A Gradient Sampling Algorithm for Stratified Maps with Applications to Topological Data Analysis

Jacob Leygonie, Mathieu Carrière, Théo Lacombe, Steve Oudot

A Gradient Sampling Algorithm for Stratified Maps with Applications to Topological Data Analysis

Jacob Leygonie[*1], Mathieu Carrière[†2], Théo Lacombe[‡3], and Steve Oudot[§4]

[1]Mathematical Institute, Oxford University, UK
[2]DataShape, Université Côte d'Azur, Inria, France
[3]LIGM, Université Gustave Eiffel, France.
[4]DataShape, Université Paris-Saclay, CNRS, Inria, Laboratoire de Mathématiques d'Orsay, France.

Abstract

We introduce a novel gradient descent algorithm extending the well-known Gradient Sampling methodology to the class of stratifiably smooth objective functions, which are defined as locally Lipschitz functions that are smooth on some regular pieces—called the strata—of the ambient Euclidean space. For this class of functions, our algorithm achieves a sub-linear convergence rate. We then apply our method to objective functions based on the (extended) persistent homology map computed over lower-star filters, which is a central tool of Topological Data Analysis. For this, we propose an efficient exploration of the corresponding stratification by using the Cayley graph of the permutation group. Finally, we provide benchmark and novel topological optimization problems, in order to demonstrate the utility and applicability of our framework.

Contents

[*]jacob.leygonie@maths.ox.ac.uk
[†]mathieu.carriere@inria.fr
[‡]theo.lacombe@univ-eiffel.fr
[§]steve.oudot@inria.fr

This research was conducted while Théo Lacombe was affiliated to DataShape, Université Paris-Saclay, CNRS, Inria, Laboratoire de Mathématiques d'Orsay, France.

1 Introduction

1.1 Motivation and related work

In its most general instance nonsmooth, non convex, optimization seek to minimize a locally Lipschitz objective or loss function $f : \mathbb{R}^n \to \mathbb{R}$. Without further regularity assumptions on f, most algorithms—such as the usual Gradient Descent with learning rate decay, or the Gradient Sampling method—are only guaranteed to produce iterates whose subsequences are asymptotically stationary, without explicit convergence rates. Meanwhile, when restricted to the class of min-max functions (like the maximum of finitely many smooth maps), stronger guarantees such as convergence rates can be obtained [43]. This illustrates the common paradigm in nonsmooth optimization: the richer the structure in the irregularities of f, the better the guarantees we can expect from an optimization algorithm. Note that there are algorithms specifically tailored to deal with min-max functions, e.g. [8].

Another example are bundle methods [37, 38, 42]. They consist, roughly, in constructing successive linear approximations of f as a proxy for minimization. Their convergence guarantees are strong, especially when an additional *semi-smoothness* property of f [10, 57] can be made. Other types of methods, like the variable metric ones, can also benefit from the semi-smoothness hypothesis [73]. In many cases, convergence properties of the algorithm are not only dependent on the structure on f, but also on the amount of information about f that can be computed in practice. For instance, the bundle method [55] assumes that the Hessian matrix, when defined locally, can be computed. For accounts of the theory and practice in nonsmooth optimization, we refer the interested reader to [5, 51, 66].

The ability to cut $\mathbb{R}^n$ in well-behaved pieces where f is regular, is another type of important structure. Examples, in increasing order of generality, are semi-algebraic, (sub)analytic, definable, tame (w.r.t. an o-minimal structure), and Whitney stratifiable functions [11]. For such objective functions, the usual gradient descent (GD) algorithm, or a stochastic version of it, converges to stationary points [31]. In order to obtain further theoretical guarantees such as convergence rates, it is necessary to design optimization algorithms specifically tailored for regular maps, since they enjoy stronger properties, e.g., tame maps are semi-smooth [48], and the generalized gradients of Whitney stratifiable maps are closely related to the (restricted) gradients of the map along the strata [11]. Besides, strong convergence guarantees can be obtained under the Kurdyka–Łojasiewicz assumption [4, 61], which includes the class of semi-algebraic maps. Our method is related to this line of work, in that we exploit the strata of $\mathbb{R}^n$ in which f is smooth.

The motivation of this work stems from Topological Data Analysis (TDA), where geometric objects such as graphs are described by means of computable and topological descriptors. Persistent Homology (PH) is one such descriptor, and has been successfully applied in various areas such as neuroscience [30, 7], material sciences [44, 69], signal analysis [63, 72], shape recognition [54], or machine learning [23, 18].

Persistent Homology describes graphs, and more generally simplicial complexes, over n nodes by means of a signature called the *barcode*, or *persistence diagram* $\mathrm{PH}(x)$. Here x is a *filter function*, that is a function on the nodes, which we view as a vector in $\mathbb{R}^n$. Loosely speaking, $\mathrm{PH}(x)$ is a finite multi-set of points in

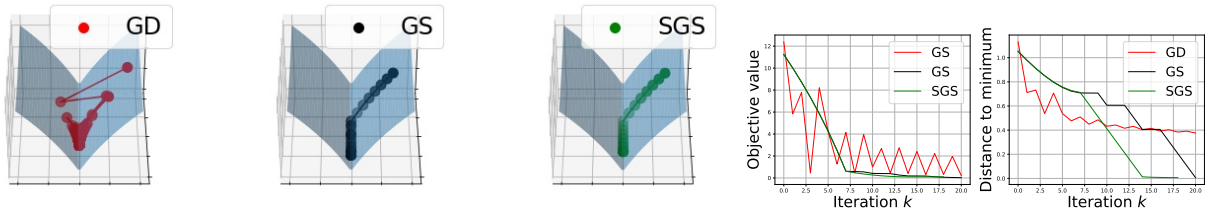

Figure 1: A proof-of-concept comparison between different gradient descent techniques. The objective function $f : (z_1, z_2) \in \mathbb{R}^2 \to 10 \log(1 + |z_1|) + z_2^2 \in \mathbb{R}$ (blue surface) attains its minimum at $x_* = (0, 0)$ and is not smooth along the line $\{z_1 = 0\}$. In particular, $\|\nabla f\| > 1$ around x_*, thus the gradient norm cannot be used as a stopping criterion. The traditional GD, for which updates are given by $x_{k+1} = x_k - \frac{\lambda_0}{k+1} \nabla f(x_k)$, oscillates around $\{z_1 = 0\}$ due to the non-smoothness of f and asymptotically converges toward x_* because of the decaying learning rate $\frac{\lambda_0}{k+1}$. In the meantime, non-smooth optimization methods that sample points around x_k in order to produce reliable descent directions converge in finite time. Namely, the classical Gradient Sampling method randomly samples 3 points and manages to reach an (ϵ, η)-stationary point of f in $\sim 20.6 \pm 3.9$ iterations (averaged over 100 experiments), while our stratified approach improves on this by leveraging the fact that we explicitly have access to the two strata $\{z_1 < 0\}$ and $\{z_1 > 0\}$ where f is differentiable. In particular, we only sample additional points when x_k is ϵ-near the line $\{z_1 = 0\}$, and reach an (ϵ, η)-stationary point in 18 iterations. Right plots showcase the evolution of the objective value $f(x_k)$ and the distance to the minimum $\|x_k - x_*\|$ across iterations. Parameters: $x_0 = (0.8, 0.8)$, $\lambda_0 = 10^{-1}$, $\epsilon = 10^{-1}$, $\eta = 10^{-2}$.

the upper half-plane $\{(b, d) \in \mathbb{R}^2,\ d \geqslant b\}$ that encodes topological and geometric information about the underlying simplicial complex and the function x.

Barcodes form a metric space **Bar** when equipped with the standard metrics of TDA, the so-called *bottleneck* and *Wasserstein* distances, and the persistence map $\mathrm{PH} : \mathbb{R}^n \to \mathbf{Bar}$ is locally Lipschitz [27, 28]. However **Bar** is not Euclidean nor a smooth manifold, thus hindering the use of these topological descriptors in standard statistical or machine learning pipelines. Still, there exist natural notions of differentiability for maps in and out of **Bar** [53]. In particular, the persistence map $\mathrm{PH} : \mathbb{R}^n \to \mathbf{Bar}$ restricts to a locally Lipschitz, smooth map on a stratification of $\mathbb{R}^n$ by polyhedra. If we compose the persistence map with a smooth and Lipschitz map $V : \mathbf{Bar} \to \mathbb{R}$, the resulting objective (or loss) function

$$f : \mathbb{R}^n \xrightarrow{\ \mathrm{PH}\ } \mathbf{Bar} \xrightarrow{\ V\ } \mathbb{R}$$

is itself Lipschitz and smooth on the various strata. From [31], and as recalled in [17], classical (Stochastic) Gradient Descent on f asymptotically converges to stationary points. Similarly, the Gradient Sampling (GS) method asymptotically converges. See [68] for an application of GS to topological optimization.

Nonetheless, it is important to design algorithms that take advantage of the structure in the irregularities of the persistence map PH, in order to get better theoretical guarantees. For instance, one can locally integrate the gradients of PH—whenever defined—to stabilize the iterates [68], or add a regularization term to f that acts as a smoothing operator [29]. In this work, we rather exploit the stratification of $\mathbb{R}^n$ induced by PH, as it turns out to be easy to manipulate. We will show in particular that we can efficiently access points x' located in neighboring strata of the current iterate x, as well as estimate the distance to these strata.

For this reason, we believe that persistent homology-based objective functions f form a rich playground for nonsmooth optimization, with many applications in point cloud inference [40], surface reconstruction [13], shape matching [64], graph classification [45, 75], topological regularization for generative models [58, 46, 39], image segmentation [47, 26], or dimensionality reduction [50], to name a few.

1.2 Contributions and outline of contents

Our new method, called *Stratified Gradient Sampling* (SGS), is a variation of the established GS algorithm, whose main steps for updating the current iterate $x_k \in \mathbb{R}^n$ we recall in Algorithm 1 below. Our method is

Algorithm 1 An update step with the Gradient Sampling algorithm

1: Sample $m \geqslant n+1$ points $x_k^1, \cdots, x_k^m$ in a ball $B(x_k, \epsilon)$
2: Compute *approximate subgradient* $G_k := \{\nabla f(x_k), \nabla f(x_k^1), \cdots, \nabla f(x_k^m)\}$
3: Compute *descent direction* $g_k := \operatorname{argmin}\{\|g\|^2,\ g \text{ in convex hull of } G_k\}$
4: Find *step size* $t_k \geqslant 0$ so that $f(x_k - t_k g_k) \leqslant f(x_k) - \beta t_k \|g_k\|^2$ ($\beta \in (0,1)$ hyperparameter)
5: Ensure that f is differentiable at $x_{k+1} := x_k - t_k g_k$ by small perturbations

motivated by the closing remarks of a recent overview of the GS methodology [15], in which the authors suggest that the GS theory and practice could be enhanced by assuming some extra structure on top of the non differentiability of f.

In this work, we deal with *stratifiably smooth maps*, for which the non differentiability is organized around smooth submanifolds that together form a stratification of $\mathbb{R}^n$. In Section 2, we review some background material in nonsmooth analysis and define stratifiably smooth maps $f : \mathbb{R}^n \to \mathbb{R}$, a variant of the Whitney stratifiable functions from [11] for which we do not impose any Whitney regularity on the gluing between adjacent strata of $\mathbb{R}^n$, but rather enforce that there exist local C^2-extensions of the restrictions of f to top-dimensional strata.

In order to update the current iterate x_k when minimizing a stratifiably smooth objective function f, we introduce a new descent direction g_k. As in GS, g_k is obtained in our new SGS algorithm by collecting the gradients of samples around x_k in an approximate subgradient G_k, and then by taking the element with minimal norm in the convex set generated by G_k. A key difference with GS is that we only need to sample as many points around x_k as there are distinct strata close by, compare with the $m \geqslant n+1$ samples of Algorithm 1. In Proposition 3, we show that we indeed obtain a descent direction, i.e., that we have the descent criterion $f(x_k - t_k g_k) \leqslant f(x_k) - \beta t_k \|g_k\|^2$ (as in Line 4 of Algorithm 1) for a suitable choice of step size t_k.

Our SGS algorithm is detailed in Section 3.1 and its analysis in Section 3.2. The convergence of the original GS methodology crucially relies on the sample size $m \geqslant n+1$ in order to apply the Carathéodory Theorem to subgradients. Differently, our convergence analysis relies on the fact that the gradients of f, when restricted to neighboring strata, are locally Lipschitz. Hence, our proof of asymptotic convergence to stationary points (Theorem 2) is substantially different. In Theorem 3, we determine a convergence rate of our algorithm that holds for any proper stratifiably smooth map, which is an improvement over the guarantees of GS for general locally Lipschitz maps. Finally, in Section 3.3, we adapt our method and results to the case where only estimated distances to nearby strata are available.

In Section 4, we introduce the persistence map PH over a simplicial complex K, which gives rise to a wide class of stratifiably smooth objectivefunctions with rich applications in TDA. We characterize strata around the current iterate (i.e., filter function) x_k by means of the permutation group over n elements, where n is the number of vertices in K. Then, the Cayley graph associated to the permutation group allows us to use Dijkstra's algorithm to efficiently explore the set of neighboring strata by increasing order of distances to x_k, that are needed to compute descent directions.

Section 5 is devoted to the implementation of the SGS algorithm for the optimization of persistent homology-based objective functions f. In Section 5.1, we provide empirical evidence that SGS behaves better than GD and GS with a simple experiment about minimization of total persistence. In Section 5.3 and Section 5.2, we consider two novel topological optimization problems which we believe are of interest in real-world applications. On the one hand, the *Topological Template Registration* of a filter function x defined on a complex K, is the task of finding a filter function x' over a smaller complex K' that preserves the barcode of x. On the other hand, given a Mapper graph G, which is a standard visualization tool for

arbitrary data sets [67], we can bootstrap the data set in order to produce multiple bootstrapped graphs G_i. The *Topological Mean* is then the task of finding a new graph G^* whose barcode is as close as possible to the mean of the barcodes associated to the graphs G_i. As a result we obtain a smoothed version G^* of the Mapper graph G in which spurious and non-relevant graph attributes are removed.

2 A direction of descent for stratifiably smooth maps

In this section, we define the class of stratifiably smooth maps whose optimization is at stake in this work. For such maps, we can define an approximate subgradient and a corresponding descent direction, which is the key ingredient of our algorithm.

2.1 Nonsmooth Analysis

We first recall some useful background in nonsmooth analysis, essentially from [25]. Throughout, $f : \mathbb{R}^n \to \mathbb{R}$ is a locally Lipschitz (non necessary smooth, nor convex) and proper (i.e., compact sublevel sets) function, which we aim to minimize.

