Persistent Homology and Nested Dissection*

Michael Kerber† Donald R. Sheehy‡ Primoz Skraba§

Abstract

Nested dissection exploits the underlying topology to do matrix reductions while persistent homology exploits matrix reductions to the reveal underlying topology. It seems natural that one should be able to combine these techniques to beat the currently best bound of matrix multiplication time for computing persistent homology. However, nested dissection works by fixing a reduction order, whereas persistent homology generally constrains the ordering according to an input filtration. Despite this obstruction, we show that it is possible to combine these two theories. This shows that one can improve the computation of persistent homology if the underlying space has some additional structure. We give reasonable geometric conditions under which one can beat the matrix multiplication bound for persistent homology.

1 Introduction

Persistent homology [ELZ02] has become the standard tool in the growing field of topological data analysis (TDA). The underlying idea is to consider a sequence of increasing shapes and track the homological changes of the shapes in this process, that is, the appearance and disappearance of holes. This information can be read off from a (generalized) LU-decomposition of the *boundary matrix* which is a combinatorial representation of the sequence of shapes. All persistent homology algorithms used in practice are based on matrix reduction through row or column operations on an initially sparse boundary matrix. Due to matrix fill-in, a cubic complexity in the size of the matrix can be achieved on worst-case examples [Mor05]. On the other hand, algorithms often show near-linear asymptotic behavior on most realistic inputs. It is likely that the structure of realistic examples keeps the matrices sparse during the computation, but how can we quantify this?

Generalized nested dissection [LRT79] is a technique to reduce the effect of fill-in in Gaussian elimination and related matrix reduction problems for instances which have certain structure. Originally, the method was developed for symmetric matrices. Nested dissection interprets the matrix as a graph and looks for a *separator*, a subset of vertices C whose removal splits the graph into two disconnected parts A and B. The missing connections between A and B correspond to areas of the matrix in which no fill-in will happen during the reduction, if the pivots are chosen in a suitable order. If the graph is guaranteed to have a "good" separator (C is small and both A and B are not too small), it can be shown that this strategy yields matrices whose fill-in is asymptotically subquadratic, and matrix reduction can be performed in subcubic time.

A natural question is: can nested dissection be applied in the context of computing persistent homology? There are several obstructions for combining these two techniques. The first is that nested dissection is based on graph separators, whereas persistent homology operates on simplicial complexes. We therefore require a separator theory on simplicial complexes that not only gives the existence of separators, but also efficient algorithms to find them in common instances. The second obstruction is that nested dissection was originally designed for symmetric square matrices, whereas the boundary matrices in persistent homology are asymmetric. While the asymmetric case has been studied (see related work below), those techniques must be adapted to the case of persistent homology. Finally, the third obstruction is that nested dissection chooses a pivot order in the matrix reduction process to guarantee a fast running time, whereas persistent homology does not permit arbitrary column swaps without changing the outcome. While the effect of a column swap is well understood for persistent homology [CSEM06], there is no previous evidence that changing the pivot order is beneficial in terms of algorithmic complexity.

Contributions We show how to overcome all the obstructions mentioned above and apply the theory of nested dissection in the context of persistent homology computation. This combination leads to variations of existing algorithms that yield improved worst-case complexity bounds for input conditions which are satisfied for many practically relevant inputs.

†Graz University of Technology, Graz, Austria, kerber@tugraz.at. The author acknowledges support by the Max Planck Center for Visual Computing and Communication.

‡Computer Science & Engineering Department, University of Connecticut, don.r.sheehy@gmail.com. This material is based upon work supported by the National Science Foundation under Grant Nos. CCF-1464379 and CCF-1525978

§Jožef Stefan Institute, Ljubljana, Slovenia, primoz.skraba@ijs.si. This work was supported by the EU project TOPOSYS (FP7-ICT-318493-STREP)

1. We extend the theory of geometric separators to simplicial complexes, proving both that they exist in a class of geometric complexes relevant to topological data analysis, and can also be computed efficiently. Thus, there is a notion of a *β-separable complex* that is analogous to a β-separable graph, i.e. one for which every subcomplex has a separator of size $O(n^\beta)$ that separates the subcomplex into pieces no bigger than αn for some fixed $\alpha, \beta < 1$.
2. We analyze a simple variant of the classic persistence algorithm and show that its fill-in can be bounded by the same techniques as used in nested dissection. In particular, if the columns are ordered according to a nested dissection ordering of $O(n^\beta)$-size separators, then the fill-in is $O(n^{1+\beta})$ and the running time is $O(n^{1+2\beta})$.[1]
3. We use the Vineyards algorithm from [CSEM06] to transform the persistent homology of the nested dissection order to that of an arbitrary order in $O(n^{2+\beta})$. The worst-case of $O(n^3)$ was the best previously known.
4. Using the framework of the output-sensitive persistence algorithm from [CK13], we show for β-separable filtered simplicial complexes with only a constant number of ε-persistent features, the persistent homology can be computed in $O(n^{\beta\omega})$ time, where $\omega \leq 2.373$ is the matrix multiplication exponent [LG14]. The best-known bound for arbitrary input complexes is $\tilde{O}(n^2)$ [CK13].

Related Work Since the first algorithm for persistent homology by Edelsbrunner et al. [ELZ02], many variants have been proposed. The only unconditional subcubic complexity bound by Milosavljević et al. [MMS11] employs fast matrix multiplication to run in $O(n^\omega)$ time. Edelsbrunner and Parsa show that computing Betti numbers, and thus also persistent homology, is at least as hard as computing ranks of sparse $n \times n$ matrices [EP14]. Chen & Kerber [CK13] give an output-sensitive algorithm whose running time depends only on the running time of matrix rank computations and the number of features that persist longer than some threshold; it is also the only variant with subquadratic worst-case space complexity. Both variants lack a proof of usefulness in application scenarios. The currently fastest approaches in practice are based on the cohomological persistence algorithm by De Silva et al. [dSMVJ11] and on heuristics of the standard reduction algorithm that exploit the structure of the boundary matrices [CK11]. Some algorithmic variants are analyzed in terms of additional parameters in addition to the input size, including the total (index) persistence of the filtration [CK11], the number of critical cells in a cubical complex [BKR14], or the maximum Betti number among all the complexes in the input filtration [DFW14, BDM13]; while these results partially explain the excellent behavior of persistent homology in practice, the worst-case for all these variants remains cubic in the input size. Several open source software libraries implement variations of these algorithms [BKRW14, Dio, MBGY14].

In this paper, we are especially interested in the geometric case. This includes primarily filtrations constructed on grids or quality meshes. Hudson et al. [HMOS10] show that the persistent homology of the Euclidean distance to a point cloud can be approximated on the type of mesh used in finite element analysis. Sheehy [She11] extends this result to a large class of Lipschitz functions. Uniform grids are also commonly used to provide a basis for function approximations on Euclidean domains, and are, for example, built into some existing software libraries such as the TDA package for R [FKLM14].

