is not identity in $\check{\mathsf{H}}_p(X)$. In other words, $h \notin \overline{\mathsf{H}}_p(X)$. Two homology classes h_1 and h_2 are *vertically homologous* if $h_1 \in h_2 + \overline{\mathsf{H}}_p(X)$.

Fact 2. *By definition,* $\text{rank}\,(\mathsf{H}_p(X)) = \text{rank}\,(\overline{\mathsf{H}}_p(X)) + \text{rank}\,(\check{\mathsf{H}}_p(X))$.

Let I be a *closed interval* of $\mathbb{R}$. We define the *height of* $I = [a, b]$ to be $\mathit{height}(I) = |b - a|$; note that the height could be 0. Given a homology class $h \in \mathsf{H}_p(X)$ and an interval I, we say that h is *supported by* I if $h \in (i_*)$ where $i_* : \mathsf{H}_p(X_I) \to \mathsf{H}_p(X)$ is the homomorphism induced by the canonical inclusion $X_I \hookrightarrow X$. In other words, X_I contains a p-cycle γ from the homology class h. We define the *height of a homology class* $h \in \mathsf{H}_p(X)$ to be

$$\mathit{height}(h) = \inf_{I \text{ supports } h} \mathit{height}(I).$$

Isomorphism between $\check{\mathsf{H}}_1(X)$ and $\mathsf{H}_1(\mathsf{R}_f)$. The surjection $\Phi : X \to \mathsf{R}_f(X)$ induces a chain map $\Phi_\#$ from the 1-dimensional singular chain group of X to the 1-dimensional singular chain group of $\mathsf{R}_f(X)$ which eventually induces a homomorphism $\Phi_* : \mathsf{H}_1(X) \to \mathsf{H}_1(\mathsf{R}_f(X))$. For the horizontal subgroup $\overline{\mathsf{H}}_1(X)$, we have that $\Phi_*(\overline{\mathsf{H}}_1(X)) = \{0\} \subseteq \overline{\mathsf{H}}_1(\mathsf{R}_f(X))$. Hence Φ_* induces a well-defined homomorphism between the quotient groups

$$\check{\Phi} : \check{\mathsf{H}}_1(X) = \frac{\mathsf{H}_1(X)}{\overline{\mathsf{H}}_1(X)} \to \frac{\mathsf{H}_1(\mathsf{R}_f(X))}{\overline{\mathsf{H}}_1(\mathsf{R}_f(X))} = \mathsf{H}_1(\mathsf{R}_f(X)).$$

The right equality above follows from that $\overline{\mathsf{H}}_1(\mathsf{R}_f(X)) = \{0\}$, which holds because every level set of $\mathsf{R}_f(X)$ consists only of a finite set of disjoint points due to the levelset-tameness of function $f : X \to$. It turns out that $\check{\Phi}$ is an isomorphism – Intuitively, this is not surprising as Φ maps each contour in the level set to a single point, which in turn also collapses every horizontal cycle.

Theorem 4. *Given a level-set tame function $f : X \to \mathbb{R}$, let $\check{\Phi} : \check{\mathsf{H}}_1(X) \to \mathsf{H}_1(\mathsf{R}_f(X))$ be the homomorphism induced by the surjection $\Phi : X \to \mathsf{R}_f(X)$ as defined above. Then the map $\check{\Phi}$ is an isomorphism. Furthermore, for any vertical homology class $h \in \check{\mathsf{H}}_1(X)$, we have that $\mathit{height}(h) = \mathit{height}(\check{\Phi}(h))$.*

Persistent homology for $f : \mathsf{R}_f \to \mathbb{R}$. We have discussed earlier that the Reeb graph of a level-set tame function $f : X \to \mathbb{R}$ can be represented by a graph whose edges have monotone function values. Then, the function $f : \mathsf{R}_f \to \mathbb{R}$ can be treated as a PL-function on the simplicial 1-complex R_f. This gives rise to the standard setting where a PL-function f is defined on a simplicial 1-complex R_f whose persistence is to be computed. We can apply algorithm ZeroPerDg from the book to compute the 0-th persistence diagram $\text{Dgm}_0(f)$. For computing one dimensional persistence diagram $\text{Dgm}_1(f)$, one can modify this algorithm slightly by registering the function values of the edges that create cycles. These are edges that connect vertices in the same component. The function values of these edges are the birth points of the 1-cycles that never die. This algorithm takes $O(n \log n + m\alpha(n))$ time where m and n are the number of vertices and edges respectively in R_f.

We can also compute the levelset zigzag persistence of f using the zigzag persistence algorithm as we described before. Only the zeroth persistence diagram $\mathrm{Dgm}_0(f)$ is nontrivial in this case. We can read the zeroth persistence diagram for the standard persistence using Theorem **??** from this level set persistence diagram. Furthermore, for every infinite bar $[a_i, \infty)$ in the standard one dimensional persistence diagram, we can get the pairing (a_j, a_i) (open-open bar) in the zeroth levelset diagram $\mathrm{Dgm}_0(f)$.

Reeb graphs can be a useful tool to compute the zeroth level set zigzag persistence diagram of a function on a topological space. Let $f : X \to \mathbb{R}$ be a continuous function whose zeroth persistence diagram we want to compute. We already observed that the function f induces a continuous function on the Reeb graph R_f. To distinguish the two domains more explicitly, we denote the former function f^X and the latter as f^R. The following observation helps computing the zeroth levelset zigzag persistence diagram $\mathrm{Dgm}_0(f^X)$ because computationally it is much harder to process a space, say the underlying space of a simplicial complex than only a graph (simplicial 1-complex).

Proposition 5. $\mathrm{Dgm}_0(f^X) = \mathrm{Dgm}_0(f^R)$ *where the diagrams are for the zeroth levelset zigzag persistence.*

The result follows from the following observation. Consider the levelset zigzag filtrations $\mathcal{F}^X$ and $\mathcal{F}^R$ for the two functions.

$$\mathcal{F}^X : X_{(a_0,a_2)} \hookleftarrow \cdots \hookrightarrow X_{(a_{i-1},a_{i+1})} \hookleftarrow X_{(a_i,a_{i+1})} \hookrightarrow X_{(a_i,a_{i+2})} \hookleftarrow \cdots \hookrightarrow X_{(a_{n-1},a_{n+1})}$$

$$\mathcal{F}^R : \mathsf{R}_{f_{(a_0,a_2)}} \hookleftarrow \cdots \hookrightarrow \mathsf{R}_{f_{(a_{i-1},a_{i+1})}} \hookleftarrow \mathsf{R}_{f_{(a_i,a_{i+1})}} \hookrightarrow \mathsf{R}_{f_{(a_i,a_{i+2})}} \hookleftarrow \cdots \hookrightarrow \mathsf{R}_{f_{(a_{n-1},a_{n+1})}}$$

Using notation for interval sets $X_i^j = X_{(a_i,a_j)}$ and $\binom{j}{i} = \mathsf{R}_{f_{(a_i,a_j)}}$, we have the following commutative diagram between the 0-th level set zigzag persistence modules.

$$\begin{array}{lccccccccc}
\mathsf{H}_0\mathcal{F}^X : & \mathsf{H}_0(X_0^0) & \longrightarrow & \mathsf{H}_0(X_0^1) & \longleftarrow & \mathsf{H}_0(X_1^1) & \cdots \longrightarrow & \mathsf{H}_0(X_{n-1}^n) & \longleftarrow & \mathsf{H}_0(X_n^n) \\
& \| & & \| & & \| & & \| & & \| \\
\mathsf{H}_0\mathcal{F}^R : & \mathsf{H}_0(\binom{0}{0}) & \longrightarrow & \mathsf{H}_0(\binom{1}{0}) & \longleftarrow & \mathsf{H}_0(\binom{1}{1}) & \cdots \longrightarrow & \mathsf{H}_0(\binom{n}{n-1}) & \longleftarrow & \mathsf{H}_0(\binom{n}{n})
\end{array}$$

All vertical maps are isomorphism because the number of components in X_j^i is exactly equal to the number of components in the quotient space $\binom{i}{j} = X_j^i/\sim$ which is used to define the Reeb graph. All horizontal maps are induced by inclusions. It follows that every square in the above diagram commutes. Therefore the above two modules are isomorphic.

10.3 Distance for Reeb graphs

Several distance measures have been proposed for Reeb graphs. In this section, we introduce two distances, one based on a natural interleaving idea, and the other based on the Gromov-Hausdorff distance idea. It has been shown that these two distance measures are strongly equivalent, that is, they are within a constant factor of each other for general Reeb graphs. For the special case of *merge trees*, the two distance measures are exactly the same.

So far, we have used R_f to denote the Reeb graph of a function f. For notational convenience, in the following we use a different notation $\mathbb{F}$ for R_f. Suppose we are given two Reeb graphs $\mathbb{F}$ and $\mathbb{G}$ with the functions $f : \mathbb{F} \to \mathbb{R}$ and $g : \mathbb{G} \to \mathbb{R}$ associated to them. To emphasize the associated functions we write $(\mathbb{F}, f)$ and $(\mathbb{G}, g)$ in place of $\mathbb{F}$ and $\mathbb{G}$ when convenient. Again, we assume that each Reeb graph is a finite simplicial 1-complex and the function is strictly monotone on each edges. Our goal is to develop a concept of distance $\mathsf{d}(\mathbb{F}, \mathbb{G})$ between them. Intuitively, if two Reeb graphs are "the same", then they are isomorphic and the function value of each point is also preserved under the isomorphism. If two Reeb graphs are not the same, we aim to measure how far it deviates from being "isomorphic". The two distances we introduce below both follow this intuition, but measures the "deviation" differently.

10.3.1 Interleaving distance

We borrow the idea of interleaving between persistence modules to define a distance between Reeb graphs. Roughly speaking, instead of requiring that there is an isomorphism between the two Reeb graphs, which would give rise to a pair of maps between them, $\phi : \mathbb{F} \to \mathbb{G}$ and $\phi^{-1} : \mathbb{G} \to \mathbb{F}$ that is function preserving, we look for the existence of a pair of "compatible" maps between appropriately "thickened" versions of $\mathbb{F}$ and $\mathbb{G}$ and the distance is measured by the minimum amount of the "thickening" needed. We make this more precise below. Given any space X, set $X_\varepsilon := X \times [-\varepsilon, \varepsilon]$.

Definition 4. Given a Reeb graph $(\mathbb{F}, f)$, its *ε-smoothing*, denoted by $\mathrm{S}_\varepsilon(\mathbb{F}, f)$, is the Reeb graph of the function $f_\varepsilon : \mathbb{F}_\varepsilon \to \mathbb{R}$ where $f_\varepsilon(x, t) = f(x) + t$, $t \in [-\varepsilon, \varepsilon]$. In other words, $\mathrm{S}_\varepsilon(\mathbb{F}, f) = \mathbb{F}_\varepsilon / \sim_{f_\varepsilon}$, where $\sim_{f_\varepsilon}$ denotes the equivalence relation where $x \sim_{f_\varepsilon} y$ iff $x, y \in \mathbb{F}_\varepsilon$ are from the same contour of f_ε.

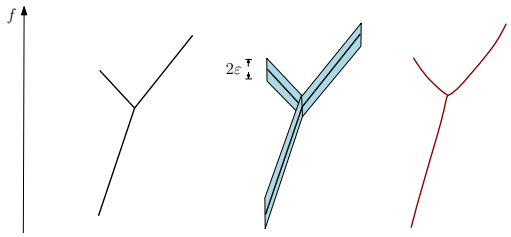


Figure 10.8: From left to right, we have the Reeb graph $(\mathbb{F}, f)$, its ε-thicking $(\mathbb{F}_\varepsilon, f_\varepsilon)$, and the Reeb graph $\mathrm{S}_\varepsilon(\mathbb{F}, f)$ of $f_\varepsilon : \mathbb{F}_\varepsilon \to \mathbb{R}$.

See Figure 10.8 for an example. As $\mathrm{S}_\varepsilon(\mathbb{F}, f)$ is the quotient space $\mathbb{F}_\varepsilon / \sim_{f_\varepsilon}$, we use $[x, t]$, $x \in \mathbb{F}, t \in [-\varepsilon, \varepsilon]$, to denote a point in $\mathrm{S}_\varepsilon(\mathbb{F}, f)$, which is the equivalent class of $(x, t) \in \mathbb{F}_\varepsilon$ under the equivalence relation $\sim_{f_\varepsilon}$. Also note that there is a natural *"quotiented-inclusion" map* $\iota : (\mathbb{F}, f) \to \mathrm{S}_\varepsilon(\mathbb{F}, f)$ defined as $\iota(x) = [x, 0]$, for any $x \in \mathbb{F}$.

