Simplification de fonctions sur les surfaces via la persistance

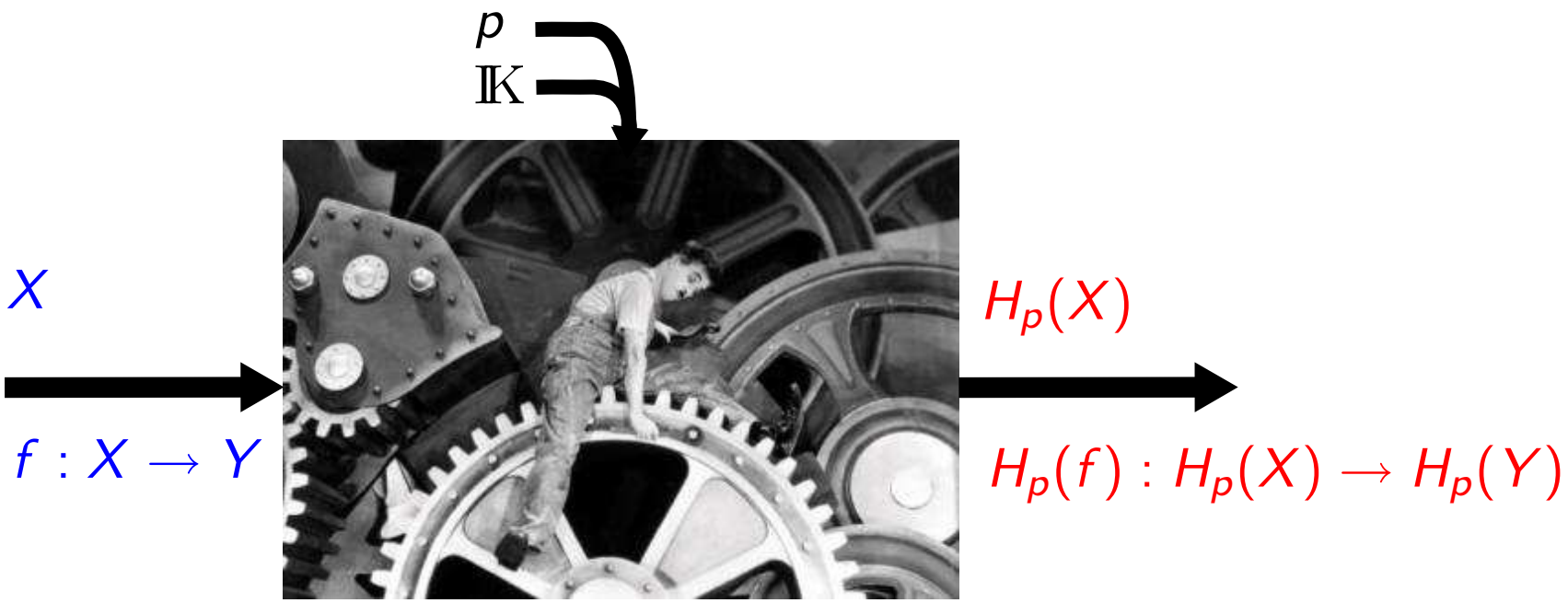
Francis Lazarus[1]

GIPSA-Lab, CNRS, INPG

26 janvier 2009

[1]En collaboration avec D. Attali, M. Glisse, S. Hornus et D. Morozov

Le foncteur homologique



$H_p(Id) = Id$ et $H_p(f \circ g) = H_p(f) \circ H_p(g)$

H_p facile à calculer si X et f sont simpliciaux.

Filtration et homologie

$$X_1 \quad \subset \quad X_2 \quad \subset \quad \dots \quad \subset \quad X_n$$

$$\Downarrow H_p$$

$$H_p(X_1) \quad \to \quad H_p(X_2) \quad \to \quad \dots \quad \to \quad H_p(X_n)$$

Filtration $\xrightarrow{H_p}$ chaîne d'applications linéaires.

Filtration et homologie

$$X_1 \quad \subset \quad X_2 \quad \subset \quad \dots \quad \subset \quad X_n$$

$$\Downarrow H_p$$

$$H_p(X_1) \quad \to \quad H_p(X_2) \quad \to \quad \dots \quad \to \quad H_p(X_n)$$

Filtration $\xrightarrow{H_p}$ chaîne d'applications linéaires.

Filtration et homologie

$$X_1 \quad \subset \quad X_2 \quad \subset \quad \dots \quad \subset \quad X_n$$

$$\Downarrow H_p$$

$$H_p(X_1) \quad \to \quad H_p(X_2) \quad \to \quad \dots \quad \to \quad H_p(X_n)$$

Filtration $\xrightarrow{H_p}$ chaîne d'applications linéaires.

Isomorphisme de chaînes d'applications

$$(f_i): \quad E_1 \xrightarrow{f_1} E_2 \xrightarrow{f_2} \cdots \xrightarrow{f_{n-1}} E_n$$

$$(g_i): \quad F_1 \xrightarrow{g_1} F_2 \xrightarrow{g_2} \cdots \xrightarrow{f_{n-1}} F_n$$

$$(f_i) \simeq (g_i)$$

Isomorphisme de chaînes d'applications

$$(f_i): \quad E_1 \xrightarrow{f_1} E_2 \xrightarrow{f_2} \cdots \xrightarrow{f_{n-1}} E_n$$

$$(g_i): \quad F_1 \xrightarrow{g_1} F_2 \xrightarrow{g_2} \cdots \xrightarrow{f_{n-1}} F_n$$

