Chapter 8

Optimal Generators

In some applications, we are not only interested in the number of holes in our data, but we also want to get representations of these holes in the data. In other words, we are interested in finding a "representative" basis of the homology group. There are two main challenges to picking such a basis: First off, there are many different choices of homology classes which form a basis of the homology group. Second off, even within a single homology class, there are many homologous cycles. Thus, there are many different choices for cycles as bases of the homology group. How do we find good bases?

To specify our optimization target, we define a weight function $w : K_p \to \mathbb{R}_{\geqslant 0}$ on the p-simplices, and the weight of a chain is simply the sum, i.e., $w(c) = \sum \alpha_i w(\sigma_i)$ for $c = \sum \alpha_i \sigma_i$. The weight of a set of cycles $\mathcal{C}$ is then the sum of weights of each cycle. We are now interested in cycles that have minimal weight in their homology class, or at bases with minimum total weight.

We look at this problem in two settings: first we look at the case where we are given a fixed simplicial complex and we want to find an optimal basis for the homology of this complex. This can be applied for example if the persistence diagram of a filtration gives us a range of values in which we expect the complex to nicely capture the shape of the data. We can then compute an optimal basis for the homology of the fixed complex for some value in this range.

In some applications, we might also want to take a closer look at single intervals in the persistence barcode, that is, understand a hole that is born at time b and dies at time d (for example, to decide whether it corresponds to a feature in the data or is just a consequence of the process). This brings us to the second setting we look at in this chapter, where we want to find an optimal representative of a persistent homology class.

8.1 Optimal Basis of a Fixed Complex

Definition 8.1. *A set $\mathcal{C}$ of cycles is an* optimal basis *for $H_p(K)$ if it is a basis and there is no other basis $\mathcal{C}'$ with $w(\mathcal{C}') < w(\mathcal{C})$.*

How can we compute such an optimal basis?

In a first step, we are going to compute a set of cycles $\mathcal{C}$ which *contains* an optimal basis. Then, we perform a greedy algorithm to find the optimal basis in a similar way to Kruskal's algorithm for Minimum Spanning Tree: we sort the cycles by increasing weight, and start our basis B with the trivial cycle 0. Then, we simply iterate through our cycles and add a cycle c_i to our basis if it cannot be written as a linear combination of our current basis. Finally, we remove 0 again from B.

The correctness of this greedy approach, as well as the correctness of many other greedy algorithms including the above-mentioned Kruskal, follows from a more general framework in *matroid theory*.

Exercise 8.2. *A* matroid *is a collection $\mathcal{I}$ of subsets of some universe U, such that*

1. *$\emptyset \in \mathcal{I}$, and if some set L is in $\mathcal{I}$, all $L' \subseteq L$ are also in $\mathcal{I}$.*
2. *If some L, L' are in $\mathcal{I}$, and $|L'| = |L| + 1$, then there exists an element $f \in L' \setminus L$, such that $L \cup \{f\} \in \mathcal{I}$.*

The sets in $\mathcal{I}$ are also called the independent sets *of the matroid. The inclusion-maximal sets in $\mathcal{I}$ are called* bases.

(a) Show that for U being any finite set of vectors in some vector space, the family $\mathcal{I}$ of subsets of U corresponding to linearly independent vectors forms the family of independent sets of a matroid.

(b) Show that for any graph $G = (V, E)$, the family $\mathcal{I}$ of subsets of E corresponding to forests in G forms the family of independent sets of a matroid.

(c) Consider a matroid on a universe U with a weight function $w : U \to \mathbb{R}$. Consider the following greedy algorithm: begin with $L = \emptyset$, and consecutively add the lowest-weight element $e \notin L$ such that $L \cup \{e\}$ remains an independent set, until reaching a basis. Show that this greedy algorithm finds a minimum-weight basis.

We now need to show that we can implement all of the pieces of the algorithm we outlined above. For the first step, we need to be able to compute our set $\mathcal{C}$. For the second greedy step, we need to be able to check linear independence of a set of cycles.

Since the problem of computing an optimal basis is NP-hard for $p > 1$, we will focus on computing a basis for $H_1(K)$. Without loss of generality, we say that K is 2-dimensional, with n triangles and $O(n)$ edges and vertices.

To compute $\mathcal{C}$, we begin with $\mathcal{C} = \emptyset$. For all vertices v, we compute the shortest path tree T_v rooted at v. We can do this for example with Dijkstra's algorithm. For every edge e that is not in T_v, we add the unique simple cycle in $T_v \cup \{e\}$ to $\mathcal{C}$. This can be implemented in $O(n^2 \log n)$, and yields a set of cycles with $|\mathcal{C}| \in O(n^2)$. But, we need to prove that it is indeed a set which contains an optimal basis.

