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

H Elowan

Sommaire

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*. Nous proposerons de plus une amélioration de l'algorithme utilisé.

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* σ 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.

On note $\sigma = [p_0, ..., p_k]$ avec les $k+1$ points définissant le k-simplexe.

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

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

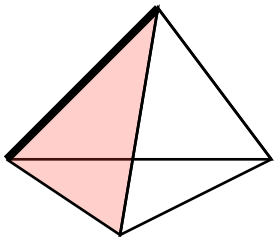

Figure 1 : Ce trièdre est un simplexe σ de dimension 3, où le triangle rouge représente un simplexe τ de dimension 2 mais aussi une face de σ dans le sens de la remarque précédente. Notons que l'arête en gras est un simplexe de dimension 1 et est une face de τ et non de σ.

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 :

$$\forall j \in [\![0, i]\!] \ , \ \ K_j \subset K_i$$

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.

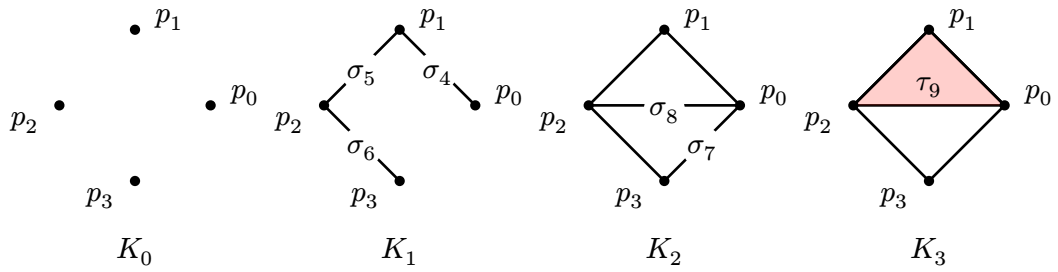

Figure 2 : Représentation d'une filtration où $K_0 \subset K_1 \subset K_2 \subset K_3$ et où chaque simplexe a été nommé : p_i pour les simplexes de dimension 0, σ_j pour la dimension 1 et τ_k pour la dimension 2.

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 $K_0 \subset K_1 \subset ... \subset K_p$ et l'ensemble S de tous les simplexes apparaissant dans la filtration. On indice S de sorte que pour tout σ_i et σ_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 $\preccurlyeq$ tel que $\sigma_i \preccurlyeq \sigma_j \Leftrightarrow i \leq j$

Observons sur la Figure 2 l'indexation des simplexes suivant l'ordre précédemment défini : les simplexes σ_8 et σ_7 sont plus grands au sens de $\preccurlyeq$ que tous les autres σ_i et p_i parce qu'ils apparaissent plus tard dans la filtration. De plus, si σ_8 était apparu dans K_3, l'ordre aurait toujours été respecté puisque σ_8 est une face de τ_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 $\left\{\sigma_{i_0}, ..., \sigma_{i_k}\right\}$ tels que :

$$\forall r \in [\![0,k]\!], \sigma_{i_r} = \left[x_{i_0}, ..., x_{i_l}\right] \text{ et } \begin{cases}\forall j \in [\![0,l]\!], w_{i_j} < t \\ \forall (p,q) \in [\![0,k]\!]^2, d\left(x_{i_p}, x_{i_q}\right) + w_{i_p} + w_{i_q} < 2t\end{cases}$$

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.

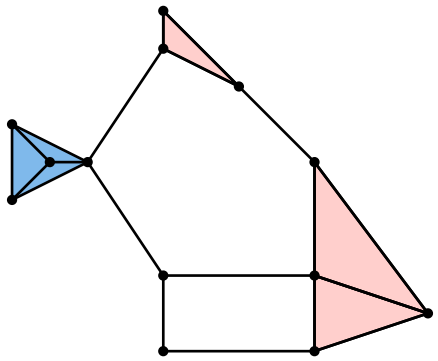

Figure 3 : Exemple de complexe simplicial de Vietoris-Rips pour un rang t où les arêtes noires sont des simplexes de dimension 1; les triangles rouges des simplexes de dimension 2 et la pyramide bleue de dimension 3.

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

$$... \overset{\delta_{k+2}}{\to} C_{k+1} \overset{\delta_{k+1}}{\to} C_k \overset{\delta_k}{\to} C_{k-1} \overset{\delta_{k-1}}{\to} ... \overset{\delta_0}{\to} \{0\}$$

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

On appelle δ_k un *opérateur de bords.*

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

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

Ces éléments représentent les "trous" en dimension k.

On peut voir que les éléments de H_0 premettent de différentier 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).

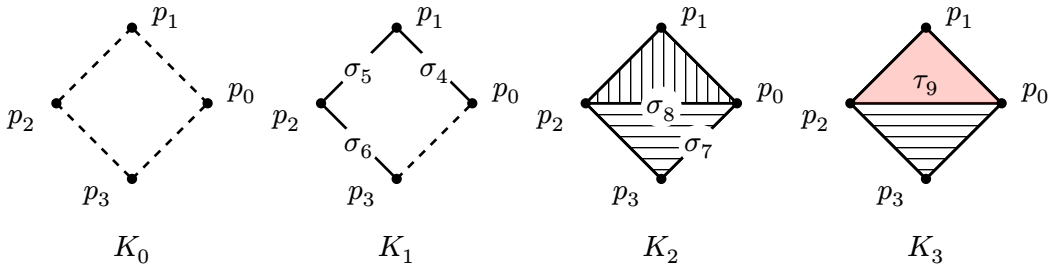

Figure 4 : Les éléments de H_0 en pointillés et ceux de H_1 achurés.

