An Algorithmic Way to Generate Simplexes for Topological Data Analysis

Krzysztof Rykaczewski[1,2], Piotr Wiśniewski[2], and Krzysztof Stencel[2,3]

[1] DLabs, Lęborska 3B, 80-230 Gdańsk
krykaczewski@gmail.com
[2] Faculty of Mathematics and Computer Science,
Nicolaus Copernicus University, Chopina 12/18,
87-100 Toruń, Poland
pikonrad@mat.umk.pl
[3] Faculty of Mathematics, Informatics and Mechanics
University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
stencel@mimuw.edu.pl

Abstract. In this article we present a new algorithm for creating simplicial Vietoris-Rips complexes that is easily parallelizable using computation models like MapReduce and Apache Spark. The algorithm does not involve any computation in homology spaces.

Keywords: data analysis, topology, simplicial complex

1 Introduction

The article of Silva and Grist [2] showed that the algebraic topology was a very practical tool. In this paper its authors analyse the coverage of an area with a network of sensors. This network is interpreted as a simplicial complex. Its homology type is then computed and exploited. If this complex is homotopy equivalent with point, the coverage is complete. Therefore, the authors translated a technical problem into a mathematical problem.

In the article [3] Grist searches for topology structures in big data sets. Again, this results in building simplicial complexes and analysing them.

The abovementioned works inspire to ask the question how to build simplicial complexes efficiently and how to analyse them. In this article w propose a novel algorithm to build the simplicial complex. This algorithm has the unique property that it can be implemented within massive parallel computational models like MapReduce and Apache Spark.

2 Background

Hereafter we assume that the data set given is a finite set of points $P := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$. This data can come up as a result of some physical experiments, psychological tests etc., and can generally be high-dimensional. We have no insight into how the data actually look like and how it is embedded, but it would be enough to know their "shape" to say something about how it is are arranged in the original space.

In recent papers of Carlsson and collaborators [1] presented ideas to explore some properties of high-dimensional data sets. They use algebraic topology tools to find certain characteristics of the data or its simpler representation. This approach allows to discover objects and features that are immersed in high-dimensional space.

We start presentation of basic facts from the notion of a simplex. Colloquially speaking, simplex is a generalization of a segment, a triangle and a tetrahedron to any dimension. More formally, suppose the $k+1$ points $p_0, \dots, p_k \in \mathbb{R}^k$ are affinely independent, which means $p_1 - p_0, \dots, p_k - p_0$ are linearly independent. Then, k-*simplex* determined by them is the set of points

$$\sigma := \left\{ \lambda_0 p_0 + \cdots + \lambda_k p_k \mid \lambda_i \geq 0,\ \ 0 \leq i \leq k,\ \sum_{i=0}^{k} \lambda_i = 1 \right\}, \tag{1}$$

i.e. k-simplex is a k-dimensional polytope which is the convex hull of its $k+1$ vertices. We often write $\sigma = [p_0, \dots, p_k]$ for short. The points $p_0, \dots, p_k$ are called *vertices* of the k-simplex. Each simplex is uniquely determined by its vertices. Any subset of vertices constitutes simplex with dimension smaller that the whole one, called a *face* of simplex.

We say two vertices v and w are *neighbors* if there is 1-simplex (edge) connecting them.

Definition 1. *A simplicial complex $\mathcal{K}$ is a set of simplexes that satisfies the following conditions:*

1. *Any face of a simplex from $\mathcal{K}$ is also in $\mathcal{K}$.*
2. *The intersection of any two simplexes σ_1, $\sigma_2 \in \mathcal{K}$ is either $\emptyset$ or a face of both σ_1 and σ_2.*

Having data set P, we can use it to build a variety of complexes. We shall now formulate two most frequent definitions.