Bounding fill-in during the Gaussian elimination of sparse matrices is in general known to be NP-hard [Yan81]. It was natural to question if it could be bounded in special cases. Nested dissection was first presented for regular grids by George [Geo73]. It was generalized to arbitrary, recursively separable graphs by Lipton et al. [LRT79], whose analysis is based on the work of Rose et al. [RTL76] which we also employ in our analysis. An alternative analysis was later given by Gilbert & Tarjan [GT87] who introduced some insights that allow one to weaken the condition on the recursive separators. Most of the work on nested dissection has been limited to the case of real, symmetric, positive definite matrices. Grigori et al. [GBDD10] consider a non-symmetric version, but do not prove explicit bounds on the fill or the work. Another work in the same spirit, though not explicitly about nested dissection, is the work of Carlsson & de Silva [CdS04] on geometric sparsity in solving linear systems. Yuster [Yus08] and Alon & Yuster [AY13] eliminate the conditions on symmetry and positive-definiteness while also allowing computations over arbitrary fields.

2 Topological Background

Simplicial Complexes A *simplicial complex* $\mathcal{K}$ is a collection of subsets, called *simplices*, of a *vertex set* V that is closed under taking subsets, i.e. $\sigma \in \mathcal{K}$ and $\tau \subset \sigma$ implies that $\tau \in \mathcal{K}$. The *dimension of a simplex* σ is defined as $\dim(\sigma) = |\sigma| - 1$, where $|\cdot|$ denotes set cardinality. The *dimension of a complex* $\mathcal{K}$

[1] An earlier draft of this paper was circulated that incorrectly reported stronger bounds of $O(n^{2\beta})$ fill-in and $O(n^{3\beta})$ running time.

is the maximum dimension of any simplex in $\mathcal{K}$. For two simplices σ and τ such that $\tau \subset \sigma$, we say τ is a *face* of σ and σ is a *coface* of τ. The *p-skeleton* of $\mathcal{K}$ is the subcomplex of $\mathcal{K}$ consisting of all simplices of dimension at most p and is denoted $\mathcal{K}^{(p)}$. In particular, the 1-skeleton, $\mathcal{K}^{(1)}$, corresponds to a graph.

Boundary Matrices and Homology A natural way to represent a simplicial complex $\mathcal{K}$ is via a sequence of sparse *boundary matrices* $\partial_0, \ldots, \partial_d$. Each matrix ∂_r is a $n_{r-1} \times n_r$ matrix, where, for each i, n_i is the number of i-simplices in $\mathcal{K}$. The columns correspond to r-simplices and the rows correspond to $(r-1)$-simplices. In the simplest case, where the underlying field is $\mathbb{Z}_2$, the integers modulo 2, each entry (i, j) of ∂_r is 1 if σ_i is a face of σ_j in codimension 1, and 0 otherwise. The case of ∂_1 is the well-known edge-vertex incidence matrix. Over other fields, it is normal to *orient* the boundary matrix. This is a generalization of orienting a graph. For example, in the case of ∂_1, each column contains two nonzero terms, one for each vertex in the edge, and they take values 1 and -1 in the matrix. This oriented boundary matrix is familiar in spectral graph theory as $\partial_1 \partial_1^\top$ is the so-called *graph Laplacian*.

A vector in $\mathbb{F}^{n_r}$ is called an r-chain and the set of r-chains is denoted C_r. Again, if $\mathbb{F} = \mathbb{Z}_2$, then we can think of an r-chain as a subset of r-simplices. This combinatorial perspective on the matrix operations is very useful and we will develop it more as we go . The boundary matrix ∂_r can be viewed as a linear operator from $C_r \to C_{r-1}$. Elements of the kernel of ∂_r are called *r-cycles* and elements of the image of ∂_r are called $(r-1)$-boundaries. The *rth homology group* of $\mathcal{K}$ is the quotient vector space

$$H_r(\mathcal{K}) := \ker \partial_r / \text{im } \partial_{r+1}.$$

It describes the r-cycles that are not boundaries of $(r+1)$-chains. Informally, it describes r-dimensional holes that have not been filled in by $(r+1)$-dimensional simplices. We will call these *homological features*. When referring to the collection of homology groups for all dimensions, we will write $H_*(\mathcal{K})$. The *Betti numbers* of a simplicial complex are defined as $\beta_r := \dim H_r(\mathcal{K})$. β_r can be directly computed from the ranks of the boundary matrices via $\beta_r = n_r - \text{rk}(\partial_r) - \text{rk}(\partial_{r+1})$.

Persistent Homology and Persistence Diagrams Given a pair of simplicial complexes, X, Y such that X is a subcomplex of Y, the inclusion map between the two induces a linear map between their homology groups, i.e.

$$X \hookrightarrow Y \text{ induces } H_*(X) \to H_*(Y).$$

The image of such a map is the *persistent homology* of the inclusion; it describes the homological features of X that persist when including X into Y. More generally, a *filtered simplicial complex* or just *filtration* is a sequence of nested complexes $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_n$. The persistent homology of this *filtered simplicial complex* describes the persistent homological features for every inclusion $X_i \hookrightarrow X_j$ for $i < j$.

The persistent homology is often encoded in a persistence diagram. This diagram is a multiset of points in $\mathbb{R}^2$, where a point (b, d) indicates a homological feature that first appears in the filtration in X_b and disappears only in X_d. b and d are referred to the birth and death times of the feature respectively. In general, $\text{rk}(H_*(X_s) \to H_*(X_t))$ equals the number of features that are born before or at X_s and that die at or after X_t; this is the number of points in the closed upper-left quadrant of the persistence diagram anchored at (s, t). Alternatively, one can look at the number μ_{st} of features born at or after X_s and dead at or before X_t; this it the number of points in closed lower-right quadrant of the persistence diagram anchored at (s, t). Moreover, μ_{st} is exactly the rank of a submatrix of ∂ composed of the lower left corner.

In practice, the common way to filter a simplicial complex is by real-valued function $f : \mathcal{K} \to \mathbb{R}$. An example is the Čech complex, where the vertices are points in $\mathbb{R}^d$, and for a k-simplex $\sigma = [v_0, \ldots, v_k]$, $f(\sigma)$ is the radius of the minimum enclosing ball of $v_0, \ldots, v_k$. The filtration is then defined by sorting the simplices with respect to the radius of their minimum enclosing ball. In such cases, all simplices in $X_i \setminus X_{i-1}$ have the same function value which we denote f_i. It is common to display the persistence diagram with respect to the function values, that is, replacing (b, d) with (f_b, f_d); sometimes, this difference is referred to as *index persistence* versus *function persistence*. We call the difference $f_d - f_b$ the *persistence* of the homological feature represented by (b, d).

3 Separating Graphs and Nested Dissection

Nested dissection is a method for ordering pivots in Gaussian elimination of sparse matrices to reduce fill-in and overall running time. The main idea is to treat the matrix as a graph and order the pivots based on recursive decomposition of the graph. We will focus first on the graph-theoretic side of the technique and review the concepts to the extent needed in later sections. The central definition is that of a *graph separator*:

DEFINITION 3.1. *A graph $G = (V, E)$ with $|V| = n$ has a $(f(n), \alpha)$-**separation** with $\alpha \in (1/2, 1)$ if V can be partitioned into 3 parts X, Y, and Z such that*

$$|X|, |Y| \leq \alpha n, \text{ and } |Z| \leq f(n),$$

and also no edge of E has one endpoint in X and one

end point in Y. The set Z is referred to as a ***separator of the graph*** *G.*

The quality of a separator depends on the growth of f. Throughout this paper, we will exclusively consider functions of the form $f(n) = \gamma n^\beta$ with $\beta \in (0,1)$ and $\gamma > 0$.

