

# TIPE : Étude de couverture de réseaux de métro, application de l'homologie persistante et optimisation

H Elowan

## Sommaire

I Définitions	1
II Les données	4
II.1 Sources	4
II.2 Points et distances	4
III Méthode	5
IV Recherche d'optimisation	7
V Résultats et conclusion	7
V.1 Résultats de l'homologie persistante	7
V.2 Résultat de l'optimisation	8
Bibliographie	9
VI Annexe	9

## Résumé

Nous nous proposons ici d'étudier les différentes disparités dans les réseaux métropolitains de plusieurs grandes villes, nous allons détecter les zones spatiales les plus en déficit de transports en commun. Pour cela, nous adopterons une approche analytique utilisant la topologie : *l'homologie persistante*.

## I Définitions

L'homologie persistante est une *méthode de calcul* qui, à partir d'une discrétisation, cherche à fournir une description des caractéristiques que l'on pourrait distinguer de cet ensemble.

Nous chercherons seulement à caractériser les “trous” dans un espace, afin de détecter les trous de couverture dans un réseau métropolitain, ici modélisé comme un nuage de points de  $\mathbb{R}^2$ , dont chaque élément est une station de métro.

Afin d'utiliser l'homologie persistante, nous devons définir certaines notions géométriques, nous noterons dans la suite  $X$  l'ensemble des points que l'on considère.

### Définition :

Un *simplexe*  $\sigma$  de dimension  $k$  (ou *k-simplexe*) correspond à l'enveloppe convexe de  $k + 1$  points de  $X$  non inclus dans un sous-espace affine de dimension  $k - 1$ .

On définit un simplexe de dimension 0 comme un point de  $X$ .

Par exemple, un simplexe de dimension 1 est un segment et un simplexe de dimension 3 est un trièdre.

*Remarque :* On dit que  $\sigma_i$  est une face de  $\sigma_j$  si et seulement si  $\sigma_i \subset \sigma_j$  et la dimension de  $\sigma_i$   $\dim(\sigma_i)$  est égale à  $\dim(\sigma_j) - 1$ .

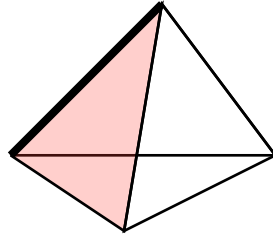


Figure 1 : Ce trièdre est un simplexe  $\sigma$  de dimension 3, où le triangle rouge représente un simplexe  $\tau$  de dimension 2 mais aussi une face de  $\sigma$  dans le sens de la remarque précédente. Notons que l'arête en gras est un simplexe de dimension 1 et est une face de  $\tau$  et non de  $\sigma$ .

**Définition :**

Un *complexe simplicial* est un ensemble de simplexes.

**Définition :**

Une *filtration* est une application qui à un entier  $i$  associe un complexe simplicial  $K_i$  de sorte que :

$$\forall j \in \llbracket 0, i \rrbracket, K_j \subset K_i$$

Observons sur la Figure 2 les notions précédemment définies. Chaque  $K_i$  est un complexe simplicial, la suite  $(K_i)_{i=0}^3$  est une filtration et tous les points, segments et faces (ici en rouge) sont des simplexes de dimension 0, 1 et 2 respectivement.

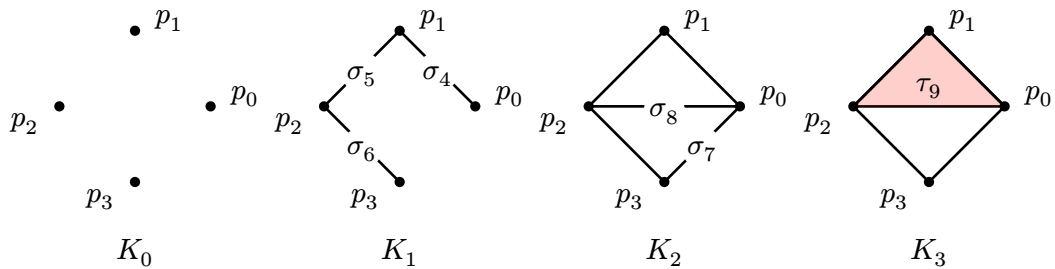


Figure 2 : Représentation d'une filtration où  $K_0 \subset K_1 \subset K_2 \subset K_3$  et où chaque simplexe a été nommé :  $p_i$  pour les simplexes de dimension 0,  $\sigma_j$  pour la dimension 1 et  $\tau_k$  pour la dimension 2.

