Master Operations Research and Combinatorial Optimization
Master Informatique / Master Mathématiques & Applications

Solving vehicle routing problem problems by finding the homological groups of the graph modified by force layout

Sidorets Kirill

June 27 2023

Research project performed at DISP laboratory

Under the supervision of:
Dr. Guillaume Bouleux, Université Jean Monnet, DISP-lab
Dr. Lorraine Trilling, INSA Lyon, DISP-lab

Defended before a jury composed of:
Professor Nadia Brauner-Vettier
Professor CUNG Van Dat
Dr. Francis Lazarus

[June] year 2023

Contents

— 3 —
Computation homology generators

We aim to investigate the use of topological features of the vehicle routing problem (VRP) input, as solution routes of this problem. This chapter reviews the methods and algorithms used and the background knowledge that is required to compute homology generators (nontrivial loops on a surface) for the most persistent features of topological space.

3.1 Preliminaries

3.1.1 Topology

Topology is a branch of mathematics that studies the property of the shapes that are invariant under continuous deformation such as stretching, twisting, or bending, but not tearing or gluing [1]. For example, a donut (Solid torus) could be transformed continuously into a coffee mug, therefore they are topologically equivalent (Figure 3.1).

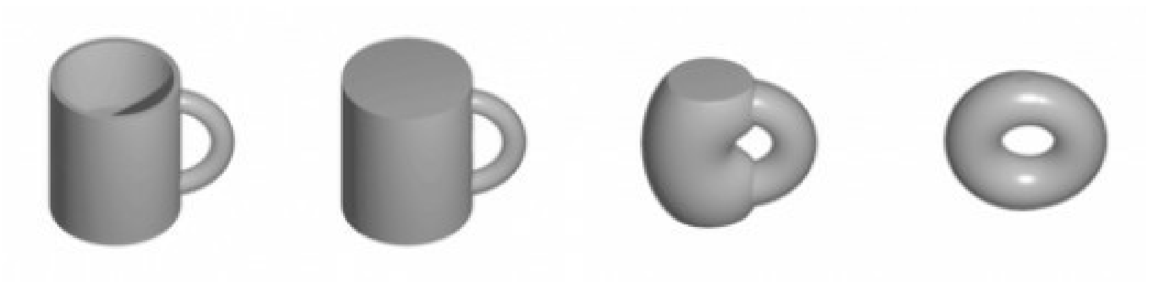


Figure 3.1: Coffee mug to donut transformation example.

Definition 3. *Topological invariants are properties of shapes that remain unchanged under continuous deformations of the space[62].*

Extracting topological invariants and counting them or associating them with algebraic structures (such as vector spaces) is studied in the algebraic topology area. Topological invariants may be used to detect whether two shapes are topologically equivalent. For example, the number of holes (number of closed intersecting curves) in a solid tour and cup (Figure 3.1) is equal to 1.

Definition 4. *In mathematics, homology is a general way of associating a sequence of algebraic objects, such as abelian [1] groups or vector spaces, with other mathematical objects such as topological spaces. [1]*

In general, it is very difficult to compute the homology of a topological object. But instead of computing homology directly, we can approximate a topological object with a simplicial complex and then compute the homology. This approach is called simplicial homology.

3.1.2 Simplicial complex

A simplicial complex is a topological object that consists of a union of points, edges, triangles, tetrahedra, and higher-dimensional polytopes [1, 55, 35] as well as information on how they glued together. A simplicial complex could be divided into parts that are called simplices. A simplex is the simplest possible polytope that can exist in a given dimension and is formally defined as follows:

Definition 5. *A k-simplex is a k-dimensional polytope which is the convex hull of its k + 1 independent points (vertices). We denote simplex with the σ symbol. For example, a 0-simplex is just a point, a 1-simplex is a segment between two points, a 2-simplex is a filled triangle and a 3-simplex is a face of tetrahedra (figure 3.2).*

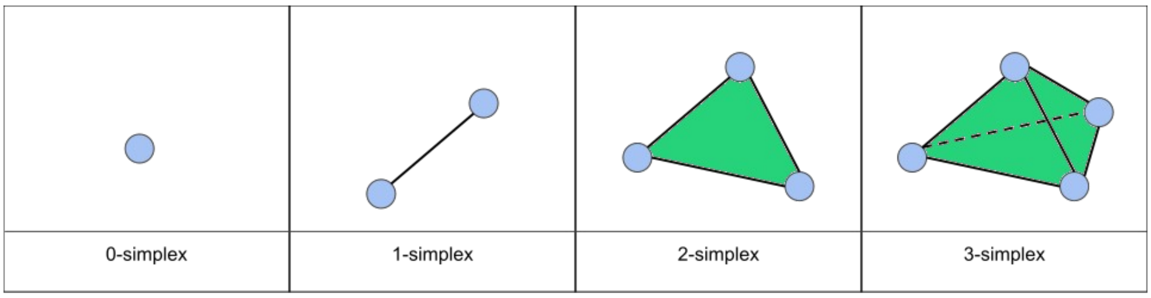


Figure 3.2: Visual example of 0,1,2 and 3 dimensional simplices

Definition 6. *We define the faces of an n-simplex σ to be the (n - 1)-simplices σ' such that the vertices of σ' are all vertices of σ.*

In other words, the faces of an n-simplex $[v_0, . . . v_n]$ are the simplices of the form $[v_0, . . . \hat{v}_i, . . . v_n]$, where $\hat{v}_i$ denotes that we take all vertices of σ except the i-th.

Now we are ready to give the formal definition of the simplicial complex :

Definition 7. *A simplicial complex is a finite collection of simplices K such that:*

- *For every face σ' of a simplex σ, if $\sigma \in K$ then $\sigma' \in K$.*
- *For any two simplices $\sigma_1, \sigma_2 \in K$, if $\sigma_1 \cap \sigma_2 \neq \emptyset$, then $\sigma_1 \cap \sigma_2$ is a common face of both σ_1 and σ_2 .*

[1] An abelian group (A), also called a commutative group, is a group in which the group operation is commutative, i.e. the group operation $\circ$ satisfies $g \circ h = h \circ g \ \forall g, h \in A$. All cyclic groups are abelian but an abelian group is necessarily cyclic.

In a simplicial complex, we can consider the holes of dimension i as voids bounded by i-simplices. For example, holes of dimension 0 are connected components, holes of dimension 1 are loops bounded by edges (1-simplices), and holes of dimension 2 are voids bounded by triangles (2-simplices). For example on figure3.3 you could see 1 dimentional hole bounded by simplexes a,b,c,d,e.