$$(f_i) \simeq (g_i)$$

Isomorphisme de chaînes d'applications

$$
\begin{array}{lccccccc}
(f_i): & E_1 & \xrightarrow{f_1} & E_2 & \xrightarrow{f_2} & \cdots & \xrightarrow{f_{n-1}} & E_n \\
 & \downarrow \simeq & & \downarrow \simeq & & & & \downarrow \simeq \\
(g_i): & F_1 & \xrightarrow{g_1} & F_2 & \xrightarrow{g_2} & \cdots & \xrightarrow{f_{n-1}} & F_n
\end{array}
$$

$$(f_i) \simeq (g_i)$$

Isomorphisme de chaînes d'applications

$$
\begin{array}{lccccccc}
(f_i): & E_1 & \xrightarrow{f_1} & E_2 & \xrightarrow{f_2} & \cdots & \xrightarrow{f_{n-1}} & E_n \\
 & \downarrow \simeq & & \downarrow \simeq & & & & \downarrow \simeq \\
(g_i): & F_1 & \xrightarrow{g_1} & F_2 & \xrightarrow{g_2} & \cdots & \xrightarrow{f_{n-1}} & F_n
\end{array}
$$

$$(f_i) \simeq (g_i)$$

Classification des chaînes d'applications linéaires

Chaînes élémentaires :

$$\mathrm{Id}_{\mathbb{K}}[a,b] : 0 \xrightarrow{1} \cdots \longrightarrow 0 \longrightarrow \mathbb{K} \xrightarrow{a} \mathbb{K} \xrightarrow{Id} \cdots \xrightarrow{Id} \mathbb{K} \xrightarrow{b} 0 \cdots \longrightarrow 0 \xrightarrow{n}$$

Décomposition canonique

$\exists!$ multi-ensemble d'intervalles, I, tq

$$(f_i) \simeq \bigoplus_{[a,b]\in I} \mathrm{Id}_{\mathbb{K}}[a,b]$$

Corollaire

I est un invariant complet pour (f_i). Ces éléments sont les intervalles de persistance de (f_i).

Classification des chaînes d'applications linéaires

Chaînes élémentaires :

$$\mathrm{Id}_{\mathbb{K}}[a,b] : 0 \xrightarrow{1} \cdots \longrightarrow 0 \longrightarrow \mathbb{K} \xrightarrow{Id} \cdots \xrightarrow{Id} \mathbb{K} \xrightarrow{b} 0 \cdots \longrightarrow 0 \xrightarrow{n}$$

Décomposition canonique

$\exists!$ multi-ensemble d'intervalles, I, tq

$$(f_i) \simeq \bigoplus_{[a,b]\in I} \mathrm{Id}_{\mathbb{K}}[a,b]$$

Corollaire

I est un invariant complet pour (f_i). Ces éléments sont les intervalles de persistance de (f_i).

Classification des chaînes d'applications linéaires

Chaînes élémentaires :

$$\mathrm{Id}_{\mathbb{K}}[a,b] : 0 \xrightarrow{1} \cdots \longrightarrow 0 \longrightarrow \mathbb{K} \xrightarrow{a} \mathbb{K} \xrightarrow{Id} \cdots \xrightarrow{Id} \mathbb{K} \xrightarrow{b} 0 \cdots \longrightarrow 0 \xrightarrow{n}$$

Décomposition canonique

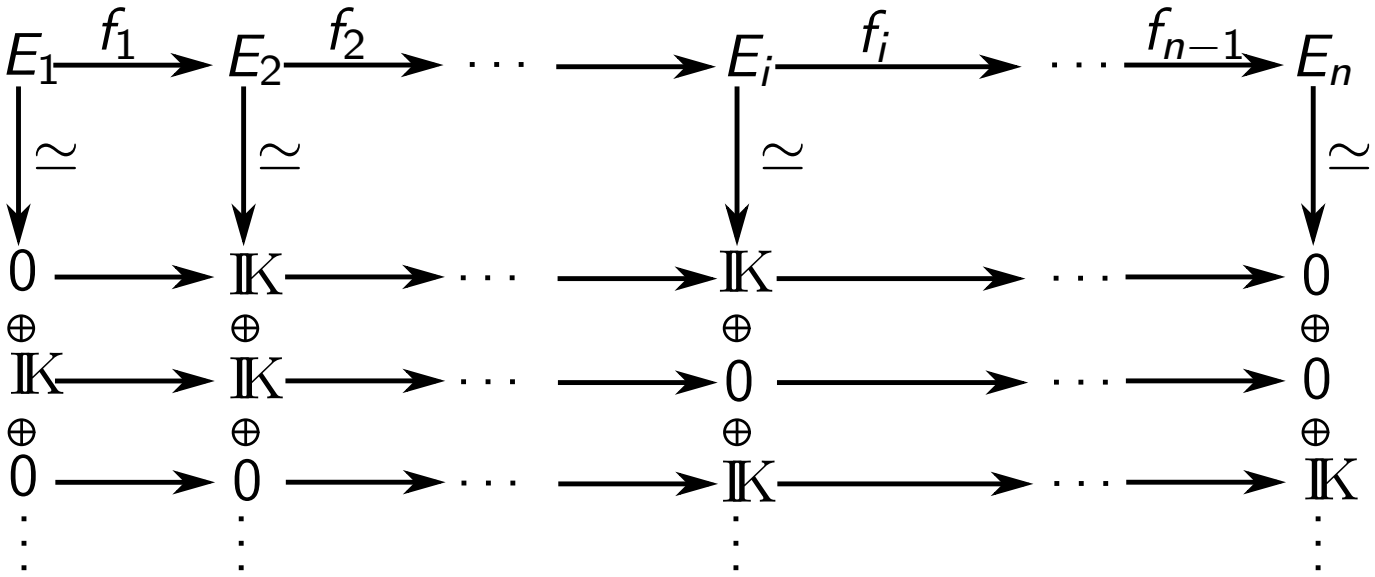
$\exists!$ multi-ensemble d'intervalles, I, tq

$$(f_i) \simeq \bigoplus_{[a,b]\in I} \mathrm{Id}_{\mathbb{K}}[a,b]$$

Corollaire

I est un invariant complet pour (f_i). Ces éléments sont les intervalles de persistance de (f_i).