DEFINITION 3.2. *A* ***graph*** *G* ***is*** *β-**separable** if for some fixed α and γ, every subgraph has a $(\gamma n^\beta, \alpha)$-separation.*

The requirement that the separations exist for all subgraphs allows us to recursively separate the graph into smaller pieces for divide-and-conquer. This is the reason for the name *nested dissection*. For brevity in some of the statements, we assume henceforth that $\beta > 1/2$; while some types of graphs are 1/2-separable (in particular, planar graphs [LT79]), larger values of β seem more common in the situations that we are considering.

$\mathcal{V}$-paths and separator trees The following concepts and results are taken from [LRT79]. Let $G = (V, E)$ be a graph and π be a numbering of its vertices, that is, a bijective map $V \to [n]$ with $[n] := \{1, \ldots, n\}$. We call a path between v_i and v_j with $i < j$ in G a *$\mathcal{V}$-path*, if no vertex is repeated and except for i and j, each vertex on the path has an index smaller than i. When the ordering is fixed, we will sometimes identify a vertex of G and its index.

The importance of $\mathcal{V}$-paths is as follows: If A is a symmetric positive definite matrix with Cholesky decomposition $A = LL^T$, and G is the adjacency graph of A, the non-zero entries of L are in one-to-one correspondence to $\mathcal{V}$-paths [LRT79, Lemma 1] (attributed to [RTL76]), if the elimination process is performed according to the ordering π.

To bound the number of $\mathcal{V}$-paths, assume that G is a β-separable graph. We say that G is in *nested dissection ordering*, if its vertices are sorted according to the following *Numbering Algorithm* [LRT79, Sec.2]. It assumes that ℓ vertices are already ordered with indices larger than b and it numbers the remaining vertices consecutively from a to b. If G has at most $n_0 = (\gamma/(1-\alpha))^{1/(1-\beta)}$ vertices, we number the vertices arbitrary from a to b. Otherwise, we split the vertex set into (A, B, C), where C is the separator of size $O(n^\beta)$ and there is no edge connecting a vertex of A and a vertex of B. Let i, j, k denote the number of unnumbered vertices of A, B, C, respectively. We number the remaining vertices of C from $b - k + 1$ to b. We apply the numbering algorithm recursively on the subgraph induced by $B \cup C$ to assign the range $b-k-j+1$ to $b - k$. Finally, we apply the numbering algorithm recursively on the subgraph induced by $A \cup C$ to assign the range a to $b - k - j$. This concludes the description of the numbering algorithm.

The numbering algorithm gives rise to a *separator tree*: it is a rooted binary tree where each node corresponds to a recursive call of the algorithm. Each node contains the subset of vertices that are numbered in the corresponding recursive call. In particular, the root contains precisely the vertices of the first separator, and each internal node contains a subset of the separator constructed for the corresponding point set (it is only a subset because part of the separator may already have been numbered in a preceding call). The leaves contain at most n_0 vertices. Moreover, all vertices stored in a node have larger indices than all vertices stored in a successor of that node in the separator tree.

LEMMA 3.1. *An internal node of the separator tree on level ℓ contains at most $\gamma(1-\varepsilon)^{\beta\ell} n^\beta$ vertices with $\varepsilon = (1 - \alpha - \gamma/(n_0+1)^{1-\beta})$.*

Proof. It suffices to show that on level ℓ, the algorithm recurses on at most $(1-\varepsilon)^\ell n$ vertices; the claim then follows from the separator size. This fact is also used in the proof of [LRT79, Theorem 2] for the case $\beta = 1/2$, and we repeat it for completeness: Indeed, the statement follows by induction using that $(n_0+1)^{1-\beta} > n_0^{1-\beta} = \gamma/(1-\alpha)$, so $\varepsilon \in (0,1)$. Hence for an internal node, $n > n_0$ and

$$\begin{aligned} |A \cup C| &\le \alpha n + \gamma n^\beta \\ &= (\alpha + \gamma/n^{1-\beta}) n \\ &\le (\alpha + \gamma/(n_0+1)^{1-\beta}) n \le (1-\varepsilon) n \end{aligned}$$

□

We say that (simplex) k is an *ancestor* of (simplex) j if either j and k are contained in the same node of the separator tree, or k lies in an ancestor node of j.

LEMMA 3.2. *For any vertex j, the number of ancestors is bounded by $O(n^\beta)$.*

Proof. Let N denote the node of j in the separator tree. We just sum up the number of vertices stored in N and all its ancestor nodes. The worst case is obtained for the case that N is a leaf, for which we can bound the number of vertices in N by n_0 by construction, and the number of nodes in each predecessor using Lemma 3.1. This yields

$$n_0 + \sum_{s=0}^{\ell} \gamma(1-\varepsilon)^{\beta s} n^\beta \le n_0 + \gamma n^\beta \sum_{s=0}^{\infty} (1-\varepsilon)^{\beta s} = O(n^\beta).$$

□

From the properties of the separator tree, a $\mathcal{V}$-path from j to k implies that k is an ancestor of j. Using Lemma 3.2, it follows that the number of $\mathcal{V}$-paths with j as lower endpoint is bounded by $O(n^\beta)$. This immediately yields a total bound of $O(n^{1+\beta})$ for all $\mathcal{V}$-paths. However, this bound is not optimal:

LEMMA 3.3. *Let G be a β-separable graph and let its vertices be in nested dissection ordering π. Then, the number of $\mathcal{V}$-paths is $O(n^{2\beta})$.*

Proof. The number of $\mathcal{V}$-paths is bounded in [LRT79, Theorem 2] to bound the fill-in of a Cholesky decomposition. More precisely, the result is exposed for $\beta = 1/2$, but extends to $\beta > 1/2$ as mentioned in [LRT79, Theorem 6-9]. □

4 Separators on Simplicial Complexes

Given a symmetric, $n \times n$ matrix M, the **graph of** M is the n vertex graph with edges corresponding to the nonzero entries of M, i.e. $\mathcal{G}_M = ([n], \{(i,j) : M_{ij} \neq 0\})$. With this definition, the notions of separators and β-separability carry over to symmetric matrices in a natural way. For simplicial complexes, we can define a symmetric graph for a fixed dimension as follows:

DEFINITION 4.1. *The **p-skeleton graph of a simplicial complex** $\mathcal{K}$ is the graph whose vertices are the p-simplices of $\mathcal{K}$ and whose edges are the pairs of p-simplices that have a common $(p-1)$-dimensional face. It is denoted $\mathcal{G}_{\mathcal{K}^{(p)}}$.*

Recall the notion of the pth boundary matrix ∂_p of $\mathcal{K}$. Clearly, $\partial_p^\top \partial_p$ is a symmetric $n_p \times n_p$-matrix, and it is not hard to see that $\mathcal{G}_{\mathcal{K}^{(p)}} = \mathcal{G}_{\partial_p^\top \partial_p}$, where the matrix multiplication is performed over an arbitrary base field $\mathbb{F}$. We remark that the same holds true when we include a diagonal matrix R with non-zero diagonal entries in the symmetrization as in $\partial_p^\top R \partial_p$.