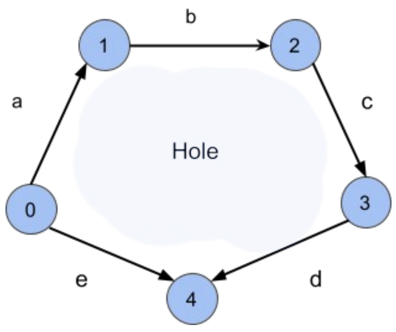


Figure 3.3: 1-dimensional hole example.

3.1.3 Simplicial homology

Informally simplicial homology is the way to find the holes in a simplicial complex, but in order to give a complete definition of simplicial homology we need to define the chains, and two special types of chains, namely cycles, and boundaries.

Definition 8. *Let K be a simplicial complex. Then q-chain is a linear combination of q-simplices with coefficients.*

$$q = \sum n_i \sigma_i$$

Where each σ_i is a q-simplex of K and the coefficients n_i come from a ring.

For example, in figure 3.4(A) We could define 1-chain that is highlighted in orange as 1-chain $o = -a + e$. Similarly 1-chain that is highlighted in black is $b = b - c + *d$ then we could sum b and o and obtain 1-chain that represents hole circle $c = a + b - c + d + e$.

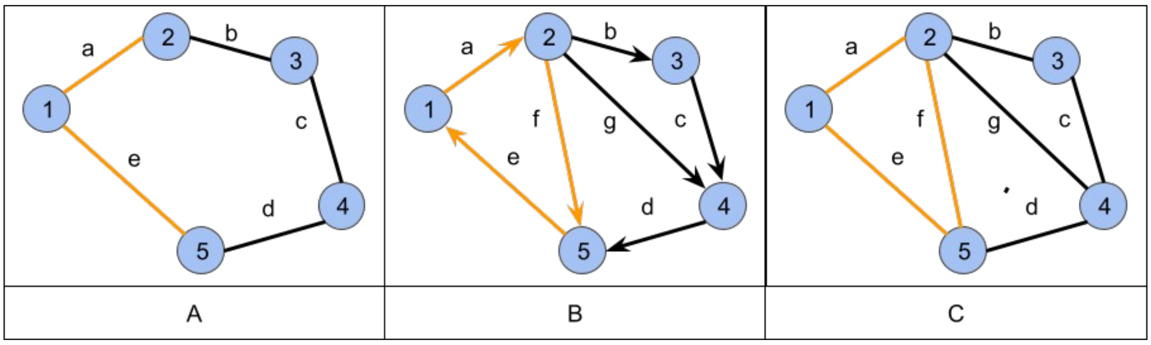


Figure 3.4: 1-dimensional chains example

In this work, we will use two types of coefficients.

- $n \in \{-1, 0, 1\}$ for coefficients that are based on the orientation of the edge of the graph. We use these coefficients to compute homology. For example, we could define chain $1-5-2$ (highlighted in orange on figure 3.4(B)) as $-1 * e - 1 * f - 1 * a$ because of the orientation of edges.

- $n \in \{0, 1\}$ for coefficients that are based on the presence of simplex inside the chain. We use it to compute the persistence of homology features. In this case, we define chain $1-5-2$ (highlighted in orange on figure 3.4(C)) as $1 * e + 1 * f + 1 * a$ because we don't take into account the orientation of edges.

We are imposing order on the vertices of the graph, and we say that edge between vertices v and v' has a direction from vertex v to v' if $ord(v) \le ord(v')$. The ordering of the vertices defines an orientation of the simplex. This orientation is chosen arbitrarily but is fixed.
We define the boundary of an i-simplex as the sum of its (i-1)-dimensional faces, in order to map an i-simplex to an (i-1)-simplex.

Definition 9. *The boundary operator $\delta_n : C_n \longrightarrow C_{n-1}$ is a homomorphism defined linearly on a chain c by its action on any simplex $\sigma = [v_0, v_1, \ldots v_k]$:*

$$\forall \sigma \in C_n;\ \delta_n(\sigma) = \sum_{0 \le i \le n} (-1)^n [v_0, \ldots \hat{v_i}, \ldots v_n]$$

where the hat indicates that v_i is omitted (deleted from the sequence)3.1.3. Thus, we take into account all $n-1$ simplices.

In order to simplify notation we may use just δ instead of δ_n.

Remark 1. *In the case when we didn't impose the direction of the simplices therefore of n-chain coefficients in $\{0, 1\}$ boundary operator have the definition:*

$$\forall \sigma \in C_q;\ \delta_q(\sigma) = \sum_{0 \le i \le n} [v_0, \ldots \hat{v_i}, \ldots v_n]$$

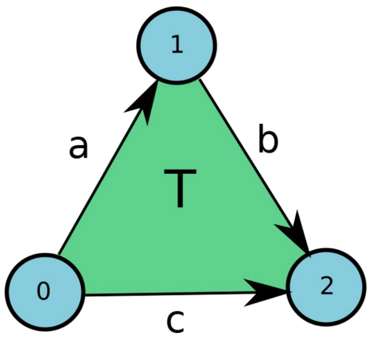


Figure 3.5: Example of simplicial complex

Example: boundary operation on 1-simplices of figure 3.5
$\delta(a) = -'0' + '1'$, $\delta(b) = -'1' + '2'$, $\delta(c) = -'0' + '2'$ and on 2-simplex T: $\delta(T) = a + b - c$. If we took boundary of $a + b - c$ we will get a 0 : $\delta_0\delta_1(T) = \delta_0(a + b - c) = -'0' + '1' - '1' + '2' + '0' - '2' = 0$. This illustrates the fundamental property of the boundary operator:

$$\delta_k \delta_{k+1} = 0$$

Definition 10. *The chain complex is the sequence of chain groups connected by boundary operation:*

$$\cdots \to C_{k+1} \xrightarrow{\delta_{k+1}} C_k \xrightarrow{\delta_k} C_{k-1} \xrightarrow{\delta_{k-1}} \cdots$$

Now we could define two special families of q-chains:

Definition 11. *A p-cycle is p-chain γ such that $\delta\gamma = 0$. The collection of p-cycles, denoted by Z_p, forms a subgroup of C_p, and is called the p-th cycle group.*