Nous observons qu'une filtration permet d'ajouter une notion de "temporalité" dans un ensemble de points. Nous sommes capable de noter quels événements surviennent entre deux complexes à la suite, par exemple l'apparition du cycle  $(\sigma_4, \sigma_5, \sigma_8)$  entre  $K_1$  et  $K_2$ .

On introduit de plus un ordre total  $\preccurlyeq$  sur l'ensemble des simplexes d'une filtration :

**Définition :**

Soient une filtration  $K_0 \subset K_1 \subset \dots \subset K_p$  et l'ensemble  $S$  de tous les simplexes apparaissant dans la filtration. On indice  $S$  de sorte que pour tout  $\sigma_i$  et  $\sigma_j$  de  $S$ :

$$\left. \begin{array}{l} \text{Si } \sigma_i \in K_{k_i} \text{ et } \sigma_j \in K_{k_j} \text{ avec } k_i < k_j \\ \text{Sinon si } \sigma_i \text{ est une face de } \sigma_j \end{array} \right\} \Rightarrow i < j$$

Si aucun des deux cas n'est réalisé alors le choix de l'ordre entre les deux simplexes est arbitraire.

On définit donc l'ordre total  $\preceq$  tel que  $\sigma_i \preceq \sigma_j \Leftrightarrow i \leq j$

Observons sur la Figure 2 l'indexation des simplexes suivant l'ordre précédemment défini : les simplexes  $\sigma_8$  et  $\sigma_7$  sont plus grands au sens de  $\preceq$  que tous les autres  $\sigma_i$  et  $p_i$  parce qu'ils apparaissent plus tard dans la filtration. De plus, si  $\sigma_8$  était apparu dans  $K_3$ , l'ordre aurait toujours été respecté puisque  $\sigma_8$  est une face de  $\tau_9$ .

Il y a cependant un problème : nous voulons analyser un ensemble de points, et non une filtration déjà existante, il nous faut alors créer une filtration depuis l'ensemble que l'on considère. Nous faisons cela via une construction incrémentale de complexes simpliciaux avec les complexes de Vietoris-Rips pondérés. Ainsi d'après [1] :

**Définition :**

Soient un ensemble  $X = (x_i)_{i=0}^n$  de points associés à des poids  $(w_i)_{i=0}^n$  et une distance  $d$ , on définit le complexe simplicial pondéré de Vietoris-Rips au rang  $t$ , noté  $V_t(X, d)$ , comme l'ensemble des simplexes  $(x_{i_0}, \dots, x_{i_k})$  tels que :

$$\begin{cases} \forall j \in \llbracket 0, k \rrbracket, w_{i_j} < t \\ \forall (j, l) \in \llbracket 0, k \rrbracket^2, d(x_{i_j}, x_{i_l}) + w_{i_j} + w_{i_l} < 2t \end{cases}$$

Ainsi plus on augmente  $t$ , plus le complexe possède des simplexes, on en donne une représentation Figure 3. Pour chaque  $t$  qui augmente le nombre de simplexes du complexe simplicial, nous ajoutons  $V_t(X, d)$  à la filtration que l'on est en train de créer.