DEFINITION 4.2. *A **simplicial complex** $\mathcal{K}$ **is** **β-separable** if $\mathcal{G}_{\mathcal{K}^{(p)}}$ is β-separable for all $p \in \mathbb{N}$.*

β-separable complexes have favorable algorithmic properties that we will analyze in later sections. The natural first question is how common they are. We show that separability of a simplicial complex is implied by the separability of its 1-skeleton, if all vertices have bounded degree.

THEOREM 4.1. *Let $\mathcal{K}$ be a d-dimensional simplicial complex for some constant d and let G be its 1-skeleton. If G is β-separable and has maximum degree $\Delta = O(1)$, then $\mathcal{K}$ is β-separable as well.*

Proof. It will suffice to show that for every positive integer $p \leq d$ and for every subgraph H of $\mathcal{G}_{\mathcal{K}^{(p)}}$, that H is β-separable. Fix any such p and H. The vertex set of H is a set T of p-simplices. The set $V = \bigcup T$ contains all vertices of $\mathcal{K}$ that are also vertices of simplices in T. Let $m = |T|$ and let $n = |V|$. Let $\Delta_p \leq \binom{\Delta}{p}$ be an upper bound on the number of p-simplices of T that contain any one vertex. It follows immediately from the definitions that

$$\frac{m(p+1)}{\Delta_p} \leq n \leq m(p+1). \tag{4.1}$$

Since G is β-separable, the subgraph of G induced on V has a $(\gamma n^\beta, \alpha)$-separation (X_V, Y_V, Z_V). We can "lift" this separation to a separation of H by letting X_T and Y_T be the simplices of T whose vertices all lie in X_V and Y_V respectively. The remaining simplices of T are assigned to Z_T. Since $\mathcal{K}$ is a simplicial complex, this partition of T gives a separation, as there can be no common $(p-1)$-simplex in X_T and Y_T since their vertex sets are disjoint. It only remains to check that the separation is balanced, i.e. $|X_T|, |Y_T| \leq \alpha' m$ for some α', and that the separator is small, i.e. $|Z_T| = O(m^\beta)$.

Without loss of generality, assume $|X_T| \geq |Y_T|$. We will prove that the separation of H is balanced for a constant $\alpha' = 1 - \frac{1-\alpha}{\Delta_p}$.

$$\begin{aligned}
|X_T| &= m - |Y_T \cup Z_T| \\
&\quad [(X_T, Y_T, Z_T) \text{ is a partition of the } m \text{ simplices of } T] \\
&\leq m - \frac{1}{p+1}|Y_V \cup Z_V| \\
&\quad [\text{each vertex of } V \text{ is in a } p\text{-simplex of } T] \\
&\leq m - \frac{1}{p+1}(1-\alpha)n \\
&\quad [(X_V, Y_V, Z_V) \text{ is } \alpha\text{-balanced}] \\
&\leq m\left(1 - \frac{1-\alpha}{\Delta_p}\right) \\
&\quad \left[\tfrac{m(p+1)}{\Delta_p} \leq n \text{ by (4.1)}\right] \\
&= \alpha' m. \\
&\quad [\text{ by our choice of } \alpha'].
\end{aligned}$$

Bounding the size of the separator is similar.

$$\begin{aligned}|Z_T| &\le \Delta_p |Z_V| \quad [\text{each } t \in Z_T \text{ contains a vertex in } Z_V] \\ &\le \Delta_p \gamma n^\beta \qquad [Z_V \text{ is an } (\gamma n^\beta, \alpha)\text{-separator}] \\ &\le \Delta_p \gamma (m(p+1))^\beta \qquad [n \le m(p+1) \text{ by (4.1)}] \\ &= O(m^\beta). \qquad [\Delta_p, \gamma, \text{ and } p \text{ are constants}]\end{aligned}$$

□

There is naturally some degradation in the quality of the separator in terms of α and γ as the dimension increases, but this degradation does not affect the exponent in the separator size.

Geometric separators. We introduce some known results on computing geometric separators and show how they apply to the class of complexes that we are most interested in. Although the results in the rest of the paper only assume the underlying complex is β-separable, we give some sources of such complexes. Indeed, they are related to complexes already considered in the topological data analysis literature, namely Delaunay triangulations of nicely spaced points. As we show below, for d-dimensional inputs $\beta = 1 - 1/d$ as might be expected from the corresponding d-dimensional geometric graph separators.

Definition 4.3. *Given a collection of interior disjoint, closed balls $B = \{\text{ball}(p_i, r_i) \mid i \in [n]\}$ in $\mathbb{R}^d$ and a constant $\alpha \ge 1$, the α-**overlap graph** of B is the graph with vertex set $\{p_1, \ldots, p_n\}$ and edge set*

$$\{(p_i, p_j) \mid \text{ball}(p_1, r_i) \cap \text{ball}(p_j, \alpha r_j) \neq \emptyset \text{ and } \text{ball}(p_1, \alpha r_i) \cap \text{ball}(p_j, r_j) \neq \emptyset\}.$$

Equivalently, (p_i, p_j) is an edge of the α-overlap graph if and only if $\|p_i - p_j\| \le \min\{r_i + \alpha r_j, \alpha r_i + r_j\}$.

Theorem 4.2. (Miller et al. [MTTV98]) *If G is a subgraph of an α-overlap graph in some fixed dimension d, then an $(O(n^{1-\frac{1}{d}}), 1 - \frac{1}{d+2})$-separation of G exists and can be computed with high probability in randomized linear time.*

The algorithm in the Miller et al. paper works by finding an approximate centerpoint (a kind of geometric median) of the centers of the balls defining the overlap graph after mapping them stereographically onto the sphere in $\mathbb{R}^{d+1}$. Then after an appropriate transformation, the separation is defined by a random great circle on the sphere. Thus, the algorithm does not need the radii, just the locations of the points. Moreover, the centerpoint can be found by sampling, because a centerpoint of a random sample is an approximate centerpoint of the whole set with high probability.

We recall some notions from computational geometry. Given a finite set of points $P \subset \mathbb{R}^d$, the *Voronoi diagram* splits $\mathbb{R}^d$ into (closed) polytopes, called *Voronoi regions*, where the Voronoi region V_q of q consists of all points in $\mathbb{R}^d$ that are at least as close to q as to any other point of P. For V_q, let the *in-radius* $r(q)$ be the maximal radius of a ball centered at q that is contained in V_q and let the *out-radius* $R(q)$ be the minimal radius of a ball centered at q that contains all boundary vertices of V_q. We say that P is *τ-well-spaced* if $R(q)/r(q) \le \tau$ for each $q \in P$. A relatively simple packing argument shows that for a set of τ-well-spaced points, each Voronoi region intersects $k(\tau, d)$ other Voronoi regions, where $k(\tau, d)$ is a constant independent of n.

The dual of the Voronoi diagram is the *Delaunay triangulation*, Del_P. We define it as a simplicial complex with vertex set P, where a simplex σ is in Del_P if the corresponding Voronoi regions intersect. When no $k+3$ points of P lie on a common k-sphere for $k < d$, the Delaunay triangulation is d-dimensional and has a natural embedding in $\mathbb{R}^d$. If this condition is not met, there are known ways to perturb the points to make it so [EM90].