Definition 2. *For a discrete set of points $P \subset \mathbb{R}^d$ and a parameter $\epsilon > 0$, we define the* Čech complex *of radius ϵ as $\check{C}(P, \epsilon)$ by*

$$\check{C}(P, \epsilon) := \left\{ [p_1, p_2, \dots, p_k] \mid \{p_1, p_2, \dots, p_k\} \subset P,\ \bigcap_i B(p_i, \epsilon/2) \neq \varnothing \right\}, \tag{2}$$

where $B(p, \epsilon)$ is the closed ball of radius ϵ centered at p.

The nerve theorem states that Čech complex $\check{\mathrm{C}}(P, \epsilon)$ has homotopy type of the union of closed balls with radius $\epsilon/2$ around data P This means that the Čech complex gives faithful representation of thickened data (they are common "shape").

Vietoris-Rips complex is closely related to the Čech complex.

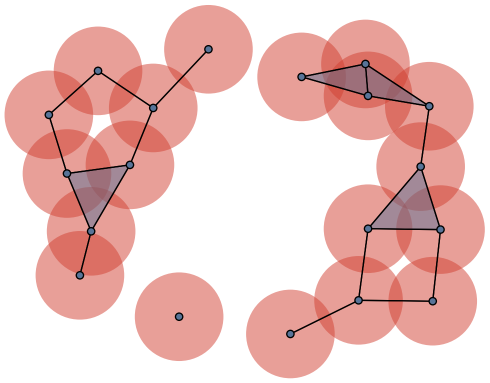

Fig. 1. Example of a Čech complex.

Definition 3. *For a given $\epsilon > 0$, the* Vietoris-Rips complex *(later called Rips complex) $R(P, \epsilon)$ is the largest simplicial complex that shares the same 1-skeleton (graph) as the Čech complex $\check{C}(P, \epsilon)$. More explicitly,*

$$R(P, \epsilon) := \{\sigma := [p_1, p_2, \ldots, p_k] \mid \{p_1, p_2, \ldots, p_k\} \subset P, diam(\sigma) \leq \epsilon\}, \quad (3)$$

where $diam(\sigma)$ is the diameter of simplex σ.

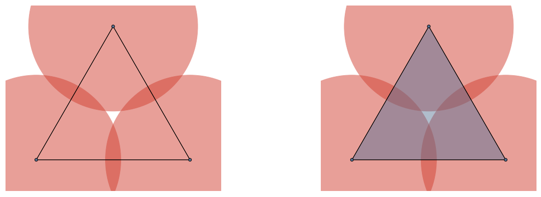

Fig. 2. Comparison of Čech (left) and Vietoris-Rips (right) complexes.

Lemma 1 (Vietoris-Rips). *For every finite set of points $P \subset \mathbb{R}^d$ and $r \geq 0$,*

$$\check{C}(P, \epsilon) \subset R(P, \epsilon) \subset \check{C}(P, 2\epsilon). \quad (4)$$

Thus, if the Čech complex at the same time for ϵ and 2ϵ approximates the data in a good way, then Vietoris-Rips complex do it as well. This estimate can be further improved [2].

Remark 1. Čech complex is difficult to calculate, but it is quite small. However, Vietoris-Rips complex is easy to calculate, but is usually very big. Both Čech and Vietoris-Rips complexes can produce simplicial complex of dimension greater than the considered space. Therefore, there are considered many alternatives for them: Delaunay Complex, Alpha Complex, (Lazy) Witness Complex, Mapper Complex, Sparsified Rips complex, Graph Induced Complex (GIC).

For small $\epsilon > 0$ we have disjoint set of balls, whereas for big ϵ the covering by balls is *simply connected* (without holes), so we are totally losing the knowledge about the data structure. Since (in general) we do not know which radius ϵ to take, we consider them all and obtain in this way a filtration of simplicial complexes, i.e.

$$\check{\mathrm{C}}(P, \epsilon_1) \subset \check{\mathrm{C}}(P, \epsilon_2) \subset \ldots \subset \check{\mathrm{C}}(P, \epsilon_n), \tag{5}$$

for $0 < \epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_n$. With the change of radius also changes the topology of the data: some holes are created, and some disappear. Roughly speaking, those holes that last longest represent the shape of data. It is usually represented in the form of the so-called *barcode* [3].