First-order information of f at $x \in \mathbb{R}^n$ in a direction $v \in \mathbb{R}^n$ is captured by its *generalized directional derivative*:

$$f^\circ(x; v) = \limsup_{y \to x, t \downarrow 0^+} \frac{f(y + tv) - f(y)}{t}, \tag{1}$$

Besides, the *generalized gradient* is the following set of linear subapproximations:

$$\partial f(x) := \big\{\zeta \in \mathbb{R}^n,\ f^\circ(x; v) \geqslant \langle \zeta, v\rangle \text{ for all } v \in \mathbb{R}^n\big\}. \tag{2}$$

Given an arbitrary set $S \subset \mathbb{R}^n$ of Lebesgue measure 0, we have an alternative description of the generalized gradient in terms of limits of surrounding gradients, whenever defined:

$$\partial f(x) = \overline{\text{co}}\big\{ \lim \nabla f(x_i) \,|\, x_i \to x,\ \nabla f(x_i) \text{ is defined },\ \lim \nabla f(x_i) \text{ exists}, x_i \notin S\big\}, \tag{3}$$

where $\overline{\text{co}}$ is the operation of taking the closure of the convex hull.[1] The duality between generalized directional derivatives and gradients is captured by the equality:

$$f^\circ(x; v) = \max\{\langle \zeta, v\rangle, \zeta \in \partial f(x)\}. \tag{4}$$

The *Goldstein subgradient* [41] is an ϵ-relaxation of the generalized gradient:

$$\partial_\epsilon f(x) = \overline{\text{co}}\big\{ \lim \nabla f(x_i) \,|\, x_i \to x',\ \nabla f(x_i) \text{ is defined },\ \lim \nabla f(x_i) \text{ exists},\ |x - x'| \leqslant \epsilon\big\}. \tag{5}$$

Given $x \in \mathbb{R}^n$, we say that:

$$x \text{ is a } \textit{stationary point} \text{ (for } f) \text{ if } 0 \in \partial f(x).$$

Any local minimum is stationary, and conversely if f is convex. We also have weaker notions. Namely, given $\epsilon, \eta > 0$,

$$\begin{gathered} x \text{ is } \textit{ϵ-stationary} \text{ if } 0 \in \partial_\epsilon f(x); \text{ and} \\ x \text{ is } \textit{(ϵ, η)-stationary} \text{ if } d(0, \partial_\epsilon f(x)) \leqslant \eta. \end{gathered}$$

[1] Here the equality holds as well (by some compactness argument) when taking the convex hull without closure. As we take closed convex hulls later on, we choose not to introduce this subtlety explicitly.

2.2 Stratifiably smooth maps

Desirable properties for an optimization algorithm is that it produces iterates $(x_k)_k$ that either converge to an (ϵ, η)-stationary point in finitely many steps, or whose subsequences (or some of them) converge to an ϵ-stationary point. For this, we work in the setting of objective functions that are smooth when restricted to submanifolds, that together partition $\mathbb{R}^n$.

Definition 1. A *stratification* $\mathcal{X} = \{X_i\}_{i \in I}$ of a closed subset $\mathbb{X} \subseteq \mathbb{R}^n$ is a locally finite partition of $\mathbb{X}$ by smooth submanifolds X_i—called *strata*—such that for $i \neq j \in I$:

$$\overline{X_i} \cap X_j \neq \varnothing \Rightarrow X_j \subseteq \overline{X_i} \backslash X_i.$$

This makes $(\mathbb{X}, \mathcal{X})$ into a *stratified space*.

Note that we do not impose any (usually needed) gluing conditions between adjacent strata, as we do not require them in the analysis. In particular, semi-algebraic, subanalytic, or definable subsets of $\mathbb{R}^n$, together with Whitney stratified sets are stratified in the above weak sense. We next define the class of maps f with smooth restrictions $f_{|X_i}$ to strata X_i of some stratification $\mathcal{X}$, inspired by the Whitney stratifiable maps of [11] (there $\mathcal{X}$ is required to be Whitney) and the *stratifiable functions* of [34], however we further require that the restrictions $f_{|X_i}$ admit local extensions of class C^2.

Definition 2. The map $f : \mathbb{R}^n \to \mathbb{R}$ is *stratifiably smooth* if there exists a stratification $\mathcal{X}$ of $\mathbb{R}^n$, such that for each stratum $X_i \in \mathcal{X}$, the restriction $f_{|X_i}$ admits an extension f_i of class C^2 in a neighborhood of X_i.

Remark 1. The slightly weaker assumption that the extension f_i is continuously differentiable with locally Lipschitz gradient would have also been sufficient for our purpose.

We denote by X_x the stratum containing x, and by $\mathcal{X}_x \subseteq \mathcal{X}$ the set of strata containing x in their closures. More generally, for $\epsilon > 0$, we let $\mathcal{X}_{x,\epsilon} \subseteq \mathcal{X}$ be the set of strata X_i such that the closure of the ball $B(x, \epsilon)$ has non-empty intersection with the closure of X_i. Local finiteness in the definition of a stratification implies that $\mathcal{X}_{x,\epsilon}$ (and $\mathcal{X}_x$) is finite.

If f is stratifiably smooth and $X_i \in \mathcal{X}_x$ is a stratum, there is a well-defined limit gradient $\nabla_{X_i} f(x)$, which is the unique limit of the gradients $\nabla f_{|X_i}(x_n)$ where $x_n \in X_i$ is any sequence converging to x. Indeed, this limit exists and does not depend on the choice of sequence since $f_{|X_i}$ admits a local C^2 extension f_i. The following result states that the generalized gradient at x can be retrieved from these finitely many limit gradients along the various adjacent top-dimensional strata.

Proposition 1. *If f is stratifiably smooth, then for any $x \in \mathbb{R}^n$ we have:*

$$\partial f(x) = \mathrm{co}\big\{\nabla_{X_i} f(x),\ X_i \in \mathcal{X}_x \text{ is of dimension } n\big\}.$$

More generally, for $\epsilon > 0$:

$$\partial_\epsilon f(x) = \overline{\mathrm{co}}\big\{\nabla_{X_i} f(x') \,|\, |x' - x| \leqslant \epsilon,\ X_i \in \mathcal{X}_{x'} \subseteq \mathcal{X}_{x,\epsilon} \text{ is of dimension } n\big\}.$$

Proof. We show the first equality only, as the second can be proven along the same lines. We use the description of $\partial f(x)$ in terms of limit gradients from Equation (3), which implies the inclusion of the right-hand side in $\partial f(x)$. Conversely, let S be the union of strata in $\mathcal{X}_x$ with positive codimension, which is of measure 0. Let x_i be a sequence avoiding S, converging to x, such that $\nabla f(x_i)$ converges as well. Since $\mathcal{X}_x$ is finite, up to extracting a subsequence, we can assume that all x_i are in the same top-dimensional stratum $X_i \in \mathcal{X}_x$. Consequently, $\nabla f(x_i)$ converges to $\nabla_{X_i} f(x)$. □

2.3 Direction of descent

Thinking of x as a current position, we look for a direction of *(steepest) descent*, in the sense that a perturbation of x in this direction produces a (maximal) decrease of f. Given $\epsilon \geqslant 0$, we let $g(x,\epsilon)$ be the projection of the origin on the convex set $\partial_\epsilon f(x)$. Equivalently, $g(x,\epsilon)$ solves the minimization problem:

$$g(x,\epsilon) = \operatorname{argmin}\big\{\|g\|,\, g \in \partial_\epsilon f(x)\big\}. \tag{6}$$

Introduced in [41], the direction $-g(x,\epsilon)$ is a good candidate of direction for descent, as we explain now. Since $g(x,\epsilon)$ is the projection of the origin on the convex closed set $\partial_\epsilon f(x)$, we have the classical inequality $\langle g(x,\epsilon), g(x,\epsilon) - g\rangle \leqslant 0$ that holds for any g in the Goldstein subgradient at x. Equivalently,

$$\forall g \in \partial_\epsilon f(x),\ \langle -g(x,\epsilon), g\rangle \leqslant -\|g(x,\epsilon)\|^2. \tag{7}$$

Informally, if we think of a small perturbation $x - tg(x,\epsilon)$ of x along this direction, for $t > 0$ small enough, then $f(x - tg(x,\epsilon)) \approx f(x) - t\langle \nabla f(x), g(x,\epsilon)\rangle$. Using Equation (7), since $\nabla f(x) \in \partial_\epsilon f(x)$, we deduce that $f(x - tg(x,\epsilon)) \leqslant f(x) - t\|g(x,\epsilon)\|^2$. So f locally decreases at linear rate in the direction $-g(x,\epsilon)$. This intuition relies on the fact that $\nabla f(x)$ is well-defined so as to provide a first order approximation of f around x, and that t is chosen small enough. In order to make this reasoning completely rigorous, we need the following well-known result (see for instance [25]):

Theorem 1 (Lebourg Mean value Theorem)**.** *Let $x, x' \in \mathbb{R}^n$. Then there exists some $y \in [x, x']$ and some $w \in \partial f(y)$ such that:*

$$f(x') - f(x) = \langle w, x' - x\rangle.$$

Let $t > 0$ be lesser than $\frac{\epsilon}{\|g(x,\epsilon)\|}$, in order to ensure that $x' := x - tg(x,\epsilon)$ is ϵ-close to x. Then by the mean value theorem (and Proposition 1), we have that

$$f(x - tg(x,\epsilon)) - f(x) = -t\langle w, g(x,\epsilon)\rangle$$

for some $w \in \partial_\epsilon f(x)$. Equation (7) yields

$$\forall t \leqslant \frac{\epsilon}{\|g(x,\epsilon)\|}, \qquad f(x - tg(x,\epsilon)) \leqslant f(x) - t\|g(x,\epsilon)\|^2, \tag{8}$$

as desired.

In practical scenarios however, it is unlikely that the exact descent direction $-g(x,\epsilon)$ could be determined. Indeed, from Equation (6), it would require the knowledge of the set $\partial_\epsilon f(x)$, which consists of infinitely many (limits of) gradients in an ϵ-neighborhood of x. We now build, provided f is stratifiably smooth, a faithful approximation $\tilde{\partial}_\epsilon f(x)$ of $\partial_\epsilon f(x)$, by collecting gradient information in the strata that are ϵ-close to x.

For each top-dimensional stratum $X_i \in \mathcal{X}_{x,\epsilon}$, let x_i be an arbitrary point in $\overline{X}_i \cap \overline{B}(x,\epsilon)$. Define

$$\tilde{\partial}_\epsilon f(x) := \overline{\operatorname{co}}\big\{\nabla_{X_i} f(x_i),\, X_i \in \mathcal{X}_{x,\epsilon}\big\}. \tag{9}$$

Of course, $\tilde{\partial}_\epsilon f(x)$ depends on the choice of each $x_i \in X_i$. But this will not matter for the rest of the analysis, as we will only rely on the following approximation result which holds for arbitrary choices of points x_i:

Proposition 2. *Let $x \in \mathbb{R}^n$ and $\epsilon > 0$. Assume that f is stratifiably smooth. Let L be a Lipschitz constant of the gradients ∇f_i restricted to $\overline{B}(x,\epsilon) \cap X_i$, where f_i is some local C^2 extension of $f_{|X_i}$, and $X_i \in \mathcal{X}_{x,\epsilon}$ is top dimensional. Then we have:*

$$\tilde{\partial}_\epsilon f(x) \subseteq \partial_\epsilon f(x) \subseteq \tilde{\partial}_\epsilon f(x) + \overline{B}(0, 2L\epsilon).$$

In particular, $d_H(\tilde{\partial}_\epsilon f(x), \partial_\epsilon f(x)) \leqslant 2L\epsilon$.

Note that, since the f_i are of class C^2, their gradients are locally Lipschitz, hence by compactness of $\overline{B}(x,\epsilon)$, the existence of the Lipschitz constant L above is always guaranteed.

Proof. From Proposition 1, we have

$$\partial_\epsilon f(x) = \overline{\text{co}}\big\{\nabla_X f(x') \,|\, |x'-x| \leqslant \epsilon,\, X \in \mathcal{X}_{x'} \subseteq \mathcal{X}_{x,\epsilon} \text{ is of dimension } n \,\big\}.$$

This yields the inclusion $\tilde{\partial}_\epsilon f(x) \subseteq \partial_\epsilon f(x)$. Now, let $x' \in \mathbb{R}^n$, $|x'-x| \leqslant \epsilon$, and let $X_i \in \mathcal{X}_{x'} \subseteq \mathcal{X}_{x,\epsilon}$ be a top-dimensional stratum touching x'. Based on how x_i is defined in Equation (9), we have that x' and x_i both belong to $\overline{B}(x,\epsilon)$, and they both belong to the stratum X_i. Therefore, $|\nabla_{X_i} f(x') - \nabla_{X_i} f(x_i)| \leqslant 2L\epsilon$, and so $\nabla_{X_i} f(x') \in \tilde{\partial}_\epsilon f(x) + \overline{B}(0, 2L\epsilon)$. The result follows from the fact that $\tilde{\partial}_\epsilon f(x) + \overline{B}(0,2L\epsilon)$ is convex and closed. □

Recall from Equation (8) that the (opposite to the) descent direction $-g(x,\epsilon)$ is built as the projection of the origin onto $\partial_\epsilon f(x)$. Similarly, we define our approximate descent direction as $-\tilde{g}(x,\epsilon)$, where $\tilde{g}(x,\epsilon)$ is the projection of the origin onto the convex closed set $\tilde{\partial}_\epsilon f(x)$:

$$\tilde{g}(x,\epsilon) = \operatorname{argmin}\big\{\|\tilde{g}\|,\, \tilde{g} \in \tilde{\partial}_\epsilon f(x)\big\}. \tag{10}$$

We show that this choice yields a direction of decrease of f, in a sense similar to Equation (8).

Proposition 3. *Let f be stratifiably smooth, and let x be a non-stationary point. Let $0 < \beta < 1$, and $\epsilon_0 > 0$. Denote by L a Lipschitz constant for all gradients of the restrictions f_i to the ball $\overline{B}(x,\epsilon_0)$ (as in Proposition 2). Then:*

- **(i)** *For $0 < \epsilon \leqslant \epsilon_0$ small enough we have $\epsilon \leqslant \frac{1-\beta}{2L}\|\tilde{g}(x,\epsilon)\|$; and*
- **(ii)** *For such ϵ, we have $\forall t \leqslant \frac{\epsilon}{\|\tilde{g}(x,\epsilon)\|}$, $f(x - t\tilde{g}(x,\epsilon)) \leqslant f(x) - \beta t \|\tilde{g}(x,\epsilon)\|^2$.*

Proof. Saying that x is non-stationary is equivalent to the inequality $\|g(x,0)\| > 0$. We show that the map $\epsilon \in \mathbb{R}^+ \mapsto \|g(x,\epsilon)\| \in \mathbb{R}^+$, which is non-increasing, is continuous at 0^+. Let ϵ be small enough such that the sets of strata incident to x are the same that meet with the ϵ-ball around x, i.e., $\mathcal{X}_{x,\epsilon} = \mathcal{X}_x$. Such an ϵ exists since there are finitely many strata, which are closed sets, that meet with a sufficiently small neighborhood of x. Of course, all smaller values of ϵ enjoy the same property. By Proposition 1, we then have the nesting

$$\partial f(x) \subseteq \partial_\epsilon f(x) \subseteq \partial f(x) + \overline{B}(0, 2L\epsilon),$$

where L is a Lipschitz constant for the gradients in neighboring strata. In turn, $0 \leqslant \|g(x,0)\| - \|g(x,\epsilon)\| \leqslant 2L\epsilon$. In particular, $\|g(x,\epsilon)\| \to \|g(x,0)\| > 0$ as ϵ goes to 0, hence $\epsilon = o(\|g(x,\epsilon)\|)$. Besides, the inclusion $\tilde{\partial}_\epsilon f(x) \subseteq \partial_\epsilon f(x)$ (Proposition 2) implies that $\|\tilde{g}(x,\epsilon)\| \geqslant \|g(x,\epsilon)\| > 0$. This yields $\epsilon = o(\|\tilde{g}(x,\epsilon)\|)$ and so item **(i)** is proved.