Theorem 4.3. *Let P be a set of τ-well-spaced points for some constant τ. Then, $\text{Del}_P^{(1)}$ is the subgraph of a $(2\tau - 1)$-overlap graph. In particular, the Delaunay triangulation of P is β-separable with $\beta = 1 - 1/d$.*

Proof. Consider the overlap graph G defined by $\{(p, r(p)) \mid p \in P\}$. Let (p, q) be an edge of Del_P. Without loss of generality, we may assume that $r(p) \ge r(q)$. We observe that

$$\|p - q\| \le 2R(q) \le 2\tau r(q) \le r(p) + (2\tau - 1) r(q),$$

so the edge belongs to G, proving the first part.

Combining the first part with Theorem 4.2, $\text{Del}_P^{(1)}$ is $(1 - 1/d)$-separable. Moreover, each vertex of $\text{Del}_P^{(1)}$ has constant degree for well-spaced points. Therefore, Theorem 4.1 asserts that the separability extends to all dimensions. □

5 Fill and Work in a Persistence Algorithm

We show that the technique of nested dissection also applies to the problem of persistence computation via matrix reduction. This gives a method for computing the homology of a large class of geometric simplicial complexes faster than matrix multiplication time.

Matrix Reduction For a matrix R, define $\text{low}_R(j)$ to be the maximum i such that R_{ij} is nonzero,

i.e. it is the lowest nonzero (also called *pivot*). We say that R is *reduced* if $\text{low}_R(j) \neq \text{low}_R(k)$ whenever $j \neq k$. If the matrix $R = \partial U$ for a boundary matrix ∂ and a unit upper triangular matrix U, then we say that R is a reduced boundary matrix of ∂. The persistence diagram of ∂ is the set of pairs $(\text{low}_R(j), j)$. The matrix reduction that computes R can be done in any of several ways, and although the matrix R is not unique, the pairs $(\text{low}_R(j), j)$ are unique [EH10]. We present a simple algorithm for computing persistence that we refer to as *the persistence algorithm* in this work (Algorithm 1). The reduction strategy resembles the annotation algorithm [DFW14, BDM13] as well as the reduction implicit in the fast matrix multiplication algorithm [MMS11].

Algorithm 1 The persistence algorithm

```
1:  procedure MATRIX_REDUCTION(∂)
2:      R ← ∂
3:      U ← I
4:      for j = 1, ..., n do
5:          if R_j ≠ 0 then
6:              i ← low_R(j)            ▷ lowest nonzero index
7:              for each k > j with R_ik ≠ 0 do
8:                  c ← R_ik / R_ij
9:                  U_k ← U_k − c U_j
10:                 R_k ← R_k − c R_j
11:     return R
```

The algorithm processes columns from left to right, finding the lowest nonzero element and zeroing out the rest of the row by column operations. Upon termination, R will be reduced and U will be unit upper triangular. The algorithm runs in $O(n^3)$ time on an $n \times n$ boundary matrix.

Define the *fill*, denoted $\text{fill}(M)$ as the number of non-zero entries of matrix M. By a careful choice of the data structures in Algorithm 1, we can bound the running time of the algorithm in terms of the fills of R and U. A major difference between this analysis and the classic setting of nested dissection is that the input matrix ∂ is not symmetric.

The complexity of three substeps of the algorithm depend on the underlying data structure of the matrix: how to find the lowest entry of a column, how to find all columns with a non-zero entry at row i, and how to perform additions efficiently. We represent the matrix as a collection of hash tables, one for each row and column, containing the non-zero entries of the corresponding row and column. We refer to them as *row tables* and *column tables*. We assume that collisions are handled through chaining, so that inserting, deleting, and searching for elements all take expected constant time (amortized) [MS08, §4.2].

We find the lowest entry of a column simply by a linear scan of the corresponding hash table. The accumulated cost for that is $\text{fill}(R)$, because the lowest entry is sought for exactly once per non-zero column of R. Finding the indices of the columns that R_j is added to is done through a scan of the row table and the cost for that is dominated by the subsequent additions. One such column addition updating R_k scans the column table of j and searches for each row index ℓ in the column table of k. If ℓ is not present, an entry is created (with the appropriate coefficient), and an index k is also added to the row table of ℓ. If present, the coefficient is updated. In case of a cancelation, the entry is removed, and k is also removed from ℓ's row table. Clearly, all these operations can be done in expected constant time per entry.

With these data structures, the running time is dominated by the column additions in the inner loop. Recall the notion of pth skeleton graphs (Definition 4.1) and let $\mathcal{G} := \mathcal{G}_{\mathcal{K}^{(p)}}$. The major insight with respect to Algorithm 1 is that the inner loop executes at most once for each $\mathcal{V}$-path of $\mathcal{G}$ as shown in the following lemma.

LEMMA 5.1. *If Algorithm 1 does a column operation for columns j and k (one execution of the inner loop), then there exists a $\mathcal{V}$-path between σ_j and σ_k in $\mathcal{G}$.*

Proof. Say that columns R_y and R_z are *adjacent* if there exists x such that both $R_{xy} \neq 0$ and $R_{xz} \neq 0$. Observe that in Algorithm 1, a column operation for j, k implies that R_j and R_k were adjacent at the start of the jth iteration of the outer loop. Specifically $R_{ij} \neq 0$ and $R_{ik} \neq 0$. We will prove the lemma by induction on j, showing that in the j-th iteration, all column additions of R_j to R_k imply a $\mathcal{V}$-path between σ_j and σ_k. For that, we maintain the stronger invariant that at the start of the j-th iteration, two columns R_y and R_z with $j \leq y < z$ are adjacent only if there is a $\mathcal{V}$-path from σ_y to σ_z in $\mathcal{G}$ such that every internal node on the path is numbered less than j.

The base of the induction is the start of the first iteration of the outer loop, at which point $R = \partial$. In this case, the only adjacent columns are those corresponding to adjacent simplices in $\mathcal{G}$. A single edge is a $\mathcal{V}$-path with no internal nodes, so the hypothesis is easily satisfied in this case.

For the inductive step, let R_y and R_z with $j \leq y < z$ be adjacent at the start of iteration $(j+1)$, and let x be such that $R_{xy} \neq 0$, $R_{xz} \neq 0$. If these columns were adjacent at the start of iteration j, then the desired $\mathcal{V}$-path exists by induction, so assume that were not adjacent at the start of iteration j. Iteration j only changes columns by adding multiples of R_j, so it must

be that R_{xj} was nonzero at the start of the iteration. It follows that R_j and R_y were adjacent at the start of iteration j because either R_{xy} was nonzero or there was a column operation adding R_j to R_y and thus R_{iy} was nonzero (with $i = \text{low}_R(j)$). By the same argument, R_j and R_z were adjacent at the start of iteration j. So, by induction there exists a pair of $\mathcal{V}$-paths, one from σ_y to σ_j and another from σ_j to σ_z, both of which have all internal nodes numbered less than j. The concatenation of these two paths is a new $\mathcal{V}$-path from σ_y to σ_z with all internal nodes numbered less than $j+1$. □

Let $\mathcal{K}$ be a β-separable simplicial complex (Definition 4.2). We call a filtration of $\mathcal{K}$ a *nested dissection filtration*, if for any dimension p, the filtration orders the p-simplices such that (the adjacency graph of) $\partial_p^T \partial_p$ is in nested dissection ordering, where ∂_p is the pth boundary matrix of $\mathcal{K}$.