Let us recall the following definition from topology.

Definition 4. *Two topological spaces X and Y are called* homotopy equivalent *if there exist continuous maps $f\colon X \to Y$ and $g\colon Y \to X$, such that the composition $f \circ g$ is homotopic (i.e. it can be deformed in a continuous way) to the identity* id_Y *on Y, and $g \circ f$ is homotopic to* id_X.

3 Related Work

Data anaylsis is a process of modeling data in order to discover useful information. Unfortunately, mining high-dimensional data is very challenging. Therefore, a number of methods was created so as to make it easier for researchers. One of the youngest but rapidly growing field now is topological data analysis (TDA).

TDA is an approach to analyse data sets using techniques from topology. This method relies on calculation of some properties of the space/data, which maintain (are invariant) under certain transformations. Although most of topological invariants are difficult to calculate, there is an algorithm which can calculate homology groups and barcodes [4, 5]. These groups are often used in applications [3].

Topological methods due to their nature can be used to cope with many problems where traditional methods fail. It has broad application in coverage problem in sensor network [2, 3], detecting breast cancer [6], computer vision [7], detection of periodicity in time series (behaviour classification) [8] etc.

Another issue addressed in this subject is the speed of algorithms. In paper [9] Dłotko et al. presented distributed algorithm for homology computation over a sensor network. Their technique involve reduction and coreduction of simplicial complexes.

Regardless of the algebraic approach, there are created algorithms that do not use calculations on homology groups to obtain some approximation of the shape of data [10]. Algorithm proposed in the last paper can have problems with more complicated structures, like the sphere, but it is shown that it does well with coverage in sensor networks.

Notice that apart from using Čech and Rips complexes there are other directions in computational topology, but we do not discuss them here [2].

4 A Method to Build Complex using MapReduce

In this Section we present an efficient method to build Vietoris-Rips complexes. We make the following assumptions:

- $\mathcal{R}$ is a Euclidean space.
- A fixed number of dimensions n is given.
- $\mathcal{M} \subset \mathcal{R}^n$ is the set of input data points.
- We assume that each input data point has a unique identifier $id(x)$. Those identifiers are items of an linearly ordered set.
- For any $i \in \{1, 2, \ldots, n\}$ we define $\Pi_i : \mathcal{R}^n \to \mathcal{R}$ to be the projection to the i-th dimension.
- In $\mathcal{R}^n$ we assume the Euclidean distance. For any two $x, y \in \mathcal{R}^n$ let $|x, y|$ denote the Euclidean distance of x and y.
- Let us choose $\epsilon > 0$. We are going to build the Vietoris-Rips complex $R(\mathcal{M}, \epsilon)$

The first step to build a Vietoris-Rips complex is to find all pair of points $x, y \in \mathcal{M}$ such that $|x, y| < \epsilon$. If the set $\mathcal{M}$ is large, computing distances between all pairs of points is too time consuming. If the number of dimensions n is equal to 1, we can assign all points to segments of the form $\langle k\epsilon, (k+1)\epsilon)$. Then, instead of computing distances for all pairs, we consider only those pairs of points that fall into the same segment or two neighbouring segments. As the result, each point is paired only with points from three segments. This can significantly accelerate the computation. Of course, in a one-dimensional space, a simpler method that encompasses sorting and sequential scanning will be faster. However, this method cannot be extended to more dimensions.

For a two-dimensional space we can apply a method similar to the former one described above. Instead of segments, we have then squares with sides of length ϵ. Eventually, each point will be paired for distance computation only with points from nine other squares. The three-dimensional case is similar, but we use cubes with side of length ϵ and each point is paired with points from 27 cubes.