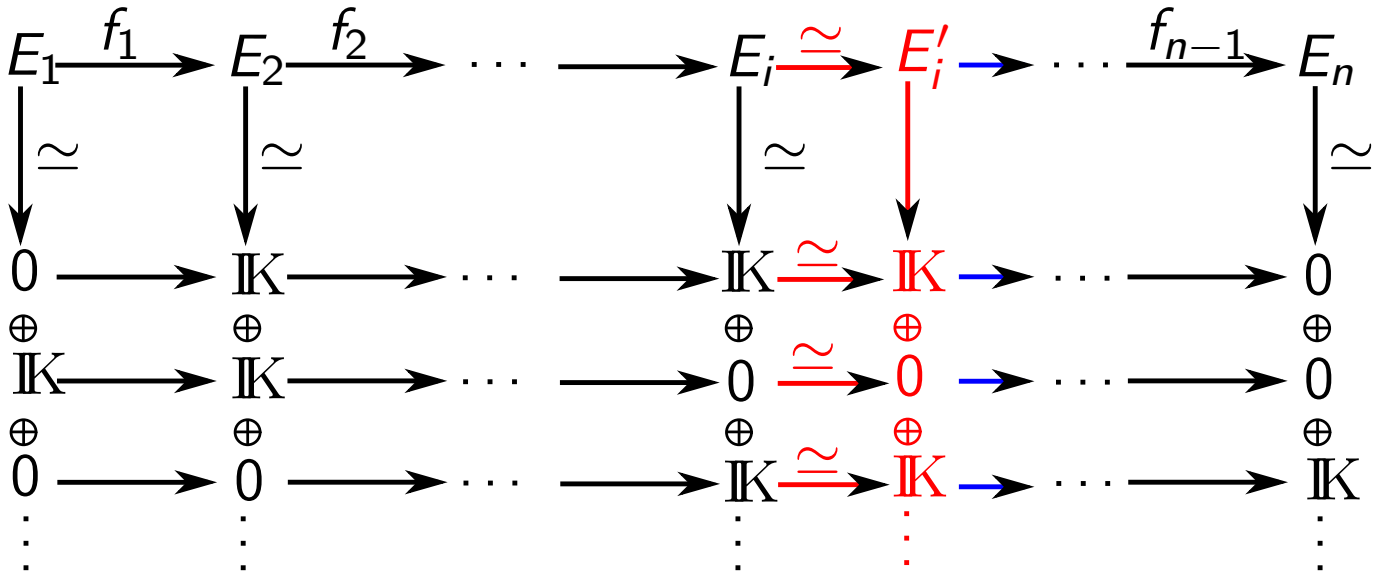
Classification des chaînes d'applications linéaires



Remarque

Ajouter un isomorphisme dans une chaîne ne change pas les intervalles de persistance (à ré-indexation près).

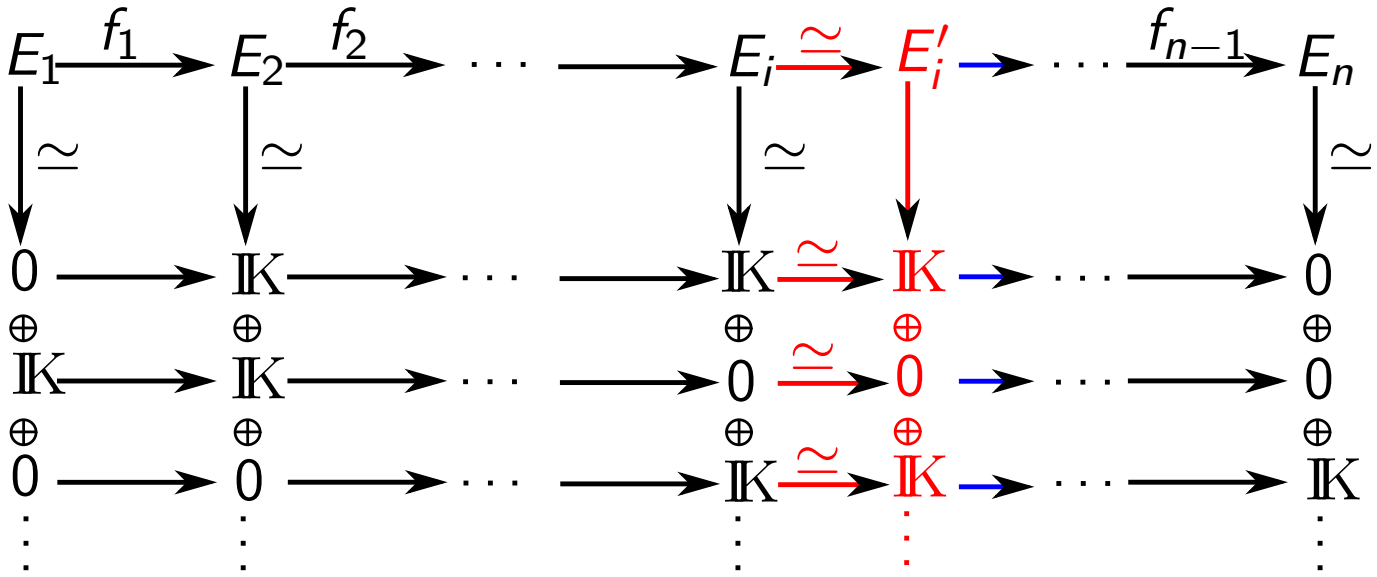
Classification des chaînes d'applications linéaires



Remarque

Ajouter un isomorphisme dans une chaîne ne change pas les intervalles de persistance (à ré-indexation près).