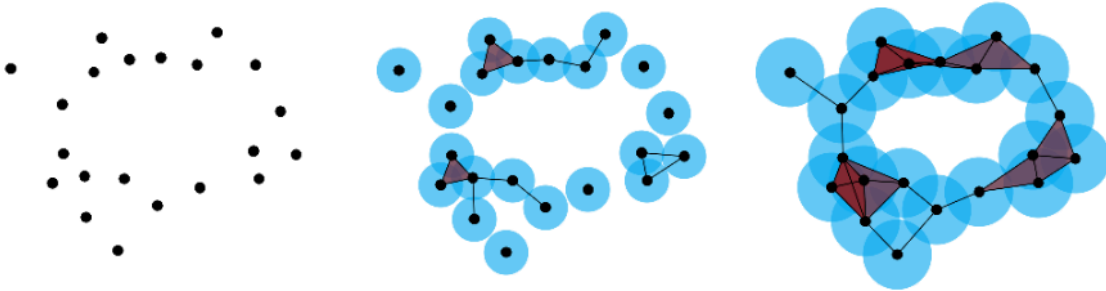


Figure 3 : Construction d'un complexe simplicial avec un rayon  $t$  grandissant de gauche à droite. Un simplexe est considéré dès lors que les boules associées à ces sommets ont une intersection non vide. Tiré de [1]

Pour définir formellement des “trous”, nous devons définir les opérateurs de bords. Ainsi selon [2] :

**Définition :**

On définit un *complexe de chaînes* comme la donnée d'une suite

$$\dots \xrightarrow{\delta_{k+2}} C_{k+1} \xrightarrow{\delta_{k+1}} C_k \xrightarrow{\delta_k} C_{k-1} \xrightarrow{\delta_{k-1}} \dots$$

Où chaque  $C_k$  est un groupe abélien libre qui a pour base les  $k$ -simplexes de  $X$  et  $\delta_k$  est un morphisme de groupes tel que  $\delta_k \circ \delta_{k+1} = 0$

On appelle  $\delta_k$  un *opérateur de bords*.

**Définition :**

On définit alors les *classes d'homologies de dimension  $k$*  comme le groupe de  $\text{Ker}(\delta_k)$  quotienté par  $\text{Im}(\delta_{k+1})$ :

$$H_k = \text{Ker}(\delta_k) / \text{Im}(\delta_{k+1})$$

Celle-ci représente les “trous” en dimension  $k$ .

On peut voir que les éléments de  $H_0$  permettent de différencier les composantes connexes de  $X$  et ceux de  $H_1$  représentent des trous qui sont entourés par un chemin fermé de points connectés (comme le cycle  $(\sigma_4, \sigma_5, \sigma_8)$  dans  $K_2$  dans la Figure 4).

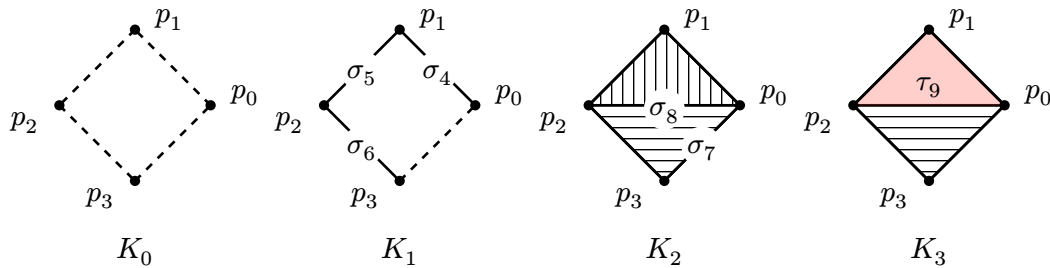


Figure 4 : Les éléments de  $H_0$  en pointillés et ceux de  $H_1$  achurés.

Pour notre usage,  $H_1$  représente les zones critiques de couverture du réseau.

## II Les données

### II.1 Sources

On choisit de se baser uniquement sur des vraies villes, que l'on nommera Ville A et Ville B par la suite, pour tester notre approche. De plus, toutes les informations relatives aux stations de métro ainsi que les temps de passages sont trouvables sur <https://transport.data.gouv.fr>.

Ces informations servent à définir nos points et notre pondération, en revanche elles ne permettent pas d'obtenir les distances entre les stations, pour cela nous utiliserons alors <https://www.geoapify.com> qui nous permet d'estimer des temps de trajet en voiture et à pied.