We now assume that ϵ satisfies the inequality of item **(i)**, and let $0 \leqslant t \leqslant \frac{\epsilon}{\|\tilde{g}(x,\epsilon)\|}$. By the Lebourg mean value theorem, there exists a $y \in [x, x - t\tilde{g}(x,\epsilon)]$ and some $w \in \partial f(y)$ such that:

$$f(x - t\tilde{g}(x,\epsilon)) - f(x) = t\langle w, -\tilde{g}(x,\epsilon)\rangle.$$

Since $t \leqslant \frac{\epsilon}{\|\tilde{g}(x,\epsilon)\|}$, y is at distance no greater than ϵ from x. In particular, w belongs to $\partial_\epsilon f(x)$. From Proposition 2, there exists some $\tilde{w} \in \tilde{\partial}_\epsilon f(x)$ at distance no greater than $2L\epsilon$ from w. We then rewrite:

$$f(x - t\tilde{g}(x,\epsilon)) - f(x) = t\langle w - \tilde{w}, -\tilde{g}(x,\epsilon)\rangle + t\langle \tilde{w}, -\tilde{g}(x,\epsilon)\rangle. \tag{11}$$

On the one hand, by the Cauchy-Schwarz inequality:

$$\langle w - \tilde{w}, -\tilde{g}(x,\epsilon)\rangle \leqslant |w - \tilde{w}| \cdot \|\tilde{g}(x,\epsilon)\| \leqslant 2L\epsilon\|\tilde{g}(x,\epsilon)\| \leqslant (1-\beta)\|\tilde{g}(x,\epsilon)\|^2, \tag{12}$$

where the last inequality relies on the assumption that $\epsilon \leqslant \frac{1-\beta}{2L}\|\tilde{g}(x,\epsilon)\|$. On the other hand, since $\tilde{g}(x,\epsilon)$ is the projection of the origin onto $\tilde{\partial}_\epsilon f(x)$, we obtain $\langle \tilde{g}(x,\epsilon) - \tilde{w}, \tilde{g}(x,\epsilon)\rangle \leqslant 0$, or equivalently:

$$\langle \tilde{w}, -\tilde{g}(x,\epsilon)\rangle \leqslant -\|\tilde{g}(x,\epsilon)\|^2. \tag{13}$$

Plugging the inequalities of Equations (12) and (13) into Equation (11) proves item **(ii)**. □

3 Stratified Gradient Sampling (SGS)

In this section we develop a gradient descent algorithm for the optimization of stratifiably smooth functions, and then we detail its convergence properties. We require that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ has the following properties:

- **(Proper)**: *f has compact sublevel sets.*
- **(Stratifiably smooth)**: *f is stratifiably smooth, and for each iterate x and $\epsilon \geqslant 0$ we have an oracle* **SampleOracle**(x, ϵ) *that samples one ϵ-close element x' in each ϵ-close top-dimensional stratum X'.*
- **(Differentiability check)** *We have an oracle* **DiffOracle**(x) *checking whether an iterate $x \in \mathbb{R}^n$ belongs to the set $\mathcal{D} \subset \mathbb{R}^n$ over which f is differentiable.*

That f is a proper map is also needed in the original GS algorithm [16], but is a condition that can be omitted as in [52] to allow the values $f(x_k)$ to decrease to $-\infty$. In our case we stick to this assumption because we need the gradient of f (whenever defined) to be Lipschitz on sublevel sets.

Similarly, the ability to check that an iterate x_k belongs to $\mathcal{D}$ is standard in the GS methodology. We use it to make sure that f is differentiable at each iterate x_k. For this, we call a subroutine **MakeDifferentiable** which slightly perturbs the iterate x_k to achieve differentiability and to maintain a descent condition. Note that these considerations are mainly theoretical because generically the iterates x_k are points of differentiability, hence **MakeDifferentiable** is unlikely to change anything.

The last requirement that f is stratifiably smooth replaces the classical weaker assumption used in the GS algorithm that f is locally Lipschitz and that the set $\mathcal{D}$ where f is differentiable is open and dense. There are many possible ways to design the oracle **SampleOracle**(x, ϵ): for instance the sampling could depend upon arbitrary probability measures on each stratum, or it could be set by deterministic rules depending on the input (x, ϵ) as will be the case for the persistence map in Section 4. However our algorithm and its convergence properties are oblivious to these degrees of freedom, as by Section 2.3 any sampling allows us to approximate the Goldstein subgradient $\partial_\epsilon f(x_k)$ using finitely many neighbouring points to compute $\tilde{\partial}_\epsilon f(x_k)$. In turn we have an approximate descent direction g_k which can be used to produce the subsequent iterate $x_{k+1} := x_k - t_k g_k$ as in the classical smooth gradient descent.

3.1 The algorithm

The details of the main algorithm **SGS** are given in Algorithm 2.

The algorithm **SGS** calls the method **UpdateStep** of Algorithm 3 as a subroutine to compute the right descent direction g_k and the right step size t_k. Essentially, this method progressively reduces the exploration radius ϵ_k of the ball where we compute the descent direction $g_k := \tilde{g}(x_k, \epsilon_k)$ until the criteria of Proposition 3 ensuring that the loss sufficiently decreases along g_k are met.

Given the iterate x_k and the radius ϵ_k, the calculation of $g_k := \tilde{g}(x_k, \epsilon_k)$ is done by the subroutine **ApproxGradient** in Algorithm 4: points x' in neighboring strata that intersect the ball $B(x_k, \epsilon_k)$ are sampled using **SampleOracle**(x_k, ϵ_k) to compute the approximate Goldstein gradient and in turn the descent direction g_k.

Much like the classical GS algorithm, our method behaves like the well-known smooth gradient descent where the gradient is replaced with a descent direction computed from gradients in neighboring strata. A key difference however is that, in order to find the right exploration radius ϵ_k and step size t_k, the **UpdateStep** needs to maintain a constant C_k to approximate the ratio $\frac{1-\beta}{2L}$ of Proposition 3, as no Lipschitz constant L may be explicitly available.

To this effect, **UpdateStep** maintains a relative balance between the exploration radius ϵ_k and the norm of the descent direction g_k, controlled by C_k, i.e., $\epsilon_k \simeq C_k \|g_k\|$. As we further maintain $C_k \simeq \frac{1-\beta}{2L}$, we know that the convergence properties of ϵ_k and g_k are closely related. Thus, the utility of this controlling

constant is mainly theoretical, to ensure convergence of the iterates x_k towards stationary points in Theorem 2. In practice, we start with a large initial constant C_0, and decrease it on line 11 of Algorithm 3 whenever it violates a property of the target constant $\frac{1-\beta}{2L}$ given by Proposition 3.

Algorithm 2 SGS$(f, x_0, \epsilon, \eta, C_0, \beta, \gamma)$

Require: Loss function f, initial iterate $x_0 \in \mathcal{D}$, exploration radius $\epsilon > 0$, initial constant $C_0 > 0$ controlling exploration radius, critical distance to origin $\eta \geqslant 0$, descent rate $0 < \beta < 1$, step size decay rate $0 < \gamma < 1$

1: $k \leftarrow 0$
2: **repeat**
3: $(t_k, g_k, C_{k+1}) \leftarrow$ **UpdateStep**$(f, x_k, \epsilon, \eta, C_k, \beta, \gamma)$ via Algorithm 3
4: $x_{k+1} \leftarrow x_k - t_k g_k$
5: $x_{k+1} \leftarrow$ **MakeDifferentiable**(x_{k+1}, x_k, t_k, g_k)
6: $k \leftarrow k + 1$
7: **until** $\|g_k\| \leqslant \eta$
8: **return** x_k

Algorithm 3 UpdateStep$(f, x_k, \epsilon, \eta, C_k, \beta, \gamma)$

1: $\epsilon_k \leftarrow \epsilon$ and $C_{k+1} \leftarrow C_k$
2: **repeat**
3: $g_k \leftarrow$ **ApproxGradient**(x_k, ϵ_k) via Algorithm 4
4: **if** $\|g_k\| \leqslant \eta$ **then**
5: Break, **return** $t_k = 0$, g_k and C_{k+1} *# Set $\eta = 0$ to reach an ϵ-stationary point*
6: **end if**
7: $t_k \leftarrow \frac{\epsilon_k}{\|g_k\|}$ *# Candidate of update step*
8: **while** $f(x_k - t_k g_k) > f(x_k) - \beta t_k \|g_k\|^2$ and $\epsilon_k \leqslant C_{k+1}\|g_k\|$ **do**
9: $C_{k+1} \leftarrow \gamma C_{k+1}$ *# Once $C_{k+1} \leqslant \frac{1-\beta}{2L}$, this loop never occurs by (ii) of Proposition 3*
10: **end while**
11: **if** $f(x_k - t_k g_k) > f(x_k) - \beta t_k \|g_k\|^2$ or $\epsilon_k > C_{k+1}\|g_k\|$ **then**
12: $\epsilon_k \leftarrow \gamma \epsilon_k$ *# Reduce ϵ_k to satisfy criterion (i) of Proposition 3*
13: **end if**
14: **until** $f(x_k - t_k g_k) < f(x_k) - \beta t_k \|g_k\|^2$ and $\epsilon_k < C_{k+1}\|g_k\|$
15: **return** t_k, g_k and C_{k+1}

Remark 2. Assume that we dispose of a common Lipschitz constant L for all gradients ∇f_i in the ϵ-neighborhood of the current iterate x_k, recall that f_i is any C^2 extension of the restriction $f_{|X_i}$ to the neighboring top-dimensional stratum $X_i \in \mathcal{X}_{x,\epsilon}$. Then we can simplify Algorithm 3 by decreasing the exploration radius ϵ_k progressively until $\epsilon_k \leqslant \frac{(1-\beta)}{2L}\|\tilde{g}(x_k, \epsilon_k)\|$ as done in Algorithm 6: This ensures by Proposition 3 that the resulting update step satisfies the descent criterion $f(x_k - t_k \tilde{g}(x_k, \epsilon_k)) < f(x_k) - \beta t_k \|\tilde{g}(x_k, \epsilon_k)\|^2$. In particular the parameter C_k is no longer needed, and the theoretical guarantees of the simplified algorithm are unchanged. Note that for objective functions from TDA (see Section 4), the stability theorems (e.g. from [27]) often provide global Lipschitz constants, enabling the use of the simplified update step described in Algorithm 6.

Remark 3. In the situation of Remark 2, let us further assume that $\epsilon \in \mathbb{R}_+ \mapsto \|\tilde{g}(x, \epsilon)\| \in \mathbb{R}_+$ is non-increasing. This monotonicity property mimics the fact that $\epsilon \in \mathbb{R}_+ \mapsto \|g(x, \epsilon)\| \in \mathbb{R}_+$ is non-increasing, since increasing ϵ grows the Goldstein generalized gradient $\partial_\epsilon f(x)$, of which $g(x, \epsilon)$ is the element with minimal norm. If the initial exploration radius ϵ does not satisfy the termination criterion (Line 8), $\epsilon \leqslant \frac{1-\beta}{2L}\|\tilde{g}(x_k, \epsilon)\|$,

Algorithm 4 **ApproxGradient**(x_k, ϵ_k)

$G_k \leftarrow \{\nabla f(x_k)\}$ *# Eventually G_k will be some approximate Goldstein subgradient $\tilde{\partial} f_{\epsilon_k}(x_k)$*
$\{x_k^1, \cdots, x_k^m\} \leftarrow$ **SampleOracle**(x_k, ϵ_k) *# ϵ-close samples from ϵ-close top dim strata*
for $1 \leqslant l \leqslant m$ **do**
  $G_k \leftarrow G_k \cup \{\nabla f(x_k^l)\}$ *# Add gradients from remote strata*
end for
Solve the quadratic minimization problem $g_k = \text{argmin}\{\|g\|^2,\ g \in \overline{\text{co}}(G_k)\}$
return g_k *# $g_k = \tilde{g}(x_k, \epsilon_k)$ is the approximate steepest descent direction*

Algorithm 5 **MakeDifferentiable**(x_{k+1}, x_k, t_k, g_k)

$r \leftarrow t_k \|g_k\|$
while $x_{k+1} \notin \mathcal{D}$ or $f(x_{k+1}) > f(x_k) - \beta t_k \|g_k\|^2$ **do**
  Replace x_{k+1} with a sample in $B(x_k - t_k g_k, r)$
  $r \leftarrow \frac{r}{2}$
end while
return x_{k+1}

Algorithm 6 **SimpleUpdateStep**$(f, x_k, \epsilon, \eta, \beta, \gamma)$

$\epsilon_k \leftarrow \epsilon$ and $g_k \leftarrow$ **ApproxGradient**(x_k, ϵ_k) via Algorithm 4
repeat
  if $\|g_k\| \leqslant \eta$ **then**
    Break, **return** $t_k = 0$ and g_k *# Set $\eta = 0$ to reach an ϵ-stationary point*
  end if
  $\epsilon_k \leftarrow \gamma \epsilon_k$ and $g_k \leftarrow$ **ApproxGradient**(x_k, ϵ_k) via Algorithm 4
until $\epsilon_k \leqslant \frac{1-\beta}{2L} \|g_k\|$
return $t_k := \frac{\epsilon_k}{\|g_k\|}$ and g_k

then one can set $\epsilon_k := \frac{1-\beta}{2L}\|\tilde{g}(x_k, \epsilon)\| \leqslant \epsilon$, yielding $\epsilon_k \leqslant \frac{1-\beta}{2L}\|\tilde{g}(x_k, \epsilon_k)\|$. This way Algorithm 6 is further simplified: in constant time we find a ϵ_k that yields the descent criterion $f(x_k - t_k\tilde{g}(x_k, \epsilon_k)) < f(x_k) - \beta t_k\|\tilde{g}(x_k, \epsilon_k)\|^2$, and the parameter γ is no longer needed. A careful reading of the proofs provided in the following section yields that the convergence rate (Theorem 3) of the resulting algorithm is unchanged, however the asymptotic convergence (Theorem 2), case (**b**), needs to be weakened: converging subsequences converge to ϵ-stationary points instead of stationary points. We omit details for the sake of concision.

3.2 Convergence

We show convergence of Algorithm 2 towards stationary points in Theorem 2. Finally, Theorem 3 provides a non-asymptotic sub linear convergence rate, which is by no mean tight yet gives a first estimate of the number of iterations required in order to reach an approximate stationary point.

Theorem 2. *If $\eta = 0$, then almost surely the algorithm either* (**a**) *converges in finitely many iterations to an ϵ-stationary point, or* (**b**) *produces a bounded sequence of iterates $(x_k)_k$ whose converging subsequences all converge to stationary points.*

As an intermediate result, we first show that the update step computed in Algorithm 3 is obtained after finitely many iterations and estimate its magnitude relatively to the norm of the descent direction.