Lemma 8.3. *$\mathcal{C}$ as computed by the method above contains an optimal basis.*

Proof. Let $\mathcal{C}^*$ be an optimal basis. Towards a contradiction, let c be a cycle contained in $\mathcal{C}^* \setminus \mathcal{C}$. Since we consider $\mathbb{Z}_2$-homology, c is a simple cycle, i.e., no edge is used multiple times.[1]

Let v be a vertex in c, and let T_v be the corresponding shortest path tree. There must be an edge $e = \{u, w\}$ in $c \setminus T_v$, since T_v is a tree. Let $\Pi_{v,u}$ and $\Pi_{v,w}$ be the shortest paths from v to u and w, respectively. These paths must be contained in T_v. Let us similarly consider $\Pi'_{v,u}$ and $\Pi'_{v,w}$ to be the paths from v to u, w in c which do not use the edge e. If we now had $\Pi'_{v,u} = \Pi_{v,u}$ and $\Pi'_{v,w} = \Pi_{v,w}$, c would be the unique simple cycle in $T_v \cup \{e\}$ and thus c would be in $\mathcal{C}$. However, since we assumed this not to be the case, w.l.o.g. we can assume that $\Pi'_{v,u} \neq \Pi_{v,u}$.

We now define the cycles $c_1 = \Pi'_{v,w} + \{e\} + \Pi_{v,u}$ and $c_2 = \Pi_{v,u} + \Pi'_{v,u}$. We can now see that as we work in $\mathbb{Z}_2$, $c_1 + c_2 = \Pi'_{v,u} + \{e\} + \Pi'_{v,w} = c$. Furthermore, we have $w(c_1) \leqslant w(c)$, since $\Pi_{v,u}$ is a shortest path (in K), while $\Pi'_{v,u}$ is not necessarily shortest. The same also holds for c_2: $w(c_2) \leqslant w(c)$ since $\{\Pi'_{v,w}, e\}$ can not be shorter than $\Pi_{v,u}$.

Let us now consider the homology classes of c_1 and c_2. If both $[c_1]$ and $[c_2]$ were dependent on $\mathcal{C}^* \setminus \{c\}$, then so would $[c]$, since $c = c_1 + c_2$. Then, $\mathcal{C}^*$ would not be a basis. Thus, at least one of $[c_1]$ and $[c_2]$ has to be independent of $\mathcal{C}^* \setminus \{c\}$. Let us consider first that c_1 is independent. Then, we could replace c by c_1 in $\mathcal{C}^*$ and get a basis which is at least as good as $\mathcal{C}^*$. If c_2 is independent, we replace c by c_2 in $\mathcal{C}^*$. After replacing, we use the same argument again on the new cycle. If we replaced c by c_1, we repeat the argument with v', the common ancestor of $\Pi_{v,u}$ and $\Pi'_{v,w}$ and the same edge e. If we replaced c by c_2, we repeat the argument with v' the common ancestor of $\Pi_{v,u}, \Pi'_{v,u}$ and e an edge incident to u. By doing this we have made some "progress": In the second iteration of the argument, one of the two paths Π' must have become equal to Π. After doing the argument twice, we must have replaced c by a cycle c' that has $w(c') \leqslant w(c)$ and c' is actually in our set $\mathcal{C}$.

At the end, we thus get a basis $\mathcal{C}'$ with $w(\mathcal{C}') \leqslant w(\mathcal{C}^*)$ with $\mathcal{C}' \subseteq \mathcal{C}$. Therefore $\mathcal{C}$ contains an optimal basis. □

So, we have finished the first step of our algorithm. It remains to figure out how to check independence. For this, we introduce *annotations*.

Definition 8.4. *An* annotation of p-simplices *is a function* $a : K^p \to \mathbb{Z}_2^g$ *giving each* p-*simplex a binary vector of size* g. *This extends to chains by sums. An annotation must fulfill:*

- $g = \beta_p(K)$
- $a(z_1) = a(z_2)$ *iff* $[z_1] = [z_2]$.

Given an annotation, we can now clearly check linear independence of cycles by simply checking linear independence of a set of vectors, for which we have existing tools such as Gaussian elimination.

[1] In general this would also follow from the weights being non-negative and $\mathcal{C}^*$ being optimal.

Proposition 8.5. *In every simplicial complex* K *and for every* $p \geqslant 0$*, there exists an annotation of* p*-simplices, and such an annotation can also be computed.*

Proof. (Sketch for $p = 1$) We can compute a spanning forest T, and let m be the number of remaining edges. We initialize annotations of length m, and set $a(e) = 0$ for every edge in the spanning forest T. For every remaining edge e_i, we set $a_j(e_i) = 1$ if and only if $j = i$, and 0 otherwise.

For every triangle $t \in K$, if the annotation of its boundary δt is not 0, we find a non-zero entry b_u in $a(\delta t)$ and add $a(\delta t)$ to every edge with $a_u(e) = 1$, and we delete the u-th entry from all annotations. One can show that this yields a valid annotation, and it can be implemented in $O(n^3)$, and more clever implementations work in $O(n^\omega)$. □

To check independence of cycles in our set $\mathcal{C}$ more efficiently, we add auxiliary annotations also to vertices in a shortest path tree T_v rooted at v. We give v the annotation 0, and for a vertex x that is the child of y, we set $a(x) := a(y) + a(e_{xy})$. For every cycle defined by the non-tree edge $e = uw$, we now have $a(c_e) = a(u) + a(w) + a(e)$. So, we never actually have to compute an explicit representation of a cycle by its edges, we only need to store its weight, the shortest path trees with the auxiliary annotations, and the non-tree edge e. Note that the auxiliary annotations can be computed in $O(gn)$ for the whole tree, thus in $O(gn^2)$ for all trees.