### II.2 Points et distances

Définissons dès lors nos objets :

**Définition :**

Un point  $x_i$ , représentant une station de métro, est défini par la donnée de sa position géographique (latitude/longitude) ainsi que son poids  $w_i$ . Le poids  $w_i$  est égal à la moyenne du temps d'attente entre deux métros en station  $x_i$  sur une semaine entière.

On définit la distance similairement à [1]:

**Définition :**

On définit la distance entre deux stations de métro  $x$  et  $y$  comme :

$$d(x, y) = \frac{1}{2}(\min(t_{\text{marche}}(x, y), t_{\text{voiture}}(x, y)) + \min(t_{\text{marche}}(y, x), t_{\text{voiture}}(y, x)))$$

Ainsi en revenant aux boules des complexes simpliciaux de Vietoris-Rips, la distance modélise le coût temporel d'un trajet "porte à porte" entre les stations (en voiture ou à pied).

### III Méthode

Pour trouver les zones critiques, nous utiliserons la méthode de *l'homologie persistante* décrite dans [1] (dans le cas de notre réseau de métro). Celle ci se décompose en 3 étapes :

- Transformation de l'ensemble des points (les stations de métro)  $x_i$  de poids  $w_i$  en une filtration;
- Création et réduction de la matrice de bordure (définie dans la suite);
- Récupération des simplexes "tueurs" de classes d'homologies

On suppose que la première étape est déjà réalisée suivant la section I.

Ainsi à partir de cette filtration, nous pouvons obtenir les classes d'homologies grâce au théorème qui suit :

**Théorème des facteurs invariants :**

D'après [3] et [4], il existe un unique ensemble  $\{d_1, \dots, d_p\}$  d'éléments de  $H_k$  définis à des inversibles près, tel que :

$$H_k \simeq \bigoplus_{i=1}^p P_k / d_i P_k$$

en notant  $P_k$  l'ensemble des parties à  $k+1$  éléments de  $X$ , formant donc un complexe simplicial constitué uniquement de  $k$ -simplexes. Cet ensemble est appelé *code barre* de  $H_k$

Informatiquement, selon [5], on calcule ce code barre en créant une matrice de bordure  $B$  après avoir défini un ordre total sur les simplexes respectant les propriétés énoncées section I.

**Définition :**

On définit la matrice de bordure, associée à un ordre total  $\sigma_0 \preccurlyeq \dots \preccurlyeq \sigma_{n-1}$  sur tous les simplexes  $(\sigma_i)_{i=0}^{n-1}$  de la filtration, suivant :

$$\forall (i, j) \in \llbracket 0, n-1 \rrbracket^2, B[i][j] = \begin{cases} 1 & \text{si } \sigma_i \text{ est une face de } \sigma_j \\ 0 & \text{sinon} \end{cases}$$

Un exemple d'une telle matrice est donnée en Table 1.

Après avoir calculé  $B$ , nous voulons la *réduire* à un code barre, dans le sens où par lecture matricielle, grâce au théorème précédent, nous pouvons donner un temps de vie à chaque simplexe par l'attribution d'un unique antécédent à chacun de ceux-ci. Le terme de *réduction* fait ici référence à la réduction de  $B$  en forme normale de Smith. Nous pouvons observer ce résultat en Table 2.

Cet algorithme de réduction est nommé *standard algorithm* et est décrit dans [5] par, en posant  $\text{low}_B(j) = \max(\{i \in \llbracket 0, n-1 \rrbracket, B[i][j] \neq 0\}) \in \mathbb{N} \cup \{-1\}$  :

```
StandardAlgorithm(B)
1  for j allant de 0 à n-1:
2    while (il existe i < j avec low[i] = low[j]):
3      ajouter colonne i de B à colonne j modulo 2
```

Notons que cet algorithme a pour complexité temporelle  $O(n^3)$  au pire.

Comparons alors nos deux matrices en Table 1 et Table 2, sur l'exemple de la Figure 2, où les cases vides remplacent les zéros et où les colonnes/lignes vides ont été omises.