Lemma 1. UpdateStep$(f, x_k, \epsilon, \eta, C_k, \beta, \gamma)$ *terminates in finitely many iterations. In addition, let L be a Lipschitz constant for the restricted gradients ∇f_i (as in Proposition 2) in the ϵ-neighborhood of x_k. Assume that $\frac{1-\beta}{2L} \leqslant C_k$. If x_k is not an (ϵ, η)-stationary point, then the returned exploration radius ϵ_k satisfies:*

$$\min\left(\frac{\gamma^2(1-\beta)}{2L}\|\tilde{g}(x_k, \frac{1}{\gamma}\epsilon_k)\|, \epsilon\right) \leqslant \epsilon_k \leqslant \min(C_k\|\tilde{g}(x_k, \epsilon_k)\|, \epsilon).$$

Moreover the returned controlling constant C_{k+1} satisfies:

$$C_{k+1} \geqslant \frac{\gamma(1-\beta)}{2L}.$$

Proof. If x_k is a stationary point, then **ApproxGradient** returns a trivial descent direction $g_k = 0$ because the approximate gradient G_k contains $\nabla f(x_k)$ (Line 1). In turn, **UpdateStep** terminates at Line 4.

Henceforth we assume that x_k is not a stationary point and that the breaking condition of Line 4 in Algorithm 3 is never reached (otherwise the algorithm terminates). Therefore, at each iteration of the main loop, either C_{k+1} is replaced by γC_{k+1} (line 9), or ϵ_k is replaced by $\gamma\epsilon_k$ (line 12), until both the following inequalities hold (line 14):

$$\textbf{(A)} \quad \epsilon_k < C_{k+1}\|\tilde{g}(x_k, \epsilon_k)| \quad \text{and}$$

$$\textbf{(B)} \quad f(x_k - t_k\tilde{g}(x_k, \epsilon_k)) < f(x_k) - \beta t_k\|\tilde{g}(x_k, \epsilon_k)\|^2.$$

Once C_{k+1} becomes lower than $\frac{1-\beta}{2L}$, inequality **(A)** implies inequality **(B)** by Proposition 3 (**ii**). It then takes finitely many replacements $\epsilon_k \leftarrow \gamma\epsilon_k$ to reach inequality **(A)**, by Proposition 3 **(i)**. At that point (or sooner), Algorithm 3 terminates. This concludes the first part of the statement, namely **UpdateStep** terminates in finitely many iterations.

Next we assume that x_k is not an (ϵ, η)-stationary point, which ensures that the main loop of **UpdateStep** cannot break at Line 5. We have the invariant $C_{k+1} \geqslant \gamma\frac{1-\beta}{2L}$: this is true at initialization ($C_{k+1} = C_k$) by assumption, and in later iterations C_{k+1} is only replaced by γC_{k+1} whenever **(A)** holds but not **(B)**, which forces $C_{k+1} \geqslant \frac{1-\beta}{2L}$ by Proposition 3 **(ii)**.

At the end of the algorithm, $\epsilon_k \leqslant C_{k+1}\|\tilde{g}(x_k, \epsilon_k)\|$ by inequality **(A)**, and so we deduce the right inequality $\epsilon_k \leqslant \min(C_k\|\tilde{g}(x_k, \epsilon_k)\|, \epsilon)$.

Besides, if both **(A)** and **(B)** hold when entering the main loop (line 11) for the first time, then $\epsilon_k = \epsilon$. Otherwise, let us consider the penultimate iteration of the main loop for which the update step is $\frac{1}{\gamma}\epsilon_k$. Then,

either condition **(A)** does not hold, namely $\frac{1}{\gamma}\epsilon_k > C_{k+1}\|\tilde{g}(x_k, \frac{1}{\gamma}\epsilon_k)\| \geqslant \gamma\frac{1-\beta}{2L}\|\tilde{g}(x_k, \frac{1}{\gamma}\epsilon_k)\|$, or condition **(B)** does not hold, which by the assertion **(ii)** of Proposition 3 implies $\frac{1}{\gamma}\epsilon_k \geqslant \frac{1-\beta}{2L}\|\tilde{g}(x_k, \frac{1}{\gamma}\epsilon_k)\|$. In any case, we deduce that

$$\epsilon_k \geqslant \min\left(\frac{\gamma^2(1-\beta)}{2L}\left\|\tilde{g}\left(x_k, \frac{1}{\gamma}\epsilon_k\right)\right\|, \epsilon\right).$$

□

Proof of Theorem 2. We assume that alternative (**a**) does not happen. By Lemma 1, Algorithm 3 terminates in finitely many iteration and by Line 14 we have the guarantee:

$$\forall k \geqslant 0,\ f(x_k - t_k\tilde{g}(x_k, \epsilon_k)) < f(x_k) - \beta t_k\|\tilde{g}(x_k, \epsilon_k)\|^2. \tag{14}$$

The subsequent iterate x_{k+1} is initialized at $x_k - t_k\tilde{g}(x_k, \epsilon_k)$ by **MakeDifferentiable** (see Algorithm 5) and replaced by a sample in a progressively shrinking ball $B(x_k - t_k\tilde{g}(x_k, \epsilon_k), r)$ until two conditions are reached. The first condition is that f is differentiable at x_{k+1}, and since $\mathcal{D}$ has full measure by Rademacher's Theorem, this requirement is almost surely satisfied in finitely many iterations. The second condition is that

$$\forall k \geqslant 0,\ f(x_{k+1}) < f(x_k) - \beta t_k\|\tilde{g}(x_k, \epsilon_k)\|^2, \tag{15}$$

which by Equation (14) and continuity of f is satisfied in finitely many iterations as well. Therefore **MakeDifferentiable** terminates in finitely many iterations almost surely. In particular, the sequence of iterates $(x_k)_k$ is infinite.

By Equation (15) the sequence of iterates' values $(f(x_k))_k$ is decreasing and it must converge by compactness of the sublevel sets below f. Using Equation (15), we obtain:

$$\epsilon_k\|\tilde{g}(x_k, \epsilon_k)\| = t_k\|\tilde{g}(x_k, \epsilon_k)\|^2 \leqslant \frac{1}{\beta}(f(x_k) - f(x_{k+1})) \longrightarrow 0^+, \tag{16}$$

so that in particular, either $\epsilon_k \to 0$ or $\|\tilde{g}(x_k, \epsilon_k)\| \to 0$. Let also L be Lipschitz constant for the restricted gradients ∇f_i (as in Proposition 2) in the ϵ-offset of the sublevel set $\{x, f(x) \leqslant f(x_0)\}$. Up to taking L large enough, there is another Lipschitz constant $L' < L$ ensuring that

$$\frac{1}{\gamma}\frac{1-\beta}{2L} \leqslant \frac{1-\beta}{2L'} \leqslant C_0.$$

By Lemma 1, upon termination of Algorithm 3, $C_1 \geqslant \gamma\frac{1-\beta}{2L'} \geqslant \frac{1-\beta}{2L}$. If $C_1 \leqslant \frac{1-\beta}{2L'}$, the **(ii)** of Proposition 3 prevents Line 9 in Algorithm 3 from ever occurring again, i.e., $C_k = C_1$ is constant in later iterations. Otherwise, C_1 satisfies $C_1 \geqslant \frac{1-\beta}{2L'}$ just like C_0. A quick induction then yields:

$$\forall k \geqslant 0,\ C_k \geqslant \frac{1-\beta}{2L}.$$

Therefore, by Lemma 1:

$$\forall k \geqslant 0,\ \min\left(\frac{\gamma^2(1-\beta)}{2L}\left\|\tilde{g}\left(x_k, \frac{1}{\gamma}\epsilon_k\right)\right\|, \epsilon\right) \leqslant \epsilon_k \leqslant \min(C_0\|\tilde{g}(x_k, \epsilon_k)\|, \epsilon). \tag{17}$$

In particular, using the rightmost inequality and Equation (16), we get $\epsilon_k \to 0^+$. In turn, using the leftmost inequality, we get that

$$\left\|\tilde{g}\left(x_k, \frac{1}{\gamma}\epsilon_k\right)\right\| \to 0^+. \tag{18}$$

The sequence of iterates $(x_k)_k$ is bounded; up to extracting a converging subsequence, we assume that it converges to some x_*. Let $\epsilon' > 0$. We claim that $0 \in \partial_{\epsilon'} f(x_*)$, that is x_* is ϵ'-stationary. As $x_k \to x_*$ and $\epsilon_k \to 0$, we have that for k large enough $B(x_k, \frac{1}{\gamma}\epsilon_k) \subseteq B(x_*, \epsilon')$, which implies that:

$$\partial_{\frac{1}{\gamma}\epsilon_k} f(x_k) \subseteq \partial_{\epsilon'} f(x_*).$$

Besides, from Proposition 2, we have $\tilde{\partial}_{\frac{1}{\gamma}\epsilon_k} f(x_k) \subseteq \partial_{\frac{1}{\gamma}\epsilon_k} f(x_k)$, so that $\tilde{g}(x_k, \frac{1}{\gamma}\epsilon_k) \in \partial_{\frac{1}{\gamma}\epsilon_k} f(x_k)$. Hence $\tilde{g}(x_k, \frac{1}{\gamma}\epsilon_k) \in \partial_{\epsilon'} f(x_*)$. In the limit, Equation (18) implies $0 \in \partial_{\epsilon'} f(x_*)$, as desired.

Following [16], the intersection of the Goldstein subgradients $\partial_{\epsilon'} f(x_*)$, over $\epsilon' > 0$, yields $\partial f(x_*)$. Hence, $0 \in \partial f(x_*)$ and x_* is a stationary point. □

Theorem 3. *If $\eta > 0$, then Algorithm 2 produces an (ϵ, η)-stationary point using at most $O\left(\frac{1}{\eta \min(\eta, \epsilon)}\right)$ iterations.*

Proof. Assume that Algorithm 2 has run over k iterations without producing an (ϵ, η)-stationary point. From Algorithm 3 (line 14), Algorithm 5 (Line 2) and the choice $t_j = \frac{\epsilon_j}{\|\tilde{g}(x_j, \epsilon_j)\|}$ of update step for $j \leqslant k$, it holds that $\beta \epsilon_j \|\tilde{g}(x_j, \epsilon_j)\| \leqslant f(x_j) - f(x_{j+1})$, and in turn

$$\sum_{j=0}^{k-1} \epsilon_j \|\tilde{g}(x_j, \epsilon_j)\| \leqslant \frac{f_0 - f^*}{\beta},$$

where $f_0 := f(x_0)$ and f^* is a minimal value of f. Besides, using Lemma 1, ϵ_j is either bigger than $\frac{\gamma^2(1-\beta)}{2L} \left\|\tilde{g}\left(x_j, \frac{1}{\gamma}\epsilon_j\right)\right\|$ or than ϵ, hence

$$\sum_{j=0}^{k-1} \epsilon_j \|\tilde{g}(x_j, \epsilon_j)\| \geqslant k \min_{j<k} \|\tilde{g}(x_j, \epsilon_j)\| \times \min_{j<k} \epsilon_j > k \times \eta \times \min\left(\epsilon, \frac{\gamma^2(1-\beta)}{2L}\eta\right).$$

The two equations cannot simultaneously hold whenever

$$k \geqslant \frac{f_0 - f^*}{\beta} \times \frac{1}{\eta \min\left(\epsilon, \frac{\gamma^2(1-\beta)}{2L}\eta\right)},$$

which allows us to conclude. □

3.3 Approximate distance to strata

The algorithm and its convergence assume that strata X that are ϵ-close to an iterate x can be detected by the oracle **SampleOracle**(x, ϵ). However in practice computing distances $d(x, X)$ to sub manifolds may be expansive or even impossible. Instead we can hope for approximate distances $\hat{d}(x, X)$. Typically when we have an assignment

$$(x, X) \in \mathbb{R}^n \times \mathcal{X} \longmapsto \tilde{x}_X \in X \subseteq \mathbb{R}^n, \text{ at our disposal, we can define } \hat{d}(x, X) := d(x, \tilde{x}_X),$$

and replace the accurate oracle **SampleOracle**(x, ϵ) with the following approximate oracle:

$$\textbf{ApproxSampleOracle}(x, \epsilon) := \left\{\tilde{x}_X \,|\, X \in \mathcal{X}, d(x, \tilde{x}_X) \leqslant \epsilon\right\} = \left\{\tilde{x}_X \,|\, X \in \mathcal{X}, \hat{d}(x, X) \leqslant \epsilon\right\}.$$

Therefore for the purpose of this section we make the following assumption: *To every iterate $x \in \mathbb{R}^n$ and stratum X we can associate an element $\tilde{x}_X$ that belongs to X, in particular we have the corresponding oracle* **ApproxSampleOracle**. *Moreover there exists a constant $a \geqslant 1$ such that the resulting approximate distance to strata $\hat{d}(x, X) := d(x, \tilde{x}_X)$ satisfies:*

$$\forall x \in \mathbb{R}^n, \forall X \in \mathcal{X},\ \hat{d}(x, X) \leqslant a d(x, X).$$

Note that we always have a reverse inequality $\hat{d}(x, X) \geqslant d(x, X)$ since $\tilde{x}_X \in X$. In the case of the persistence map this will specialize to $d(x, X) \leqslant \hat{d}(x, X) \leqslant 2d(x, X)$, that is $a = 2$, see Proposition 7.

We then replace the approximate Goldstein subgradient $\tilde{\partial}_\epsilon f(x)$ with $\hat{\partial}_\epsilon f(x)$, defined in the exact same way except that it is computed from strata satisfying $\hat{d}(x, X) \leqslant \epsilon$, that is, $\hat{\partial}_\epsilon f(x)$ contains $\nabla f(\tilde{x}_X)$ for each such stratum. The proof of Proposition 2 adapts straightforwardly to the following statement:

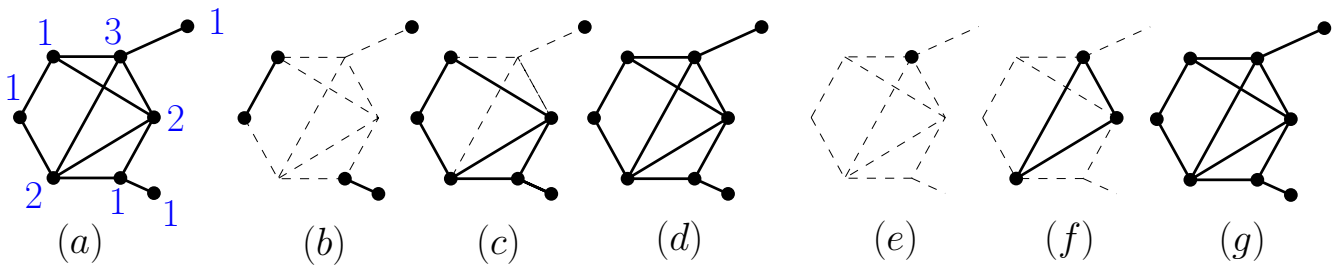

Figure 2: Sublevel sets and superlevel sets filtrations illustrated on graphs. (a) Input graph (V,E) along with the values of a function $x : V \to \mathbb{R}$ (blue). (b, c, d) Sublevel sets for t = 1,2,3 respectively. (e, f, g) Superlevel sets for t = 3,2,1 respectively.