Unfortunately, enormous growth of the number of dimensions prohibits direct usage of this method. In case of a 100-dimentional space, the "segments" will be 100-dimensional hypercubes with ϵ-side. Each point will be paired for distance computation with points from 3^{100} hypercubes. Obviously the naïve method will amount to be better for sure in this case.

However, we can apply the abovementioned three-dimensional method in multi-dimensional spaces *indirectly*. If we choose three dimensions i, j, k, we can consider projecting the whole space to them and further search for close pairs like in a three-dimensional space. Let us consider the projection $\Pi = \Pi_i \times \Pi_j \times \Pi_k : \mathcal{R}^n \to \mathcal{R}^3$. Obviously for each pair $x, y \in \mathcal{R}^n$ the following inequality holds:

$$|\Pi(x), \Pi(y)|_3 \leq |x, y|,$$

By $|\ldots|_3$ we denote the Euclidean distance in $\mathcal{R}^3$.

We will assign all points to ϵ-sided cubes according to their projections onto the chosen dimensions. Then, it is enough to pair each point for distance computation only with points residing in the same cube or in 26 neighbouring cubes. If two points are not assigned to the same cube or neighbouring cubes, the distance of their projections onto the chosen dimensions is larger than ϵ. Therefore, so is their distance in $\mathcal{R}^n$ and their pair does not have to be considered.

The efficiency of this method notably depends on the choice of the three projected dimension. Chosen dimensions may reduce the number of distance computations required. Our first idea was to compare the projection onto each single dimension with the uniform distribution using the algorithm based on *Kullback–Leibler divergence* [11]. However, this approach required two scans of input data. Moreover, it is hard to distinguish more dense and more sparse distributions. The problem lies rather in sparseness (with respect to ϵ) that uniformity. As the result, we invented the following quality measure of dimensions.

For a given dimension i and an integer t we define:

$$\sigma(i, t) = |\{x \in \mathcal{M}; t\epsilon \leq \Pi_i(x) < (t+1)\epsilon\}|$$

The number $\sigma(i, t)$ tells how many points will fall into the segment identified by t if we project to the dimension i. Note that if a point x satisfies the condition in the definition above, then $t = \lfloor \Pi_i(x)/\epsilon \rfloor$. $\lfloor a \rfloor$ is the largest integer not greater than a.

For a dimension $i \in \{1, 2, \ldots, n\}$ and an input data point $x \in \mathcal{M}$ we define $t_i(x)$ to be the integer:

$$t_i(x) = \lfloor \Pi_i(x)/\epsilon \rfloor.$$

Observe that the computation of all non-zero values of σ is possible using only a single scan of input data. Moreover, it is easily implementable in the MapReduce computation model. The mapper emits $\langle i, t_i(x) \rangle$ for each input data point $x \in \mathcal{M}$ and for each dimension i. The reducer simply counts the number of occurrences of pairs $\langle i, t \rangle$.

Non-dispersion measure of a dimension i is the number

$$\Delta_i = \sum_{t \in \mathcal{Z}} \sigma(i, t)^2$$

Now, for each dimension we compute its non-dispersion measure by means of the MapReduce procedure presented above. Then we chose three dimensions with smallest non-dispersions.

Assume i, j, k to be the three dimensions with smallest non-dispersion. The algorithm to find all pairs of points closer that ϵ using these dimensions can be easily parallelized using the MapReduce computation model.