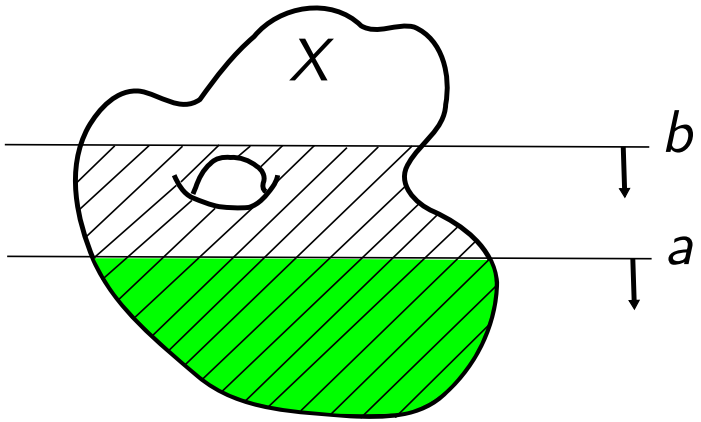
Classification des chaînes d'applications linéaires



Remarque

Ajouter un isomorphisme dans une chaîne ne change pas les intervalles de persistance (à ré-indexation près).

Fonctions et persistance

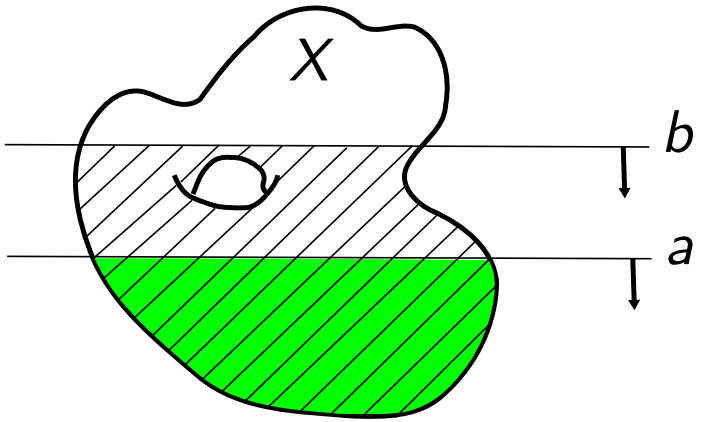


Soit $f : X \to \mathbb{R}$.

$a < b \implies \{f \leq a\} \subset \{f \leq b\}$

Fonction $f \longrightarrow$ filtration $\xrightarrow{H_p}$ chaîne d'applications linéaires $\longrightarrow$ intervalles de persistance $I(f)$

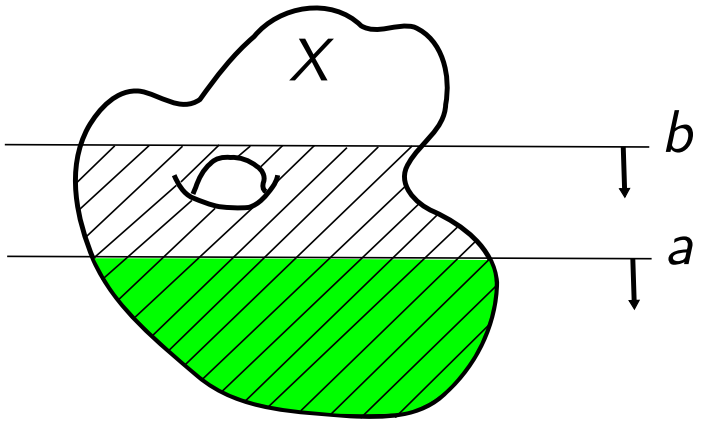
Fonctions et persistance



Soit $f : X \to \mathbb{R}$.
$a < b \implies \{f \leq a\} \subset \{f \leq b\}$

Fonction $f \longrightarrow$ filtration $\xrightarrow{H_p}$ chaîne d'applications linéaires $\longrightarrow$ intervalles de persistance $I(f)$

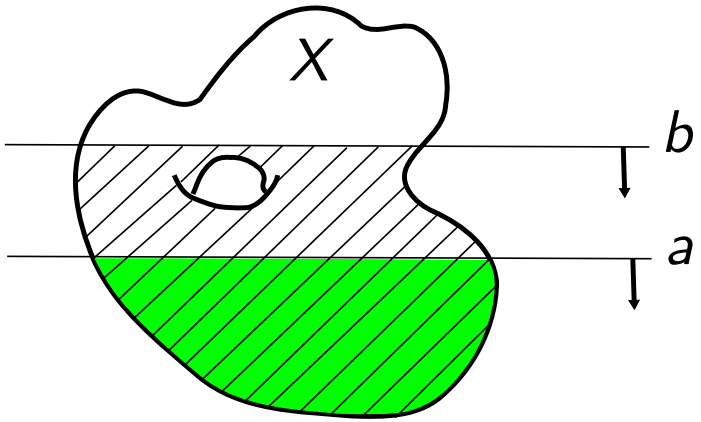
Fonctions et persistance



Soit $f : X \to \mathbb{R}$.
$a < b \implies \{f \leq a\} \subset \{f \leq b\}$

Fonction $f \longrightarrow$ filtration $\xrightarrow{H_p}$ chaîne d'applications linéaires
$\longrightarrow$ intervalles de persistance $I(f)$