Proposition 4. *Let $x \in \mathbb{R}^n$ and $\epsilon > 0$. Assume that f is stratifiably smooth. Let L be a Lipschitz constant of the gradients ∇f_i restricted to $\overline{B}(x, a\epsilon) \cap X_i$, where f_i is some local C^2 extension of $f_{|X_i}$, and $X_i \in \mathcal{X}_{x,\epsilon}$ is top dimensional. Then we have:*

$$\hat{\partial}_\epsilon f(x) \subseteq \partial_\epsilon f(x) \quad \text{and} \quad \partial_\epsilon f(x) \subseteq \hat{\partial}_{a\epsilon} f(x) + \overline{B}(0, (a+1)L\epsilon).$$

Proof. The inclusion $\hat{\partial}_\epsilon f(x) \subseteq \partial_\epsilon f(x)$ is clear. Conversely, let $\nabla_X f(x') \in \partial_\epsilon f(x)$, where X is a top-dimensional stratum incident to x' and $|x' - x| \leqslant \epsilon$. We then have $\hat{d}(x, X) \leqslant a\epsilon$ and hence $\tilde{x}_X$ is a point in $\hat{\partial}_{a\epsilon} f(x)$ which is $(a+1)\epsilon$-close to x'. Therefore $\nabla_X f(x') \in \hat{\partial}_{a\epsilon} f(x) + \overline{B}(0, (a+1)L\epsilon)$. The rest of the proof is then conducted as in Proposition 2. □

The vector $\hat{g}(x, \epsilon)$ with minimal norm in $\hat{\partial}_\epsilon f(x)$ can then serve as the new descent direction in place of $\tilde{g}(x, \epsilon)$:

Proposition 5. *Let f be stratifiably smooth, and x be a non stationnary point. Let $0 < \beta < 1$, and $\epsilon_0 > 0$. Denote by L a Lipschitz constant for all gradients of the restriction f_i in the ball $\overline{B}(x, a\epsilon_0)$ (as in Proposition 4).*

- **(i)** *For $0 < \epsilon \leqslant \epsilon_0$ small enough we have $\epsilon \leqslant \frac{1-\beta}{2L}\|\hat{g}(x, \epsilon)\|$; and*
- **(ii)** *For such ϵ, we have $\forall t \leqslant \frac{\epsilon}{a\|\hat{g}(x,\epsilon)\|}$, $f(x - t\hat{g}(x, \epsilon)) \leqslant f(x) - \beta t\|\hat{g}(x, \epsilon)\|^2$.*

Proof. The proof of Proposition 3 can be replicated by replacing ϵ with $\frac{\epsilon}{a}$ and using Proposition 4 instead of Proposition 2. □

In light of this result, we can use $g_k = \hat{g}(x_k, \epsilon_k)$ as a descent direction, which in practice simply amounts to replace the accurate oracle **SampleOracle**(x_k, ϵ_k) in Algorithm 4 with the approximate oracle **ApproxSampleOracle**(x_k, ϵ_k). The only difference is that the assignment of update step in Algorithm 3 (Line 7) should take the constant a into account, namely:

$$\text{(Line 7')} \qquad t_k \leftarrow \frac{\epsilon}{a\|g_k\|}.$$

The convergence analysis of Section 3.2 holds as well for this algorithm, and the proofs of Theorem 2 and Theorem 3 are unchanged.

4 Application to Topological Data Analysis

In this section, we define the persistence map PH : $\mathbb{R}^n \to$ **Bar** which is a central descriptor in TDA that gives rise to prototypical stratifiably smooth objective functions f in this work. We refer the reader to [36, 62, 76] for full treatments of the theory of Persistence. We then introduce the stratification that

makes PH a stratifiably smooth map, by means of the permutation group. Using the associated Cayley graph we give a way to implement the oracle **SampleOracle**(x, ϵ) that samples points in nearby top dimensional strata, which is the key ingredient of Algorithm 4 for computing descent directions.

4.1 The Persistence Map

Persistent Homology and Barcodes Let $n \in \mathbb{N}$, and let $\{v_1, \dots, v_n\}$ be a (finite) set of vertices. A *simplicial complex* K is a subset of the power set $\mathcal{P}(\{v_1, \dots, v_n\})$ whose elements are called *simplices*, and which is closed under inclusion: if $\sigma \in K$ is a simplex and $\sigma' \subseteq \sigma$, then $\sigma' \in K$. The *dimension* of the complex is the maximal cardinality of its simplices minus one. In particular a 1-dimensional simplicial complex is simply an undirected graph.

A *filter function* is a function on the vertices of K, which we equivalently view as a vector $x \in \mathbb{R}^n$. Given $t \in \mathbb{R}$, we have the sub complexes $K_{\leqslant t} = \{\sigma \in K,\ \forall v \in \sigma,\ x(v) \leqslant t\}$. This yields a nested sequence of sub complexes called the *sublevel sets* filtration of K by x:

$$\varnothing \longrightarrow \cdots \longrightarrow K_{\leqslant s} \xrightarrow{s \leqslant t} K_{\leqslant t} \longrightarrow \cdots \longrightarrow K, \tag{19}$$

See Figure 2 for an illustration on graphs. The *(Ordinary) Persistent Homology* of x in degree $p \in \{0, \cdots, \dim K\}$ records topological changes in Equation (19) by means of points $(b, d) \in \mathbb{R}^2$, here $b < d$, called *intervals*. For instance, in degree $p = 0$, an interval (b, d) corresponds to a connected component that appears in $K_{\leqslant b}$ and that is merged with an older component in $K_{\leqslant d}$. In degree $p = 1$ and $p = 2$, intervals track loops and cavities respectively, and more generally an interval (b, d) in degree p tracks a p-dimensional sphere that appears in $K_{\leqslant b}$ and persists up to $K_{\leqslant d}$.

Note that there are possibly infinite intervals (b, ∞) for p-dimensional cycles that persist forever in the filtration Equation (19). Such intervals are not easy to handle in applications, and it is common to consider the *(Extended) Persistent Homology* of x, for which they do not occur, i.e. we append the following sequence of pairs involving superlevel sets $K_{\geqslant t} := \{\sigma \in K |\ \forall v \in \sigma,\ x(v) \geqslant t\}$ to Equation (19):

$$K \cong (K, \varnothing) \longrightarrow \cdots \longrightarrow (K, K_{\geqslant s}) \xrightarrow{s \geqslant t} (K, K_{\geqslant t}) \longrightarrow \cdots \longrightarrow (K, K). \tag{20}$$

Together intervals (b, d) form the (extended) *barcode* $\mathrm{PH}_p(x)$ of x in degree p, which we simply denote by $\mathrm{PH}(x)$ when the degree is clear from the context.

Definition 3. A *barcode* is a finite multi-set of pairs $(b, d) \in \mathbb{R}^2$ called *intervals*, with $b \leqslant d$. Two barcodes differing by intervals of the form (b, b) are identified. We denote by **Bar** the set of barcodes.

The set **Bar** of barcodes can be made into a metric space as follows. Given two barcodes $D := \{(b, d)\}$ and $D' := \{(b', d')\}$, a *partial matching* $\gamma : D \to D'$ is a bijective map from some subset $A \subseteq D$ to some $B \subseteq D'$. For $q \geqslant 1$ the *q-th diagram distance* $W_q(D, D')$ is the following cost of transferring intervals (b, d) to intervals (b', d'), minimized over partial matchings γ between D and D':

$$\begin{aligned} W_q(D, D') := \inf_{\gamma} \Bigg(& \sum_{(b,d) \in A} \|\gamma(b, d) - (b, d)\|_2^q \\ & + \sum_{(b,d) \in D \setminus A} \left(\frac{d - b}{\sqrt{2}} \right)^q + \sum_{(b',d') \in D' \setminus B} \left(\frac{d' - b'}{\sqrt{2}} \right)^q \Bigg)^{\frac{1}{q}}. \end{aligned} \tag{21}$$

In particular the intervals that are not in the domain A and image B of γ contribute to the total cost relative to their distances to the diagonal $\{b = d\} \subset \mathbb{R}^2$.

The Stability Theorem [27, 28] implies that the map $\mathrm{PH} : \mathbb{R}^n \to$ **Bar**, which we refer to as the *persistence map* in what follows, is Lipschitz continuous.

Differentiability of Persistent Homology Next we recall from [53] the notions of differentiability for maps in and out of **Bar** and the differentiability properties of PH. Note that the results of [53] focus on ordinary persistence, yet they easily adapt to extended persistence, see e.g. [75].

Given $r \in \mathbb{N}$, we define a *local $\mathcal{C}^r$-coordinate system* as a collection of C^r real-valued maps coming in pairs $b_i, d_i : U \to \mathbb{R}$ defined on some open Euclidean set $U \subseteq \mathbb{R}^n$, indexed by a finite set I, and satisfying $b_i(x) \leqslant d_i(x)$ for all $x \in U$ and $i \in I$. A local $\mathcal{C}^r$-coordinate system is thus equally represented as a map valued in barcodes

$$\tilde{B} : x \in U \mapsto \left\{b_i(x), d_i(x)\right\}_{i \in I} \in \mathbf{Bar},$$

where each interval $(b_i(x), d_i(x))$ is identified and tracked in a $\mathcal{C}^r$ manner.

Definition 4. A map $B : \mathbb{R}^n \to \mathbf{Bar}$ is *r-differentiable* at $x \in \mathbb{R}^n$ if $B_{|U} = \tilde{B}_{|U}$ for some local $\mathcal{C}^r$-coordinate system $\tilde{B}$ defined in a neighborhood U of x.

Similarly,

Definition 5. A map $V : \mathbf{Bar} \to \mathbb{R}^m$ is *r-differentiable* at $D \in \mathbf{Bar}$ if $V \circ \tilde{B} : \mathbb{R}^n \to \mathbb{R}^m$ is of class $\mathcal{C}^r$ in a neighborhood of the origin for all $n \in \mathbb{N}$ and local $\mathcal{C}^r$-coordinate system $\tilde{B}$ defined around the origin such that $\tilde{B}(0) = D$.

These notions compose together via the chain rule, so for instance an objective function $f = V \circ B : \mathbb{R}^n \to \mathbb{R}^m$ is differentiable in the usual sense as soon as B and V are so.

We now define the stratification $\mathcal{X}$ of $\mathbb{R}^n$ such that the persistence map $B = \mathrm{PH}$ is r-differentiable (for any r) over each stratum. Denote by Σ_n the group of permutations on $\{1, \cdots, n\}$. Each permutation $\pi \in \Sigma_n$ gives rise to a closed polyhedron

$$\mathcal{S}_\pi := \left\{ x \in \mathbb{R}^n \,|\, \forall 1 \leqslant i < n,\, x_{\pi(i)} \leqslant x_{\pi(i+1)} \right\}, \tag{22}$$

which is a *cell* in the sense that its (relative) interior is a top-dimensional stratum of our stratification $\mathcal{X}$. The (relative) interiors of the various faces of the cells $\mathcal{S}_\pi$ form the lower dimensional strata. In terms of filter functions, a stratum is simply a maximal subset whose functions induce the same pre-order on vertices of K. We then have that any persistence based loss is stratifiably smooth w.r.t. this stratification.

Proposition 6. *Let $V : \mathbf{Bar} \to \mathbb{R}$ be a 2-differentiable map. Then the objective function $f := V \circ \mathrm{PH}$ is stratifiably smooth for the stratification $\mathcal{X}$ induced by the permutation group Σ_n.*

Proof. From Proposition 4.23 and Corollary 4.24 in [53], on each a cell $\mathcal{S}_\pi$ we can define a local C^2 coordinate system that consists of linear maps $b_i, d_i : \mathcal{S}_\pi \to \mathbb{R}$, in particular it admits a C^2 extension on a neighborhood of $\mathcal{S}_\pi$. Since V is globally 2-differentiable, by the chain rule, we incidentally obtain a local C^2 extension f_i of $f_{|\mathcal{S}_\pi} = (V \circ \mathrm{PH})_{|\mathcal{S}_\pi}$. □

Remark 4. Note that the condition that $f = V \circ \mathrm{PH}$ is a proper map, as required for the analysis of Algorithm 2, sometimes fails because PH may not have compact level-sets. The intuitive reason for this is that a filter function x can have an arbitrarily large value on two distinct entries—one simplex creates a homological cycle that the other destroys immediately—that may not be reflected in the barcode $\mathrm{PH}(x)$. Hence the fiber of PH is not bounded. However, when the simplicial complex K is (homeomorphic to) a compact oriented manifold, any filter function must reach its maximum at the simplex that generates the fundamental class of the manifold (or one of its components), hence PH has compact level-sets in this case. Finally, we note that it is always possible to turn a loss function f based on PH into a proper map by adding a regularization term that controls the norm of the filter function x.

4.2 Exploring the space of strata

In the setting of Proposition 6, the objective function $f = V \circ \mathrm{PH} : \mathbb{R}^n \to \mathbb{R}$ is a stratifiably smooth map, where the stratification involved is induced by the group Σ_n of permutations on $\{1, \cdots, n\}$, with cells $\mathcal{S}_\pi$ as in Equation (22). In order to calculate the approximate subgradient $\partial_\epsilon f(x)$, we need to compute the set $\mathcal{X}_{x,\epsilon}$ of cells $\mathcal{S}_\pi$ that are at Euclidean distance no greater than ϵ from x:

$$d_2^2(x, \mathcal{S}_\pi) := \min_{p \in \mathcal{S}_\pi} \|x - p\|_2^2 \leqslant \epsilon^2. \tag{23}$$

Estimating distances to strata In practice however, solving the quadratic problem of Equation (23) to compute $d_2(x, \mathcal{S}_\pi)$ can be done in $O(n \log n)$ time using solvers for isotonic regression [9]. Since we want to approximate many such distances to neighboring cells, we rather propose the following estimate which boils down to $O(1)$ computations to estimate $d_2(x, \mathcal{S}_\pi)$. For any $\pi \in \Sigma_n$, we consider the *mirror of* x *in* $\mathcal{S}_\pi$, denoted by $x^\pi \in \mathbb{R}^n$ and obtained by permuting the coordinates of x according to π:

$$\forall 1 \leqslant i \leqslant n, \qquad x^\pi_{\pi(i)} := x_i. \tag{24}$$

In the rest of this section, we assume that the point x is fixed and has increasing coordinates, $x_i \leqslant x_{i+1}$, which can always be achieved after a suitable re-ordering of these coordinates. The proxy $d_2(x, x^\pi)$ then yields a good estimate of $d_2(x, \mathcal{S}_\pi)$, as expressed by the following result.