Definition 12. *A p-boundary is a p-chain γ that is the boundary of a (p + 1)-chain; i.e, there exists $c \in C_{p+1}$ such that $\gamma = \delta_{p+1}(c)$. The collection of p-boundaries, denoted by B_p, forms a subgroup of C_p called the p-th boundary group.*

Property 1. $B_p \subseteq Z_p \subseteq C_p$

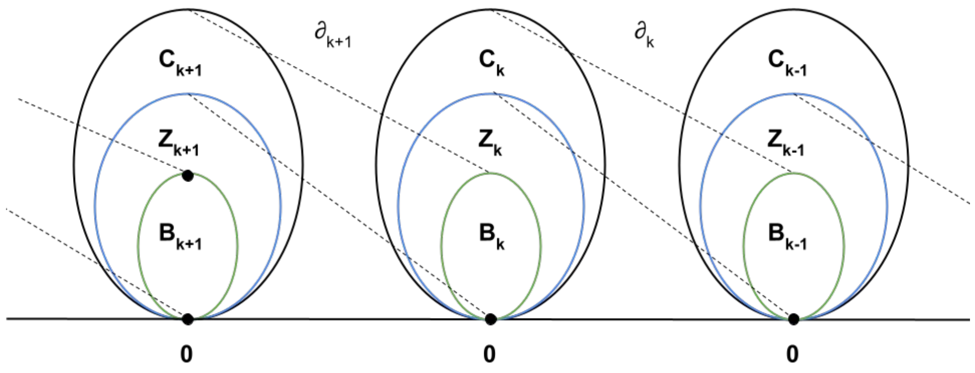


Figure 3.6: Relation between B_p,Z_p and C_p

Claim 1. *$Z_p = Ker(\delta_p)$ and $B_{p-1} = im(\delta_p)$ for the boundary operator $\delta_p : C_p \to C_{p-1}$.*

Proof of this claim goes directly from the definition of image and kernel.
Since B_p is a subgroup of Z_p, we can take the quotient of the two, and the resulting structure still has a group structure and is called the homology groups, denoted by $H_p = Z_p/B_p$. A quotient group could be described as, if B is a subgroup of A, then one can subdivide A into portions based on B. In particular, two elements $x, y \in A$ are considered to be in the same portion if $x - y \in B$. Each such collection is called a coset. The collection of cosets is the quotient group.

Definition 13. *The p homology group is $H_p = Z_p/B_p$. We call Betti number, the rank of p homology group the p, denoted by $\beta_p = rank(H_p)$.*

3.2 Computation of homology generators

The computation of homology generators consists of multiple steps. This section contains a description of each step as well as the method of their application in order to obtain homology generators.

3.2.1 Executive summary

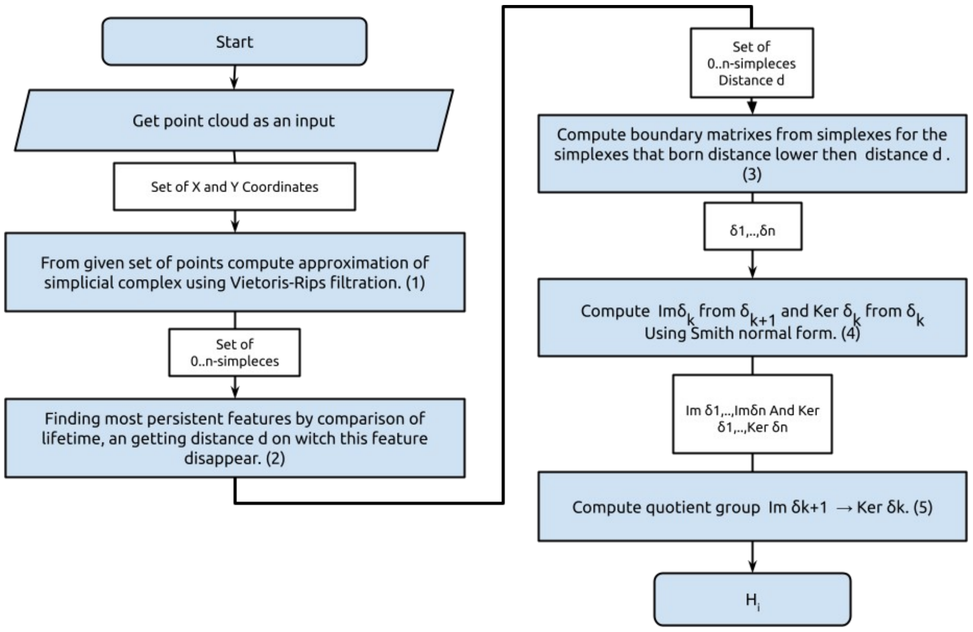

Start
Get point cloud as an input
Set of X and Y Coordinates
From given set of points compute approximation of simplicial complex using Vietoris-Rips filtration. (1)
Set of 0..n-simpleces
Finding most persistent features by comparison of lifetime, an getting distance d on witch this feature disappear. (2)
Set of 0..n-simpleces Distance d
Compute boundary matrixes from simplexes for the simplexes that born distance lower then distance d . (3)
δ1,..,δn
Compute Imδk from δk+1 and Ker δk from δk Using Smith normal form. (4)
Im δ1,..,Imδn And Ker δ1,..,Ker δn
Compute quotient group Im δk+1 → Ker δk. (5)
Hi

Figure 3.7: Sequence of actions required to obtain persistent homology generators.

Computing a homology generator from space of points consists of several steps (Figure 3.7). We start with (Figure 3.7(1)) filtration in order to obtain simplicial complex 3.2.2. This simplicial complex approximately defines the topological space of the input point space. Then (Figure 3.7(2)) we detect the most persistent features of this space 3.2.5. (Figure 3.7(3)) knowing when features in which we are interested are persisting we could compute boundary operation matrix 3.2.3 for these features and after computing (Figure 3.7(4)) image and kernel of this matrices 3.2.6 we could (Figure 3.7(5))compute the homology generators 3.2.8 of the most persistent features, which we then try to use as routes in VRP solution.