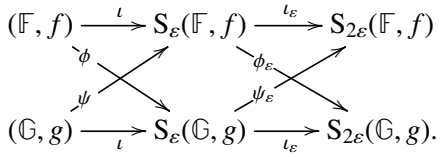
Suppose we have two Reeb graphs (A, f_a) and (B, f_b). A map $\mu : (A, f_a) \to (B, f_b)$ between them is *function-preserving* if $f_a(x) = f_b(\mu(x))$ for each $x \in A$. A function-preserving map μ between (A, f_a) and $\mathrm{S}_\varepsilon(B, f_b)$ induces a function-preserving map μ_ε between $\mathrm{S}_\varepsilon(A, f_a)$ and $\mathrm{S}_{2\varepsilon}(B, f_b)$ as follows:

$$\mu_\varepsilon : \mathrm{S}_\varepsilon(A, f_a) \to \mathrm{S}_{2\varepsilon}(B, f_b), \qquad \text{s. t,} \quad [x, t] \mapsto [\mu(x), t].$$

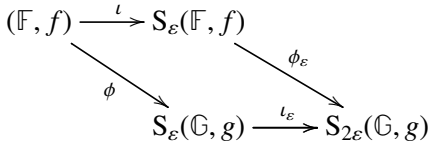
Now consider the "quotiented-inclusion" map ι introduced earlier, and suppose we also have a pair of function-preserving maps $\phi : (\mathbb{F}, f) \to \mathrm{S}_\varepsilon(\mathbb{G}, g)$ and $\psi : (\mathbb{G}, g) \to \mathrm{S}_\varepsilon(\mathbb{F}, f)$. Using the above construction, we then obtain the following maps:

$$\begin{aligned} \iota_\varepsilon &: \mathrm{S}_\varepsilon(\mathbb{F}, f) \to \mathrm{S}_{2\varepsilon}(\mathbb{F}, f), & [x, t] &\mapsto [x, t], \\ \phi_\varepsilon &: \mathrm{S}_\varepsilon(\mathbb{F}, f) \to \mathrm{S}_{2\varepsilon}(\mathbb{G}, g), & [x, t] &\mapsto [\phi(x), t] \\ \psi_\varepsilon &: \mathrm{S}_\varepsilon(\mathbb{G}, g) \to \mathrm{S}_{2\varepsilon}(\mathbb{F}, f), & [y, t] &\mapsto [\psi(y), t] \end{aligned}$$

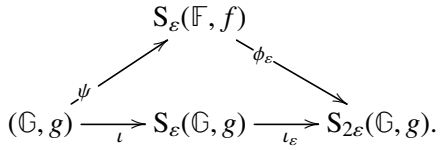
Definition 5 (Reeb graph interleaving)**.** A pair of continuous maps $\phi : (\mathbb{F}, f) \to \mathrm{S}_\varepsilon(\mathbb{G}, g)$ and $\psi : (\mathbb{G}, g) \to \mathrm{S}_\varepsilon(\mathbb{F}, f)$ are *ε-interleaved* if (i) both of them are function preserving, and (ii) the following diagram commutes:



One can recognize that the above requirements of commutativity mirror the rectangular and triangular commutativity in case of persistence modules. It is easy to verify the rectangular commutativity, that is, to verify that the following diagram (and its symmetric version involving maps ψ and ψ_ε) commutes.



Rectangular commutativity however does not embody the interaction between maps ϕ and ψ. The key technicality lies in verifying the triangular commutativity, that is, ϕ and ψ make the diagram below (and its symmetric version) commute:



For sufficiently large ε, $\mathrm{S}_\varepsilon(A, f_a)$ for any Reeb graph becomes a single segment with monotone function values on it. Hence one can always find maps ϕ and ψ that are ε-interleaved for sufficiently large ε. On the other hand, if $\varepsilon = 0$, then this implies $\psi = \phi^{-1}$. Hence the smallest ε accommodating ε-interleaved maps indicates how far the input Reeb graphs are from being identical. This forms the intuition behind defining the following distance between Reeb graphs.

Definition 6 (Interleaving distance)**.** Given two Reeb graphs $(\mathbb{F}, f)$ and $(\mathbb{G}, g)$, the interleaving distance between them is defined as:

$$\mathsf{d}_I(\mathbb{F}, \mathbb{G}) = \inf\{\varepsilon \mid \text{there exists a pair of } \varepsilon\text{-interleaved maps between } (\mathbb{F}, f) \text{ and } (\mathbb{G}, g)\}. \quad (10.1)$$

10.3.2 Functional distortion distance

We now define another distance between Reeb graphs called *functional distortion distance* which takes a metric space perspective. It views a Reeb graph as an appropriate metric space, and measures the distance between two Reeb graphs via a construction used for defining Gromov-Hausdorff distances.

Definition 7 (Function-induced metric)**.** Given a path π from u to v in a Reeb graph (A, f_a), the *height of* π is defined as

$$height(\pi) = \max_{x \in \pi} f_a(x) - \min_{x \in \pi} f_a(x).$$

Let $\Pi(u, v)$ denote the set of all paths between two points $u, v \in A$. The *function-induced metric* $\mathsf{d}_{f_a} : A \times A \to \mathbb{R}$ *on* A induced by f_a is defined as

$$\mathsf{d}_{f_a}(u, v) = \min_{\pi \in \Pi(u,v)} height(\pi).$$

In other words, $\mathsf{d}_{f_a}(u, v)$ is the minimum length of any closed interval $I \subset \mathbb{R}$ such that u and v are in the same path component of $f_a^{-1}(I)$. It is easy to verify for a finite Reeb graph, the function-induced distance d_{f_a} is indeed a proper metric on it, and hence we can view the Reeb graph (A, f_a) as a metric space (A, d_{f_a}).

Definition 8 (Functional distortion distance)**.** Given two Reeb graphs $(\mathbb{F}, f)$ and $(\mathbb{G}, g)$, and a pair of continuous maps $\Phi : \mathbb{F} \to \mathbb{G}$ and $\Psi : \mathbb{G} \to \mathbb{F}$, set

$$C(\Phi, \Psi) = \{(x, y) \in \mathbb{F} \times \mathbb{G} \mid \Phi(x) = y, \text{or } x = \Psi(y)\}$$

and

$$D(\Phi, \Psi) = \sup_{(x,y),(x',y') \in C(\Phi,\Psi)} \frac{1}{2} \left| \mathsf{d}_f(x, x') - \mathsf{d}_g(y, y') \right|.$$

The *functional distortion distance* between $(\mathbb{F}, f)$ and $(\mathbb{G}, g)$ is defined as:

$$\mathsf{d}_{FD}(\mathbb{F}, \mathbb{G}) = \inf_{\Phi,\Psi} \max\{ D(\Phi, \Psi), \ \|f - g \circ \Phi\|_\infty, \ \|g - f \circ \Psi\|_\infty \}. \tag{10.2}$$

Note that the maps Φ and Ψ are not required to preserve function values; however the terms $\|f - g \circ \Phi\|_\infty$ and $\|g - f \circ \Psi\|_\infty$ bound the difference in function values under the maps Φ and Ψ. If we ignore these two terms $\|f - g \circ \Phi\|_\infty$ and $\|g - f \circ \Psi\|_\infty$, and if we do not assume that Φ and Ψ have to be continuous, then d_{FD} is the simply the Gromov-Hausdorff distance between the metric spaces $(\mathbb{F}, \mathsf{d}_f)$ and $(\mathbb{G}, \mathsf{d}_g)$ [44]. The above definition is thus a function-adapted version of the continuous Gromov-Hausdorff distance [2].

[2] if one removes the requirement of continuity on Φ and Ψ, the resulting functional distortion distance takes values within a constant factor of d_{FD} we defined.

Properties of the distances. It turns out that the two distances we introduced are strongly equivalent.

Theorem 6 (Bi-Lipschitz equivalence)**.** $\mathsf{d}_{FD} \leq 3\mathsf{d}_I \leq 3\mathsf{d}_{FD}$.

Furthermore, it is known that for Reeb graphs $\mathsf{R}_f, \mathsf{R}_g$ derived from two "nice" functions $f, g : X \to \mathbb{R}$ defined on the same domain X, both distances are *stable* [22, 4].

Definition 9 (Stable distance)**.** Given $f, g : X \to \mathbb{R}$, let $(\mathsf{R}_f, \tilde{f})$ and $(\mathsf{R}_g, \tilde{g})$ be the Reeb graph of f and g, respectively. We say that a Reeb graph distance d_R is *stable* if

$$\mathsf{d}_R((\mathsf{R}_f, \tilde{f}), (\mathsf{R}_g, \tilde{g})) \leq \|f - g\|_\infty.$$

Finally, it is also known that these distances are bounded from below (up to a constant factor) by the bottleneck distance between the persistence diagrams associated to the two input Reeb graphs. In particular, given $(\mathbb{F}, f)$ (and similarly for $(\mathbb{G}, g)$), consider the 0-th persistence diagram $\mathrm{Dgm}_0(f)$ induced by the levelset zigzag-filtration of f as in previous section. We consider only the 0-th persistence homology as each level set $f^{-1}(a)$ consists of only a finite set of points. We have the following result (see Theorem 3.2 of [8]).

Theorem 7. $\mathsf{d}_b(\mathrm{Dgm}_0(f), \mathrm{Dgm}_0(g)) \leq 2\mathsf{d}_I(\mathsf{R}_f, \mathsf{R}_g) \leq 2\mathsf{d}_{FD}(\mathsf{R}_f, \mathsf{R}_g)$.

Universal Reeb graph distance. We introduced two Reeb graph distances above. There are other possible distances for Reeb graphs, such as the *edit distance* originally developed for Reeb graphs induced by functions on curves and surfaces. All these distances are stable, which is an important property to have. The following concept allows one to identify the most "discriminative" Reeb graph distance among all stable distances.

Definition 10. A Reeb graph distance d_U is *universal* if and only if (i) d_U is stable; and (ii) for any other stable Reeb graph distance d_S, we have $\mathsf{d}_S \leq \mathsf{d}_U$.

It has been shown that neither the interleaving distance nor the functional distortion distance is universal. On the other hand, for Reeb graphs of piecewise-linear functions defined on compact triangulable spaces, such universal Reeb graph distance indeed exists. In particular, one can construct a universal Reeb graph distance via a pullback idea to a common space; see [62]. The authors of [62] propose two further edit-like distances for Reeb graphs, both of which are universal.

Computation. Unfortunately, except for the bottleneck distance d_b, the computation of any of the distances mentioned above is at least as hard as graph isomorphism. In fact, even for merge trees (which are simpler variant of the Reeb graph, described in Definition 2 at the end of Section 10.1), it is NP-hard to compute the interleaving distance between them [3]. But for this special case, a fixed-parameter tractable algorithm exists [60].

10.4 Notes and Exercises

The Reeb graph was originally introduced for Morse functions [54]. It was naturally extended to more general spaces as it does not rely on smooth / differential structures. This graph, as a summary of a scalar field, has found many applications in graphics, visualization and more recently in data analysis; see e.g. [6, 7, 14, 24, 38, 47, 48, 56, 58, 61, 63]. Its loop free version, the contour tree, has many applications of its own. Properties of the Reeb graph has been studied in [19, 28, 33]. The concept of Reeb space was introduced in [34].

An $O(m \log m)$ algorithm to compute the Reeb graph of a function on a triangulation of a 2-manifold is given in [19], where m is the size of the triangulation: In particular, it follows a similar high level framework as in Reeb-SweepAlg() described Algorithm 1. For the case where K represents the triangulation of a 2-manifold, the pre-image graph G_a has a simpler structure (a collection of disjoint loops for a generic a value). Hence the connectivity of G_as can be maintained efficiently in $O(\log n_v)$ time, rendering an $O(m \log n_v) = O(m \log m)$ time algorithm to compute the Reeb graph [19]. Several subsequent algorithms are proposed to handle more general cases; e.g, [30, 31, 52, 59]. The best existing algorithm for computing the Reeb graph of a PL-function defined on a simplicial complex, as described in Section 10.2.1, was proposed by Parsa in [51]. The randomized algorithm with the same time complex (in expectation) described in Section 10.2.2 was given in [46]. The loop-free version of the Reeb graph, namely, the contour tree, can be computed much more efficiently in $O(n \log n)$ time, where n is the total number of vertices and edges in the input simplicial complex domain [12]. As a by-product, this algorithm also computes both the merge tree and split tree of the input PL-function within the same time complexity.

The concepts of horizontal and vertical homology groups were originally introduced in [18] for any dimensions. The specific connection of the 1-dimensional case to the Reeb graphs (e.g., Theorem 4) was described in [28]. The discussion of persistent homology for $f : \mathsf{R}_f \to \mathbb{R}$ follows from [23]. The 0-th level set zigzag persistence (or equivalently, the 0-th and 1-st extended persistence) for the Reeb graph can be computed in $O(n \log n)$ time using the mergeable tree algorithm of [39].

The interleaving distance of merge trees was originally introduced by Morozov et al. in [50]. The interleaving distance for the Reeb graphs is more complicated, and was introduced by de Silva et al. [22]. There is also an equivalent cosheave-theoretical way of defining the interleaving distance. Its description involves the sheaf theory [21]. The functional distortion distance for Reeb graphs was originally introduced in [4], and its relation to interleaving distance was studied in [5]. The lower-bound in Theorem 7 was proven in [8]; while some weaker bounds were earlier given in [9, 5]. An interesting distance between Reeb graphs can be defined by mapping its levelset zigzag persistence module to a 2-parameter persistence module. See the Notes in Chapter 12 of the book for more details. The edit distance for Reeb graphs induced by functions on curves or surfaces has been proposed in [35, 36]. Finally, the universality of Reeb graph distance and universal (edit-like) distance for Reeb graphs was proposed and studied in [62]. It remains an interesting open question whether the interleaving distance (and thus functional distortion distance) is within a constant factor of the universal Reeb graph distance.