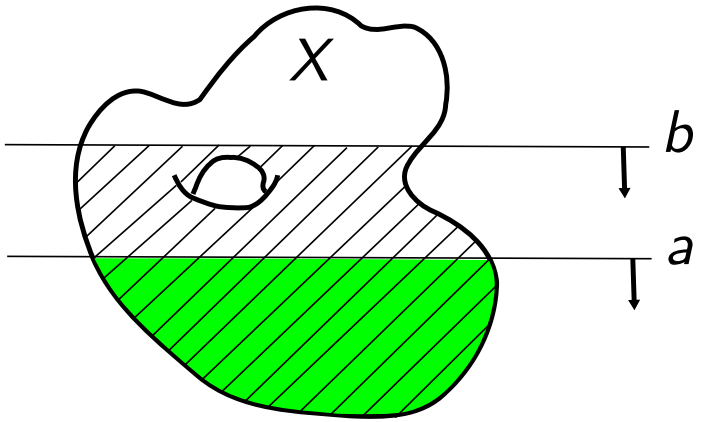
Fonctions et persistance



Soit $f : X \to \mathbb{R}$.
$a < b \implies \{f \leq a\} \subset \{f \leq b\}$

Fonction $f \longrightarrow$ filtration $\xrightarrow{H_p}$ chaîne d'applications linéaires
$\longrightarrow$ intervalles de persistance $I(f)$

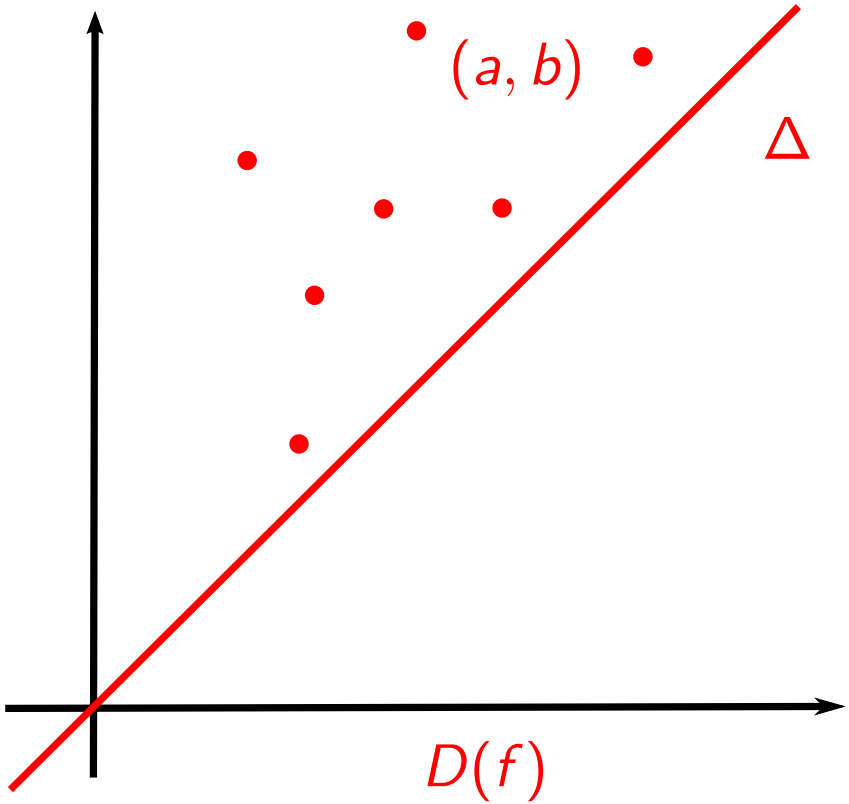
Fonctions et persistance



Soit $f : X \to \mathbb{R}$.
$a < b \implies \{f \leq a\} \subset \{f \leq b\}$

Fonction $f \longrightarrow$ filtration $\xrightarrow{H_p}$ chaîne d'applications linéaires $\longrightarrow$ intervalles de persistance $I(f)$

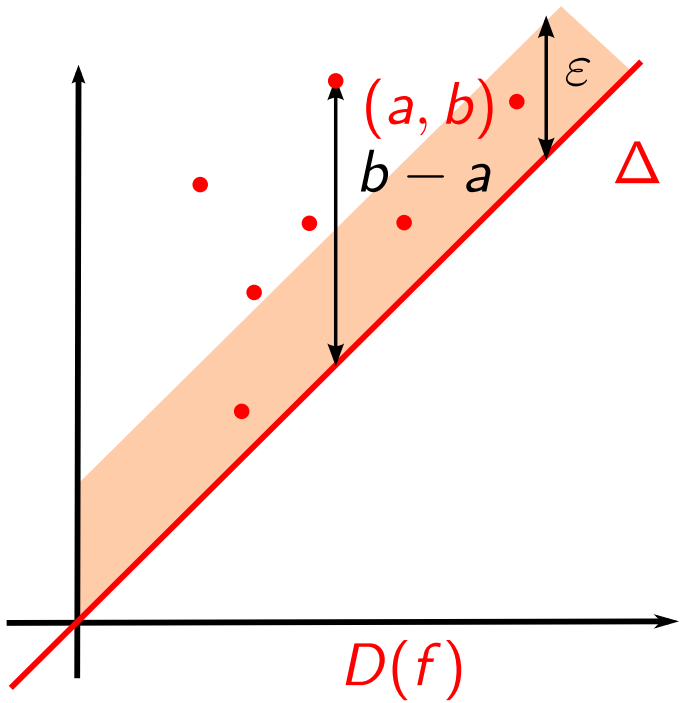
Diagramme de persistance



$(a, b) \in D(f) \Leftrightarrow [a, b[\in I(f)$ ou $a = b$.