3.2.2 Obtaining simplices

The input of the vehicle routing problem contains point cloud X in which each point represents the geographical position of the client that we need to visit. However, homology does not give interesting information about the cloud of points. Because features of this topological space represent connected components, which are just the points defined in the input and we don't have simplices of dimension other than 0. From point cloud X , we need to induce a family of simplicial complexes K_X^k for a range of values of $k \in \mathbf{R}, k \geq 0$. Considering that the complex at step m is embedded in the complex at n for $\forall n, m \in \mathbf{R}$ such that $m \leq n$, i.e. $K_X^m \subseteq K_X^n$. This nested family of simplicial complexes is called filtration. During the construction of this family, some homological features may appear and then disappear. The persistence of these features can be considered as the features of the input. There exist multiple variations of simplicial complex filtration: Vietoris-Rips filtration[61], Dowker sink and source [15] filtration, Cech filtration [40] and etc. [1]. One of the most popular is Vietoris-Rips Filtration.

Definition 14. *In topological data analysis, the Vietoris-Rips filtration is the collection of nested Vietoris-Rips complexes on a metric space created by taking the sequence of Vietoris-Rips complexes over an increasing scale parameter.*

Definition 15. *Suppose that X is a metric space, with metric d that represents the distance between two vertices. For any finite subset S of X , and any $\varepsilon > 0$, we define the ε-Rips complex of the subset S to be the abstract simplicial complex whose vertex set is S, and where a subset $s_0, s_1, \dots s_k$, is a simplex if and only if $d(x_i, x_j) \leq \varepsilon \quad \forall i, j$ so that $0 \leq i, j \leq k$. [13]*

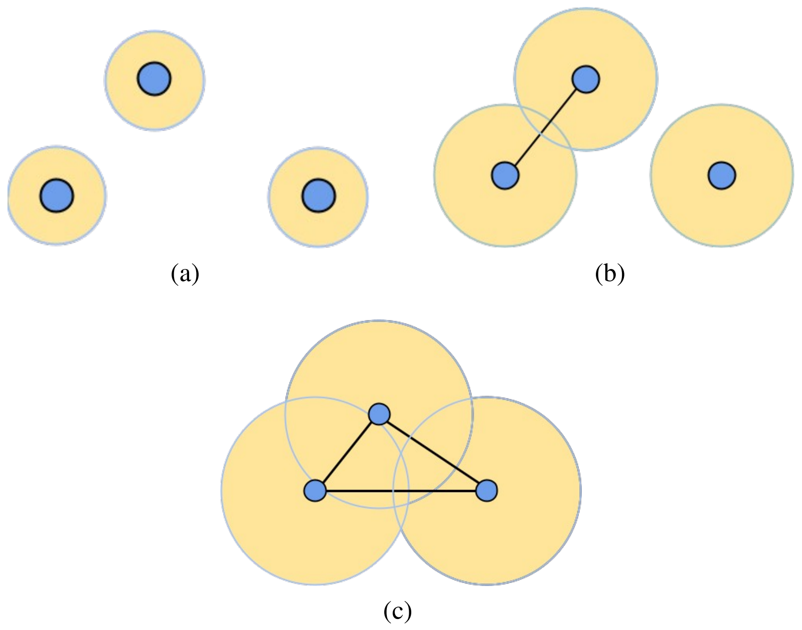


Figure 3.8: Visualization of Vietoris-Rips filtration intermediate steps

More informally around each vertex, we could imagine circle with some radius r (3.8(a)). N vertices form an n-simplex when each pair of this circle have common point. We also say that n-simplex is born at the radius r if $r = max_{\forall i,j \in V}(distance(v_i, v_j))$.

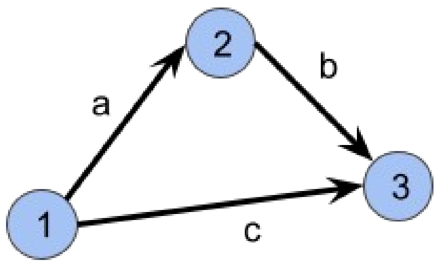
For example on figure 3.8(b) 1-simplex was born, and on figure 3.8(c) 2-simplex was born. We could compute n-simplices, using algorithm 1. This algorithm c computes all simplexes and stores the birth time of each one of them.

Algorithm 1 1-simplex generation

```
Require: $X, Y \leftarrow$ Set of x and y coordinates
  $\forall n \in [0, |X|]\ n-simplices = \emptyset$
  $\forall n \in [0, |X|]\ n-bornDistance = \emptyset$
  for $i \in [0, |X|]$ do
      for each unique combination of i vertices c=($v_1, \dots v_i$) do:
          $i-simplices = i-simplices \cup c$
          $i-bornDistance = i-bornDistance \cup \max_{(v_i,v_j) \subset c}\{distance(v_i, v_j)\}$
      end for
  end for
```

3.2.3 Constructing boundary matrix

We represent the boundary operator $\delta_k : C_k \rightarrow C_{k-1}$ relative to the standard bases of the chain groups as an integer matrix M_k with entries in $\{-1,0,1\}$ [74] or $\{0,1\}$. The matrix M_k is the standard matrix representation of δ_k. The number of columns in M_k equals the number of k-simplices, and the number of rows equals to the number of $(k-1)$-simplices.



2-simplex / 1-simplex	a	b	c
"1"	-1	0	1
"2"	1	-1	0
"3"	0	1	1

Figure 3.9: Boundary matrix example.

For example, we have a small simplicial complex that consists of 3 0-simplices and 3 1-simplices (figure 3.9). Than we could construct boundary matrix that represents $\delta_1 : C_1 \rightarrow C_0$ (figure 3.9).

3.2.4 Matrix operations

The action of a linear map is most transparent when it is presented in terms of bases in which it has a diagonal matrix [36]. With this in mind, we need to be able to represent the boundary matrix in diagonal form. We could achieve it using Smith's normal form algorithm 3.2.4. But first, we define elementary matrix operation that we may apply to the matrix.

- Row operations correspond to changing a basis for the co-domain.
  - exchange row_i and row_j
  - multiply row_i by -1.
  - replace row_i by $rowi$ + q*row_j.
- Column operations correspond to changing a basis for the domain
  - exchange $column_i$ and $column_j$.
  - multiply $column_i$ by -1.
  - replace $column_i$ by $column_i$ + q*$column_j$.

Smith Normal Form

For any matrix A, Smith's normal form allows to obtain basis representation in terms of matrix B, where B has a diagonal form.