Exercise

1. Suppose we are given a triangulation K of a 2-dimensional square. Let $f : |K| \to \mathbb{R}$ be a PL-function on K induced by a vertex function $f|_V : V(K) \to \mathbb{R}$. Assume that all vertices have distinct function values.

   (1.a) Given a value $a \in \mathbb{R}$, describe the topology of the contour $f^{-1}(a)$.

   (1.b) As we vary a continuously from $-\infty$ to $+\infty$, show that the connectivity of $f^{-1}(a)$ can only change when a equals $f(v)$ for some $v \in V(K)$.

   (1.c) Enumerate all cases of topological changes of contours when a passes through $f(v)$ for some $v \in V$.

2. Given a finite simplicial complex K and a PL-function f induced by $f|_V : V(K) \to \mathbb{R}$, let R_f be the Reeb graph w.r.t. f. Suppose we add a new simplex σ of dimension 1 or 2 to K, and let K' be the new simplicial complex. Describe how to obtain the new Reeb graph $\mathsf{R}_f(K')$ from $\mathsf{R}_f(K)$.

3. Given a finite simplicial complex K, let n_d denote the number of d-dimensional simplices in K. Let f be a PL-function on K induced by $f : V(K) \to \mathbb{R}$, and assume that all n_0 vertices in $V(K)$ are already sorted in non-decreasing order of f. Describe an algorithm to compute the merge tree for K w.r.t. f, and give the time complexity of your algorithm. (Make your algorithm as efficient as possible.)

Topic 11: Topological Analysis of Graphs

In this chapter, we present some examples of topological tools that help analyze or summarize graphs. In the previous chapter, we already discussed one specific type of graph, the Reeb graph, obtained by quotienting a space with the connected components of levelsets of a given function. Abstractly, a Reeb graph can also be considered as a graph equipped with a height function. In this chapter, we focus on general graphs. Structures such as cliques in a graph correspond to simplices as we have seen in Vietoris-Rips complexes. They can help summarizing or characterizing graph data. See Figure 10.9 for an example [55], where a directed graph is used to model the synaptic network of neurons built by taking neurons as the vertices and the synaptic connections directed from pre- to postsynaptic neurons as the directed edges. It is observed that there are unusually

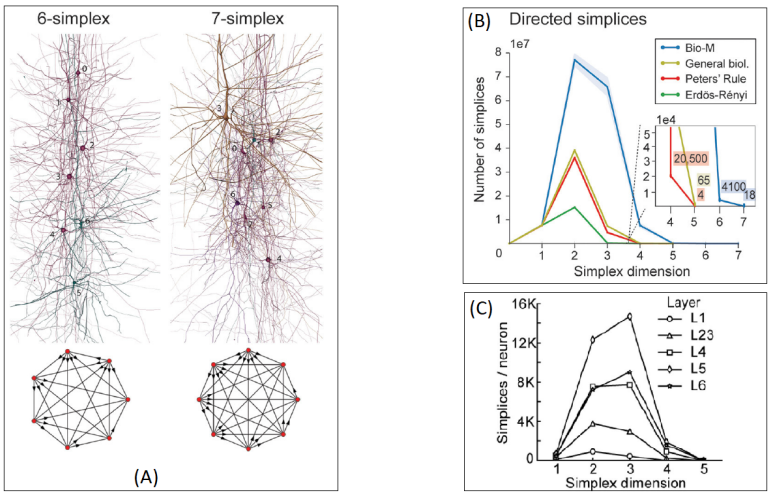


Figure 10.9: (A) shows examples of two directed cliques (simplices) formed in the synaptic network. (B) shows the number of p-simplices for different types of graphs, where "Bio-M" is the synaptic network from reconstructed neurons. Note that this neuronal network has far more directed cliques than other biological or random graphs. (C) shows that the count of directed cliques further differ depending on which layers neurons reside. Image taken from [55].

high number of directed cliques (viewed as a simplex as we show in Section 10.7.1) in such networks, compared to other biological networks or random graphs, see Figure 10.9. Topological analysis such as the one described in Section 10.7 can facilitate such applications.

Specifically, Sections 10.5 and 10.6 focus on topological analysis of undirected graphs. We present topological approaches to summarize undirected graphs as well as to compare them. In Section 10.7, we discuss how to obtain topological invariant of directed graphs. In particular, we describe two ways to define homology for directed graphs. The first is by constructing an appropriate simplicial complex over an input directed graph and then taking the corresponding simplicial homology (Section 10.7.1). The second approach is the so-called path homology for directed graphs, which is not rooted in a simplicial homology. Rather, it is based on constructing a specific chain complex directly from directed paths in the input graph, and defining a homology

group using the boundary operators associated to the resulting chain complex (Section 10.7.2). It turns out that both path homology and the persistent version of it can be computed via a similar matrix reduction algorithm as the standard persistence algorithm (for simplicial complexes) though with some key differences. We describe this algorithm in Section 10.7.3, and mention an improved algorithm for the 1-st homology.

10.5 Topological summaries for graphs

We have seen graphs in various contexts so far. Here we consolidate some of the persistence results specifically suited for graphs. Sometimes graphs appear as abstract objects devoid of any geometry where they are described only combinatorially. At other times, graphs are equipped with a function or a metric. Reeb graphs studied in the previous chapter fall into this latter category. We present results specifically developed for different viewpoints on graphs. Nevertheless, (weighted or unweighted) combinatorial graphs can also be viewed as metric graphs, by associating with it an appropriate shortest path metric (see Section 10.5.2).

10.5.1 Combinatorial graphs

A graph G is combinatorially presented as a pair $G = (V, E)$, where V is a set called nodes/vertices of G and $E \subseteq V \times V$ is a set of pairs of vertices called edges of G. We now introduce two common ways to obtain a persistence-based topological summary for G.

Graphs viewed as a simplicial 1-complex. We can view G as a simplicial 1-complex with V and E being the set of 0-simplicies and 1-simplices respectively. Using tools developed for persistence, we can then summarize G w.r.t. a given PL-function $f : |G| \to \mathbb{R}$ by the persistence diagram $\mathrm{Dgm} f$. This is what was done in Section 10.2.3 in the previous chapter while describing persistent homology for Reeb graphs. In practice, the chosen PL-functions are sometimes called *descriptor functions*. For example, we can choose $f : |G| \to \mathbb{R}$ to be given by a vertex function called the degree-function, where $f(v)$ equals the degree of the graph node v in G. Some other choices for the descriptor function include the heat-kernel signature function [13] and the Ollivier Ricci curvature of graphs [64]. Note that, under this view, given that the domain is a simplicial 1-complex, there is only zeroth and 1-st persistent homology to consider.

Clique complex view. Given a graph $G = (V, E)$, its induced *clique complex*, also called the *flag complex*, is defined as follows.

Definition 11 (Clique complex)**.** Given a graph $G = (V, E)$, a *clique simplex* σ of dimension k is

$$\sigma = \{v_{i_0}, \ldots, v_{i_k}\} \text{ where either } k = 0 \text{ or for any } \; j \neq j' \in [0, k], \;\; (v_{i_j}, v_{i_{j'}}) \in E.$$

By definition, every face of a clique simplex is also a clique simplex. Therefore, the collection of all clique simplices form a simplicial complex C_G called the *clique complex* of G. In other words, the vertices of any $(k+1)$-clique in G spans a k-simplex in C_G.

Given a weighted graph $G = (V, E, \omega)$ with $\omega : E \to \mathbb{R}$, let G^a denote the subgraph of G spanned by all edges with weight at most a; that is, $G^a = (V^a, E^a)$ where $E^a = \{(u, v) \mid \omega(u, v) \leq a\}$ and V^a is the vertex set adjoining E^a. Let C_{G^a} be the clique complex induced by G^a. It is easy to see that $\mathrm{C}_{G^a} \subseteq \mathrm{C}_{G^b}$ for any $a \leq b$. Assuming all edges $E = \{e_1, \ldots, e_m\}$ are sorted in non-decreasing order of their weights and setting $a_i = \omega(e_i)$, we thus obtain the following clique-complex filtration:

$$\mathrm{C}_{G^{a_1}} \subseteq \mathrm{C}_{G^{a_2}} \subseteq \cdots \subseteq \mathrm{C}_{G^{a_m}}.$$

The persistent homology induced by the clique-complex filtration can be used to summarize the weighted graph $G = (V, E, \omega)$. Here one can consider the k-th homology groups for k upto $|V| - 1$.

10.5.2 Graphs viewed as metric spaces

A finite *metric graph* is a metric space $(|G|, \mathsf{d}_G)$ where the space is the underlying space of a finite graph G, equipped with a length metric d_G [11]. We have already seen metric graphs in the previous chapter where Reeb graphs are equipped with a function induced metric (Definition 7). We can also obtain a metric graph from a (positively) weighted combinatorial graph.

Given a graph $G = (V, E, \omega)$ where the weight of each edge is positive[3], we can view it as a metric graph $(|G|, \mathsf{d}_G)$ obtained by gluing a set of length segments (edges), where intuitively d_G is the shortest path metric on $|G|$ induced by edge lengths $\omega(e)$s.

Fact 3. *A positively weighted graph $G = (V, E, \omega)$ induces a metric graph $(|G|, \mathsf{d}_G)$.*

Indeed, let $|G|$ be the underlying space of G, viewing G as a simplicial 1-complex. For every edge $e \in E$, consider the arclength parameterization $e : [0, \omega(e)] \to |e|$, and define $\mathsf{d}_G(x, y) = |e^{-1}(y) - e^{-1}(x)|$ for every pair $x, y \in |e|$. The length of any path $\pi(u, v)$ between two points $u, v \in |G|$ is the sum of the lengths of the restrictions of π to edges in G. The distance $\mathsf{d}_G(u, v)$ between any two points $u, v \in |G|$ is the minimum length of any path connecting u to v in $|G|$ which is a metric. The metric space $(|G|, \mathsf{d}_G)$ is the metric graph of G.

Intrinsic Čech and Vietoris-Rips filtrations. Given a metric graph $(|G|, \mathsf{d}_G)$, let $B^o_{|G|}(x; r) := \{y \in |G| \mid \mathsf{d}_G(x, y) < r\}$ denote the radius-r *open* metric ball centered at $x \in |G|$. Recall that[4], the intrinsic Čech complex $\mathbb{C}^r(|G|)$ and intrinsic Vietoris-Rips complex $\mathbb{VR}^r(|G|)$ are defined as:

$$\mathbb{C}^r(|G|) := \{\{x_0, \ldots, x_p\} \mid \bigcap_{i \in [0,p]} B^o_{|G|}(x_i; r) \neq \emptyset\};$$

$$\mathbb{VR}^r(|G|) := \{\{x_0, \ldots, x_p\} \mid \mathsf{d}_G(x_i, x_j) < 2r \text{ for any } i \neq j \in [0, p]\}.$$

Remark 1. *We note that alternatively, $G = (V, E, \omega)$ can also be viewed as a* discrete metric space $(V, \hat{\mathsf{d}})$ *where $\hat{\mathsf{d}} : V \times V \to \mathbb{R}^+ \cup \{0\}$ is the restriction of d_G to graph nodes V of G. We can thus build* discrete intrinsic Čech or Vietoris-Rips complexes *spanned by only vertices in G. If G is a complete graph, then the discrete Vietoris-Rips complex at scale r is equivalent to the clique complex for G^r as introduced in Section 10.5.1. Most of our discussions below have analogous results for the discrete case.*

[3] If G is unweighted, then $\omega : E \to \mathbb{R}$ is the constant function $\omega(e) = 1$ for any $e \in E$.

[4] Note that here we use open metric balls instead of closed metric balls to define the Čech and Rips complexes, so that the theoretical result in Theorem 8 is cleaner to state.

We now consider the intrinsic Čech filtration $\mathcal{C} := \{\mathbb{C}^r\}_{r\in\mathbb{R}}$ and intrinsic Vietoris-Rips filtration $\mathcal{R} := \{\mathbb{VR}^r\}_{r\in\mathbb{R}}$, and their induced persistence modules $\mathsf{H}_p\mathcal{C} := \{\mathsf{H}_p(\mathbb{C}^r)\}_{r\in\mathbb{R}}$ and $\mathsf{H}_p\mathcal{R} := \{\mathsf{H}_p(\mathbb{VR}^r)\}_{r\in\mathbb{R}}$. We have (see [15]):

Fact 4. *Given a finite metric graph $(|G|, \mathsf{d}_G)$ induced by $G = (V, E, \omega)$, the persistence modules $\mathsf{H}_p\mathcal{C}$ and $\mathsf{H}_p\mathcal{R}$ are both q-tame.*

Hence both the intrinsic Čech and intrinsic Vietoris-Rips filtrations induce well-defined persistence diagrams, which can be used as summaries (signatures) for the input graph $G = (V, E, \omega)$.

In what follows, we present some results on the homotopy types of these simplicial complexes, as well as their induced persistent homology.