The mapper reads all input data points and for each point x emits 27 key-value pairs:

$$\langle\langle t_1, t_2, t_3\rangle, x\rangle$$

where t_1, t_2, t_3 satisfy the following conditions:

$$\begin{array}{l} t_1 \in \{\ t_i(x) - 1,\ t_i(x),\ t_i(x) + 1\ \} \\ t_2 \in \{\ t_j(x) - 1,\ t_j(x),\ t_j(x) + 1\ \} \\ t_3 \in \{\ t_k(x) - 1,\ t_k(x),\ t_k(x) + 1\ \} \end{array}$$

A single reducer collects all pairs that came with a given key $\langle t_1, t_2, t_3\rangle$. If a pair x, y satisfies the conditions $id(x) < id(y)$, $t_1 = t_i(x)$, $t_2 = t_j(x)$, $t_3 = t_k(x)$, then the distance of points x, y is computed. The three dimension equalities assure that x lies in the very ϵ-sided cube $\langle t_1, t_2, t_3\rangle$. If $|x, y| < \epsilon$, the pair $\langle x, y\rangle$ is added to the result.

The collection of all pairs $\langle x, y\rangle$ that satisfy $|x, y| < \epsilon$ is the most time-consuming phase of the construction of the Vietoris-Rips complex. Those pairs are single-dimensional simplexes. The second phase computes triplets $\langle x, y, z\rangle$ such that the three points are pairwise closer than ϵ and $id(x) < id(y) < id(z)$. Following phases consist in identifying longer and longer lists of points that satisfy analogous conditions. It can be done as in the Apriori algorithm to find frequent subsets, because if all points in a set are closer than ϵ that the same must hold for all its subsets. If ϵ is not too big, these phases are not time-consuming. However, it is much simpler to build higher-dimensional simplexes using the following lemma.

Lemma 2 (Raising the dimension of simplexes). *Let R be a Vietoris-Rips complex. A simplex $a := \langle x_1, ..., x_n, x_{n+1}, x_{n+2}\rangle$ belongs to R if and only if all the three following simplexes are also in R:*

$$\begin{array}{l} b := \langle x_1, ..., x_n, x_{n+1}\rangle \\ c := \langle x_1, ..., x_n, x_{n+2}\rangle \\ d := \langle x_{n+1}, x_{n+2}\rangle \end{array}$$

Proof. The "only if" part is straightforward. If $a \in R$, then all its subsimplexes obviously belong to R. Thus, so do b, c, d.

The proof of the "if" part is based on the following property of Vietoris-Rips complexes. If a simplex a belongs to R, then all its one-dimensional subsimplexes also belong to R. Assume a one-dimensional simplex $e := \langle x_j, x_k\rangle \subset a$, where $j, k \in \{1, \ldots, n+2\}$. We will show that $e \in R$.

We have to consider three cases. Firstly, if $j = n+1 \wedge k = n+2$, then $e = d \in R$. Secondly, if $j < n+1 \wedge k = n+2$, then $e \subset c \in R$, thus $e \in R$. Thirdly, if $k < n+2$, then $j < n+1$, thus $e \subset b \in R$ and also $e \in R$.

Since the choice of j and k is arbitrary, the "if" part has been proven. So has been the whole lemma. $\square$

As the result of the first phase of the algorithm, we have a collection *pairs* that contains all pairs $\langle x_i, x_j\rangle$ such that

$$|x_i, x_j| < \epsilon \wedge id_j < id_k$$

We recall that id_i is the identifier of the point x_i.

In the second phase of the algorithm, we build collections of simplexes of subsequent dimensions n that belong to the Vietoris-Rips complex R. The algorithm stops when for a given n we get the empty list of simplexes.

For a given n, we search for all b, c and d that satisfy the constraints of Lemma 2. The symbols introduced in Lemma 2 will prove to be handy again. Therefore, we perform the following steps:

1. We convert the collection of simplexes $\langle x_1, x_2, \ldots, x_{n+1}\rangle$ of the dimension n to the collection of pairs $\langle\langle x_1, x_2, \ldots, x_n\rangle, x_{n+1}\rangle$.
2. We compute a natural self-join of this set of pairs using the first coordinate. As the result we get the set of all triplets: $\langle\langle x_1, x_2, \ldots, x_n\rangle, x_j, x_k\rangle$ (a as in Lemma 2) such that there are pairs $\langle\langle x_1, x_2, \ldots, x_n\rangle, x_j\rangle$ (b) and $\langle\langle x_1, x_2, \ldots, x_n\rangle, x_k\rangle$ (c) in the previous step.
3. We leave (select) only those triples that satisfy $id_j < id_k$.
4. We equi-join the remaining triplets with the set of all pairs using the last two coordinates of the triplets. We do it to assure that the last two coordinates of triplets (d) form a one-dimensional simplex belonging to the complex.
5. We flatten remaining triplets to get complexes of the dimension $n + 1$ of the form $\langle x_1, x_2, \ldots, x_n, x_j, x_k\rangle$.

Note, that this procedure contains only operations that are expressible in relational algebra (or calculus). Therefore, the whole algorithm is formulated in a highly abstract language that is known to be easy to optimise and execute in parallel. We exploited this possibility and implemented it in the Apache Spark for efficient execution on large computing infrastructures of commodity computers. An example Python code that implements the whole algorithm is shown below.

```
komplex = []
komplex.append(points.map(lambda p: p[0]))
komplex.append(pairs)

# initial
base = pairs
pairsAsKey = pairs.map(lambda p: (p, 1))
noAnylizedObjects = pairs.count()

#iterated
while noAnylizedObjects > 0:
    base = base.map(lambda a:(a[:-1], a[-1]))
    base = base.join(base) \
               .filter(lambda a: a[1][0] < a[1][1]) \
               .map(lambda a: (a[1] , a[0])) \
               .join(pairsAsKey) \
            .map(lambda a:rigthTupleSum(a[1][0], a[0]))
    noAnylizedObjects = base.count()
    if noAnylizedObjects > 0:
        komplex.append(base)
```

5 Conclusions

In this article we have presented an efficient method to build Vietoris-Rips complexes. Moreover, this method is easily parallelizable using the MapReduce computation model or Apache Spark. Further research will focus on algorithms that reduce constructed complexes. It will facilitate finding homology types of complexes. As a result it will also allow assessing of the correctness of the selection of the ϵ value.

References

1. Carlsson, G., Ishkhanov, T., De Silva, V., Zomorodian, A.: On the local behavior of spaces of natural images. International journal of computer vision **76** (2008) 1–12
2. De Silva, V., Ghrist, R.: Coverage in sensor networks via persistent homology. Algebraic & Geometric Topology **7** (2007) 339–358
3. Ghrist, R.: Barcodes: the persistent topology of data. Bulletin of the American Mathematical Society **45** (2008) 61–75
4. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. Discrete and Computational Geometry **28** (2002) 511–533
5. Zomorodian, A., Carlsson, G.: Computing persistent homology. Discrete & Computational Geometry **33** (2005) 249–274
6. Nicolau, M., Levine, A.J., Carlsson, G.: Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. Proceedings of the National Academy of Sciences **108** (2011) 7265–7270
7. Freedman, D., Chen, C.: Algebraic topology for computer vision. Computer Vision (2009) 239–268
8. Vejdemo-Johansson, M., Pokorny, F.T., Skraba, P., Kragic, D.: Cohomological learning of periodic motion. Applicable Algebra in Engineering, Communication and Computing **26** (2015) 5–26
9. Dłotko, P., Ghrist, R., Juda, M., Mrozek, M.: Distributed computation of coverage in sensor networks by homological methods. Applicable Algebra in Engineering, Communication and Computing **23** (2012) 29–58
10. Ćwiszewski, A., Wiśniewski, P.: Coverage verification algorithm for sensor networks. In: Computer Applications for Bio-technology, Multimedia, and Ubiquitous City. Springer (2012) 397–405
11. Amari, S.I., Cichocki, A., Yang, H.H., et al.: A new learning algorithm for blind signal separation. Advances in neural information processing systems (1996) 757–763