For the next lemma, recall the definition of the separator tree from Section 3. We defined a (simplex) k to be an ancestor of (simplex) j if either j and k are contained in the same node of the separator tree, or k lies in an ancestor node of j.

LEMMA 5.2. *For a β-separable simplicial complex with a nested dissection filtration, if $U_{jk} \neq 0$, k is an ancestor of j.*

Proof. We prove the statement by induction on k. For $k = j$, k is clearly an ancestor of itself. Let $k > j$ and $U_{jk} \neq 0$. When the algorithm starts, $U_{jk} = 0$ because U starts as diagonal matrix. Let $\ell < k$ be the iteration of the algorithm for which U_{jk} becomes non-zero for the first time. This implies that $U_{j\ell} \neq 0$ and that the algorithm adds column ℓ to column k. By induction, $U_{j\ell} \neq 0$ means that ℓ is an ancestor of j. Because column ℓ is added to column k, Lemma 5.1 implies that there is a $\mathcal{V}$-path from ℓ to k. By the properties of the separator tree, k is an ancestor of ℓ. By transitivity k is an ancestor of j. □

LEMMA 5.3. *For a β-separable simplicial complex with a nested dissection filtration*

$$\text{fill}(U) = O(n^{1+\beta}).$$

Proof. By Lemma 5.2, $U_{jk} \neq 0$ implies that k is an ancestor of j. Lemma 3.2 shows that each j has $O(n^\beta)$ ancestors, which proves that each row of U has $O(n^\beta)$ non-zero entries. □

THEOREM 5.1. *For a β-separable simplicial complex with a nested dissection filtration and any dimension p, Algorithm 1 runs in $O(n^{1+2\beta})$ time and $O(n^{1+\beta})$ space.*

Proof. For the running time, it suffices to apply Lemma 5.1 and Lemma 3.3 to conclude that one must do at most $O(n^{2\beta})$ column operations. In the worst case, each column operation requires $O(n)$ time, so the $O(n^{1+2\beta})$ total running time follows.

The bound on the space comes from bounding the fill in the matrices R and U. Lemma 5.3 bounds fill(U). We can bound fill(R) by $(p+1) \cdot \text{fill}(U)$. Since $R = \partial U$, the k-th column of R is equal to a linear combination of columns of ∂, given by the k-th column of U. Since each column of ∂ has at most $p+1$ non-zero entries, the k-th column of R has at most $p+1$ times the number of non-zero entries of the k-th column of U. Summing over all the columns yields the result □.

6 Vineyards in Separable Complexes

In the preceding section, we showed that if the input filtration corresponds to a nested dissection order, then the nested dissection analysis can be used to bound the fill-in and running-time of a simple persistence algorithm. However, one is usually interested in the persistent homology of a fixed filtration which is in general different from a nested filtration ordering. We show that converting a reduced matrix for a nested dissection order into a reduced matrix for any other filtration can be done in $O(n^{2+\beta})$ time. Even though this bound is worse than the best known complexity bound of $O(n^\omega)$ for persistent homology [MMS11], it leads to a persistent homology algorithm that is purely based on elementary reductions and yields a subcubic bound for a large class of instances.

We consider the following task in this section. Assume that we want to compute the persistent homology of $\partial := \partial_p$ with respect to an arbitrary filtration f. Moreover, assume that the simplicial complex is β-separable, and a nested filtration ordering π of the simplices is known. We order the rows of ∂ in f-order and the columns of ∂ in π-order. Observe that the results of the preceding section hold for any *row* order of the matrix. Therefore, using Algorithm 1, the reordered boundary matrix can be reduced in $O(n^{1+2\beta})$ time.

To get the columns from π into f-order, we employ the vineyard algorithm of Cohen-Steiner et al. [CSEM06] which updates a reduced matrix in worst-case linear time after transposing the ordering of a single pair of adjacent simplices in a filtration. Because $O(n^2)$ transpositions might be necessary from π to f, it seems that $O(n^3)$ is the best one could hope for. However, $O(n)$ time is not necessary for many transpositions.

To show that, we have to define the sequence of transpositions carefully. Recall the separator tree of nested dissection from Section 3. Let C denote the simplices stored in some tree node, and let A, B

denote the simplices stored in the left and right subtree, respectively. Note that in π, all columns of A precede all columns of B, which in turn precede all columns in C. We first recursively bring A in f-order, then we bring B in f-order. Next, we bring $A \cup B$ in f-order, by transposing the columns in B with predecessors until they end in the correct position. Note that this step only transposes A-columns with B-columns, and we call this an *A-B-swap*. Finally, we bring $(A \cup B) \cup C$ in f-order by transposing the columns in C with predecessors until they are in the correct position. That ends the description of the transformation sequence. To bound the cost, we only have to note:

LEMMA 6.1. *An A-B-swap takes $O(1)$ time.*

Proof. Let R, U be the matrices constructed in the persistent homology algorithm for nested dissection order in Section 5. Note that $R = \partial U$.

Let $i < j$ be the indices of the simplices involved in the swap. Since i and j lie in disjoint subtrees of the separator tree, Lemma 5.2 implies that U_{ij} is zero. Inspecting the case distinction of the Vineyard algorithm [CSEM06, Sec.3], we observe that case 1 can be disregarded because it repairs only the effect of a row swap, and the only possible cases are 2.2, 3.2, and 4. All these cases require only constant time. □

Let $\mathrm{Cost}(n)$ denote the cost function for the transpositions with $n = |A \cup B \cup C|$. The recursive calls cost $\mathrm{Cost}(|A|) + \mathrm{Cost}(|B|)$, and the A-B-swaps cost $O(n^2)$ at most. Finally, to move a column of C to the correct position, at most $n-1$ transpositions are needed, and each such transposition costs $O(n)$ in the worst case. Since $|C| = O(n^\beta)$, the total cost for moving the C-columns is $O(n^{2+\beta})$. Therefore, the transposition cost satisfies the recurrence

$$\mathrm{Cost}(n) = cn^{2+\beta} + \mathrm{Cost}(n_1) + \mathrm{Cost}(n_2),$$

with $n_1 + n_2 \leq n$ and $n_1, n_2 \leq \alpha n$ for some $\alpha < 1$. A simple inductive proof shows that $\mathrm{Cost}(n) \leq c' n^{2+\beta}$ with $c' = c/(1 - \alpha^{2+\beta} - (1-\alpha)^{2+\beta})$. We can thus summarize as follows.

THEOREM 6.1. *Given a β-separable complex $\mathcal{K}$ with a nested dissection ordering π and any filtration f on $\mathcal{K}$. There is a sequence of transpositions that transforms the columns from π-order to f-order for which the Vineyard algorithm only requires $O(n^{2+\beta})$ time.*

As the complexity of the vineyard transformation dominates the complexity of persistence computation for π-order, we arrive at the following running time bound.