Topology of Čech and Vietoris-Rips complexes. Interestingly, the intrinsic Čech and Vietoris-Rips complexes induced by a metric graph may have non-trivial high-dimensional homology groups. The following results from [1] provide a precise characterization of the homotopy groups of these complexes for a metric graph whose underlying space is a circle. Specifically, let $\mathbb{S}^1$ denote the circle of unit circumference which is assumed for simplicity. The results below can be extended to a circle of any length by appropriate scaling. Let $\mathbb{S}^d$ denote the d-dimensional sphere.

Theorem 8. *Let $0 < r < \frac{1}{2}$. There are homotopy equivalences: for $\ell = 0, 1, \ldots,$*

$$\mathbb{C}^r(\mathbb{S}^1) \simeq \mathbb{S}^{2\ell+1} \quad \text{if } \frac{\ell}{2(\ell+1)} < r \leq \frac{\ell+1}{2(\ell+2)}; \quad \text{and}$$
$$\mathbb{VR}^{r/2}(\mathbb{S}^1) \simeq \mathbb{S}^{2\ell+1} \quad \text{if } \frac{\ell}{2\ell+1} < \frac{r}{2} \leq \frac{\ell+1}{2\ell+3}.$$

We remark that if one uses the closed ball to define these complexes, then the statements are similar and but with some additional technicalities; see [1].

Much less is known for more general metric graphs. Below we present two sets of results: Theorem 9 characterizes the intrinsic Vietoris-Rips complexes for a certain family of metric graphs [2]; while Theorem 10 characterizes only the 1-st persistent homology induced by the intrinsic Čech complexes, but for any finite metric graph [37]. Recall that $\tilde{\mathsf{H}}_p$ denotes the p-th reduced homology group.

Theorem 9. *Let G be a finite metric graph, with each edge of length one, that can be obtained from a vertex by iteratively attaching (i) an edge along a vertex or (ii) a k-cycle graph along a vertex or a single edge for $k > 2$ (see, e.g., Figure 10.10). Then we have that $\tilde{\mathsf{H}}_p(\mathbb{VR}(G; r)) \approx \oplus_{i=1}^{n} \tilde{\mathsf{H}}_p(\mathbb{VR}(C_{k_i}; r))$ where $\oplus$ stands for the direct sum, n is the number of times operation (ii) is performed, and C_{k_i} is a loop of k_i edges (and thus C_{k_i} is of length k_i) which was attached in the ith time that operation (ii) is performed.*

The above theorem can be relaxed to allow for different edge lengths though one needs to define the "gluing" more carefully in that case. See [2] for details. Graphs described in Theorem 9 are intuitively generated by iteratively gluing a simple loop along a "short" simple path in the existing graph. Note that the above theorem implies that the Vietoris-Rips complex for a metric tree has isomorphic reduced homology groups as a point.

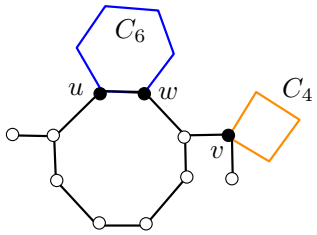


Figure 10.10: A 4-cycle C_4 is attached to the base graph along vertex v; whlie a 6-cycle C_6 is attached to the base graph along edge (u, w).

Persistent homology induced by Čech complexes. Instead of a fixed scale, Theorem 10 below provides a complete characterization for the 1-st persistent homology of intrinsic Čech complex filtration of a general finite metric graph. To present the result, we recall the concept of the shortest cycle basis (optimal generators) for $\mathsf{H}_1(G)$ while treating $G = (V, E, \omega)$ as a simplicial 1-complex. Specifically, in our setting, given any 1-cycle $\gamma = e_{i_1} + e_{i_2} + \cdot + e_{i_s}$, define *the length of* γ to be $length(\gamma) = \sum_{j=1}^{s} \omega(e_{i_j})$. A *cycle basis* of G refers to a set of g 1-cycles $\Gamma = \{\gamma_1, \ldots, \gamma_g\}$ that form a basis for the 1-dimensional cycle group $\mathsf{Z}_1(G)$. Notice that we can replace $\mathsf{H}_1(G)$ with the cycle group $\mathsf{Z}_1(G)$ because the two are isomorphic in case of graphs. Given a cycle basis Γ, its *length-sequence* is the sequence of lengths of elements in the basis in non-decreasing order. A cycle basis of G is a *shortest cycle basis* if its length-sequence is lexicographically minimal among all cycle basis of G.

Theorem 10. *Let $G = (V, E, \omega)$ be a finite graph with positive weight function $\omega : E \to \mathbb{R}$. Let $\{\gamma_1, \ldots, \gamma_g\}$ be a shortest cycle basis of G where $g = \text{rank}\,(\mathsf{Z}_1(G))$, and for each $i = 1, \ldots, g$, let $\ell_i = length(\gamma_i)$. Then, the 1-st persistence diagram $\text{Dgm}_1\mathcal{C}$ induced by the intrinsic Čech filtration $\mathcal{C} := \{\mathbb{C}^r(|G|)\}_{r\in\mathbb{R}}$ on the metric graph $(|G|, \mathsf{d}_G)$ consists of the following set of points on the y-axis:*

$$\text{Dgm}_1\mathcal{C} = \{(0, \frac{\ell_i}{4}) \mid 1 \le i \le g\}.$$

Unfortunately, no such characterization is available for high-dimensional cases. Some partial results on the higher-dimensional persistent homology induced by intrinsic Čech filtration are given in [26].

10.6 Graph comparison

The topological invariants described in the previous section can be used as signatures to compare graphs. For example, given two graphs $G_1 = (V_1, E_1, \omega_1)$ and $G_2 = (V_2, E_2, \omega_2)$ with positive weight functions, let $\mathcal{C}(G_1)$ and $\mathcal{C}(G_2)$ denote the intrinsic Čech filtrations for $(|G_1|, \mathsf{d}_{G_1})$ and $(|G_2|, \mathsf{d}_{G_2})$, respectively. We can then define $\mathsf{d}_{IC}(G_1, G_2) = \mathsf{d}_b(\text{Dgm}_1\mathcal{C}(G_1), \text{Dgm}_1\mathcal{C}(G_2))$ and d_{IC} gives rise to a pseudo-distance (a metric for which the first axiom may hold only with 'if' condition) for the family of finite graphs with positive weights. Furthermore, this pseudo-distance is stable w.r.t. the Gromov-Hausdorff distance.

Persistence-distortion distance. In what follows, we introduce another pseudo-distance for metric graphs, called the *persistence distortion* distance, which, instead of mapping the entire graph into a single persistence diagram, maps each point in the graph to such a summary.

First, given a finite metric graph $(|G|, \mathsf{d}_G)$, for any point $\mathrm{s} \in |G|$, consider the shortest path distance function $f_{\mathrm{s}} : |G| \to \mathbb{R}$ defined as: $x \mapsto \mathsf{d}_G(x, \mathrm{s})$. Let

$$\mathrm{P}_{\mathrm{s}} := \mathrm{Dgm}_0 f_{\mathrm{s}}, \text{ the 0-th persistence diagram induced by the function } f_{\mathrm{s}}. \tag{10.3}$$

Let $\mathbb{G}$ and $\mathbb{D}$ denote the space of finite metric graphs, and the space of finite persistence diagrams, respectively; and let $2^{\mathbb{D}}$ denote the space of all subsets of $\mathbb{D}$. We define:

$$\phi : \mathbb{G} \to 2^{\mathbb{D}} \quad \text{where for any } |G| \in \mathbb{G}, \;\; \phi(|G|) \;\mapsto\; \{\, \mathrm{P}_{\mathrm{s}} \mid \mathrm{s} \in |G| \,\}. \tag{10.4}$$

In other words, ϕ maps a metric graph $|G|$ to a set of (infinitely many) points $\phi(|G|)$ in the space of persistence diagrams $\mathbb{D}$. The image $\phi(|G|)$ is another graph in the space of persistence diagrams though this map is not injective.

Now let $(|G_1|, \mathsf{d}_{G_1})$ and $(|G_2|, \mathsf{d}_{G_2})$ denote the metric graphs induced by finite graphs $G_1 = (V_1, E_1, \omega_1)$ and $G_2 = (V_2, E_2, \omega_2)$ with positive edge weights.

Definition 12 (Persistence distortion distance)**.** The *persistence-distortion distance* between $(|G_1|, \mathsf{d}_{G_1})$ and $(|G_2|, \mathsf{d}_{G_2})$, denoted by $\mathsf{d}_{PD}(G_1, G_2)$, is the Hausdorff distance $\mathsf{d}_H(\phi(|G_1|), \phi(|G_2|)$ between the two image sets $\phi(|G_1|)$ and $\phi(|G_2|)$ in the space of persistence diagrams $(\mathbb{D}, \mathsf{d}_b)$ equipped with the bottleneck distance d_b. In other words, setting $\mathcal{A} := \phi(|G_1|)$ and $\mathcal{B} := \phi(|G_2|)$, we have

$$\mathsf{d}_{PD}(G_1, G_2) := \mathsf{d}_H(\phi(|G_1|), \phi(|G_2|)) = \max\{ \max_{P \in \mathcal{A}} \min_{Q \in \mathcal{B}} \mathsf{d}_b(P, Q);\; \max_{Q \in \mathcal{B}} \min_{P \in \mathcal{A}} \mathsf{d}_b(P, Q) \}.$$

The persistence distortion d_{PD} is a pseudo-metric. It can be computed in polynomial time for finite input graphs. It is stable w.r.t. the Gromov-Hausdorff distance between the two input metric graphs.

Theorem 11. $\mathsf{d}_{PD}(G_1, G_2) \leq 6\mathsf{d}_{GH}(|G_1|, |G_2|)$.

One can also define a *discrete persistence distortion* distance $\widehat{\mathsf{d}}_{PD} = \mathsf{d}_H(\widehat{\phi}(G_1), \widehat{\phi}(G_2))$, where $\widehat{\phi}(G) := \{\mathrm{P}_{\mathrm{s}} \mid \mathrm{s} \in V\}$ for a graph $G = (V, E, \omega)$. Both the persistence distortion distance and its discrete variant can be computed in time polynomial in the size (number of vertices and edges) of the combinatorial graphs G_1 and G_2 generating the metric graphs $|G_1|$ and $|G_2|$ respectively.

10.7 Topological invariants for directed graphs

In this section, we assume that we are given a directed graph $G = (V, \vec{E}, \omega)$ where $\vec{E} \subseteq V \times V$ is the directed edge set, and $\omega : \vec{E} \to \mathbb{R}$ is the edge weight function (if the input graph is unweighted, we assume that all weights equal to 1). Each directed edge (u, v) is an ordered pair, and thus edge $(u, v) \neq (v, u)$. For simplicity, we assume that there is no self-loop (v, v) in $\vec{E}$, and also there is at most one directed edge between an ordered pair of nodes. Given a node $v \in V$, its in-degree is $indeg(v) = |\{u \mid (u, v) \in \vec{E}\}|$, and its out-degree is $outdeg(v) = |\{u \mid (v, u) \in \vec{E}\}|$.

10.7.1 Simplicial complexes for directed graphs

Treating a directed graph as an asymmetric network (as it may be that $\omega(u, v) \neq \omega(v, u)$), one can extend ideas in the previous section to this asymmetric setting. We give two examples below: both cases lead to simplicial complexes from an input directed graph (weighted or unweighted), and one can then compute (persistent) homological information of (filtrations of) this simplicial complex as summaries of input directed graphs.

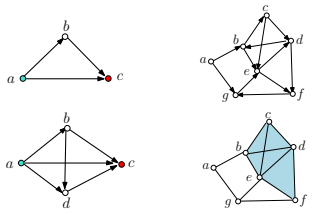


Figure 10.11: (a) a 3-clique and a 4-clique with the blue point being its source and the red point being its sink. (b) A directed graph (left) and its directed clique complex (right). The set of triangles in this complex are: $\{bce, ced, edf\}$. There is no higher dimensional simplices. Note that if the edge (b, d) is also in the directed graph in (b), then the tetrahedron $bcde$ will be in its corresponding directed clique complex.

Directed clique complex. A node in a directed graph is a *source node* if it has in-degree 0; and it is a *sink node* if it has out-degree 0. A graph $(\{v_1, \ldots, v_k\}, E')$ is a *directed k-clique* if (i) there are $\binom{k}{2}$ edges in E', and (ii) it has a unique source node, as well as a unique sink node. See 10.11 (a) for examples. A set of vertices $\{v_{i_1}, \ldots, v_{i_k}\}$ spans a directed clique in $G = (V, \vec{E})$ if there is a subset of edges of $E' \subseteq \vec{E}$ such that $(\{v_{i_1}, \ldots, v_{i_k}\}, E')$ is a directed k-clique. It is easy to see that given a directed clique, any subset of its vertices also form a directed clique (Exercise **??**).