We extend matrix A with identity matrix I_r on the left ($|I_r| =$number of rows) to keep track of row operations, and with identity matrix I_c down ($|I_c| =$number of columns) to keep track of column operation. It means that we will work with the following augmented matrix:

$$\begin{array}{cc} I_r & A \\ 0 & I_c \end{array}$$

We perform the operation only on A. The zero matrix in the lower left corner is there to complete the expression to a square matrix, and it never changes. At the final stage, we obtain a matrix:

$$\begin{array}{cc} P & B \\ 0 & R \end{array}$$

Where B is the diagonal form matrix, Q is the row operation matrix and R is the Column operation matrix. Then we could express B as $B = QAR$. Note: we will as well store matrices $\bar{Q}$ and $\bar{R}$ that will contain invert matrices of Q and R. Hence $\bar{Q} = Q^{-1}$ and $\bar{R} = R^{-1}$. $\bar{Q}$,$\bar{R}$ are initialized as diagonal matrices with the same dimension as Q and R respectively. If we apply an elementary operation on Q, we apply an inverse operation on $\bar{Q}$, for example, if we apply row exchange on Q then we apply column exchange on$\bar{Q}$. We have the same behavior between $\bar{R}$ and R. Then we reduce A to diagonal form using algorithm 6 that was proposed in [36]. Note: this is a simplified version of the algorithm that shows the reduction process with the omission of details about saving operations. For more detailed information, the reader can refer to [36]. First of all, we define some utility functions such as :

- partRowReduce(A,k,l) - Reduce sub row R of matrix A_{nxm} , $R = A[k, l:]$ to form in witch $\forall i \in \{1, |R|\} R[i] < R[0]$ 2.

Algorithm 2 partRowReduce

```
Require: A ← matrix.
Require: k ← index of the row.
Require: l ← index of the column from which the sub-string starts.
  for from i = k+1 to numberOfRows(A) do
      q = floor($A[i,l]/A[k,l]$)
      To row $i$ add row k multiplied by -q.
  end for
```

- partColumnReduce(A,k,l) - Reduce sub column C of matrix A_{nxm} , $C = A[k:, l]$ to form in witch $\forall i \in \{1, |C|\} C[i] < C[0]$ 3.

Algorithm 3 partColumnReduce

```
Require: A ← matrix.
Require: k ← index of the row from which the sub-string starts.
Require: l ← index of the column.
  for from i = l+1 to numberOfColumns(A) do
      q = floor(A[k,i]/A[k,l])
      To column $i$ add column l multiplied by -q.
  end for
```

- checkForDivisibility(A,k,l) - get matrix A_{nxm} and integer k,l and return false and i,j if $A[k][l]$ do not divide $A[i][j]$ $\forall i \in \{k, n\}$ and $\forall j \in \{l, m\}$, or return true if $A[k][k]$ dived any entry in sub matrix $A[k:][k:]$ 0.

Algorithm 4 checkForDivisibility

```
Require: A ← matrix.
Require: k ← index of the row.
Require: l ← index of the column.
  for from i = k to numberOfRows(A) do
    for from j = k to numberOfColumns(A) do
      if $A[i][j] \mod A[k][k] \neq 0$ then
        return (False,i,j)
      end if
    end for
  end for
  return True
```

With these functions, we now can define function partSmithNormalForm 5 which reduces the segment of the matrix (highlighted with orange color on figure 3.10). The function takes matrix A and index k as input after execution of all elements of sub row $A\ [k,k+1:]$ and sub-column $A[k+1:,k]$ equal to zero, and $A[k][k]$ divides all entry's of sub-matrix $A[k:,k:]$.

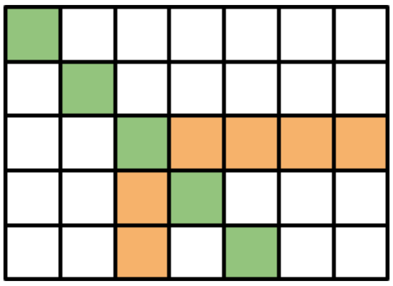


Figure 3.10: Smith normal form explanation support image.

In order to achieve this result we move entry with minimum non-zero absolute value at the position k,k, and reduce sub-row $A\ [k,k:]$. Since on each iteration of partRowReduce 2 all entries of $A\ [k,k+1:]$ will be lower the $A\ [k,k]$ and on each iteration, we move entry with minimum non zero absolute value at the position k,k, eventually all entries of $A\,[k,k+1:]$ will become equal to 0. Similarly, we then reduce sub-column $A\ [k:,k]$. And then we verify that $A[k][k]$ divides all entries of sub-matrix $A[k:,k:]$, if $A[k,k]$ don't divide $A[i,j]$ using elementary matrix operations we change it to be equal to $A[i,j] mod A[k,k]$.

```
Algorithm 5 partSmithNormalForm
Require: A ← matrix.
Require: k ← index of the row.
Require: l ← index of the column.
  do
      i,j = coordinates of entry with minimum non-zero absolute value
      Exchange rows i and k
      Exchange columns j and k
      partRowReduce(A)
      if $B[k+1:,k] \neq 0$ then
          continue
      end if
      partColumnReduce(A)
      if $B[k,k+1:] \neq 0$ then
          continue
      end if(divisible, i, j, q) := checkForDivisibility(A, k)
      if not divisible then
          add row i to row K
          add column $i * q$ to row K
      end if
  while divisible
```

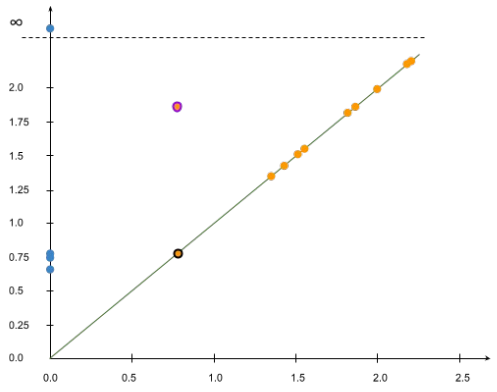
Then we apply partSmithNormalForm 5 for each diagonal entry of matrix A (highlighted with green color on figure 3.10).

```
Algorithm 6 SmithNormalForm
  $t \leftarrow -1$
  while $A[t+1:,t+1:] \neq 0$ do
      t++
      partSmithNormalForm(A)
      if $A[t,t] < 0$ then
          multiply row t by -1
      end if
  end while
```

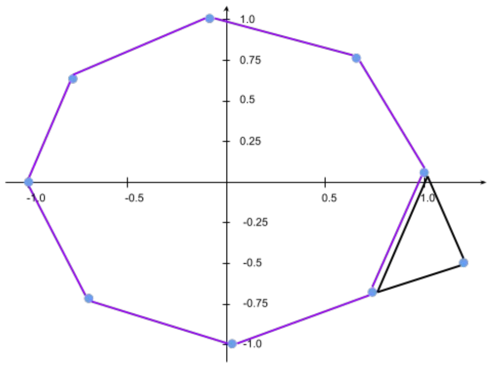
This approach is proven to give Smith normal form [36] however could be inefficient in terms of time complexity. There exists a more efficient implementation of Smith normal form computation [63, 38].