Proposition 7. *For any permutation $\pi \in \Sigma_n$, we have:*

$$d_2(x, \mathcal{S}_\pi) \leqslant d_2(x, x^\pi) \leqslant 2 d_2(x, \mathcal{S}_\pi).$$

Proof. The left inequality is clear from the fact that x^π belongs to the cell $\mathcal{S}_\pi$. To derive the right inequality, let $\hat{x}^\pi$ be the projection of x onto $\mathcal{S}_\pi$. It is a well-known fact in the discrete optimal transport literature, or alternatively a consequence of Lemma 2 below, that

$$d_2(x^\pi, \hat{x}^\pi) = \min_{\tau \in \Sigma_n} d_2(x^\tau, \hat{x}^\pi),$$

so that in particular $d_2(x^\pi, \hat{x}^\pi) \leqslant d_2(x, \hat{x}^\pi)$. Consequently,

$$d_2(x, x^\pi) \leqslant d_2(x, \hat{x}^\pi) + d_2(x^\pi, \hat{x}^\pi) \leqslant 2 d_2(x, \hat{x}^\pi) = 2 d_2(x, \mathcal{S}_\pi).$$

□

Our approximate oracle for estimating the Goldstein subgradient, see Section 3.3, computes the set of mirrors x^π that are at most ϵ-away from the current iterate $x := x_k$, that is:

$$\textbf{ApproxSampleOracle}(x, \epsilon) := \left\{ x^\pi \,|\, d_2(x, x^\pi) \leqslant \epsilon, \pi \in \Sigma_n \right\}.$$

Remark 5. Recall that the oracle is called several times in Algorithm 3 when updating the current iterate x_k with a decreasing exploration radius ϵ_k. However, for the oracle above we have

$$\textbf{ApproxSampleOracle}(x, \epsilon') \subseteq \textbf{ApproxSampleOracle}(x, \epsilon) \text{ whenever } \epsilon' \leqslant \epsilon,$$

so that once we have computed $\textbf{ApproxSampleOracle}(x_k, \epsilon_k)$ for an initial value ϵ_k and the current x_k, we can retrieve $\textbf{ApproxSampleOracle}(x_k, \epsilon')$ for any $\epsilon' < \epsilon_k$ in a straightforward way, avoiding re-sampling neighboring points around x_k and computing the corresponding gradient each time ϵ_k decreases, saving an important amount of computational resources.

Sampling in nearby strata In order to implement the oracle **ApproxSampleOracle**(x, ϵ), we consider the Cayley graph with permutations Σ_n as vertices and edges between permutations that differ by elementary transpositions (those that swap consecutive elements). In other words, the Cayley graph is the dual of the stratification of filter functions: a node corresponds uniquely to a cell $\mathcal{S}_\pi$ and an edge corresponds to a pair of adjacent cells.

We explore this graph starting at the identity permutation using an arbitrary exploration procedure, for instance the Depth-First Search (DFS) algorithm. During the exploration, assume that the current node, permutation π, has not yet been visited (otherwise we discard it). If $d_2(x, x^\pi) \leqslant \epsilon$, then we record the mirror point x^π. Else, $d_2(x, x^\pi) > \epsilon$, and in this case we do not explore the children of π. Note that given a child π' of π, we retrieve $x^{\pi'}$ and $d_2(x, x^{\pi'})$ from x^π and $d_2(x, x^\pi)$ in $O(1)$ time. The following result entails that this procedure indeed computes **ApproxSampleOracle**(x, ϵ).

Proposition 8. *Let $\pi' \in \Sigma_n$ be a permutation differing from the identity. Then there must be at least one parent π of π' in the Cayley graph such that $d_2(x, x_\pi) \leqslant d_2(x, x_{\pi'})$.*

Proposition 8 is a straight consequence of the following well-known, elementary lemma.

Lemma 2. *Let $x, y \in \mathbb{R}^n$ be two vectors whose coordinates are sorted in the same order, namely $x_i \leqslant x_j \Leftrightarrow y_i \leqslant y_j$. Given $\pi \in \Sigma_n$ a permutation, let* $\mathrm{inv}(\pi)$ *be the set of inversions, i.e. pairs (i, j) satisfying $(j - i)(\pi(j) - \pi(i)) < 0$. Then*

$$\mathrm{inv}(\pi) \subseteq \mathrm{inv}(\pi') \Rightarrow \sum (x_i - y_{\pi(i)})^2 \leqslant \sum (x_i - y_{\pi'(i)})^2.$$

Remark 6. For an arbitrary filter function x, the computation of the barcode $\mathrm{PH}(x)$ has complexity $O(\#K^3)$, here $\#K$ is the number of vertices and edges in the graph K (or the number of simplices if K is a simplicial complex). In the SGS algorithm we need to compute $\mathrm{PH}(x_\pi)$ for each cell $\mathcal{S}_\pi$ near the current iterate x_k, which can quickly become too expansive. Below we describe two heuristics that we implemented in some of our experiments (see Section 5.3) to reduce time complexity.

The first method bounds the number of strata that can be explored with a hyper-parameter $N \in \mathbb{N}$, enabling a precise control of the memory footprint of the algorithm. In this case exploring the Cayley graph of Σ_n using Dijkstra's algorithm is desirable, since it allows to retrieve the N strata that are the closest to the current iterate x_k. Note that in the original Dijkstra's algorithm for computing shortest-path distances to a source node, each node is reinserted in the priority queue each time one of its neighbors is visited. However in our case we dispose of the exact distances $d(x, x_\pi)$ to the source each time we encounter a new node, permutation π, so we can simplify Dijkstra's algorithm by treating each node of the graph at most once. The second approach is *memoization*: inside a cell $\mathcal{S}_\pi$, all the filter functions induce the same pre-order on the n vertices of K, hence the knowledge of the barcode $\mathrm{PH}(x_\pi)$ of one of its filter functions allows to compute $\mathrm{PH}(x'_\pi)$ for any other $x'_\pi \in \mathcal{S}_\pi$ in $O(\#K)$ time. We can take advantage of this fact by recording the cells $\mathcal{S}_\pi$ (and the barcode $\mathrm{PH}(x_\pi)$ of one filter function x_π therein) that are met by the SGS (or GS) algorithm during the optimization, thereby avoiding redundant computations whenever the cell $\mathcal{S}_\pi$ is met for a second time.

5 Experiments

In this section we apply our approach to the optimisation of objective functions based on the persistence map PH, and compare it with other methods. There are two natural classes of objective functions that we can build on top of the barcode $\mathrm{PH}(x)$. One consists in turning $\mathrm{PH}(x)$ into a vector using one of the many existing vectorisation techniques for barcodes [14, 2, 24, 18] and then to apply any standard objective function defined on Euclidean vector space. In this work we focus on the second type of objective functions which are based on direct comparisons of barcodes by means of metrics W_q on **Bar** as introduced in Section 4.

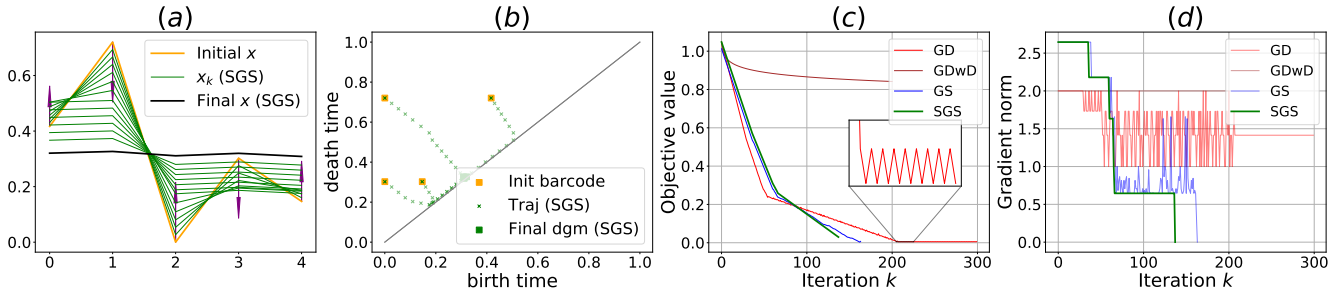

Figure 3: Comparison of vanilla Gradient Descent (GD), Gradient Descent with decay (GDwD), Gradient Sampling (GS) and our Stratified Gradient Sampling (SGS) on a toy example. (a) The evolution of filter functions $(x_k)_k$ as the total extended persistence Pers is minimised with the SGS method. Purple arrows indicate descent direction at $k = 0$. As expected, the minimization tends to make $(x_k)_k$ topologically as trivial as possible, that is flat in this context (b) The barcodes $\mathrm{PH}(x_k)$ represented as *persistence diagrams* extended with the point $(\min(x), \max(x))$. (c) The value $\mathrm{Pers}(x_k)$ of the objective function across iterations k. (d) The corresponding gradient norms $(\|g_k\|)_k$. Only GS and SGS reach the stopping criterion $\|g_k\| < \eta$.

We consider three experimental settings in increasing level of complexity. Section 5.1 is dedicated to the optimization of an elementary objective function in TDA that allows for explicit comparisons of SGS with other optimization techniques. Section 5.2 and Section 5.3 introduce two novel topological optimization tasks: that of *topological registration* for translating filter functions between two distinct simplicial complexes, and that of *topological Fréchet mean* for smoothing the *Mapper graphs* built on top of a data set.

Implementation Our implementation is done in `Python 3` and relies on `TensorFlow` [1] for automatic-differentiation, `Gudhi` [56] for TDA-related computations (barcodes, distances W_q, Mapper graphs), `cvxpy` [32] for solving the quadratic minimization problem involved in Algorithm 4. Our implementation handles both ordinary and extended persistence, complexes of arbitrary dimension, and can easily be tuned to enable general objective functions (assuming those are provided in an automatic differentiation framework). Our code is publicly available at `https://github.com/tlacombe/topt`.

5.1 Proof-of-concept: Minimizing total extended persistence

The goal of this experiment is to provide a simple yet instructive framework where one can clearly compare different optimization methods. Here we consider the vanilla Gradient Descent (GD), its variant with learning-rate Decay (GDwD), the Gradient Sampling (GS) methodology and our Stratified Gradient Sampling (SGS) approach. Recall that GD is very well-suited to smooth optimization problems, while GS deals with objective functions that are merely locally Lipschitz with a dense subset of differentiability. To some extent, SGS is specifically tailored to functions with an intermediate degree of regularity since their restrictions to strata are assumed to be smooth, and this type of functions arise naturally in TDA.

We consider the elementary example of filter functions x on the graph obtained from subdividing the unit interval with n vertices and the associated (extended) barcodes $\mathrm{PH}(x) = \mathrm{PH}_0(x)$ in degree 0.[2] When the target diagram is empty, $D = \varnothing$, the objective $x \mapsto W_1(\mathrm{PH}(x), \varnothing)$ to minimize is also known in the TDA literature as the *total extended persistence* of $\mathrm{PH}(x)$:

$$\mathrm{Pers} : x \in \mathbb{R}^n \longmapsto \sum_{(b,d)\in \mathrm{PH}(x)} |d - b| \in \mathbb{R}.$$

[2] In this setting the extended barcode can be derived from the ordinary barcode by adding the interval $(\min(x), \max(x))$.

Throughout the minimization, the sublevel sets of x are simplified until they become topologically trivial: $\mathrm{Pers}(x)$ is minimal if and only if x is constant. This elementary optimization problem enables a clear comparison of the GD, GS and SGS methods.

For each mode $\in \{\mathrm{GD}, \mathrm{GDwD}, \mathrm{GS}, \mathrm{SGS}\}$ we get a gradient g_k^{mode} and thus build a sequence of iterates

$$x_{k+1} := x_k - \epsilon_k g_k^{\mathrm{mode}},\ k \geqslant 0.$$

For GD, the update step $\epsilon_k = \epsilon$ is constant, for GDwD it is set to be $\epsilon_k = \epsilon/(1+k)$, and for mode $\in \{\mathrm{GS}, \mathrm{SGS}\}$ it is reduced until $\mathrm{Pers}(x_k - \epsilon_k g_k^{\mathrm{mode}}) < \mathrm{Pers}(x_k) - \beta\epsilon_k \|g_k^{\mathrm{mode}}\|^2$ (and in addition $\epsilon_k < C_k \|g_k^{\mathrm{SGS}}\|$ for SGS). In each case the condition $\|g_k^{\mathrm{mode}}\| \leqslant \eta$ is used as a stopping criterion.

For the experiments, the graph consists of $n = 5$ vertices, $x_0 = (0.4, 0.72, 0, 0.3, 0.14)$, $\epsilon = \eta = 0.01$, and we also have the hyper-parameters $\gamma = 0.5$ and $\beta = 0.5$ for the GS and SGS algorithm. The results are illustrated in Figure 3.

Whenever differentiable, the objective Pers has gradient norm greater than 1, so in particular it is *not* differentiable at its minima, which consists of constant functions. Therefore GD oscillates around its optimal value: the stopping criterion $\|g_k^{\mathrm{GD}}\| \leqslant \eta$ is never met which prevents from detecting convergence. Setting ϵ_k to decay at each iteration in GDwD theoretically ensures the convergence of the sequence $(x_k)_k$, but comes at the expense of a dramatic decrease of the convergence rate.

In contrast, the GS and SGS methods use a fixed step-size ϵ_k yet they converge since they compute a descent direction by minimizing $\|g\|$ over the convex hull of the surrounding gradients $\{\nabla\mathrm{Pers}(x_k), \nabla\mathrm{Pers}(x^{(1)}), \ldots, \nabla\mathrm{Pers}(x^{(m)})\}$, as described in Algorithm 1 and Algorithm 4. Here $x^{(1)}, \ldots, x^{(m)}$ are either sampled randomly around the current iterate x_k (with $m = n + 1$) for GS or in the strata around x_k (if any) for SGS. We observe that it takes less iterations for SGS to converge: 137 iterations versus ~ 165 iterations for GS (averaged over 10 runs). This is because in GS the convex hull of the random sampling $\{x^{(1)}, \ldots, x^{(m)}\}$ may be far from the actual generalized gradient $\partial_\epsilon f$, incidentally producing sub-optimal descent directions and missing local minima, while in SGS the sampling takes all nearby strata into account which guarantees a reliable direction (as in Proposition 3), and in fact since the objective Pers restricts to a linear map on each stratum the approximate gradient $\tilde{\partial}_\epsilon f(x_k)$ equals $\partial_\epsilon f(x_k)$.

Another difference is that GS samples $n+1 = 6$ nearby points at each iteration k, while SGS samples as many points as there are nearby strata, and for early iterations there is just one such stratum. In practice, this results in a total running time of ~ 2.7s for GS vs. 2.4s for SGS to reach convergence.[3]

5.2 Topological Registration

We now present the more sophisticated optimization task of *topological registration*. This problem takes inspiration from registration experiments in shapes and medical images analysis [49, 33, 35], where we want to translate noisy real-world data (e.g. MRI images of a brain) into a simpler and unified format (e.g. a given template of the brain).

Problem formulation In a topological analog of this problem the observation consists of a filter function F defined on a simplicial complex K which may have, for instance, a large number of vertices, and the template consists of a simplicial complex K' simpler than K (e.g. with fewer vertices). The goal is then to find a filter function x on K' such that (K', x) faithfully recovers the topology of the observation (K, F). Formally we minimise the q-th distance ($q \in [1, +\infty]$) between their barcodes

$$x \mapsto W_q(\mathrm{PH}(x, K'), \mathrm{PH}(F, K)), \tag{25}$$

where we include the complexes in the notations $\mathrm{PH}(F, K)$ of the barcodes to make a clear distinction between filter functions defined on K and K'.