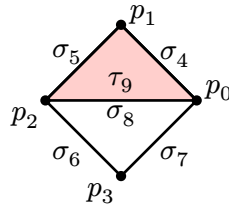


Figure 5 : Rappel du nommage des simplexes ( $p_i$  pour la dimension 0,  $\sigma_j$  pour la dimension 1,  $\tau_k$  pour la dimension 2)

En regardant la matrice  $\overline{B}$ , nous remarquons que l'opération de réduction a permis d'avoir la ligne *low* sans répétition de nombres positifs. Cette propriété s'interprète comme il suit :

Enfants		4	5	6	7	8	9
Parents	0	1			1	1	
	1	1	1				
	2		1	1		1	
	3			1	1		
	4						1
	5						1
	8						1
	low	1	2	3	3	2	8

Table 1 : Matrice  $B$

Enfants		4	5	6	7	8	9
Parents	0	1					
	1	1	1				
	2		1	1			
	3			1			
	4						1
	5						1
	8						1
	low	1	2	3	-1	-1	8

Table 2 : Matrice  $\overline{B}$ ,  $B$  après réduction

Si  $\text{low}_{\overline{B}}(j) = i \neq -1$ , alors la paire de simplexes  $(\sigma_i, \sigma_j)$  représente le temps de vie d'une classe d'homologie : l'apparition de  $\sigma_i$  fait apparaître une nouvelle classe d'homologie tandis que  $\sigma_j$  va la *tuer* en apparaissant.

Ainsi, sur la Figure 2 : dans  $K_2$ ,  $\sigma_8$  cause l'apparition d'une classe dans  $H_1$  (car elle crée un cycle) cependant l'apparition du simplexe  $\tau_9$  dans  $K_3$  tue la classe de  $\sigma_8$  dans  $H_1$  (car elle "remplit" le contenu du cycle).

En revanche, si  $\text{low}_{\overline{B}}(j) = -1$ , alors l'apparition de  $\sigma_j$  crée une classe d'homologie : s'il existe  $k$  tel que  $\text{low}_{\overline{B}}(k) = j$  on est dans le cas précédent, sinon la classe d'homologie n'est jamais tuée.

C'est depuis cette matrice que nous sommes capables de déterminer toutes les classes d'homologies ainsi que leur durée de vie, et donc de générer des représentations graphiques comme montré en Figure 6 et Figure 7.

## IV Recherche d'optimisation

La motivation de cette section vient de l'observation de la complexité du standard algorithm :  $O(n^3) = O(2^{|X|})$  ( $n$  étant le nombre total de simplexes possibles  $= 2^{|X|}$ ). Nous cherchons donc à optimiser la complexité de celui ci.

L'optimisation que l'on propose provient de deux observations :

- $B$  est une matrice creuse ;
- L'opération de somme de colonnes (ligne 3) agit seulement sur une matrice extraite  $B_d$  ne dépendant que de la dimension des simplexes.

Le premier point influence le choix d'utilisation de listes d'adjacences à une matrice d'adjacence, en particulier par des doubles listes chaînées ordonnées.

La justification du second point se trouve en annexe.

On en déduit cet algorithme où  $B$  est modifié par effet de bords sur les  $B_d$  :

```
StandardAlgorithmUpgrade(B)
1  dims <- Tableau des simplexes où dims[i] contient la liste des simplexes de dim=i
2  for toute dimension d à considérer:
3      for chaque simplexe j de dims[d] de façon croissante
4          while (il existe i dans dims[d] tel que low[j] = low[i]):
5              ajouter colonne i de B à colonne j modulo 2
6
```

Le calcul de la complexité ne permettant pas d'avoir une meilleure borne, on note que cette algorithme est en  $O(2^{|X|})$  au pire, mais bien moins en pratique, voir Figure 8.

## V Résultats et conclusion

### V.1 Résultats de l'homologie persistante

Les triangles représentés en Figure 6 et Figure 7 montrent les zones où il est le plus difficile de rejoindre une station de métro. Pour les plus gros triangles, il peut être cohérent de croire qu'il est difficile de se rendre à ces stations de métro. En revanche, l'interprétation est plus dure pour les plus petits triangles.