3.2.5 Compute persistence of features

During the construction of the Vietoris-Rips simplicial complex multiple 1-simplex holes appeared but all of them eventually disappear due to the creation of higher diminution simplices. We want to study the persistence of chains to find the moment when they disappear. This will give us information about the length of the life of chains. In order to visualize the length of the life of the feature we use a persistence diagram, in which features are represented by points, features of different dimensions have different colors, abscissa corresponds to the distance on which the feature was born, and ordinate corresponds to the distance on which feature disappears. Chains with long life lengths represent persistent holes and chains with relatively short life lengths represent noise.



(a) Persistence diagram



(b) Graph.

Figure 3.11

For example in figure 3.11(b) we could see that a more persistent hole is highlighted with purple color, which characterizes space better than a hole that is highlighted with black color. And we could see that the purple feature has the best persistence (point with purple border on 3.11(a)) when other features disappear immediately after they are born. In order to compute the life length of the features we will use the algorithm presented in [74].

Boundary matrix

First of all, we need to compute the boundary matrix. The paper proposes a Z modulo 2 coefficient to describe chains in finding the persistence of the features. We construct the boundary matrix for the hole simplicial complex, which means that the domain and co-domain of this matrix contain all simplices. Simplices are arranged in ascending order of born distance. If two simplices have the same born distance we arrange them in ascending order of dimension of the simplex.

Reduction

To obtain bar-codes [2] we need to reduce the boundary matrix that we initialize in the previous section in such form that if any column i has a pivot value at row j this means that there is no column k such that $k < i$ and pivot of column k is in a row j.
In order to reduce the matrix to this form we iterate through all columns from left to right, and verify that the pivot of this column is unique. If we find that column i has the same pivot as column k and $k < i$, then we change column i to (column i - column k) and verify again that after the summation pivot of column i is unique if no we repeat the process.

[2]We call bar-codes - segments description that has start and end positions, which in this case describe the distance at which the feature is born and the distance at which the feature vanishes.

Algorithm 7 Reduction

```
Require: B = boundary matrix
  i = 0
  while i < |B| do
    if There exists column k (k<i) with a pivot in the same row as column i then
      Column i = Column i - Column k
    else
      i++
    end if
  end while
```

Obtaining bar-codes from reduced matrix

A column that equals zero indicates the birth of the feature (start of the bar code). If column *i* equals zero in order to get the death of this feature (end of bar code) we need to find the column that has a pivot in the row *i*, if there is no such column that means that there is no death point and bar-code goes to infinity. Hence we present an algorithm that computes all bar codes.

Algorithm 8 Barcodes

```
  $L = \emptyset$
  for i from 0 to |B| do
    if column i = 0 then
      q = Column With pivot in row i
      if $q \neq -1$ then
        $L = L \cup \{Birth_i, Birth_q\}$
      else
        $L = L \cup \{Birth_i, infinity\}$
      end if
    end if
  end for
```

3.2.6 Finding image and kernel of boundary matrices

From definition (13) we know that $H_p = Z_p/B_{p-1}$. Hence knowing Z_p and B_p, we could obtain H_p by computing the quotient group. Z_p is collection of cycles and equals to $Ker(\delta_p)$, B_{p-1} is collection of boundaries and equals to $im(\delta_p)$ 3.1.3. Therefore we need to compute the image and kernel of the boundary matrix.

Using 6 we could compute Smith normal form of δ_{p-1} ($B = P\delta_{p-1}R$). In this case, the kernel of δ_{p-1} is represented by R[:,k+1:] where k - number of non-zero columns (highlighted with green color on figure 3.12).

An important note we should remember is that $C_{k-1} \rightarrow C_k \rightarrow C_{k+1}$ that means that domain of δ_{k-1}, is co-domain of δ_k, Which means that before finding an image from delta k we need to apply on rows of delta k transformations that we apply on columns of δ_{k-1}. In order to do this we could simply multiply δ_k by $\bar{R}$. Then using 6 we could compute Smith normal form of δ_k', δ_p ($B = P\delta_p R$). In this case image δ_{p-1} is represented by P[:r,:] where r is the number of non-zero rows (highlighted with yellow color on figure 3.12).

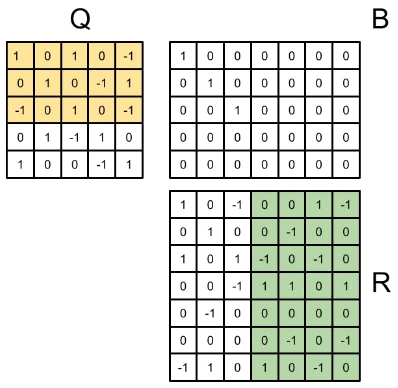


Figure 3.12: Kernel and image.

3.2.7 Computing quotient group

Now we want to define the last tool that we will need to compute quotients of finitely generated free abelian groups. This will be the application of the Smith normal form algorithm to the issue of solvability of linear systems of equations Ax = b under the restriction that all the terms must be integers.

Algorithm 9 Solve

```
Require: A = matrix
Require: b = vector
  $m = numberOfRows(A)$
  $(B, Q, \bar{Q}, R, \bar{R}, s, t)$ = smithForm(A);
  c = $\bar{Q}$* b;
  vector u;
  for i := 1 to t do
     if B[i, i] divides c[i] then
        u[i] = c[i]/B[i, i];
     else
        return "Failure";
     end if
  end for
  for i := t + 1 to m do
     if c[i] $\neq 0$ then then
        return "Failure"
     else
        u[i]:= 0;
     end if
  end for;
  return R*u;
```

In [36] Theorem 3.55 was proven that this algorithm 9 returns solution to linear system of equation Ax=b if it exists, otherwise returns "Failure".

Now we could introduce algorithm 10 that computes the quotient group of finitely generated free abelian groups.