THEOREM 6.2. *Given a β-separable complex $\mathcal{K}$ with a known nested dissection ordering π and any filtration f on $\mathcal{K}$. There is a combination of Algorithm 1 and the Vineyard algorithm which computes persistent homology in $O(n^{2+\beta})$ time.*

7 The Output-Sensitive Algorithm

The output-sensitive algorithm from [CK13] reduces persistence computation to rank queries over submatrices of the boundary matrix. We will show in this section that for β-separable complexes, nested dissection can improve the asymptotic running time for the output-sensitive algorithm. When the number of highly persistent features is small, it gives an asymptotic improvement over the algorithm in the previous section.

Persistence via rank computation. We first revisit the algorithm from [CK13]. Recall our notation of ∂ for the boundary matrix in fixed dimension p. Assume for simplicity that ∂ is a $m \times n$ matrix with $m \leq n$. For row indices $i_1 \leq i_2$ and column indices $j_1 \leq j_2$, let $\mu_{i_1,i_2}^{j_1,j_2}$ denote the number of homology classes which are born in the index range $[i_1, i_2]$ and die in the index range $[j_1, j_2]$. The key observation is that μ can be computed with four rank computations of submatrices of ∂ using the following formula.

$$\mu_{i_1,i_2}^{j_1,j_2} = \mathrm{rk}\left(\partial_{i_1,j_2}^{i_1,j_2}\right) - \mathrm{rk}\left(\partial_{i_1+1,j_1-1}^{i_1+1,j_1-1}\right) - \mathrm{rk}\left(\partial_{i_2+1,j_2}^{i_2+1,j_2}\right) + \mathrm{rk}\left(\partial_{i_2+1,j_1-1}^{i_2+1,j_1-1}\right).$$

Moreover, if $\mu > 0$, the birth-death index pairs in the range can be computed by binary search, where the cost in each iteration corresponds (asymptotically) to the cost of computing μ. Therefore, the cost to compute one persistence pair is roughly bounded by $R(n) \log n$, where n is the size of the matrix and $R(n)$ is the cost to compute the rank of ∂.

Finally, the algorithm avoids the computation of low-persistence points. This is achieved by computing $\mu_{i_1,i_2}^{j_1,j_2}$ only if the birth and death range have sufficient difference in function value. For fixed $\Gamma > 0$ and $0 < \delta < 1$, the algorithm detects all persistence pairs of persistence at least Γ in time $O((1/\delta + C_{(1-\delta)\Gamma} \log n) R(n))$, where $C_{(1-\delta)\Gamma}$ is the number of persistence pairs of persistence at least $(1-\delta)\Gamma$. The main primitive computed by the algorithm is ranks of submatrices. In [CK13], deterministic and randomized variants are discussed. In the deterministic case, $R(n) = O(n^\omega)$ [IMH82], resulting in a complexity of $O(C_{(1-\delta)\Gamma} n^{2.373})$. This is asymptotically inferior to the asymptotically best known persistence algorithm that runs in $O(n^\omega)$ time [MMS11]. However, using Wiedemann's algorithm [KDS91] to compute ranks yields a randomized, Monte-Carlo algorithm for the Γ-persistent homology that runs in

$\tilde{O}(C_{(1-\delta)\Gamma} n^2)$ time, where $\tilde{O}$ means that logarithmic factors in n are ignored.

Yuster's method. Fix an arbitrary $m \times n$ matrix M over a finite field, and consider the task to compute $\mathrm{rk}(M)$. A first idea is to instead compute the rank of the symmetric matrix $M^T M$ using nested dissection. Over $\mathbb{R}$ or $\mathbb{Q}$, it is always true that $\mathrm{rk}(M) = \mathrm{rk}(M^T M)$, but this is not generally true over finite fields. Moreover, the nested dissection algorithm can get stuck if a zero appears in a diagonal entry during the Gaussian elimination, but the corresponding row/column is not completely zero.

Yuster [Yus08] proposes the following technique to overcome these problems. Instead of computing the rank of $M^T M$, he proposes to compute $\mathrm{rk}(M^T R M)$, where R is a diagonal matrix with diagonal elements chosen uniformly at random. To have sufficiently many elements to chose from, an extension field of the base field must be chosen. It is known that the rank does not change when switching to an extension field. Specifically, with the base field containing q elements, we need the extension field to contain at least $2\max\{n, m\}^2$ elements. We slightly rephrase Yuster's result [Yus08, Lemma 3.5].

LEMMA 7.1. *A randomly chosen matrix R yields with probability at least $1/2$ a matrix $M^T R M$ such that $\mathrm{rk}(M) = \mathrm{rk}(M^T R M)$ and $M^T R M$ is* pivoting-free*, that is, the algorithm will never encounter a zero diagonal entry with a non-zero row/column.*

Consequently, the nested dissection algorithm applied to $M^T R M$ will compute $\mathrm{rk}(M)$ correctly. It is also not hard to prove that in all cases, the nested dissection algorithm returns a number that is at most $\mathrm{rk}(M)$. Thus, by choosing λ random matrices R independently, the success probability can be boosted to $1 - 2^{-\lambda}$.

Fast ranks for β-separable complexes. Fix a β-separable simplicial complex and consider the case that $M = \partial^{j_1,j_2}_{i_1,i_2}$ as encountered in the output-sensitive algorithm. We compute the rank of M with Algorithm 2. By the previous paragraph, the algorithm returns the correct rank with probability $1 - 2^{-\lambda}$. We will analyze its running time next.

The first observation is that the ordering π is in fact a nested dissection ordering for the graph of $M^T R M$, just because the graph is the same for each R (if no r_i is chosen to be zero). Consequently, the nested dissection algorithm terminates in $O(n^{\omega\beta})$ steps [Yus08, Lemma 3.6], where n is the number of columns of M.

The second observation is that $\mathcal{G} = \mathcal{G}_{M^T R M}$ can be computed in $O(n + \#\{\text{edges of } \mathcal{G}\})$, for any random matrix R. Indeed, traverse the rows of M in any order. For a row i, compute the column set C_i that has a non-zero entry in row i. The corresponding simplices all share the same codimension one face; consequently, the vertices of C_i form a clique in the graph of $M^T R M$. Moreover, each edge of $\mathcal{G}$ is constructed only once because two simplices share at most one codimension one face. This implies the runtime bound.