[CEH07] stabilité : $|D(f) - D(g)| \leq \|f - g\|_\infty$

ε-simplification



Définition

g est une ε-simplification de f si $\|f - g\|_\infty \leq \varepsilon$ et $D(g) = D(f) \setminus \{$ les points à moins de ε de $\Delta\}$.

[EMP06] "Persistence-sensitive simplification..." cadre affine par morceaux

Cadre combinatoire

Données : Une filtration $\mathcal{K}$ d'un complexe simplicial K

$$\mathcal{K} : K_1 \subset K_2 \subset \ldots \subset K_n = K$$

avec $K_i = K_{i-1} \cup \{\sigma_i\}$.

Affirmation

On peut attribuer un signe à chaque simplexe de K tq tout intervalle de $\mathcal{K}$ est de la forme

- $[i, j[$, $j \leq n$ avec σ_i positif, σ_j négatif et $\dim \sigma_j = \dim \sigma_i + 1 = p + 1$,
- ou $[k, n+1[$ avec σ_k positif et $\dim \sigma_k = p$.

Définition

On dit alors que σ_i et σ_j sont appariés et que σ_k est essentiel ou non-apparié.

Cadre combinatoire

Données : Une filtration $\mathcal{K}$ d'un complexe simplicial K

$$\mathcal{K} : K_1 \subset K_2 \subset \ldots \subset K_n = K$$

avec $K_i = K_{i-1} \cup \{\sigma_i\}$.

Affirmation

On peut attribuer un signe à chaque simplexe de K tq tout intervalle de $\mathcal{K}$ est de la forme

- $[i, j[$, $j \leq n$ avec σ_i positif, σ_j négatif et $\dim \sigma_j = \dim \sigma_i + 1 = p + 1$,
- ou $[k, n+1[$ avec σ_k positif et $\dim \sigma_k = p$.

Définition

On dit alors que σ_i et σ_j sont appariés et que σ_k est essentiel ou non-apparié.

Calcul de la persistance en dimension 0

Pb: Calculer $I_0(\mathcal{K})$, i.e. les intervalles de persistance de

$$H_0(K_1) \to H_0(K_2) \to \ldots \to H_0(K_n)$$

Rappel : $H_0(K) \simeq \mathbb{K}^{\beta_0}$ où β_0 est le nb. de composantes de K.

Calcul de la persistance en dimension 0

On montre que

- Tout sommet est positif
- Si K est connexe, σ_1 est le seul sommet essentiel
- Une arête σ_j est négative ssi elle joint 2 composantes V et W de K_{j-1}. Dans ce cas σ_j est appariée avec le sommet σ_i qui réalise $\max\{\min V, \min W\}$.

Calcul de la persistance en dimension 0

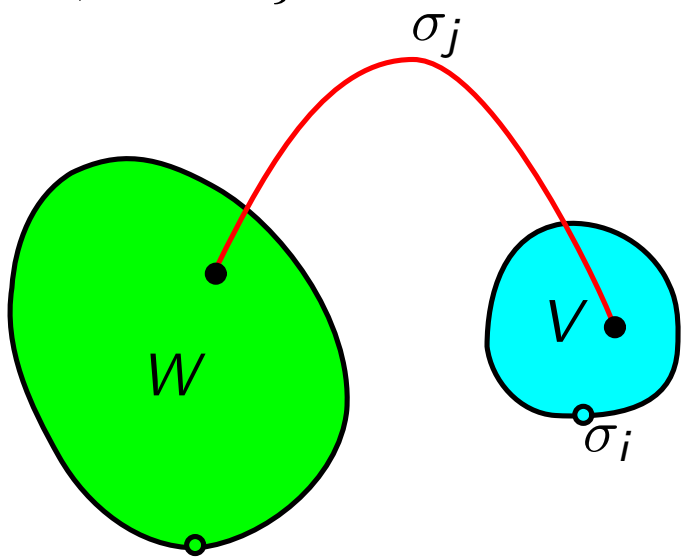
On montre que

- Tout sommet est positif
- Si K est connexe, σ_1 est le seul sommet essentiel
- Une arête σ_j est négative ssi elle joint 2 composantes V et W de K_{j-1}. Dans ce cas σ_j est appariée avec le sommet σ_i qui réalise $\max\{\min V, \min W\}$.

Calcul de la persistance en dimension 0

On montre que

- Tout sommet est positif
- Si K est connexe, σ_1 est le seul sommet essentiel
- Une arête σ_j est négative ssi elle joint 2 composantes V et W de K_{j-1}. Dans ce cas σ_j est appariée avec le sommet σ_i qui réalise $\max\{\min V, \min W\}$.



Calcul de la persistance en dimension 0

- test de négativité des arêtes = test insertion MST dans algo. de Kruskal

$\Longrightarrow$ Les arêtes négatives forment un MST dans K pour les poids égaux aux indices.

Remarque : Les paires de persistance de $H_0(\mathcal{K})$ ne dépendent que de ce MST.

Calcul de la persistance en dimension 0

- test de négativité des arêtes = test insertion MST dans algo. de Kruskal

$\Longrightarrow$ Les arêtes négatives forment un MST dans K pour les poids égaux aux indices.