Definition 13 (Directed clique complex)**.** Given a directed graph $G = (V, \vec{E})$, the *directed clique complex* induced by G is a simplicial complex K defined as

$$\widehat{\mathrm{C}}(G) := \{\sigma = \{v_{i_1}, \ldots, i_{i_k}\} \mid \{v_{i_1}, \ldots, i_{i_k}\} \text{ spans a directed } \ k \text{ clique in } G\}.$$

See Figure 10.11 (b) for a simple example. Now given a weighted directed graph $G = (V, \vec{E}, \omega)$, for any scale a, let G^a be the subgraph of G spanned by all directed edges whose weight is at most a. Assuming all edges $e_1, \ldots, e_m$, $m = |\vec{E}|$, are sorted by their weights in a non-decreasing order, set $a_i = \omega(e_i)$. Similar to the clique complex filtration for undirected graphs introduced in Section 10.5.1, this gives rise to the following filtration of simplicial complexes induced by the directed clique complexes:

$$\widehat{\mathrm{C}}(G^{a_1}) \subseteq \widehat{\mathrm{C}}(G^{a_2}) \subseteq \cdots \subseteq \widehat{\mathrm{C}}(G^{a_m}).$$

One can then use the persistence diagram induced by the above filtration as a topological invariant for the input directed graph G.

Definition 14 (Dowker complex)**.** Given a weighted directed graph $G = (V, \vec{E}, \omega)$ and a threshold δ, the *Dowker δ-sink complex* is the following simplicial complex:

$$D_\delta^{si}(G) := \{\sigma = \{v_{i_0}, \ldots, v_{i_d}\} \mid \text{ there exists } v \in V \text{ so that } \omega(v_{i_j}, v) \leq \delta \text{ for any } j \in [0, d]\}. \quad (10.5)$$

In the above definition, v is called a δ-sink for the simplex σ. In the right example of Figure 10.11 (a), assume all edges have weight 1. If we now remove edge (b, d), then abd is not a 3-clique any more in $G^{\delta=1}$. However, abd still forms a 2-simplex in the Dowker sink complex D_1^{si} with sink c.

In general, as δ increases, we obtain a sequence of Dowker complexes connected by inclusions, called the *Dowker sink filtration* $\mathcal{D}^{si}(G) = \{D_\delta^{si} \hookrightarrow D_{\delta'}^{si}\}_{\delta \leq \delta'}$.

Alternatively, one can define the *Dowker δ-source complex* in a symmetric manner:

$$D_\delta^{so}(G) := \{\sigma = \{v_{i_0}, \ldots, v_{i_d}\} \mid \text{ there exists } v \in V \text{ so that } \omega(v, v_{i_j}) \leq \delta \text{ for any } j \in [0, d]\} \quad (10.6)$$

resulting in a *Dowker source filtration* $\mathcal{D}^{so}(G) = \{D_\delta^{so} \hookrightarrow D_{\delta'}^{so}\}_{\delta \leq \delta'}$. It turns out that by the duality theorem of Dowker [32], the two Dowker complexes have isomorphic homology groups. It can be further shown that the choice of Dowker complexes does not matter when persistent homology is considered [16].

Theorem 12 (Dowker Duality)**.** *Given a directed graph $G = (V, \vec{E}, \omega)$, for any threshold $\delta \in \mathbb{R}$ and dimension $p \geq 0$, we have $\mathsf{H}_p(D_\delta^{si}) \cong \mathsf{H}_p(D_\delta^{so})$. Furthermore, the persistence modules induced by the Dowker sink and the Dowker source filtrations are isomorphic as well, that is,*

$$\mathrm{Dgm}_p\, \mathcal{D}^{si} = \mathrm{Dgm}_p\, \mathcal{D}^{so}, \text{ for any } p \geq 0.$$

10.7.2 Path (persistent) homology for directed graphs

In this subsection, we introduce the so-called *path homology*, which is different from the simplicial homology that we defined for clique complex and Dowker complex. Instead of constructing a simplicial complex from an input directed graph and considering its simplicial homology group, here, we use the directed graph to define a chain complex directly. The resulting path homology group has interesting mathematical structures behind, e.g., there is a concept of homotopy in directed graphs under which the path homology is preserved, and it accommodates the Künneth formula.

Note that in this chapter, we have assumed that a given directed graph $G = (V, \vec{E})$ does not contain self-loops (where a *self-loop* is an edge (u, u) from u to itself). For notational simplicity, below we sometimes use index i to refer to vertex $v_i \in V = \{v_1, \ldots, v_n\}$.

Let $\mathbf{k}$ be a field with 0 and 1 being the additive and multiplicative identities respectively. We use $-a$ to denote the additive inverse of a in $\mathbf{k}$. An *elementary p-path* on V is simply a sequence $v_{i_0}, v_{i_1}, \cdots, v_{i_p}$ of $p + 1$ of vertices of V, which we denote by $e_{v_{i_0}, v_{i_1}, \ldots, v_{i_d}}$, or just $e_{i_0, i_1, \cdots, i_p}$ for simplicity. Let $\Lambda_p = \Lambda_p(G, \mathbf{k})$ denote the $\mathbf{k}$-linear space of all linear combinations of elementary p-paths with coefficients from $\mathbf{k}$. The set $\{e_{i_0, \cdots, i_p} \mid i_0, \cdots, i_p \in V\}$ forms a basis for Λ_p. Each element c of Λ_d is called a *p-path* or *p-chain*, and it can be written as

$$c = \sum\nolimits_{i_0, \cdots, i_p \in V} a_{i_0 \cdots i_p} e_{i_0 \cdots i_p}, \text{ where } a_{i_0 \cdots i_p} \in \mathbf{k}.$$

Similar to the case of simplicial complexes, we can define *boundary map* $\partial_p : \Lambda_p \to \Lambda_{p-1}$ as:

$$\partial_p\, e_{i_0\cdots i_p} = \sum_{i_0,\cdots,i_p \in V} (-1)^k e_{i_0\cdots \hat{i}_j \cdots i_p}, \quad \text{for any elementary } p\text{ -path } e_{i_0\cdots i_p},$$

where $\hat{i}_k$ means the removal of index i_k. The boundary of a p-path $c = \sum a_{i_0\cdots i_p} \cdot e_{i_0\cdots i_p}$, is thus $\partial_p c = \sum a_{i_0\cdots i_p} \cdot \partial_p e_{i_0\cdots i_p}$. For convenience, we set $\Lambda_{-1} = 0$ and note that Λ_0 is the set of $\mathbf{k}$-linear combinations of vertices in V. It is easy to show that $\partial_{p-1} \cdot \partial_p = 0$, for any $p > 0$. In what follows, we often omit the dimension p from ∂_p when it is clear from the context.

Next, we restrict the consideration to real paths in directed graphs formed by consecutive directed edges. Specifically, given a directed graph $G = (V, \vec{E})$, call an elementary p-path $e_{i_0,\cdots,i_p}$ *allowed* if there is an edge from i_k to i_{k+1} for all $k \in [0, p-1]$. Define $\mathcal{A}_p$ as the space spanned by all allowed elementary p-paths, that is, $\mathcal{A}_p := \text{span}\{e_{i_0\cdots i_p} : e_{i_0\cdots i_p} \text{ is allowed}\}$. An elementary p-path $i_0 \cdots i_p$ is called *regular* if $i_k \neq i_{k+1}$ for all k, and is *irregular* otherwise. Clearly, every allowed path is regular since there is no self-loop. However, applying the boundary map ∂ to Λ_p may create irregular paths. For example, $\partial e_{uvu} = e_{vu} - e_{uu} + e_{uv}$ is irregular because of the term e_{uu}. To deal with this case, the term containing consecutive repeated vertices is taken as 0. Thus, for the previous example, we have $\partial e_{uvu} = e_{vu} - 0 + e_{uv} = e_{vu} + e_{uv}$. The boundary map ∂ on $\mathcal{A}_p$ is now taken to be the boundary map for Λ_p restricted on $\mathcal{A}_p$ with this modification, where all terms with consecutive repeated vertiecs created by the boundary map ∂ are replaced with 0's. For simplicity, we still use the same symbol ∂ to represent this modified boundary map on the space of allowed paths.

After restricting the boundary operator to the space of allowed paths $\mathcal{A}_p$s, the inclusion that $\partial\mathcal{A}_p \subset \mathcal{A}_{p-1}$ may not hold; that is, the boundary of an allowed p-path is not necessarily an allowed $(p-1)$-path. To this end, we adopt a stronger notion of allowed paths: an allowed path c is *∂-invariant* if ∂c is also allowed. Let $\Omega_p := \{c \in \mathcal{A}_p \mid \partial c \in \mathcal{A}_{p-1}\}$ be the space generated by all ∂-invariant p-paths. Note that $\partial\Omega_p \subset \Omega_{p-1}$ (as $\partial^2 = 0$). This gives rise to the following *chain complex of ∂-invariant allowed paths*:

$$\cdots \Omega_p \xrightarrow{\partial} \Omega_{p-1} \xrightarrow{\partial} \cdots \Omega_1 \xrightarrow{\partial} \Omega_0 \xrightarrow{\partial} 0.$$

We can now define the homology groups of this chain complex.

Definition 15 (Path homology)**.** The p-th cycle group is defined as $\mathsf{Z}_p = \ker \partial|_{\Omega_p}$, and elements in Z_p are called p-cycles. The p-th boundary group is defined as $\mathsf{B}_p = \text{Im}\, \partial|_{\Omega_{p+1}}$, with elements of B_p called p-boundary cycles (or simply p-boundaries). The *p-th path homology group* is defined as $\mathsf{H}_p(G, \mathbf{k}) = \mathsf{Z}_p/\mathsf{B}_p$.

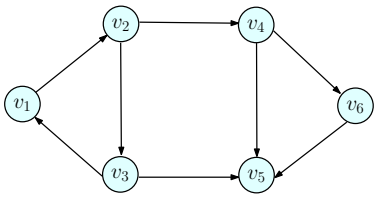


Figure 10.12: A directed graph G.

Examples. Consider the directed graph in Figure 10.12, and assume that the coefficient field $\mathbf{k} \neq \mathbb{Z}_2$: Examples of elementary 1-path include: e_{12}, e_{24}, e_{13}, e_{14}, and so on. However, e_{13} and e_{14} are not an allowed 1-path. More examples of allowed 1-path include: $e_{12} + e_{46}$, $e_{12} + e_{31}$, $e_{46} + e_{65} + e_{45}$ and $e_{46} + e_{65} - e_{45}$. Note that any allowed 1-path is also ∂-invariant; that is, $\Omega_1 = \mathcal{A}_1$, as all 0-paths are allowed. Observe that $\partial(e_{46} + e_{65} + e_{45}) = e_6 - e_4 + e_5 - e_6 + e_5 - e_4 = 2e_5 - 2e_4$, which is not 0 (unless the coefficient field $\mathbf{k} = \mathbb{Z}_2$). However, $\partial(e_{46} + e_{65} - e_{45}) = 0$, meaning that $e_{46} + e_{65} - e_{45} \in \mathsf{Z}_1$. Other 1-cycle examples include

$$e_{12} + e_{23} + e_{31},\ e_{24} + e_{45} - e_{23} - e_{35},\ \text{ and } e_{12} + e_{24} + e_{45} - e_{53} + e_{31} \in \mathsf{Z}_1$$

Examples of elementary 2-paths include: $e_{123}, e_{245}, e_{256}$ and e_{465}. However, e_{256} is not allowed. Consider the allowed 2-path e_{245}, its boundary $\partial e_{245} = e_{45} - e_{25} + e_{24}$ is not allowed as e_{25} is not allowed. Hence the allowed 2-path e_{245} is not ∂-invariant; similarly, we can see that neither e_{235} nor e_{123} is in Ω_2. It is easy to check that $e_{465} \in \Omega_2$ as $\partial e_{465} = e_{65} - e_{45} + e_{46}$. Also note that while neither e_{235} nor e_{245} is in Ω_2, the allowed 2-path $e_{245} - e_{235}$ is ∂-invariant as

$$\partial(e_{245} - e_{235}) = e_{45} - e_{25} + e_{24} - e_{35} + e_{25} - e_{23} = e_{45} + e_{24} - e_{35} - e_{23} \in \mathcal{A}_1.$$

This example suggests that elementary ∂-invariant p-paths *do not* necessarily form a basis for Ω_p – this is rather different from the case of simplicial complex, where the set of p-simplices form a basis for the p-th chain group.

The above discussion also suggests that $e_{46} + e_{65} - e_{45}, e_{24} + e_{45} - e_{23} - e_{35} \in \mathsf{B}_1$.

For this example,

$\{e_{12} + e_{23} + e_{31}, e_{46} + e_{65} - e_{45}, e_{24} + e_{45} - e_{23} - e_{35}\}$ is a basis for the 1-cycle group Z_1;

$\{e_{46} + e_{65} - e_{45}, e_{24} + e_{45} - e_{23} - e_{35}\}$ is a basis for the 1-boundary group B_1; while

$\{e_{245} - e_{235}, e_{465}\}$ is a basis for the space of ∂-invariant 2-paths Ω_2.

Persistent path homology for directed graphs. Given a weighted directed graph $G = (V, \vec{E}, \omega)$, let G^a denote the subgraph of G containing all directed edges with weight at most a. This gives rise to a filtration of graphs $\mathcal{G} : \{G^a \hookrightarrow G^b\}_{a \leq b}$. Let $\mathsf{H}_p(G^a)$ denote the p-th path homology induced by graph G^a. It can be shown [17] that the inclusion $G^a \hookrightarrow G^b$ induces a well-defined homormorphism $\xi_p^{a,b} : \mathsf{H}_p(G^a) \to \mathsf{H}_p(G^b)$, and the sequence $\mathcal{G} : \{G^a \hookrightarrow G^b\}_{a \leq b}$ leads to a persistence module $\mathsf{H}_p\mathcal{G} : \{\mathsf{H}_p(G^a) \to \mathsf{H}_p(G^b)\}_{a \leq b}$.