Algorithm 2 Rank algorithm

```
1: procedure NESTED_DISSECTION_RANK(∂^{j1,j2}_{i1,i2})
2:     Compute M^T M and a nested dissection ordering π for it
3:     r = 0
4:     for j = 1, ..., λ do
5:         Choose R = diag(r_1, ..., r_n) u.a.r.
6:         Compute M^T R M
7:         Apply nested dissection with ordering π on M^T R M. Let r' be the rank obtained.
8:         r ← max{r, r'}
9:     return r
```

Finally, we observe that $\mathcal{G}$ is β-separable by assumption, and Lemma 3.3 implies that its number of edges is bounded by $O(n^{2\beta})$. Consequently, since $\omega \geq 2$, the complexity of computing $M^T R M$ is dominated by the cost of the nested dissection algorithms. We summarize:

LEMMA 7.2. *Algorithm 2 computes the rank of M in $O(S(n) + \lambda \cdot n^{\omega\beta})$ time with success probability $1 - 2^{-\lambda}$, where $S(n)$ is the cost to compute the nested dissection ordering for M.*

To apply the rank predicate in the output-sensitive persistence algorithm, λ must be set large enough to guarantee that no rank computation will fail in the execution. To ensure this property with a constant probability, roughly $\lambda = O(\log n)$ is sufficient (see the discussion preceding [CK13, Thm.14] for the precise statement). We can therefore summarize:

THEOREM 7.1. *For a filtration on a β-separable simplicial complex of size n, there is a Monte-Carlo algorithm with fixed success probability to compute all persistence pairs with persistence at least Γ in time*

$$\tilde{O}(C_{(1-\delta)\Gamma}(S(n) + n^{\omega\beta})),$$

where $\delta > 0$ is a fixed constant, $C_{(1-\delta)\Gamma}$ is the number of persistence pairs with persistence at least $(1-\delta)\Gamma$, $S(n)$ is the complexity of computing a nested dissection order,and $\tilde{O}$ means that we ignore logarithmic factors in n.

In particular, if Γ is chosen such that $C_{(1-\delta)\Gamma}$ is small (say, a constant or $O(\log n)$), if separators can be

computed efficiently (say in $O(n^{\omega\beta})$), and if β is smaller than $\frac{2}{\omega} \approx 0.843$, this algorithm computes the relevant persistence pairs in subquadratic time.

8 Conclusion

We have drawn a connection between nested dissection, a method for ordering pivots in Gaussian elimination, and persistent homology. This method takes advantage of additional geometric structure in the form of separators, which occurs in a natural class of geometric complexes to improve running time. We showed that, under this condition, a simple variant of the persistence algorithm provably runs in subcubic time. Since this approach fixes the ordering of computation, it is only generally applicable for computing static homology. We extended these results to show that the vineyard algorithm for updating persistence, can turn the restricted ordering into any desired filtration ordering in subcubic time on separable complexes, thus giving a guaranteed subcubic time persistence algorithm without resorting to dense matrix methods. Finally, we showed that nested dissection can also be combined with the Chen-Kerber algorithm for output-sensitive persistence computation, as the individual rank computations in that algorithm can be computed by nested dissection. There are several open problems and directions to consider:

- Can the running time bound for persistence be improved to $O(n^{3\beta})$ or $O(n^{\omega\beta})$ on β-separable filtrations? Can a single dissection ordering be used in the output sensitive algorithm, rather than computing separators for each submatrix?
- Approximation of persistence has been shown to improve space and time bounds. If we consider the L_∞ distance on filtrations, are there a large classes of filtrations which are ε-close to nested dissection orderings? That is, can we always find a nested dissection ordering which is ε-close to any filtration, or at least a general class of filtrations?

Acknowledgements

We would like to acknowledge Kirk Gardner for his help in correcting proofs in Section 5.

References

[AY13] Noga Alon and Raphael Yuster. Matrix sparsification and nested dissection over arbitrary fields. *Journal of the ACM*, 60(4), 2013.

[BDM13] Jean-Daniel Boissonnat, Tamal K. Dey, and Clément Maria. The compressed annotation matrix: An efficient data structure for computing persistent cohomology. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 695–706, 2013.

[BKR14] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III*, Mathematics and Visualization, pages 103–117. Springer International Publishing, 2014.

[BKRW14] Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. PHAT - persistent homology algorithms toolbox. In *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, pages 137–143, 2014.

[CdS04] Gunnar Carlsson and Vin de Silva. A geometric framework for sparse matrix problems. *Advances in Applied Mathematics*, 33(1):1–25, 2004.

[CK11] Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *27th European Workshop on Computational Geometry*, 2011.

[CK13] Chao Chen and Michael Kerber. An output-sensitive algorithm for persistent homology. *Computational Geometry: Theory and Applications*, 46(4):435–447, 2013.

[CSEM06] David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, SCG '06, pages 119–126, New York, NY, USA, 2006. ACM.

[DFW14] Tamal K. Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. In *30th Annual Symposium on Computational Geometry, SOCG'14, Kyoto, Japan, June 08 - 11, 2014*, page 345, 2014.

[Dio] Dionysus. By Dmitriy Morozov (`http://www.mrzv.org/software/dionysus/`).

[dSMVJ11] Vin de Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co)homology. *Inverse Problems*, 27, 2011.

[EH10] H. Edelsbrunner and J. Harer. *Computational Topology. An Introduction.* Amer. Math. Soc., 2010.

[ELZ02] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 4(28):511–533, 2002.

[EM90] Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, January 1990.

[EP14] Herbert Edelsbrunner and Salman Parsa. On the computational complexity of betti numbers: Reductions from matrix rank. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 152–160, 2014.

[FKLM14] Brittany Terese Fasy, Jisu Kim, Fabrizio Lecci, and Clément Maria. Introduction to the R package TDA. *CoRR*, abs/1411.1830, 2014.

[GBDD10] Laura Grigori, Erik G. Boman, Simplice Donfack, and Timothy A. Davis. Hypergraph-based unsymmetric nested dissection ordering for sparse LU factorization. *SIAM Journal on Scientific Computing*, 32(6):3426–3446, 2010.

[Geo73] Alan George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, 1973.

[GT87] John Russell Gilbert and Robert Endre Tarjan. The analysis of a nested dissection algorithm. *Numerische Mathematik*, 50(4):377–404, 1987.

[HMOS10] Benoît Hudson, Gary L. Miller, Steve Y. Oudot, and Donald R. Sheehy. Topological inference via meshing. In *Proceedings of the 26th ACM Symposium on Computational Geometry*, pages 277–286, 2010.

[IMH82] Oscar H. Ibara, Shlomo Moran, and Roger Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3:45–56, 1982.

[KDS91] Erich Kaltofen and B. David Saunders. On Wiedemann's method of solving sparse linear systems. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 539, pages 29–38. Springer, 1991.

[LG14] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM.

[LRT79] Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16(2):346–358, 1979.

[LT79] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36:177–189, 1979.

[MBGY14] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, pages 167–174, https://project.inria.fr/gudhi/software/, 2014.

[MMS11] Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the Twenty-seventh Annual Symposium on Computational Geometry*, SoCG '11, pages 216–225, New York, NY, USA, 2011. ACM.

[Mor05] Dmitriy Morozov. Persistence algorithm takes cubic time in worst case. *BioGeometry News, Dept. Comput. Sci., Duke Univ*, 2005.

[MS08] Kurt Mehlhorn and Peter Sanders. *Algorithms and Data Structures – The Basic Toolbox*. Springer, 1st edition edition, 2008.

[MTTV98] Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Geometric separators for finite-element meshes. *SIAM Journal of Scientific Computing*, 19(2):364–386, 1998.

[RTL76] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal of Computing*, 5(2):266–283, 1976.

[She11] Donald R. Sheehy. *Mesh Generation and Geometric Persistent Homology*. PhD thesis, Carnegie Mellon University, 2011.

[Yan81] Mihalis Yannakakis. Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981.

[Yus08] Raphael Yuster. Matrix sparsification for rank and determinant computations via nested dissection. In *FOCS: IEEE Symposium on Foundations of Computer Science*, pages 137–145, 2008.