Remarque : Les paires de persistance de $H_0(\mathcal{K})$ ne dépendent que de ce MST.

Calcul de la persistance en dimension 0

- test de négativité des arêtes = test insertion MST dans algo. de Kruskal

$\Longrightarrow$ Les arêtes négatives forment un MST dans K pour les poids égaux aux indices.

Remarque : Les paires de persistance de $H_0(\mathcal{K})$ ne dépendent que de ce MST.

ε-simplification en dimension 0

Soit $f : K \to \mathbb{R}$.
$f(\sigma_1) < f(\sigma_2) \ldots < f(\sigma_n) \to \mathcal{K}_f \to T_f$.

Lemme

On peut calculer une ε-simplification h de la restriction de f à T_f tq
Si $g : K \to \mathbb{R}$ avec $g|_{T_f} = h$ et $g|_{K \setminus T_f} = f$, alors

- $D_0(g) = D_0(f) \setminus \{$ points de persistance $\leq \varepsilon\}$
- $D_1(g) = D_1(f)$.

Principe de la simplification

Soit K une surface triangulée et $f : K \longrightarrow \mathbb{R}$.

- On calcule une ε-simplification de $D_0(f)$ par le lemme précédent.
- K est une surface $\Longrightarrow$ K admet une surface duale K^*.
- par dualité (cf. [CEH08]) (σ, τ) est une paire pour f ssi (τ^*, σ^*) est une paire pour $-f$
- Pour simplifier $D_1(f)$, il suffit de simplifier $D_0(-f)$ sur K^*.

Principe de la simplification

Soit K une surface triangulée et $f : K \longrightarrow \mathbb{R}$.

- On calcule une ε-simplification de $D_0(f)$ par le lemme précédent.
- K est une surface $\Longrightarrow$ K admet une surface duale K^*.
- par dualité (cf. [CEH08]) (σ, τ) est une paire pour f ssi (τ^*, σ^*) est une paire pour $-f$
- Pour simplifier $D_1(f)$, il suffit de simplifier $D_0(-f)$ sur K^*.

Principe de la simplification

Soit K une surface triangulée et $f : K \longrightarrow \mathbb{R}$.

- On calcule une ε-simplification de $D_0(f)$ par le lemme précédent.
- K est une surface $\Longrightarrow$ K admet une surface duale K^*.
- par dualité (cf. [CEH08]) (σ, τ) est une paire pour f ssi (τ^*, σ^*) est une paire pour $-f$
- Pour simplifier $D_1(f)$, il suffit de simplifier $D_0(-f)$ sur K^*.

Principe de la simplification

Soit K une surface triangulée et $f : K \longrightarrow \mathbb{R}$.

- On calcule une ε-simplification de $D_0(f)$ par le lemme précédent.
- K est une surface $\Longrightarrow$ K admet une surface duale K^*.
- par dualité (cf. [CEH08]) (σ, τ) est une paire pour f ssi (τ^*, σ^*) est une paire pour $-f$
- Pour simplifier $D_1(f)$, il suffit de simplifier $D_0(-f)$ sur K^*.

Principe de la simplification

Soit K une surface triangulée et $f : K \longrightarrow \mathbb{R}$.

- On calcule une ε-simplification de $D_0(f)$ par le lemme précédent.
- K est une surface $\implies$ K admet une surface duale K^*.
- par dualité (cf. [CEH08]) (σ, τ) est une paire pour f ssi (τ^*, σ^*) est une paire pour $-f$
- Pour simplifier $D_1(f)$, il suffit de simplifier $D_0(-f)$ sur K^*.

Principe de la simplification

Il suffit de savoir calculer une ε-simplification d'une fonction définie sur un arbre.

Structure des paires à simplifier

K est un arbre.
$f(\sigma_1) < f(\sigma_2) \ldots < f(\sigma_n) \rightarrow \mathcal{K}_f$

On se donne $\mathcal{P} \subset I(\mathcal{K}_f)$ fermé par inclusion (e.g. les paires de persistance $< \varepsilon$).

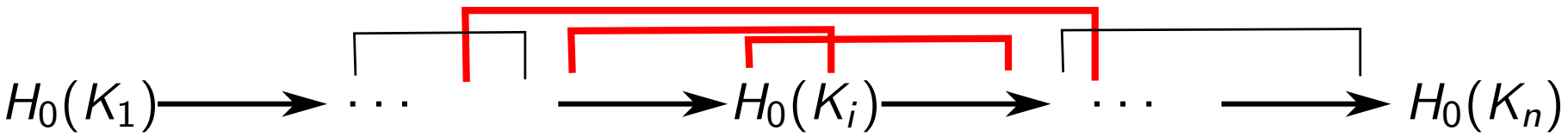
Définition

g est une $\mathcal{P}$-simplification de f si $D(g) = D(f) \setminus f(\mathcal{P})$.

Structure des paires à simplifier

K est un arbre.
$f(\sigma_1) < f(\sigma_2) \ldots < f(\sigma_n) \rightarrow \mathcal{K}_f$

On se donne $\mathcal{P} \subset I(\mathcal{K}_f)$ fermé par inclusion (e.g. les paires de persistance $< \varepsilon$).