10.7.3 Computation of (persistent) path homology

The example in the previous section (recall Figure 10.12) illustrates the challenge of computing path homology induced by a directed graph G when compared to the case of simplicial homology. In particular, the set of elementary allowed d-paths may no longer form a basis for the space of the ∂-invariant d-paths Ω_d: Indeed, recall that $\{e_{465}, e_{245} - e_{235}\}$ form a basis for Ω_2, yet, neither e_{245} nor e_{235} belongs to Ω_2.

We now present an algorithm to compute the persistent path homology of a given weighted directed graph $G = (V, \vec{E}, \omega)$. Note that as a byproduct, this algorithm can also compute the path homology of a directed graph.

Given a p-path τ, its *allowed-time* is set to be the smallest value (weight) a when it belongs to $\mathcal{A}_p(G^a)$; and we denote it by $\mathtt{at}(\tau) = a$. Let $A_p = \{\tau_1, \ldots, \tau_t\}$ denote the set of elementary allowed p-paths, sorted by their allowed-times in a non-decreasing order. Similarly, set $A_{p-1} = \{\sigma_1, \ldots, \sigma_s\}$ to be the sequence of elementary allowed $(p-1)$-paths sorted by their allowed-times in a non-decreasing order. Let $a_1 < a_2 < \cdots < a_{\hat{t}}$ be the sequence of *distinct* allowed-times of elementary p-paths in A_p in increasing orders. Obviously, $\hat{t} \leq t = |A_p|$. Similarly, set $b_1 < b_2 < \cdots < b_{\hat{s}}$ be the sequence of *distinct* allowed-times for $(p-1)$-paths in A_{p-1} sorted in increasing order.

Note that A_p (resp. A_{p-1}) forms a basis for $\mathcal{A}_p(G)$ (resp. $\mathcal{A}_{p-1}(G)$). In fact, for any i, set $A_p^{a_i} := \{\tau_j \mid \mathtt{at}(\tau_j) \leq a_i\}$. It is easy to see that $A_p^{a_i}$ equals $\{\tau_1, \ldots, \tau_{\rho_i}\}$, where

$$\rho_i \in [1, t] \text{ is the largest index of any elementary } p\text{-path whose allowed-time is at most } a_i; \tag{10.7}$$

and $A_p^{a_i}$ forms a basis for $\mathcal{A}_p(G^{a_i})$. Note that the cardinality of $A_p^{a_i} \setminus A_p^{a_{i-1}}$ could be larger than 1 and that is why ρ_i is not necessarily equal to i. A symmetric statement holds for $A_{p-1}^{b_j}$ and $\mathcal{A}_{p-1}(G^{b_j})$.

From now on, we fix a dimension p. On the high level, the algorithm for computing the p-th persistent path homology has the following three steps, which looks similar to the algorithm that computes standard persistent homology for simplicial complexes. However, there are key differences in the implementation of these steps.

Step 1. Set up a "boundary matrix" $M = M_p$.

Step 2. Perform left-to-right matrix reduction to transform M to a reduced form $\widehat{M}$.

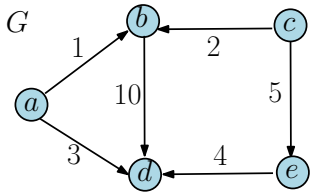
Step 3. Construct the persistence diagram from the reduced matrix $\widehat{M}$.

The details of these steps are given as follows.

Description of Step 1. The columns of M correspond to A_p, ordered by their allowed-times. We would like $\mathtt{col}_M[i] = \partial\tau_i$. However, the boundary of an allowed path may not be allowed. Hence the rows of the matrix need to correspond to not only the elementary allowed $(p-1)$-paths in A_{p-1} (ordered by their allowed-times), but also any elementary (non-allowed) $(p-1)$-path that appears in the boundary of any $\tau_j \in A_p$: we assign the allowed-times for such paths to be $+\infty$. The rows of M are ordered in non-decreasing allowed-times from top to bottom. Let $\widehat{A}_{p-1} = \{\sigma_1, \ldots, \sigma_s, \sigma_{s+1}, \ldots, \sigma_\ell\}$ be the final set of elementary $(p-1)$-paths corresponding to rows of M. Note that the first s elements are from A_{p-1}, while those in $\{\sigma_{s+1}, \ldots, \sigma_\ell\}$ are not-allowed, and have allowed-time $+\infty$. See an example in Figure 10.13 (a) and (b).

The matrix M represents the boundary operator ∂_p restricted to $\mathcal{A}_p$. In other words, the i-th column of M, denoted by $\mathtt{col}_M[i]$, contains the boundary of τ_i, represented using the basis elements in $\widehat{A}_{p-1}$; that is, $\partial_p\tau_i = \sum_{j=1}^{\ell} \mathtt{col}_M[i][j]\sigma_j$. From a vector representation point of view, we will also simply say that $\partial_p\tau_i = \mathtt{col}_M[i]$. The allowed time for the $(p-1)$-path represented by a column vector C is simply the allowed time $\mathtt{at}(\sigma_j)$ associated to the lowest-id $j = lowId(C)$ in this vector. It is important to note that the rows of M are ordered in increasing indices from top down. Hence *lowId* of a column means the largest index in $\widehat{A}_{p-1}$ for which this column contains a non-zero entry. We further associate a p-path γ_i with the ith column of M for each $i \in [1, t]$,

with the property $\partial_p \gamma_i = \texttt{col}_M[i]$. At the beginning of the algorithm, γ_i is initialized to be τ_i and later updated through the reduction process in (Step 2) below.



(a) graph G

$$\begin{array}{c} \\ e_{ab} \\ e_{cb} \\ e_{ad} \\ e_{ed} \\ e_{ce} \\ e_{bd} \\ e_{cd} \end{array} \begin{array}{c} \begin{array}{ccc} e_{ced} & e_{abd} & e_{cbd} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ -1 & 0 & -1 \end{bmatrix} \end{array}$$

(b) original boundary matrix M

$$\begin{array}{c} \\ e_{ab} \\ e_{cb} \\ e_{ad} \\ e_{ed} \\ e_{ce} \\ e_{bd} \\ e_{cd} \end{array} \begin{array}{c} \begin{array}{ccc} e_{ced} & e_{abd} & e_{cbd} \end{array} \\ \begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \end{array}$$

(c) reduced matrix $\widehat{M}$

Figure 10.13: The input is the weighted directed graph in (a). Its 1-dimensional boundary matrix M as constructed in (Step 1) is shown in (b). Note that $\texttt{at}(e_{cd}) = +\infty$ (so $e_{cd} \notin \mathcal{A}_1(G)$). For each edge (i.e, elementary allowed 1-path) in G, its allowed-time is simply its weight. There are only three elementary allowed 2-paths, and their allowed-times are: $\texttt{at}(e_{ced}) = 5$, $\texttt{at}(e_{abd}) = 10$ and $\texttt{at}(e_{cbd}) = 10$. (c) shows the reduced matrix. From this matrix, we can deduce that the 1-th persistence diagram (for path homologyg) includes two points: $(10, 10)$ and $(5, 10)$ (generated by the second and third columns). Note that for the first column (corresponding to e_{ced}), as $\texttt{at}(\texttt{col}_{\widehat{M}}[1]) = \infty$; hence the corresponding γ_1 is not ∂-invariant.

Description of Step 2. We now perform the standard left-to-right matrix reduction to M, where the only allowed operation is to add a column to some column on its right. We convert M to its *reduced form* $\widehat{M}$ (recall Definition from earlier topic); and through this process, we also update γ_i accordingly so that at any moment, $\partial_p \gamma_i = \texttt{col}_{M'}[i]$ where M' is the updated boundary matrix at that point. In particular, if we add column j to column $i > j$, then we will update $\gamma_i = \gamma_i + \gamma_j$. We note that other than the additional maintenance of γs, this reduction step of M is the same as the reduction in MatPersistence presented earlier. The following claim follows easily from that there are only left-to-right column additions, and that the allowed-times of γ_is are initially sorted in non-decreasing order.

Claim 1. *For any $i \in [1, t]$, the allowed-time of γ_i remains the same through any sequence of left-to-right column additions.*

Let Ω_p^i denote the space of ∂-invariant p-paths w.r.t G^{a_i}; that is, $\Omega_p^i = \Omega_p(G^{a_i})$. Given a p-path τ, let $\texttt{ent}(\tau)$ be its *entry-time*, which is the smallest value a such that $\tau \in \Omega_p(G^a)$. It is easy to see that for any p-path τ, we have that

$$\texttt{ent}(\tau) = max\{\texttt{at}(\tau), \texttt{at}(\partial_p \tau)\}. \tag{10.8}$$

Recall that each column vector $\texttt{col}_{\widehat{M}}[i]$ is in fact the vector representation of a $(p-1)$-path (with respect to basis elements in $\widehat{A} = \{\sigma_1, \ldots, \sigma_\ell\}$). Also, the allowed time for a column $\texttt{col}_{\widehat{M}}[i]$ is given by $\texttt{at}(\texttt{col}_{\widehat{M}}[i]) = \texttt{at}(\sigma_h)$ where $h = lowId(\texttt{col}_{\widehat{M}}[i])$.

Claim 2. *Given a reduced matrix $\widehat{M}$, let $C = \sum_{i=1}^{t} c_i \mathtt{col}_{\widehat{M}}[i]$ be a $(p-1)$-path. Let $\mathtt{col}_{\widehat{M}}[j]$ be the column with lowest (i.e, largest) lowId among all columns $\mathtt{col}_{\widehat{M}}[i]$s such that $c_i \neq 0$, and set $h = lowId(\mathtt{col}_{\widehat{M}}[j])$. It then follows that $\mathtt{at}(C) = \mathtt{at}(\sigma_h)$.*

Now for the reduced matrix $\widehat{M}$, given any $i \in [1, \hat{t}]$, we set ρ_i to be the largest index $j \in [1, t]$ such that $\mathtt{at}(\gamma_j) \leq a_i$. By Claim 1, each column j has a fixed allowed time associated to the p-path γ_j associated to it, which stays invariant through the reduction process. So this quantity ρ_i is well defined, consistent with what we defined earlier in Eqn (10.7), and remains invariant through the reduction process. Set:

$$\begin{aligned}
\Gamma^i &:= \{\gamma_1, \ldots, \gamma_{\rho_i}\},\\
I_i &:= \{j \leq \rho_i \mid \mathtt{at}(\mathtt{col}_{\widehat{M}}[j]) \leq a_i\}, \text{ and}\\
\Sigma^i &:= \{\gamma_j \mid \mathtt{ent}(\gamma_j) \leq a_i\} = \{\gamma_j \mid j \in I_i\}.
\end{aligned}$$

Theorem 13. *For any $k \in [1, \hat{t}]$, Γ^k forms a basis for $\mathcal{A}_p^k := \mathcal{A}(G^{a_k})$; while Σ^k forms a basis for $\Omega_p^k = \Omega_p(G^{a_k})$.*

Proof. That Γ^k forms a basis for $\mathcal{A}_p^k$ follows easily from the facts that originally, $\{\tau_1, \ldots, \tau_{\rho_k}\}$ form a basis for $\mathcal{A}_p^k$, and the left-to-right column additions maintain this. In what follows, we prove that Σ^k forms a basis for Ω_p^k. First, note that all elements in Σ^k represent paths in Ω_p^k and they are linearly independent by construction (as their lowest ids are distinct). So we only need to show that any element in Ω_p^k can be represented by a linear combination of vectors in Σ^k.

Let ξ_k denote the largest index $j \in [1, s]$ such that $\mathtt{at}(\sigma_j) \leq a_k$. In other words, an equivalent formulation for I_k is that $I_k = \{j \leq \rho_k \mid lowId(\mathtt{col}_{\widehat{M}}[j]) \leq \xi_k\}$.