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

Pour cette situation sur les stations de métros, 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 simplicaux de Vietoris-Rips, la distance modélise le coût temporel d'un trajet "porte à porte" entre les stations (en voiture ou à pied).

II 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 x_i (les stations de métro) de poids w_i (égale à la moyenne de temps d'attente en station sur une semaine) 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 calculer les classes d’homologies grâce au théorème qui suit :

Théorème des facteurs invariants :
D’après [3] et [4], il existe un unique ensemble $\{d_1, ..., d_p\}$ d’éléments de $\mathbb{Z}$ définis à des inversibles près et $\beta \in \mathbb{N}$ tels que :

$$H_k \simeq \mathbb{Z}^\beta \bigoplus_{i=1}^{p} \mathbb{Z}/d_i\mathbb{Z}$$

Où β est le rang de la partie libre du groupe abélien de type fini H_k (celui ci est de plus le nombre de trous de dimension k, appelé *nombre de Betty*).

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 $\sigma_0 \preccurlyeq ... \preccurlyeq \sigma_{n-1}$ sur tous les simplexes $(\sigma_i)_{i=0}^{n-1}$ de la filtration, suivant :

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

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. Nous pouvons observer ce résultat en Table 2.

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

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

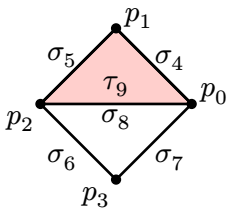

Figure 5 : Rappel du nommage des simplexes (p_i pour la dimension 0, σ_j pour la dimension 1, τ_k pour la dimension 2)

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 (σ_i, σ_j) représente le temps de vie d'une classe d'homologie : l'apparition de σ_i fait apparaitre une nouvelle classe d'homologie tandis que σ_j va la *tuer* en apparaissant.

Ainsi, sur la Figure 2 : dans K_2, σ_8 cause l'apparition d'une classe dans H_1 (car elle crée un cycle) cependant l'apparition du simplexe τ_9 dans K_3 tue la classe de σ_8 dans H_1 (car elle "remplit" le contenu du cycle).

En revanche, si $\text{low}_{\overline{B}}(j) = -1$, alors l'apparition de σ_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.

III Recherche d'optimisation

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

On propose une amélioration suite à la propriété suivante :

Propriété :
L'opération de somme de colonnes (ligne 3) agit seulement sur une matrice extraite B_d ne dépendant que de la dimension des simplexes.

Preuve

Soit $d \in [\![0, |X|-1]\!]$, considérons la matrice extraite :

$$B_d = \left(B_{i,j}\right)_{(i,j)\in I} \text{ telle que } \; I = \left\{(i,j) \in [\![0, n-1]\!], \dim(\sigma_i) = d \text{ et } \dim\left(\sigma_j\right) = d+1\right\}$$

On note φ 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 σ_i, σ_j et σ_k les simplexes associés. Donc σ_k est une face de σ_i et σ_j, donc par définition

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

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.

Vu que la matrice est creuse, nous choisissons d'utiliser une liste d'adjacence pour représenter la matrice.

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 $O\left(2^{3|X|}\right)$ au pire, mais bien moins en pratique, voir Figure 8.

IV Résultats et conclusion

IV.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

IV.2 Résultat de l'optimisation

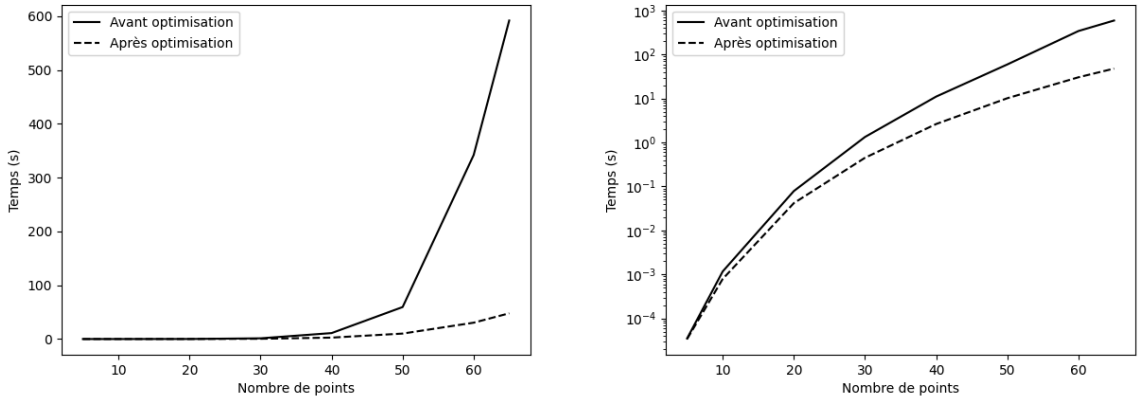

Figure 8 : Temps d'exécution des deux algorithmes précédents, avec une ordonnée linéaire à gauche et logarithmique à droite.

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 realités 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*.