Algorithm 10 quotientGroup

```
Require: W,V matrices
  n= numberOfColumns(V);
  matrix A;
  for i := 1 to numberOfColumns(V) do
      A[i] := Solve(W, V[i]);
  end for
  $(B,Q,\bar{Q},R,\bar{R},s,t)$ := SmithForm(A);
  U:= W* Q;
  return (U,B,s);
```

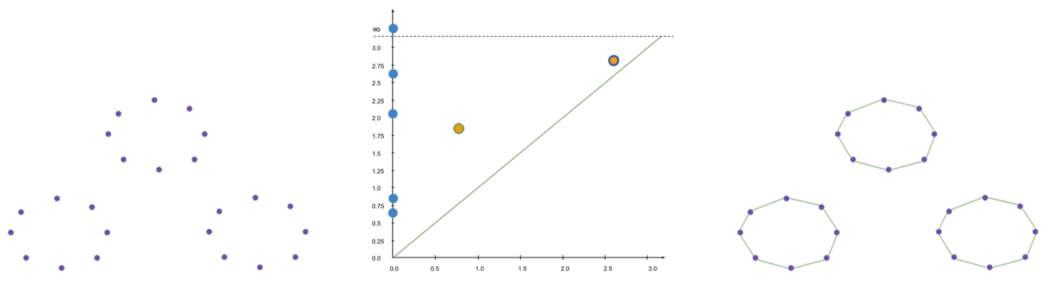
3.2.8 Getting homology generators

In [36] Theorem 3.59 was proven that this algorithm 10 indeed computes the quotient group. And that columns from $n+1$ to m where n is a rank of B, and m is a rank of U represent bases of homology generators. Therefore we could obtain a homology group H_k by computing the quotient group of $Ker(\delta_k)$ and $Im(\delta_{k+1})$.

Now we could define the whole sequence of actions that we need to commit in order to obtain homology generators from space of points (figure 3.7).

1. We approximate the simplicial complex of point space using Vietoris-Rips filtration.
2. We compute the persistence of features of simplicial complex and chose the maximal birth distance of simplex in order to look at homology generators of the most persistent features (holes).
3. We compute δ_1 and δ_1 boundary matrix for simplices with birth distance lower than the distance chosen in the previous step.
4. We compute the image and kernel of δ_1
5. We obtain a homology generator from the quotient group of kernel and image.

For example, if for point space shown in figure3.13 (a), we compute the persistence diagram (figure3.13 (b)), we could see that born persistent holes were born at a distance 0.8. Then by computing homology generators for this distance, we could detect 3 circles(figure3.13 (c)) that form this point space.



(a) (b) (c)

Figure 3.13: Example

Bibliography

[1] Mehmet E. Aktas, Esra Akbas, and Ahmed El Fatmaoui. Persistence homology of networks: methods and applications. *Applied Network Science*, 4, 12 2019.

[2] Enrique Alba and Bernabé Dorronsoro. Computing nine new best-so-far solutions for capacitated vrp with a cellular genetic algorithm. *Information Processing Letters*, 98(6):225–230, 2006.

[3] İ K Altınel and T Öncan. A new enhancement of the clarke and wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 56:954–961, 8 2005.

[4] T. Antal and P. L. Krapivsky. Weight-driven growing networks. *Physical Review E*, 71:026103, 2 2005.

[5] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, and D. Naddef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106(2):546–557, 1998.

[6] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52:723–738, 2004.

[7] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218:1–6, 4 2012.

[8] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. Modeling the evolution of weighted networks. *Physical Review E*, 70:066149, 12 2004.

[9] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4:235–282, 10 1994.

[10] Omer Bobrowski and Matthew Kahle. Topology of random geometric complexes: a survey. *Journal of Applied and Computational Topology*, 1:331–364, 6 2018.

[11] Zuzana Borčinova. Two models of the capacitated vehicle routing problem. *Croatian Operational Research Review*, 8:463–469, 12 2017.

[12] Alex Van Breedam. Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86:480–490, 11 1995.

[13] ERIK CARLSSON, GUNNAR CARLSSON, and VIN DE SILVA. An algebraic topological method for feature identification. *International Journal of Computational Geometry & Applications*, 16:291–314, 8 2006.

[14] Ruey-Maw Chen, Fu-Ren Hsieh, and Di-Shiun Wu. Heuristics based ant colony optimization for vehicle routing problem. In *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1039–1043, 2012.

[15] Samir Chowdhury and Facundo Mémoli. Persistent homology of asymmetric networks: An approach based on dowker filtrations. *arXiv preprint arXiv:1608.05432*, 2016.

[16] Nicos Christofides. The vehicle routing problem. *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, 10(V1):55–70, 1976.

[17] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 8 1964.

[18] Carlos D. Correa, Deborah Silver, and Min Chen. Constrained illustrative volume deformation. *Computers & Graphics*, 34(4):370–377, 2010. Procedural Methods in Computer Graphics Illustrative Visualization.

[19] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 10 1959.

[20] Tamal K. Dey, Anil N. Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. 1 2010.

[21] Jean-Guillaume Dumas, Frank Heckenbach, David Saunders, and Volkmar Welker. Computing simplicial homology based on efficient smith normal form algorithms. pages 177–206. Springer Berlin Heidelberg, 2003.

[22] Cody Dunne and Ben Shneiderman. Improving graph drawing readability by incorporating readability metrics: A software tool for network analysts. *University of Maryland, HCIL Tech Report HCIL-2009*, 13, 2009.

[23] Emerson G. Escolar and Yasuaki Hiraoka. Optimal cycles for persistent homology via linear programming. pages 79–96. 2016.

[24] Sami Faiz, Saoussen Krichen, and Wissem Inoubli. A dss based on gis and tabu search for solving the cvrp: The tunisian case. *The Egyptian Journal of Remote Sensing and Space Science*, 17(1):105–110, 2014.

[25] Robin Forman. A user's guide to discrete morse theory. *Séminaire Lotharingien de Combinatoire [electronic only]*, 48:B48c–35, 2002.

[26] David Forrester, Stephen G Kobourov, Armand Navabi, Kevin Wampler, and Gary V Yee. graphael: A system for generalized force-directed layouts. In *Graph Drawing: 12th International Symposium, GD 2004, New York, NY, USA, September 29-October 2, 2004, Revised Selected Papers 12*, pages 454–464. Springer, 2005.

[27] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[28] Petr Gajdoš, Tomáš Ježowicz, Vojtěch Uher, and Pavel Dohnálek. A parallel fruchterman–reingold algorithm optimized for fast visualization of large graphs and swarms of data. *Swarm and evolutionary computation*, 26:56–63, 2016.