Nous devons revenir à la définition de notre distance : celle ci prend en compte le temps minimal entre un trajet en voiture et le même trajet à pied. Les plus petites zones, comme à gauche sur la ligne bleue dans la Ville A ou en bout de ligne rouge dans la Ville B, correspondent en fait à des espaces uniquement piétons dont le temps de trajet est plus court à pied qu'en voiture. Ainsi, les plus petites zones indiquent donc la même information (difficulté d'accès à ces stations) que les grandes mais à une échelle différente.

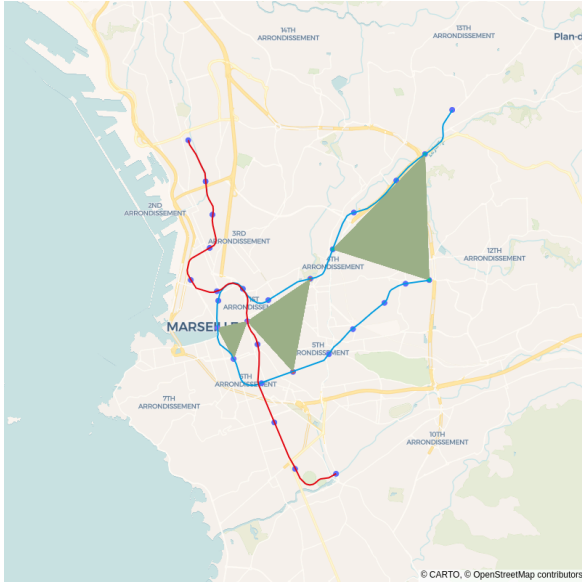


Figure 6 : Carte de Ville A

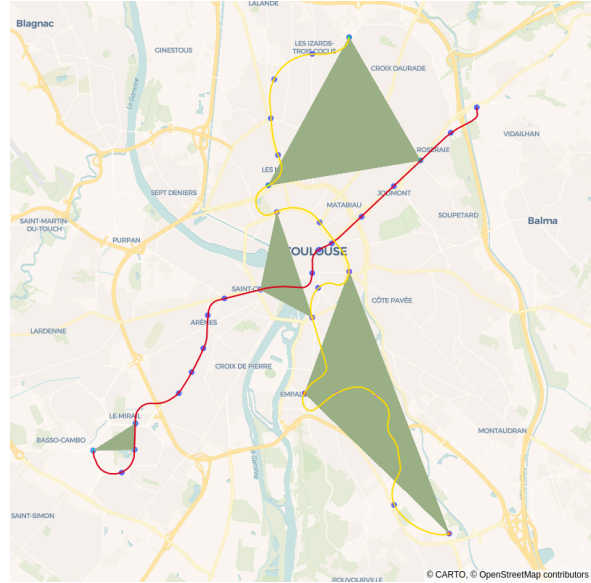


Figure 7 : Carte de Ville B

## V.2 Résultat de l'optimisation

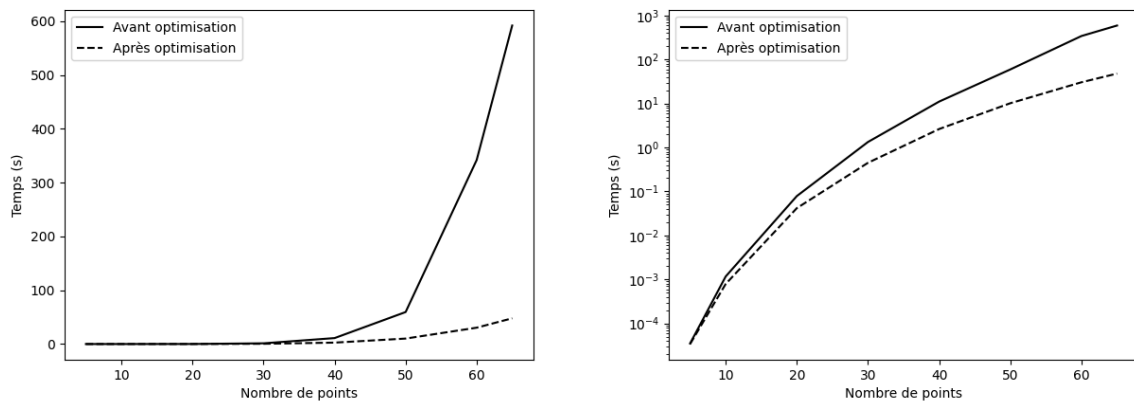


Figure 8 : Temps d'exécution des deux algorithmes précédents, avec une ordonnée linéaire à gauche et logarithmique à droite.