Now consider any $\gamma \in \Omega_p^k \subseteq \mathcal{A}_p^k$. As Γ^k forms a basis for $\mathcal{A}_p^k$, we have that

$$\gamma = \sum_{i=1}^{\rho_k} c_i \gamma_i \quad \text{and} \quad \partial\gamma = \sum_{i=1}^{\rho_k} c_i \partial\gamma_i = \sum_{i=1}^{\rho_k} c_i \mathtt{col}_{\widehat{M}}[i].$$

As $\gamma \in \Omega_p^k$ and $\mathtt{ent}(\gamma) = \max\{\mathtt{at}(\gamma), \mathtt{at}(\partial\gamma)\}$ (see Eqn (10.8)), we have $\mathtt{at}(\gamma) \leq a_k$ and $\mathtt{at}(\partial\gamma) \leq a_k$. By Claim 2, it follows that for any $j \in [1, \rho_k]$ with $c_j \neq 0$, its lowId satisfies $lowId(\mathtt{col}_{\widehat{M}}[j]) \leq \xi_k$. Hence each such index j with $c_j \neq 0$ must belong to I_k, and as a result, γ can be written as a linear combination of p-paths in Σ^k. Combined with that all vectors in Σ^k are in Ω_p^k and are linearly independent, it follows that Σ^k forms a basis for Ω_p^k. □

Corollary 14. *Set $J_k := \{j \in I_k \mid \mathtt{col}_{\widehat{M}}[j]$ is all zeros$\}$. Further we set $Z^k := \{\gamma_j \mid j \in J_k\}$; and $B^k := \{\mathtt{col}_{\widehat{M}}[j] \mid j \in I_k \setminus J_k\}$. Then (i) Z^k forms a basis for the p-dimensional cycle group $\mathsf{Z}_p(G^{a_k})$; and (ii) B^k forms a basis for the $(p-1)$-dimensional boundary group $\mathsf{B}_{p-1}(G^{a_k})$.*

Proof. Let $\hat{\partial}_p$ denote the restriction of ∂_p over Ω_p. Recall that $\mathsf{Z}_p = \mathrm{Ker}\hat{\partial}_{\mathrm{p}}$, while $\mathsf{B}_{p-1} = \mathrm{Im}\hat{\partial}_{\mathrm{p}}$. Easy to see that by construction of Z^k, we have $Z^k \subseteq \mathrm{Span}(Z^k) \subseteq \mathsf{Z}_p(G^{a_k})$. Since all Γ_is are linearly independent, we thus have that vectors in Z^k are linearly independent. It then follows that $|Z^k| \leq \mathrm{rank}\,(\mathsf{Z}_p(G^{a_k}))$ where $|Z^k|$ stands for the cardinality of Z^k.

Similarly, as the matrix $\widehat{M}$ is reduced, all non-zero columns of $\widehat{M}$ are linearly independent, and thus vectors in B^k are linearly indepdent. Furthermore, by Theorem 13, each vector in B^k is in

$\mathsf{B}_{p-1}(G^{a_k})$ (as it is the boundary of a p-path from Ω_p^k). Hence we have that $\text{Span}(B^k) \subseteq \mathsf{B}_{p-1}(G^{a_k})$, and $|B^k| \leq \text{rank}\,(\mathsf{B}_{p-1}(G^{a_k}))$.

On the other hand, let $\hat{\partial}_p|_{\Omega_p^k}$ denote the restriction of $\hat{\partial}_p$ to only $\Omega_p^k \subseteq \Omega_p$. Note that by Rank Nullity Theorem,

$$|\Sigma_p^k| = \text{rank}\,(\Omega_p^k) = \text{rank}\,(\ker(\hat{\partial}_p|_{\Omega_p^k})) + \text{rank}\,((\hat{\partial}_p|_{\Omega_p^k})) = \text{rank}\,(\mathsf{Z}_p(G^{a_k})) + \text{rank}\,(\mathsf{B}_{p-1}(G^{a_k})).$$

As $\text{rank}\,(\Sigma_p^k) = |Z^k| + |B^k|$, and combining the above equation with the inequalities obtained in the previous paragraphs, it follows that it must be that $|Z^k| = \text{rank}\,(\mathsf{Z}_p(G^{a_k}))$ and $|B^k| = \text{rank}\,(\mathsf{B}_{p-1}(G^{a_k}))$. The claim then follows. □

Description of Step 3: constructing persistence diagram from the reduced matrix $\widehat{M}$. Given a weighted directed graph $G = (V, \vec{E}, \omega)$, for each dimension $p \geq 0$, construct the boundary matrix M_{p+1} as described above. Perform the left-to-right column reduction to M_{p+1} to obtain a reduced form $\widehat{M} = \widehat{M}_{p+1}$. The p-th persistence diagram $\text{Dgm}_p\mathcal{G}$ where $\mathcal{G} : \{G^a \hookrightarrow G^b\}_{a \leq b}$ can be computed as follows.

Let $\mu_p^{a,b}$ denote the persistence pairing function: that is, the persistence point (a, b) is in $\text{Dgm}_p\mathcal{G}$ with multiplicity $\mu_p^{a,b}$ if and only if $\mu_p^{a,b} > 0$. At the beginning, $\mu_p^{a,b}$ is initialized to be 0 for all $a, b \in \mathbb{R}$. We then inspect every non-zero column $\texttt{col}_{\widehat{M}}[i]$, and take the following actions.

- If $\texttt{at}(\texttt{col}_{\widehat{M}}[i]) \neq \infty$, then we increase the pairing function $\mu^{\texttt{at}(\texttt{col}_{\widehat{M}}[i]),\texttt{ent}(\gamma_i)}$ by 1. Observe that, $\texttt{at}(\texttt{col}_{\widehat{M}}[i]) \leq \texttt{ent}(\gamma_i)$ because

$$\texttt{ent}(\gamma_i) = \max\{\texttt{at}(\gamma_i), \texttt{at}(\partial\gamma_i)\} = \max\{\texttt{at}(\tau_i), \texttt{at}(\texttt{col}_{\widehat{M}}[i])\},$$

  where γ_i is the allowed elementary $(p+1)$-path corresponding to this column.

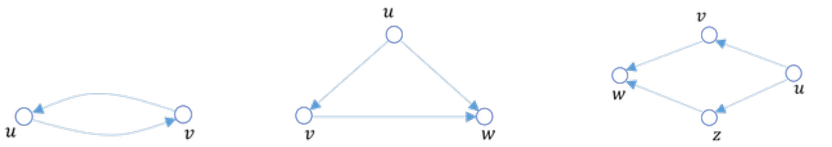
- Otherwise, the path γ_i corresponds to this column is not ∂-invariant (i.e, not in Ω_p), and we do nothing.

- Finally, consider the reduced matrix $\widehat{M}_p$ for the p-th boundary matrix M_p as constructed above. Recall the construction of J_k as in Corollary 14. For any $j \in J_k$ such that j is not appearing as the lowest-id of any column in $\widehat{M}_{p+1}$, we increase the pairing function $\mu^{\texttt{at}(\tau),\infty}$ by 1, where τ is the elementary p-path corresponding to this column.

See Figure 10.13 for an example. Let N_p denote the number of allowed elementary p-paths in G: obviously, $N_p = O(n^{p+1})$. However, as we see earlier, the number of rows of M_{p+1} is not necessarily bounded by N_p; and we can only bound it by the number of elementary p-paths in G, which we denote by $\widehat{N}_p$. If we use the standard Gaussian elimination for the column reduction as in the Algorithm MatPersistence, then the time complexity to compute the reduced matrix $\widehat{M}_{p+1}$ is $O(\widehat{N}_p^2 N_{p+1})$. One can further improve it using the fast matrix multiplication time.

We note that due to Theorem 13 and Corollary 14, the above algorithm is rather similar to the matrix reduction for the standard persistent homology induced by simplicial complexes. However, the example in Figure 10.13 shows the difference.

Improved computation for 1-st persistent path homology. The time complexity can be improved for computing the 0-th and 1-st persistent path homology. In particular, the 0-th persistence path homology coincides with the 0-th persistence homology induced by the persistence of clique complexes, and thus can be computed in $O(m\alpha(n) + n \log n)$ time using the union-find data structure, where $n = |V|$ and $m = |\vec{E}|$.



bigon boundary triangle boundary quadrangle

Figure 10.14: Boundary bigon, triangle and quadrangle. Such boundary cycles generate all 1-dimensional boundary cycles.

For the 1-dimensional case, it turns out that the boundary group has further structures. In particular, the 1-dimensional boundary group is generated by only the specific forms of bigons, triangles and quadrangles as shown in Figure 10.14. The 1-st persistent path homology can thus be computed more efficiently by a different algorithm (from the above matrix reduction algorithm) by enumerating certain family of boundary cycles of small cardinality which generates the boundary group. In particular, the cardinality of this family depends on the so-called *arboricity* $\mathrm{a}(G)$ *of* G: Ignoring the direction of edges in graph G (i.e., viewing it as an undirected graph), its arboricity $\mathrm{a}(G)$ is the minimum number of edge-disjoint spanning forests into which G can be decomposed [45]. An alternative definition of the arboricity is that:

$$\mathrm{a}(G) = \max_{H \text{ is a subgraph of } G} \frac{|E(H)|}{|V(H)| - 1}. \tag{10.9}$$

Without describing the algorithm developed in [25], we present its computational complexity for the 1-st persistent path homology in the following theorem.

Theorem 15. *Given a directed weighted graph $G = (V, \vec{E}, w)$ with $n = |V|$, $m = |E|$, and $N_p = O(n^{p+1})$ the number of allowed elementary p-paths, assume that the time to compute the rank of a $r \times r$ matrix is r^ω. Let $d_{in}(v)$ and $d_{out}(u)$ denote the in-degree and out-degree of a node $v \in V$, and $\mathrm{a}(G)$ be the arboricity of G. Set $K = \min\{\mathrm{a}(G)m, \sum_{(u,v)\in\vec{E}} (d_{in}(u) + d_{out}(u))\}$. Then we can compute the p-th persistent path homology:*

- *in $O(m\alpha(n) + n \log n)$ time when $p = 0$;*
- *in $O(Km^{\omega-1} + \mathrm{a}(G)m)$ time when $p = 1$; and*

In particular, the arboricity $\mathrm{a}(G) = O(1)$ for plannar graphs, thus it takes $O(n^\omega)$ time to compute the 1-st persistent path homology for a planar directed graph G.

10.8 Notes and Exercises

The clique complex (also called the flag complex) is one of the most common ways to construct a simplicial complex from a graph. Recent years have seen much work on using the topological

profiles associated with the clique complex for network analysis; e.g, one of the early applications in [53]. Most materials covered in Section 10.5.2 come from [1, 2, 15]; note that [15] provides a detailed exposition for the intrinsic Čech and Vietoris-Rips filtrations of general metric spaces (beyond merely metric graphs). Theorem 9 comes as a corollary of Proposition 4 in [2], which is a stronger result than this theorem: In particular, Proposition 4 of [2] characterizes the homotopy type of the family of graphs described in Theorem 9.

The comparison of graphs via persistence distortion was proposed in [27].

Topological analysis for directed graphs and asymmetric networks is more recent. Nevertheless, the clique complex for directed graphs has already found applications in practical domains; e.g, [29, 49, 55]. The path homology was originally proposed in [40], and further studied and developed in [41, 42, 43] and the persistent version is proposed and studied in [17]. Note that as mentioned earlier, the path homology is not a simplicial homology, nevertheless, we have shown in this chapter that there is still a matrix reduction algorithm to compute it for any dimensions, with the same time complexity for computing the homology for simplicial complexes. The path homology also has a rich mathematical structure: There is a concept of homotopy theory for digraphs under which the path homology is preserved [41], and it is also dual to the cohomology theory of diagrams introduced in [42]. Note that in this chapter, we have assumed that the input directed graph does not have self-loops. Additional care is needed to handle such loops.

The matrix reduction algorithm for computing the persistent path homology that we described in Section 10.7.3 is based on the work in [17]. The algorithm of [17] assumes that the input graph is a complete and weighted directed graph; or equivalently, is a finite set V with a weight function $w : V \times V \to \mathbb{R}$ that may be asymmetric. We modify it so that the algorithm works with an arbitrary weighted graph. Finally, a hypergraph $G = (V, E)$ consists of a finite set of nodes V, and a collection $E \subseteq 2^V$ of subsets of V, each such subset called a hyper-edge. (In other words, a graph is a hypergraph where every hyper-edge has cardinality 2.) We remark that the idea behind path homology has also been extended to defining the so-called *embedded homology* for hypergraphs [10].

Exercise

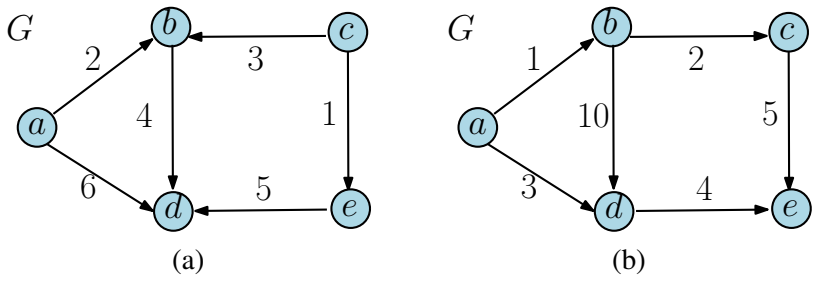


Figure 10.15: (a) graph for Exercise 4. (b) graph for Exercise 5. Edge weights are marked.

1. Consider a metric tree $(|T|, \mathsf{d}_T)$ induced by a weighted finite tree $T = (V, E, w)$. Suppose the largest edge weight is $w_0 > 0$. Consider the discrete intrinsic Čech complex $\mathbb{C}^r(V)$ spanned

by vertices in V. That is, let $B_{|T|}(x; r) := \{y \in |T| \mid \mathsf{d}_T(x, y) < r\}$ denote the open radius-r ball around a point x. Then, we have

$$\mathbb{C}^r(V) := \{\langle v_0, \ldots, v_p\rangle \mid v_i \in V, \text{ for } i \in [0, p], \text{ and } \bigcap_{i\in[0,p]} B_T(v_i; r) \neq \emptyset\}.$$

Prove that for any $r > w_0$, $\mathbb{C}^r(V)$ is homotopy equivalent to a point.