[3] Experiment run on a `Intel(R) Core(TM) i5-8350U @ 1.70GHz` CPU

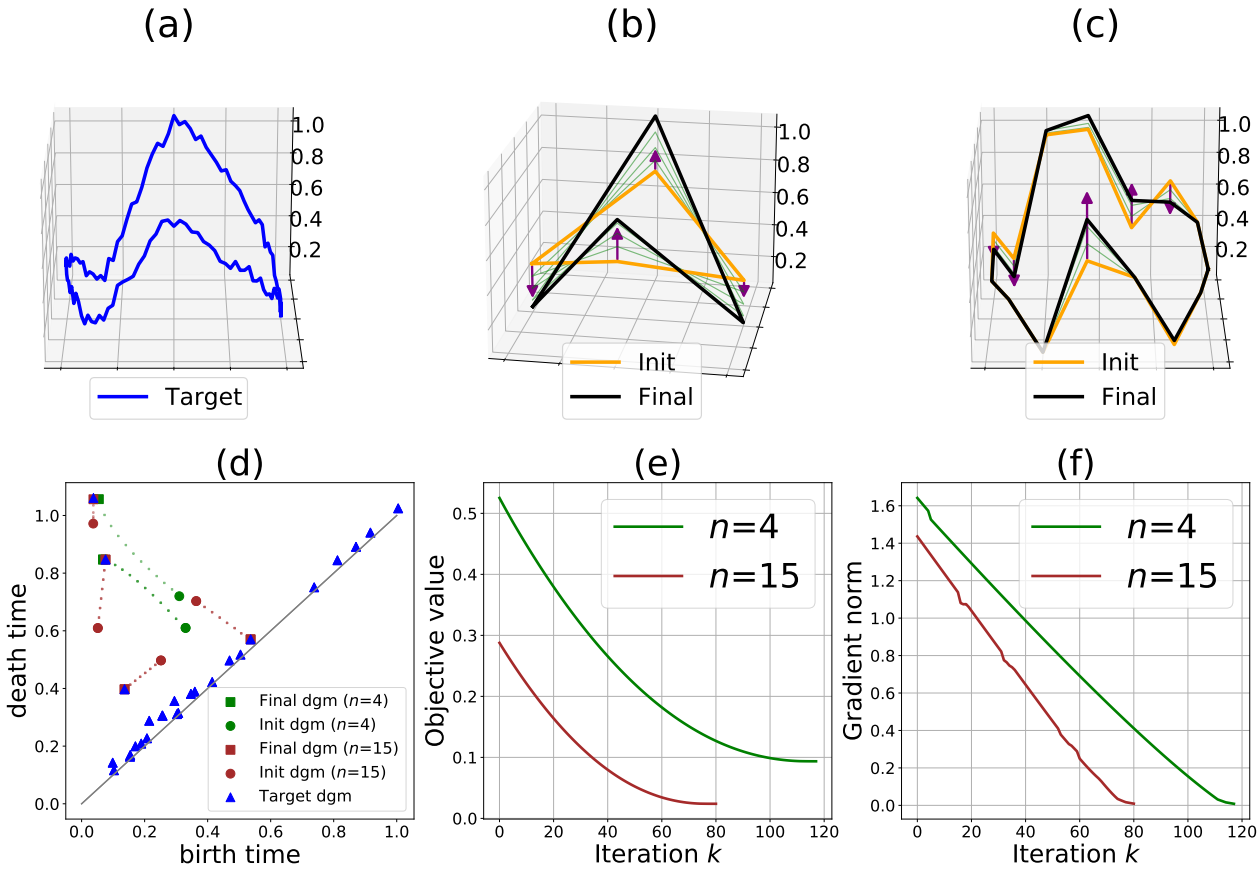

Figure 4: Illustration of topological registration (a) The target function defined on a (circular) simplicial complex of which we want to reproduce the topology. (b) The registration obtained when using a template with $n = 4$ vertices. Purple arrows indicate descent direction at $k = 0$. (c) The registration obtained when using a template with $n = 15$ vertices. (d) The target persistence diagram (blue triangles) along with the diagram trajectories through iterations for both cases (green and brown, respectively). (e) The values of the objective function across iterations. Using a larger template allows to attain lower objective values. (f) The corresponding gradient norms, both reaching the stopping criterion $\|g_k\| \leqslant \eta$.

Experiment We minimise (25) using our SGS approach. The observed simplicial complex K is taken to be the subdivision of the unit circle with 120 vertices, see Figure 4. Let $F = F_0 + \zeta$ where $F_0 \in \mathbb{R}^{120}$ is a piecewise linear map satisfying $F_0[0] = 0, F_0[30] = 1, F_0[45] = 0.05, F_0[60] = 0.35, F_0[75] = 0.1, F_0[90] = 0.8$ and ζ is a uniform random noise in $[0, 0.1]^{120}$. The (extended) barcode of F_0 consists of two long intervals $(0, 1), (0.05, 0.9)$ and one smaller interval $(0.1, 0.35)$ that corresponds to the small variation of F_0 (Figure 4 (a, left of the plot)). The stability of persistent homology implies that the barcode of F, which is a noisy version of F_0, contains a perturbation of these three intervals along with a collection of small intervals of the form (b, d) with $(d - b) < 0.1$, since 0.1 is the amplitude of the noise ζ. The persistence diagram representation of this barcode can be seen on Figure 4 (d, blue triangles): the three intervals are represented by the points away from the diagonal $\{x = y\} \subset \mathbb{R}^2$ and the topological noise is accounted by the points close to the diagonal.

We propose to compute a topological registration x of (K, F) for two simpler circular complexes with $n = 4$ and $n = 15$ vertices respectively (Figure 4, (b,c)). We initialize the vertex values x_0 randomly (uniformly in $[0, 1]^n$), and minimize (25) via SGS. We use $q = 2$, and the parameters of Algorithm 4 are set to $\epsilon = 0.01$, $\eta = 0.01$, $\beta = 0.5$, $\gamma = 0.5$.

With $n = 4$ vertices, the final filter function x returned by Algorithm 4 reproduces the two main peaks of F that correspond to the long intervals $(0, 1), (0.05, 0.9)$, but it fails to reproduce the small bump corresponding to $(0.1, 0.35)$ as it lacks the degrees of freedom to do so. *A fortiori* the noise appearing in F is completely absent in x, as observed in Figure 4 (d) where the two points appearing in the barcode of x_0 are pushed towards the two points of the target barcode of F as it is the best way to reduce the distance W_q. Using $n = 15$ vertices the barcode $\mathrm{PH}(x)$ retrieves the third interval $(0.1, 0.35)$ as well and thus the final filter function x reaches a lower objective value. However x also fits some of the noise, as one of the interval in the diagram of x_k is pushed toward a noisy interval close to the diagonal (see Figure 4 (d)).

5.3 Topological Mean of Mapper graphs

In our last experiment, we provide an application of our SGS algorithm to the *Mapper* data visualization technique [67]. Intuitively, given a data set X, Mapper produces a graph $\mathrm{Map}(X)$, whose attributes, such as its connected components and loops, reflect similar structures in X. For instance, the branches of the Mapper graph in [65] correspond to the differentiation of stem cells into specialized cells. Besides its potential for applications, Mapper enjoys strong statistical and topological properties [12, 21, 20, 19, 59, 6].

In this last experiment, we propose an optimization problem to overcome one of the main Mapper limitations, i.e., the fact that Mapper sometimes contains irrelevant features, and solve it with the SGS algorithm. For a proper introduction to Mapper and its main parameters we refer the reader to the appendix (Appendix A).

Problem formulation It is a well-known fact that the Mapper graph is not robust to certain changes of parameters which may introduce artificial graph attributes, see [3] for an approach to curate $\mathrm{Map}(X)$ from its irrelevant attributes. In our case we assume that $\mathrm{Map}(X)$ is a graph embedded in some Euclidean space $\mathbb{R}^d$ ($d = 2$ in our experiments), which is typically the case when the data set X is itself in $\mathbb{R}^d$, and we modify the embedding of the nodes in order to cancel geometric outliers. For notational clarity we distinguish between the embedded graph $\mathrm{Map}(X) \subseteq \mathbb{R}^d$ and its underlying abstract graph K. Let n be the number of vertices of K.

We propose an elementary scheme inspired from [20] in order to produce a simplified version of $\mathrm{Map}(X)$. For this, we consider a family of bootstrapped data sets $\hat{X}_1, \ldots, \hat{X}_k$ obtained by sampling the data set $\mathrm{card}(X)$ times with replacements, from which we derive new mapper graphs $K_1, \cdots, K_k$, whose embeddings $\mathrm{Map}(\hat{X}_1), \ldots, \mathrm{Map}(\hat{X}_k)$ in $\mathbb{R}^d$ are fixed during the experiment. In particular, given a fixed unit vector $\mathbf{e}$ in $\mathbb{R}^d$, the projection $F_{\mathbf{e}}$ onto the line parametrized by $\mathbf{e}$ induces filter functions for each K_i, hence barcodes $\mathrm{PH}(F_{\mathbf{e}}, K_i)$.

We minimize the following objective over filter functions $\tilde{F}_{\mathbf{e}} \in \mathbb{R}^n$:

$$\tilde{F}_{\mathbf{e}} \in \mathbb{R}^n \mapsto \sum_{i=1}^{k} W_2(\mathrm{PH}(\tilde{F}_{\mathbf{e}}, K), \mathrm{PH}(F_{\mathbf{e}}, K_i))^2 \in \mathbb{R}. \tag{26}$$

By viewing the optimized filter function $\tilde{F}_{\mathbf{e}}$ as the coordinates of the vertices of K along the $\mathbf{e}$-axis, we obtain a novel embedding of the mapper graph $\mathrm{Map}(X)$ in $\mathbb{R}^d$ that is the *topological barycenter* of the family $(F_{\mathbf{e}}, \mathrm{Map}(\hat{X}_i))$.

To further improve the embedding $\mathrm{Map}(X)$, we jointly optimize Eq. (26) over a family $\{\mathbf{e}_j\}_j$ of directions. Intuitively, irrelevant graph attributes do not appear in most of the subgraphs $\mathrm{Map}(\hat{X}_i)$ and thus are removed in the optimized embedding of $\mathrm{Map}(X)$.

Remark 7. In some sense, the minimization Equation (26) corresponds to pulling back to filter functions the well-known minimization problem $\mathbf{Bar} \ni D \mapsto \sum_{i=1}^{k} W_2(D, D_i)^2$ that defines the *barycenter* or *Fréchet mean* of barcodes $D_1, \ldots, D_k$, see [70]. Indeed, a *topological mean* of a set of filter functions $x^1, \ldots, x^k$ on simplicial complexes $K_1, \ldots, K_k$ can be defined as a minimizer of $x \in \mathbb{R}^n \mapsto \sum W_2(\mathrm{PH}(x, K), \mathrm{PH}(x^i, K_i))^2$. In our experiment, x is interpreted as a radial projection onto the $\mathbf{e}$-axis, and in fact when considering several directions $\{\mathbf{e}_j\}_j$ the mean resulting from the optimization is actually that of the so-called Persistent Homology Transform from [71].

Experiment To illustrate this new method for Mapper, we consider a data set X of single cells characterized by chromatin folding [60]. Each cell is encoded by the squared distance matrix M of its DNA fragments. This data set was previously studied in [22], in which it was shown that the Mapper graph could successfully capture the cell cycle, represented as a big loop in the graph. However, this attribute could only be observed by carefully tuning the parameters. Here we start with a Mapper graph computed out of arbitrary parameters, and then curate the graph using bootstrap iterates as explained in the previous paragraphs.

Specifically, we processed the data set X with the stratum-adjusted correlation coefficient (SCC) [74], with 500kb and convolution parameter $h = 1$ on all chromosomes. Then, we run a kernel PCA on the SCC matrix to obtain two lenses and computed a Mapper graph from these lenses using resolution 15, gain 0.4 on both lenses, and hierarchical clustering with threshold 2 on Euclidean distance. See Appendix A for a description of these parameters. The resulting Mapper graph $\mathrm{Map}(X)$ displayed in Figure 5 (upper left) contains the expected main loop associated to the cell cycle, but it also contains many spurious branches. However computing the Mapper graph with same parameters on a bootstrap iterate results in less branches but also in a coarser version of the graph (Figure 5, upper middle).

After using the SGS algorithm capped at 150 strata (see Remark 6), $\epsilon = 0.01$, $\eta = 0.01$, $\beta = 0.5$, $\gamma = 0.5$, initialized with $\mathrm{Map}(X)$, and with loss computed out of 10 bootstrap iterates and 4 directions with angles $\{0, \pi/2, \pi/4, -\pi/4\}$, the resulting Mapper, shown in Figure 5 (upper right), offers a good compromise: its resolution remains high and it is curated from irrelevant and artifactual attributes.

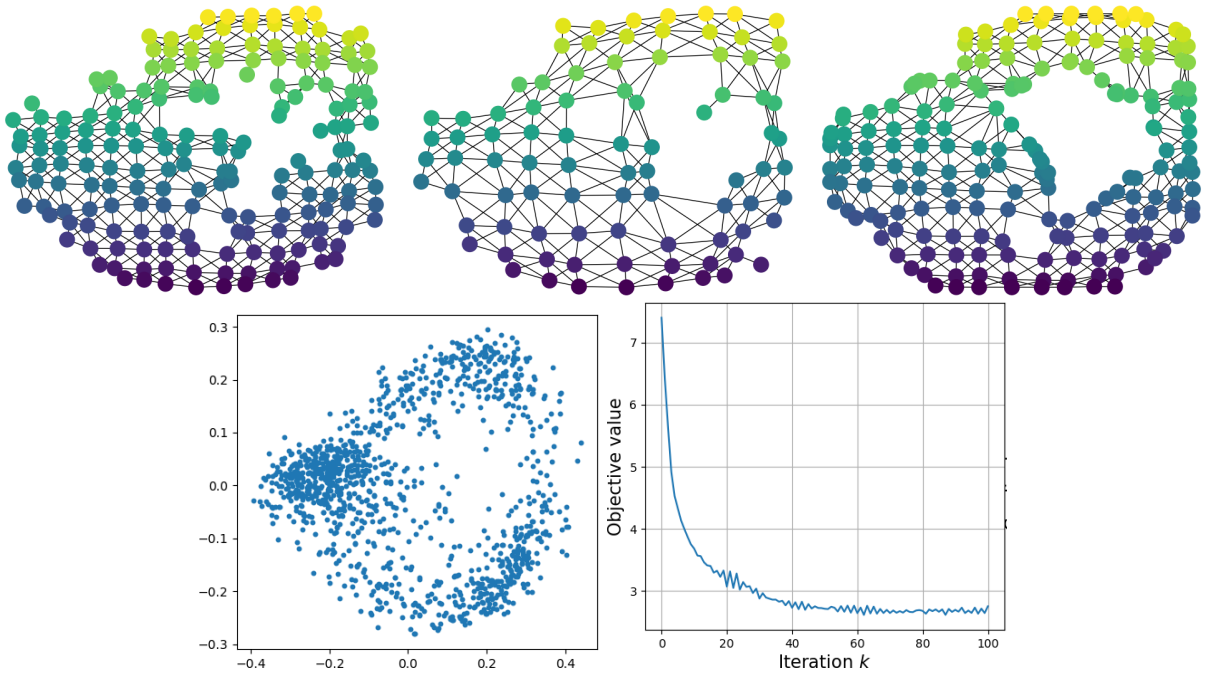

Figure 5: Different Mapper graphs colored with the first kernel PCA component. *(Top row)* Left: original Mapper graph computed with a set of arbitrary parameters with many spurious branches. Middle: Mapper graph obtained from bootstrap with very low resolution. Right: curated Mapper graph obtained as the Fréchet mean of the bootstrap iterates. *(Bottom row)* Left: visualization of the data set with kernel PCA. Right: the evolution of the loss (26) during the optimization process.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18, 2017.