[29] Yuvraj Gajpal and P.L. Abad. Multi-ant colony system (macs) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, 196(1):102–117, 2009.

[30] Michel Gendreau, Alain Hertz, and Gilbert Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40:1086–1094, 12 1992.

[31] Helen Gibson, Joe Faith, and Paul Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information visualization*, 12(3-4):324–357, 2013.

[32] Geir Hasle and Oddvar Kloster. Industrial vehicle routing. pages 397–435. Springer Berlin Heidelberg.

[33] Stefan Irnich, Birger Funke, and Tore Grünert. Sequential search and its application to vehicle-routing problems. *Computers & Operations Research*, 33:2405–2429, 8 2006.

[34] David S Johnson and Lyle A McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1(1):215–310, 1997.

[35] Jakob Jonsson. *Simplicial complexes of graphs*, volume 3. Springer, 2008.

[36] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. Computing homology groups. pages 93–141. 2004.

[37] Matthew Kahle. Random geometric complexes. *Discrete & Computational Geometry*, 45:553–573, 4 2011.

[38] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8(4):499–507, 1979.

[39] R.H. Katz. A database approach for managing vlsi design data. pages 274–282. IEEE, 1982.

[40] Michael Kerber and R. Sharathkumar. Approximate čech complex in low and high dimensions. In Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam, editors, *Algorithms and Computation*, pages 666–676, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[41] Suresh Nanda Kumar and Ramasamy Panneerselvam. A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, 04:66–74, 2012.

[42] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 6 1992.

[43] Gilbert Laporte, Stefan Ropke, and Thibaut Vidal. Chapter 4: Heuristics for the vehicle routing problem. pages 87–116. Society for Industrial and Applied Mathematics, 11 2014.

[44] SEUNG-YONG LEE, KYUNG-YONG CHWA, JAMES HAHN, and SUNG YONG SHIN. Image morphing using deformation techniques. *The Journal of Visualization and Computer Animation*, 7:3–23, 1 1996.

[45] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

[46] Fei Liu, Chengyu Lu, Lin Gui, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. Heuristics for vehicle routing problem: A survey and recent advances. 3 2023.

[47] Silvia Mazzeo and Irene Loiseau. An ant colony algorithm for the capacitated vehicle routing. *Electronic Notes in Discrete Mathematics*, 18:181–186, 2004.

[48] Mazin Abed Mohammed, Mohd Sharifuddin Ahmad, and Salama A. Mostafa. Using genetic algorithm in implementing capacitated vehicle routing problem. In *2012 International Conference on Computer & Information Science (ICCIS)*, volume 1, pages 257–262, 2012.

[49] Mazin Abed Mohammed, Mohd Khanapi Abd Ghani, Raed Ibraheem Hamed, Salama A. Mostafa, Dheyaa Ahmed Ibrahim, Humam Khaled Jameel, and Ahmed Hamed Alallah. Solving vehicle routing problem by using improved k-nearest neighbor algorithm for best solution. *Journal of Computational Science*, 21:232–240, 2017.

[50] Habibeh Nazif and Lai Soon Lee. Optimised crossover genetic algorithm for capacitated vehicle routing problem. *Applied Mathematical Modelling*, 36(5):2110–2117, 2012.

[51] Andreas Noack. Energy models for graph clustering. *J. Graph Algorithms Appl.*, 11(2):453–480, 2007.

[52] Ippei Obayashi. Volume-optimal cycle: Tightest representative cycle of a generator in persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 2:508–534, 1 2018.

[53] Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 12 1993.

[54] Paola Pellegrini. Application of two nearest neighbor approaches to a rich vehicle routing problem. In *Technical report, TR/IRIDIA/2005-15*, page 2005. IRIDIA, Université Libre de Bruxelles, 2005.

[55] Jovan Popović and Hugues Hoppe. Progressive simplicial complexes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 217–224, 1997.

[56] Jean-Yves Potvin and Jean-Marc Rousseau. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46:1433–1446, 12 1995.

[57] E. Promayon, P. Baconnier, and C. Puech. Physically-based deformations constrained in displacements and volume. *Computer Graphics Forum*, 15:155–164, 8 1996.

[58] HELEN C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13:501–516, 10 2002.

[59] T.K. Ralphs, L. Kopman, W.R. Pulleyblank, and L.E. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94:343–359, 1 2003.

[60] Nicholas A Scoville. *Discrete Morse Theory*, volume 90. American Mathematical Soc., 2019.

[61] Donald R. Sheehy. Linear-size approximations to the vietoris-rips filtration. pages 239–248. ACM, 6 2012.

[62] David S Simon. Topological invariants. In *Tying Light in Knots*, 2053-2571, pages 5–1 to 5–11. Morgan & Claypool Publishers, 2018.

[63] Arne Storjohann. Near optimal algorithms for computing smith normal forms of integer matrices. In *Proceedings of the 1996 international symposium on Symbolic and algebraic computation*, pages 267–274, 1996.

[64] Ashley Suh, Mustafa Hajij, Bei Wang, Carlos Scheidegger, and Paul Rosen. Persistent homology guided force-directed graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):697–707, 2019.

[65] N Suthikarnnarunai. A sweep algorithm for the mix fleet vehicle routing problem. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 2, pages 19–21. Citeseer, 2008.

[66] Yi Tao and Fan Wang. An effective tabu search approach with improved loading algorithms for the 3l-cvrp. *Computers & Operations Research*, 55:127–140, 2015.

[67] J. E. M. Teoh, J. An, C. K. Chua, M. Lv, V. Krishnasamy, and Y. Liu. Hierarchically self-morphing structure through 4d printing. *Virtual and Physical Prototyping*, 12:61–68, 1 2017.

[68] Daniela Thyssens, Jonas Falkner, and Lars Schmidt-Thieme. Supervised permutation invariant networks for solving the cvrp with bounded fleet size. 1 2022.

[69] Paolo Toth and Daniele Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15:333–346, 11 2003.

[70] William Thomas Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767, 1963.

[71] Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643, 4 2022.

[72] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60:611–624, 6 2012.

[73] Wenbin Zhu, Hu Qin, Andrew Lim, and Lei Wang. A two-stage tabu search algorithm with enhanced packing heuristics for the 3l-cvrp and m3l-cvrp. *Computers & Operations Research*, 39(9):2178–2195, 2012.

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

[75] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31:170–186, 5 1997.