Finally, we have to check independence. Given an $(n \times m)$ matrix M, we can find the lexicographic leftmost set of independent columns in time $O(\max(n, m)^\omega)$. Instead of naively doing this n^2 times (once for every cycle), we group our cycles of $\mathcal{C}$ into groups A_i of size g, and compute the leftmost set for $[B|A_i]$, and thus we get $O(n^2 g^{\omega-1})$ runtime for this step.

To summarize, computing $\mathcal{C}$ takes $O(n^2 \log n)$, sorting the $O(n^2)$ cycles also takes $O(n^2 \log n)$, and for checking linear independence we need $O(n^\omega)$ for the annotations of the edges, $O(gn^2)$ for the auxiliary annotations, and $O(n^2 g^{\omega-1})$ for the block-wise linear independence checking. Overall, we thus get a runtime of $O(n^\omega + n^2 g^{\omega-1})$.

Theorem 8.6. *Given a 2-dimensional simplicial complex* K *with* n *triangles and a weight function* w *on its edges, we can compute an optimal basis of* $H_1(K)$ *in time* $O(n^\omega + n^2 g^{\omega-1})$.

8.2 Persistent Cycles

In the persistent setting, given a simplex-wise filtration $\mathcal{F}$ and an interval $[b, d]$, can we find an optimal persistent p-cycle c that is born at b and dies at d?

Sadly, this problem is already known to be NP-hard for $d < \infty$ and $p \geqslant 1$. However, if we assume that K is a weak $(p+1)$-pseudomanifold, i.e., a simplicial complex in which each p-simplex is a face of at most 2 $(p+1)$-simplices, then there exists a polynomial-time algorithm, which we will describe in this section.

If we consider cycles that live until ∞, we can solve the problem in polynomial time for $p = 1$, but it is NP-hard for $p \geqslant 2$. Here, the assumption of K being a weak $(p+1)$-pseudomanifold does not save us. However, if we further assume that the complex can be embedded in $\mathbb{R}^{p+1}$, then it is again polynomial.

To solve the problem for $d < \infty$ in a weak $(p+1)$-pseudomanifold, we consider undirected flow networks: We have a graph, where every edge has a capacity in $[0, \infty]$, some sources, and some sinks, and we want to find the maximum flow we can send from the sources to the sinks without sending too much flow through any edge. Recall that if we consider a cut which separates the sources from the sinks, the capacity of this cut is an upper bound on the value of the maximum flow. Furthermore, if we consider the minimum such cut, its capacity is equal to the value of the maximum flow. This can be solved in polynomial time.

We can build a dual graph G, by placing a vertex into every $(p+1)$-simplex and adding an edge whenever they share a p-simplex. We furthermore add a dummy vertex which gets connected to all vertices which only have one neighbor. We are going to make the vertex belonging to the $(p+1)$-simplex which is the destructor of our desired cycle the source. Furthermore, we make the dummy vertex as well as all vertices belonging to $(p+1)$-simplices added after the destructor into sinks. Edges added at or before the birth are getting the capacity equal to their weight, while all other edges get capacity ∞. Then, it turns out that the p-simplices belonging to the edges in a minimum cut separating the sources from the sinks are an optimal persistent cycle.

Exercise 8.7. *Consider a simplex-wise filtration on a simplicial complex that is a weak $(p+1)$-pseudomanifold, and consider some interval $[b, d]$ (for $d < \infty$) such that there exists a p-cycle born at b and dying at d. We look at the dual graph G with source and sinks defined as in the lecture. Consider a cut with finite capacity that separates the source from the sinks. Let c be the chain corresponding to the p-simplices dual to the edges going over this cut. Show that c is a p-cycle born at b and dying at d, and show that its weight is equal to the capacity of the cut.*

This exercise proves one direction of the correctness of the algorithm described above. The other direction is similar. We get the following result.

Theorem 8.8. *Given a a simplex-wise filtration on a simplicial complex that is a weak $(p+1)$-pseudomanifold and an interval $[b, d]$ (for $d < \infty$), we can compute an optimal p-cycle born at b and dying at d in polynomial time.*

For details, we refer to Chapter 5 in the book of Dey and Wang [1].

Questions

32. *How can we compute an optimal basis given a set of cycles that contain one?* Explain the algorithm described in Section 8.1. Further, explain annotations and how they can be used to check linear independence.

33. *How can we compute a set of 1-cycles that contain an optimal basis of* H_1*?* Describe the algorithm to do this and prove its correctness.

34. *How can we compute an optimal persistent cycle?* Explain the algorithm described in Section 8.2.

References

[1] Tamal Krishna Dey and Yusu Wang, *Computational topology for data analysis*, Cambridge University Press, 2022.