On observe sur la Figure 8 une nette amélioration entre la version avant optimisation et celle après, cependant, même si la figure de gauche montre la différence ressentie lors de l'exécution, celle de droite montre que nous restons quand même avec une complexité "de même forme" en le nombre d'éléments de  $X$ .

Notons que le stockage en liste d'adjacence permet un plus grand nombres d'éléments tandis que la matrice d'adjacence pose beaucoup plus de problèmes. Ici, l'étude s'arrête pour un nombre d'éléments égal à 65 puisqu'il faut plus de 16go de ram pour stocker la matrice d'une taille supérieure (ma limite physique).

L'homologie persistante est donc une méthode nous permettant de mettre en lumière des zones mal desservies en prenant en compte des réalités plus complexes que seul le temps de trajet. Par exemple, nous prenons en compte les temps d'attente en station mais nous aurions pu aussi prendre en compte la densité de population autour de ces stations. Cette caractéristique peut être une possibilité d'ouverture de ce sujet car celle ci joue intuitivement un rôle dans le temps d'attente en station et donc dans la difficulté de prendre un métro.



## Bibliographie

- [1] ABIGAIL HICKOK, BENJAMIN JARMAN, MICHAEL JOHNSON, JIAJIE LUO, and MASON A. PORTER, *Persitent Homology for Resource Coverage: A Case Study of Access to Polling Sites*.
- [2] Jean-Yves Welschinger, Introduction aux théories homologiques, (n.d.).
- [3] Henri Paul de Saint-Gervais, Une invitation à l'homologie persistante, (n.d.).
- [4] Afra Zomorodian and Gunnar Carlsson, *Computing Persistent Homology*.
- [5] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington, *A Roadmap for the Computation of Persistent Homology*.

## VI Annexe

Justifions le deuxième point annoncé dans section IV, soit  $d \in \llbracket 0, |X| - 1 \rrbracket$ , considérons la matrice extraite :

$$B_d = (B_{i,j})_{(i,j) \in I} \text{ telle que } I = \{(i,j) \in \llbracket 0, n-1 \rrbracket, \dim(\sigma_i) = d \text{ et } \dim(\sigma_j) = d+1\}$$

On note  $\varphi$  la correspondance entre les indices des deux matrices :  $(B_d)_{\varphi(i,j)} = B_{i,j}$

Supposons que l'on exécute la ligne 3 de l'algorithme, alors  $\text{low}(j) = \text{low}(i) = k$ , on pose  $\sigma_i$ ,  $\sigma_j$  et  $\sigma_k$  les simplexes associés. Donc  $\sigma_k$  est une face de  $\sigma_i$  et  $\sigma_j$ , donc par définition

$$\dim(\sigma_k) + 1 = \dim(\sigma_i) = \dim(\sigma_j)$$

La ligne  $L_k$  ainsi que les deux colonnes  $C_i$  et  $C_j$  sont alors considérées dans  $B_d$  ( $d = \dim(\sigma_k)$ ). De plus, toutes les lignes ayant un coefficient non nul dans les colonnes  $C_i$  ou  $C_j$  le sont aussi puisqu'un coefficient non nul revient à être une face, donc de dimension  $d$ .

Ainsi l'opération de somme des colonnes  $C_i + C_j$  dans  $B$  (ligne 3) est équivalent à celle de  $C_{i'} + C_{j'}$  dans  $B_d$  avec  $\varphi(i,j) = (i',j')$ .

Ainsi, au lieu d'exécuter l'algorithme sur la matrice creuse  $B$ , on peut l'exécuter sur les matrices extraites  $B_d$  plus petites et moins creuses, on localise ainsi les modifications.