2. Consider a finite graph G=(V,E) with unit edge length, and its induced metric d_G on it. For a base point $v \in V$, let $f_v : |G| \to R$ be the shortest path distance function to v; that is, for any $x \in |G|$, $f_v(x) = \mathsf{d}_G(x, v)$.

   (2.a) Characterize the maxima of this function f_v.

   (2.b) Show that the total number of critical values of f_v is bounded from above by $O(n+m)$.

   (2.c) Show that this shortest path distance function can be described by $O(n + m)$ functions whose total descriptive complexity is $O(n + m)$.

3. Consider a finite metric graph $G = (V, E, w)$. Recall for each basepoint s $\in |G|$, it is mapped to the persitsence diagram $\mathrm{P_s}$ as in Eqn (10.3) (which is a point in the space of persistence diagrams). Show that this map is 1-Liptschiz w.r.t. the bottleneck metric on the space of persistence diagrams; that is, $\mathsf{d}_b(\mathrm{P_s}, \mathrm{P_t}) \leq \mathsf{d}_G(\mathrm{s}, \mathrm{t})$ for any two s, t $\in |G|$.

4. Consider the graph in Figure 10.15 (a). Compute the 0-th and 1-st persistence diagrams for the filtrations induced by (i) the directed clique complexes; (ii) the Dowker-sink complexes; and (iii) the Dowker-source complexes.

5. Consider the graph in Figure 10.15 (b). Compute the 1-st persistence diagram for the filtrations (i) induced by directed clique complexes; and (ii) induced by path homology.

6. Given an example of a graph which is a 2-boundary w.r.t. path homology, but is not a boundary cycle w.r.t. directed clique complex.

Bibliography

[1] M. Adamaszek and H. Adams. The Vietoris-Rips complexes of a circle. *Pacific Journal of Mathematics*, 290:1–40, 2017.

[2] Michal Adamaszek, Henry Adams, Ellen Gasparovic, Maria Gommel, Emilie Purvine, Radmila Sazdanovic, Bei Wang, Yusu Wang, and Lori Ziegelmeier. On homotopy types of Vietoris–Rips complexes of metric gluings. *J Appl. and Comput. Topology*, 2020. https://doi.org/10.1007/s41468-020-00054-y.

[3] Pankaj K. Agarwal, Kyle Fox, Abhinandan Nath, Anastasios Sidiropoulos, and Yusu Wang. Computing the Gromov-Hausdorff distance for metric trees. *ACM Trans. Algorithms*, 14(2):24:1–24:20, 2018.

[4] Ulrich Bauer, Xiaoyin Ge, and Yusu Wang. Measuring distance bewteen Reeb graphs. In *Proc. 30th Annu. Sympos. Comput. Geom. (SoCG)*, pages 464–473, 2014. See the full version at arXiv:1307.2839.

[5] Ulrich Bauer, Elizabeth Munch, and Yusu Wang. Strong equivalence of the interleaving and functional distortion metrics for Reeb graphs. In *Proc. 31rd Annu. Sympos. Comput. Geom. (SoCG)*, pages 461–475, 2015.

[6] Silvia Biasotti, Bianca Falcidieno, and Michela Spagnuolo. Extended Reeb graphs for surface understanding and description. In *Proc. 9th Internat. Conf. Discrete Geom. for Computer Imagery*, pages 185–197, 2000.

[7] Silvia Biasotti, Daniela Giorgi, Michela Spagnuolo, and Bianca Falcidieno. Reeb graphs for shape analysis and applications. *Theor. Comput. Sci.*, 392(1-3):5–22, 2008.

[8] Håvard Bjerkevik. Stability of higher-dimensional interval decomposable persistence modules. *arXiv preprint arXiv:1609.02086*, 2020.

[9] M. Botnan and M. Lesnick. Algebraic stability of zigzag persistence modules. *Algebraic & geometric topology*, 18:3133–3204, 2018.

[10] Stephane Bressan, Jingyan Li, Shiquan Ren, and Jie Wu. The embedded homology of hypergraphs and applications. *Asian Journal of Mathematics*, 23(3):479–500, 2019.

[11] D. Burago, Y. Burago, and S. Ivanov. *A course in metric geometry*. volume 33 of *AMS Graduate Studies in Math*. American Mathematics Society, 2001.

[12] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Comput. Geom*, 24(2):75–94, 2003.

[13] M. Carriere, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda. Perslay: a neural network layer for persistence diagrams and new graph topological signatures. In *23rd Intl. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2020. to appear.

[14] F. Chazal, R. Huang, and J. Sun. Gromov–Hausdorff approximation of filamentary structures using Reeb-type graphs. *Discrete Comput. Geom.*, 53:621—649, 2015.

[15] Frédéric Chazal, Vin de Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1):193–214, Dec 2014.

[16] Samir Chowdhury and Facundo Mémoli. Persistent homology of asymmetric networks: An approach based on dowker filtrations, 2018. arXiv:1608.05432.

[17] Samir Chowdhury and Facundo Mémoli. Persistent path homology of directed networks. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1152–1169. SIAM, 2018.

[18] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using poincaré and lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

[19] Kree Cole-McLaughlin, Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Loops in Reeb graphs of 2-manifolds. *Discrete & Computational Geometry*, 32(2):231–244, 2004.

[20] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[21] Justin Curry. Sheaves, cosheaves and applications, 2014.

[22] Vin de Silva, Elizabeth Munch, and Amit Patel. Categorified reeb graphs. *Discrete & Computational Geometry*, 55(4):854–906, Jun 2016.

[23] Tamal K. Dey. Computing height persistence and homology generators in $\mathbb{R}^3$ efficiently. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2649–2662, 2019.

[24] Tamal K. Dey, Fengtao Fan, and Yusu Wang. An efficient computation of handle and tunnel loops via reeb graphs. *ACM Trans. Graph.*, 32(4):32, 2013.

[25] Tamal K. Dey, Tianqi Li, and Yusu Wang. An efficient algorithm for 1-dimensional (persistent) path homology. pages 36:1–36:15, 2020.

[26] Tamal K. Dey, Facundo Mémoli, and Yusu Wang. Topological analysis of nerves, reeb spaces, mappers, and multiscale mappers. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 36:1–36:16, 2017.

[27] Tamal K. Dey, Dayu Shi, and Yusu Wang. Comparing graphs via persistence distortion. In *Proc. 31rd Annu. Sympos. Comput. Geom. (SoCG)*, pages 491–506, 2015.

[28] Tamal K. Dey and Yusu Wang. Reeb graphs: Approximation and persistence. *Discrete Comput. Geom.*, 49(1):46–73, 2013.

[29] Pawel Dlotko, Kathryn Hess, Ran Levi, Max Nolte, Michael Reimann, Martina Scolamiero, Katharine Turner, Eilif Muller, and Henry Markram. Topological analysis of the connectome of digital reconstructions of neural microcircuits. *arXiv preprint arXiv:1601.01580*, 2016.

[30] Harish Doraiswamy and Vijay Natarajan. Efficient output-sensitive construction of Reeb graphs. In *Proc. 19th Internat. Sym. Alg. and Comput.*, pages 556–567, 2008.

[31] Harish Doraiswamy and Vijay Natarajan. Efficient algorithms for computing Reeb graphs. *Computational Geometry: Theory and Applications*, 42:606–616, 2009.

[32] C. H. Dowker. Homology groups of relations. *Annals of Maths*, 56:84—95, 1952.

[33] H. Edelsbrunner and J. Harer. Persistent homology — a survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry. Twenty Years Later*, pages 257–282. Amer. Math. Soc., Providence, Rhode Island, 2008. Contemporary Mathematics 453.

[34] Herbert Edelsbrunner, John Harer, and Amit K. Patel. Reeb spaces of piecewise linear mappings. In *ACM Sympos. Comput. Geom. (SoCG)*, pages 242–250, 2008.

[35] Barbara Di Fabio and Claudia Landi. Reeb graphs of curves are stable under function perturbations. *Mathematical Methods in Applied Sciences*, 35:1456–1471, 2012.

[36] Barbara Di Fabio and Claudia Landi. The edit distance for Reeb graphs of surfaces. *Discrete Comput. Geom.*, 55:423—-461, 2016.

[37] E. Gasparovic, M. Gommel, E. Purvine, R. Sazdanovic, B. Wang, Y. Wang, and L. Ziegelmeier. The relationship between the intrinsic Čech and persistence distortion distances for metric graphs. *Journal of Computational Geometry (JoCG)*, 10(1), 2019. DOI: https://doi.org/10.20382/jocg.v10i1a16.

[38] X. Ge, I. Safa, M. Belkin, and Y. Wang. Data skeletonization via Reeb graphs. In *Proc. 25th Annu. Conf. Neural Information Processing Systems (NIPS)*, pages 837–845, 2011.

[39] Loukas Georgiadis, Robert Endre Tarjan, and Renato Fonseca F. Werneck. Design of data structures for mergeable trees. In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 394–403, 2006.

[40] Alexander Grigor'yan, Yong Lin, Yuri Muranov, and Shing-Tung Yau. Homologies of path complexes and digraphs. *arXiv preprint arXiv:1207.2834*, 2012.

[41] Alexander Grigor'yan, Yong Lin, Yuri Muranov, and Shing-Tung Yau. Homotopy theory for digraphs. *arXiv preprint arXiv:1407.0234*, 2014.

[42] Alexander Grigor'yan, Yong Lin, Yuri Muranov, and Shing-Tung Yau. Cohomology of digraphs and (undirected) graphs. *Asian J. Math*, 19(5):887–931, 2015.

[43] Alexander Grigor'yan, Yuri Muranov, and Shing-Tung Yau. Homologies of digraphs and künneth formulas. *Communications in Analysis and Geometry*, 25(5):969–1018, 2017.

[44] Mikhail Gromov. Groups of polynomial growth and expanding maps (with an appendix by Jacques Tits). *Publications Mathématiques de I'Institut des Hautes Études Scientifiques*, 53(1):53–78, 1981.

[45] F. Harary. *Graph Theory*. Addison Wesley series in mathematics. Addison-Wesley, 1971.

[46] W. Harvey, R. Wenger, and Y. Wang. A randomized $o(m \log m)$ time algorithm for computing Reeb graph of arbitrary simplicial complexes. In *Proc. 25th Annu. ACM Sympos. Compu. Geom.*, pages 267–276, 2010.

[47] Franck Hétroy and Dominique Attali. Topological quadrangulations of closed triangulated surfaces using the Reeb graph. *Graph. Models*, 65(1-3):131–148, 2003.

[48] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212. ACM, 2001.

[49] Paolo Masulli and Alessandro EP Villa. The topology of the directed clique complex as a network invariant. *SpringerPlus*, 5(1):388, 2016.

[50] Dmitriy Morozov, Kenes Beketayev, and Gunther H. Weber. Interleaving distance between merge trees. In *Workshop on Topological Methods in Data Analysis and Visualization: Theory, Algorithms and Applications*, 2013.

[51] Salman Parsa. A deterministic $O(m \log m)$ time algorithm for the Reeb graph. *Discrete & Computational Geometry*, 49(4):864–878, Jun 2013.

[52] Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. Graph.*, 26(3):58, 2007.

[53] Giovanni Petri, Martina Scolamiero, Irene Donato, and Francesco Vaccarino. Topological strata of weighted complex networks. *PLOS ONE*, 8:1–8, 06 2013.

[54] Geoge Reeb. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 222:847–849, 1946.

[55] Michael W Reimann, Max Nolte, Martina Scolamiero, Katharine Turner, Rodrigo Perin, Giuseppe Chindemi, Paweł Dłotko, Ran Levi, Kathryn Hess, and Henry Markram. Cliques of neurons bound into cavities provide a missing link between structure and function. *Frontiers in computational neuroscience*, 11:48, 2017.

[56] Yoshihisa Shinagawa, Tosiyasu L. Kunii, and Yannick L. Kergosien. Surface coding based on morse theory. *IEEE Comput. Graph. Appl.*, 11(5):66–78, 1991.

[57] Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, June 1983.

[58] Julien Tierny. *Reeb graph based 3D shape modeling and applications*. PhD thesis, Universite des Sciences et Technologies de Lille, 2008.

[59] Julien Tierny, Attila Gyulassy, Eddie Simon, and Valerio Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1177–1184, 2009.

[60] Elena Farahbakhsh Touli and Yusu Wang. FPT-algorithms for computing gromov-hausdorff and interleaving distances between trees. *CoRR*, abs/1811.02425, 2018. A conference version appeared in European Symposium on Algorithms (ESA) 2019.

[61] Tony Tung and Francis Schmitt. The augmented multiresolution Reeb graph approach for content-based retrieval of 3d shapes. *Internat. J. Shape Modeling*, 11(1):91–120, 2005.

[62] Claudia Landi Ulrich Bauer and Facundo Mémoli. The Reeb graph edit distance is universal, 2020.

[63] Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. Removing excess topology from isosurfaces. *ACM Trans. Graph.*, 23(2):190–208, 2004.

[64] Qi Zhao and Yusu Wang. Learning metrics for persistence-based summaries and applications for graph classification. In *33rd Annu. Conf. Neural Inf. Processing Systems (NeuRIPS)*, pages 9855–9866, 2019.