[3] Dominique Attali, Marc Glisse, Samuel Hornus, Francis Lazarus, and Dmitriy Morozov. Persistence-sensitive simplication of functions on surfaces in linear time. In *Topological Methods in Data Analysis and Visualization (TopoInVis 2009)*. Springer, 2009.

[4] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. *Math. Program.*, 137(1-2, Ser. A):91–129, 2013.

[5] Adil Bagirov, Napsu Karmitsa, and Marko M Mäkelä. *Introduction to Nonsmooth Optimization: theory, practice and software*. Springer, 2014.

[6] Francisco Belchí, Jacek Brodzki, Matthew Burfitt, and Mahesan Niranjan. A numerical measure of the instability of Mapper-type algorithms. *Journal of Machine Learning Research*, 21(202):1–45, 2020.

[7] Paul Bendich, James S Marron, Ezra Miller, Alex Pieloch, and Sean Skwerer. Persistent homology analysis of brain artery trees. *The annals of applied statistics*, 10(1):198, 2016.

[8] Dimitri P Bertsekas. *Nondifferentiable optimization via approximation*. Springer, 1975.

[9] Michael J Best and Nilotpal Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1):425–439, 1990.

[10] A Bihain. Optimization of upper semidifferentiable functions. *Journal of Optimization Theory and Applications*, 44(4):545–568, 1984.

[11] Jérôme Bolte, Aris Daniilidis, Adrian Lewis, and Masahiro Shiota. Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2):556–572, 2007.

[12] Adam Brown, Omer Bobrowski, Elizabeth Munch, and Bei Wang. Probabilistic convergence and stability of random Mapper graphs. *Journal of Applied and Computational Topology*, 5:99–140, 2021.

[13] Rickard Brüel-Gabrielsson, Vignesh Ganapathi-Subramanian, Primoz Skraba, and Leonidas J Guibas. Topology-aware surface reconstruction for point clouds. In *Computer Graphics Forum*, volume 39 No. 5, pages 197–207. Wiley Online Library, 2020.

[14] Peter Bubenik et al. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16(1):77–102, 2015.

[15] James V Burke, Frank E Curtis, Adrian S Lewis, and Michael L Overton. The gradient sampling methodology. *invited survey for* INFORMS Computing Society Newsletter*, Research Highlights*, 2019.

[16] James V Burke, Adrian S Lewis, and Michael L Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[17] Mathieu Carriere, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hariprasad Kannan, and Yuhei Umeda. Optimizing persistent homology based functions. In *International Conference on Machine Learning*, pages 1294–1303. PMLR, 2021.

[18] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics*, pages 2786–2796. PMLR, 2020.

[19] Mathieu Carrière and Bertrand Michel. Statistical analysis of Mapper for stochastic and multivariate filters. In *CoRR*. arXiv:1912.10742, 2019.

[20] Mathieu Carrière, Bertrand Michel, and Steve Oudot. Statistical analysis and parameter selection for Mapper. *Journal of Machine Learning Research*, 19(12):1–39, 2018.

[21] Mathieu Carrière and Steve Oudot. Structure and stability of the one-dimensional Mapper. *Foundations of Computational Mathematics*, 18(6):1333–1396, 2017.

[22] Mathieu Carrière and Raúl Rabadán. Topological data analysis of single-cell Hi-C contact maps. In *Abel Symposia*, volume 15, pages 147–162. Springer-Verlag, 2020.

[23] Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2573–2582, 2019.

[24] Yu-Min Chung and Austin Lawson. Persistence curves: A canonical framework for summarizing persistence diagrams. *arXiv preprint arXiv:1904.07768*, 2019.

[25] Frank H Clarke. *Optimization and nonsmooth analysis*. SIAM, 1990.

[26] James R Clough, Ilkay Oksuz, Nicholas Byrne, Julia A Schnabel, and Andrew P King. Explicit topological priors for deep-learning based image segmentation using persistent homology. In *International Conference on Information Processing in Medical Imaging*, pages 16–28. Springer, 2019.

[27] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[28] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have l p-stable persistence. *Foundations of computational mathematics*, 10(2):127–139, 2010.

[29] Padraig Corcoran and Bailin Deng. Regularization of persistent homology gradient computation. *arXiv preprint arXiv:2011.05804*, 2020.

[30] Yuri Dabaghian, Vicky L Brandt, and Loren M Frank. Reconceiving the hippocampal map as a topological template. *Elife*, 3:e03476, 2014.

[31] Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1):119–154, 2020.

[32] Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

[33] Lijin Ding, Ardeshir Goshtasby, and Martin Satter. Volume image registration by template matching. *Image and Vision Computing*, 19(12):821–832, 2001.

[34] Dmitriy Drusvyatskiy, Alexander D Ioffe, and Adrian S Lewis. Clarke subgradients for directionally lipschitzian stratifiable functions. *Mathematics of Operations Research*, 40(2):328–349, 2015.

[35] Stanley Durrleman, Pierre Fillard, Xavier Pennec, Alain Trouvé, and Nicholas Ayache. Registration, atlas estimation and variability analysis of white matter fiber bundles modeled as currents. *NeuroImage*, 55(3):1073–1090, 2011.

[36] Herbert Edelsbrunner and John Harer. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008.

[37] Antonio Fuduli, Manlio Gaudioso, and Giovanni Giallombardo. A DC piecewise affine model and a bundling technique in nonconvex nonsmooth minimization. *Optimization Methods and Software*, 19(1):89–102, 2004.

[38] Antonio Fuduli, Manlio Gaudioso, and Giovanni Giallombardo. Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. *Siam journal on optimization*, 14(3):743–756, 2004.

[39] Rickard Brüel Gabrielsson, Bradley J Nelson, Anjan Dwaraknath, and Primoz Skraba. A topology layer for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1553–1563. PMLR, 2020.

[40] Marcio Gameiro, Yasuaki Hiraoka, and Ippei Obayashi. Continuation of point clouds via persistence diagrams. *Physica D: Nonlinear Phenomena*, 334:118–132, 2016.

[41] AA Goldstein. Optimization of lipschitz continuous functions. *Mathematical Programming*, 13(1):14–22, 1977.

[42] Napsu Haarala, Kaisa Miettinen, and Marko M Mäkelä. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, 109(1):181–205, 2007.

[43] Elias Salomão Helou, Sandra A Santos, and Lucas EA Simões. On the local convergence analysis of the gradient sampling method for finite max-functions. *Journal of Optimization Theory and Applications*, 175(1):137–157, 2017.

[44] Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G Escolar, Kaname Matsue, and Yasumasa Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016.

[45] Christoph Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *International Conference on Machine Learning*, pages 4314–4323. PMLR, 2020.

[46] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Mandar Dixit. Connectivity-optimized representation learning via persistent homology. In *International Conference on Machine Learning*, pages 2751–2760. PMLR, 2019.

[47] Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *Advances in Neural Information Processing Systems*, pages 5658–5669, 2019.

[48] AD Ioffe. An invitation to tame optimization. *SIAM Journal on Optimization*, 19(4):1894–1917, 2009.

[49] Sarang C Joshi and Michael I Miller. Landmark matching via large deformation diffeomorphisms. *IEEE transactions on image processing*, 9(8):1357–1370, 2000.

[50] Oleg Kachan. Persistent homology-based projection pursuit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 856–857, 2020.

[51] Krzysztof C Kiwiel. *Methods of descent for nondifferentiable optimization*, volume 1133. Springer, 2006.

[52] Krzysztof C Kiwiel. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

[53] Jacob Leygonie, Steve Oudot, and Ulrike Tillmann. A framework for differential calculus on persistence barcodes. *Foundations of Computational Mathematics*, pages 1–63, 2021.

[54] Chunyuan Li, Maks Ovsjanikov, and Frederic Chazal. Persistence-based structural recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1995–2002, 2014.

[55] Ladislav Lukšan and Jan Vlček. A bundle-Newton method for nonsmooth unconstrained minimization. *Mathematical Programming*, 83(1-3):373–391, 1998.

[56] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *International congress on mathematical software*, pages 167–174. Springer, 2014.

[57] Robert Mifflin. An algorithm for constrained optimization with semismooth functions. *Mathematics of Operations Research*, 2(2):191–207, 1977.

[58] Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological autoencoders. In *International conference on machine learning*, pages 7045–7054. PMLR, 2020.

[59] Elizabeth Munch and Bei Wang. Convergence between categorical representations of Reeb space and Mapper. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51, pages 53:1–53:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[60] Takashi Nagano, Yaniv Lubling, Csilla Várnai, Carmel Dudley, Wing Leung, Yael Baran, Netta Mendelson-Cohen, Steven Wingett, Peter Fraser, and Amos Tanay. Cell-cycle dynamics of chromosomal organization at single-cell resolution. *Nature*, 547:61–67, 2017.

[61] Dominikus Noll. Convergence of non-smooth descent methods using the Kurdyka-Lojasiewicz inequality. *J. Optim. Theory Appl.*, 160(2):553–572, 2014.

[62] Steve Y Oudot. *Persistence theory: from quiver representations to data analysis*, volume 209. American Mathematical Society Providence, RI, 2015.

[63] Jose A Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15(3):799–838, 2015.

[64] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, volume 37 No. 5, pages 13–25. Wiley Online Library, 2018.

[65] Abbas Rizvi, Pablo Cámara, Elena Kandror, Thomas Roberts, Ira Schieren, Tom Maniatis, and Raúl Rabadán. Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development. *Nature Biotechnology*, 35:551–560, 2017.

[66] Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

[67] Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *4th Eurographics Symposium on Point-Based Graphics (SPBG 2007)*, pages 91–100. The Eurographics Association, 2007.

[68] Yitzchak Solomon, Alexander Wagner, and Paul Bendich. A fast and robust method for global topological functional optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 109–117. PMLR, 2021.

[69] Jacob Townsend, Cassie Putman Micucci, John H Hymel, Vasileios Maroulas, and Konstantinos D Vogiatzis. Representation of molecular structures with persistent homology for machine learning applications in chemistry. *Nature communications*, 11(1):1–9, 2020.

[70] Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet means for distributions of persistence diagrams. *Discrete & Computational Geometry*, 52(1):44–70, 2014.

[71] Katharine Turner, Sayan Mukherjee, and Doug M. Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, 2014.

[72] Yuhei Umeda. Time series classification via topological data analysis. *Information and Media Technologies*, 12:228–239, 2017.

[73] Jan Vlček and Ladislav Lukšan. Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications*, 111(2):407–430, 2001.

[74] Tao Yang, Feipeng Zhang, Galip Yardımcı, Fan Song, Ross Hardison, William Noble, Feng Yue, and Qunhua Li. HiCRep: assessing the reproducibility of Hi-C data using a stratum-adjusted correlation coefficient. *Genome Research*, 27(11):1939–1949, 2017.

[75] Ka Man Yim and Jacob Leygonie. Optimization of spectral wavelets for persistence-based graph classification. *Frontiers in Applied Mathematics and Statistics*, 7:16, 2021.

[76] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

A Background on Mapper

The Mapper is a visualization tool that allows to represent any data set X equipped with a metric and a continuous function $f : X \to \mathbb{R}$ with a graph. It is based on the Nerve Theorem, which essentially states that, under certain conditions, the nerve of a cover of a space has the same topology of the original space, where a cover is a family of subspaces whose union is the space itself, and the (1-skeleton of a) nerve is a graph whose nodes are the cover elements and whose edges are determined by the intersections of cover elements. The whole idea of Mapper is that since covering a space is not always simple, an easier way is to cover the image of a continuous function defined on the space with regular intervals, and then pull back this cover to obtain a cover of the original space.

More formally, Mapper has three parameters: the *resolution* $r \in \mathbb{N}^*$, the *gain* $g \in [0, 1]$, and a *clustering method* $\mathcal{C}$. Essentially, the Mapper is defined as $\mathrm{Map}(X) = \mathcal{N}(\mathcal{C}(f^{-1}(\mathcal{I}(r, g))))$, where $\mathcal{I}(r, g)$ stands for a cover of $\mathrm{im}(f)$ with r intervals with $g\%$ overlap, and $\mathcal{N}$ stands for the nerve operation, which is applied on the cover $\mathcal{C}(f^{-1}(\mathcal{I}(r, g)))$ of X. This cover is made of the connected components (assessed by $\mathcal{C}$) of the subspaces $f^{-1}(I), I \in \mathcal{I}(r, g)$. See Figure 6.

The influence of the parameters $r, g, \mathcal{C}, f$ on the Mapper shape is still an active research area. For instance, the number of Mapper nodes increases with the resolution, and the number of edges increases with the gain, but these parameters, as well as the function f and the clustering method $\mathcal{C}$, can also have more subtle effects on the Mapper shape. We refer the interested reader to the references mentioned in this article for a more detailed introduction to Mapper.

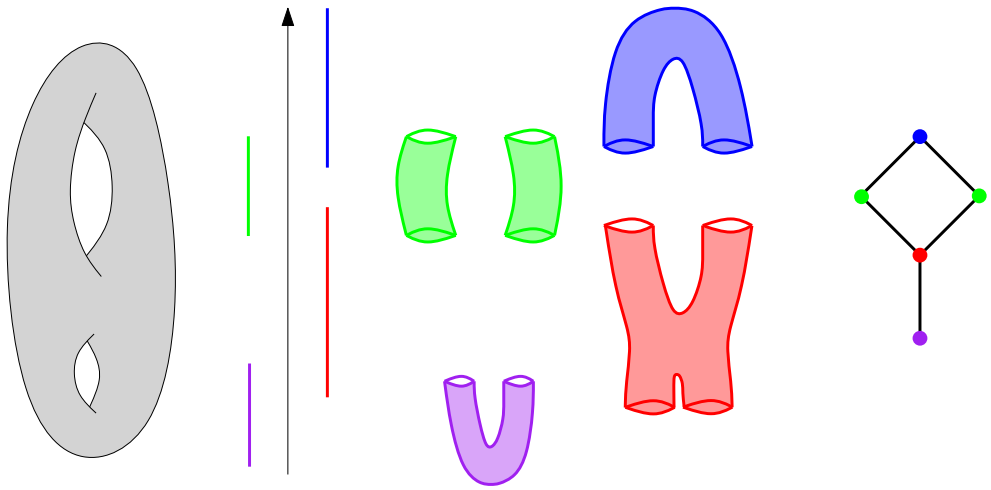

Figure 6: Example of Mapper computation on a double torus with height function covered by four intervals. Ech interval is pulled back in the original space through f^{-1} and then separated into its connected components with $\mathcal{C}$. In particular, the cover element obtained with the preimage of the green interval is separated into its two connected components. The nerve of this new cover is then computed to obtain the Mapper. One can see that with only four intervals, the topology of the double torus is only partially captured since only one loop is present in the Mapper instead of two.