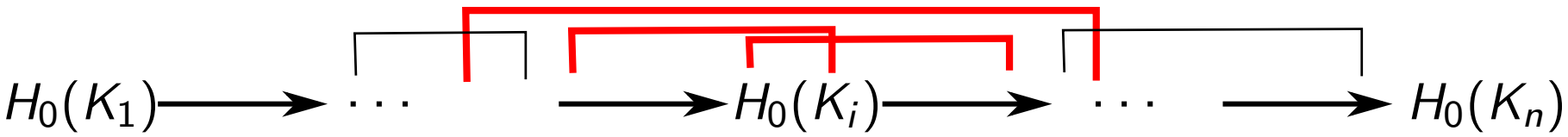
Définition

g est une $\mathcal{P}$-simplification de f si $D(g) = D(f) \setminus f(\mathcal{P})$.

Structure des paires à simplifier

K est un arbre.
$f(\sigma_1) < f(\sigma_2) \ldots < f(\sigma_n) \rightarrow \mathcal{K}_f$

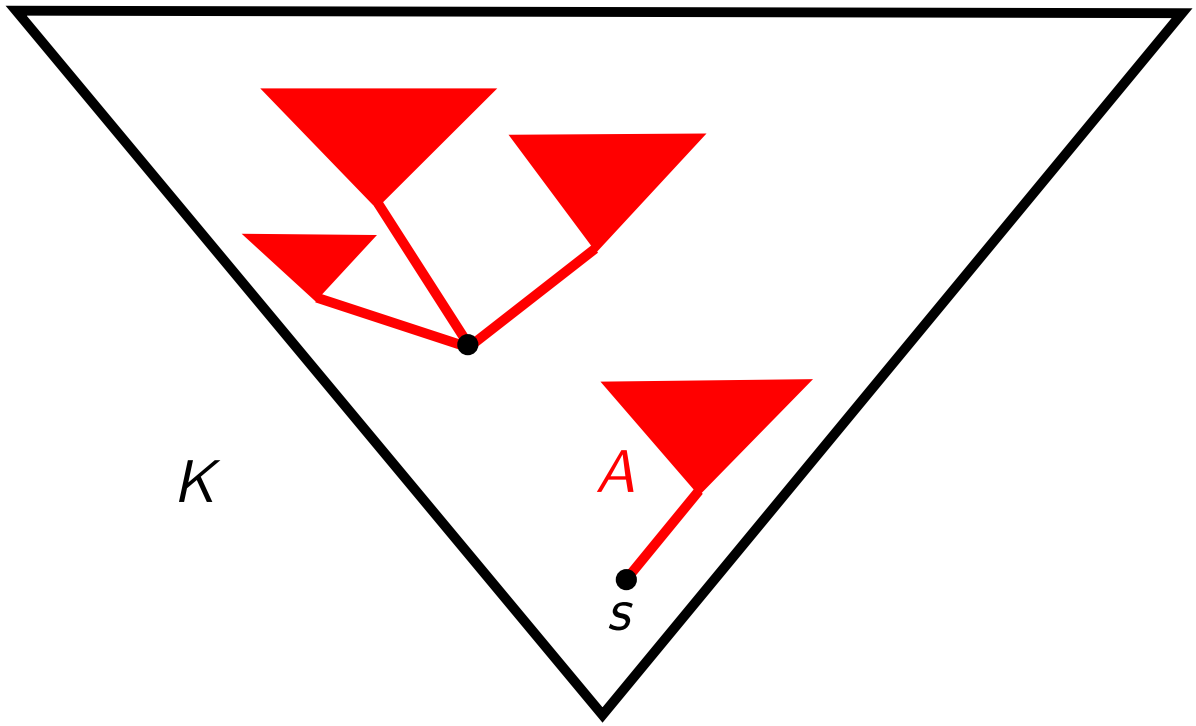
On se donne $\mathcal{P} \subset I(\mathcal{K}_f)$ fermé par inclusion (e.g. les paires de persistance $< \varepsilon$).



Définition

g est une $\mathcal{P}$-simplification de f si $D(g) = D(f) \setminus f(\mathcal{P})$.

Structure des paires à simplifier



Lemme

Chaque composante de $\mathcal{P}$ est un arbre A privé de sa racine s tel que $f(s) < f(A)$.

Simplification d'une composante de $\mathcal{P}$

On peut simplifier chaque composante A de $\mathcal{P}$ indépendamment.

- Le nombre de paires de $H_0(\mathcal{K}_f)$ ne dépend que de K.
- Les paires de A "à supprimer" deviennent des paires (σ_i, σ_j) de persistance nulle : $g(\sigma_i) = g(\sigma_j)$
  ($\Longrightarrow$ g ne peut être injective).
- Si on impose qu'au plus deux simplexes ont la même valeur, alors les paires nulles sont de la forme (σ_i, σ_{i+1}).

Définition

Une paire (σ_i, σ_{i+1}) est dite locale. Nécessairement $\sigma_i \prec \sigma_{i+1}$.

Simplification d'une composante de $\mathcal{P}$

On peut simplifier chaque composante A de $\mathcal{P}$ indépendamment.

- Le nombre de paires de $H_0(\mathcal{K}_f)$ ne dépend que de K.
- Les paires de A "à supprimer" deviennent des paires (σ_i, σ_j) de persistance nulle : $g(\sigma_i) = g(\sigma_j)$ ($\Longrightarrow$ g ne peut être injective).
- Si on impose qu'au plus deux simplexes ont la même valeur, alors les paires nulles sont de la forme (σ_i, σ_{i+1}).

Définition

Une paire (σ_i, σ_{i+1}) est dite locale. Nécessairement $\sigma_i \prec \sigma_{i+1}$.

Simplification d'une composante de $\mathcal{P}$

On peut simplifier chaque composante A de $\mathcal{P}$ indépendamment.

- Le nombre de paires de $H_0(\mathcal{K}_f)$ ne dépend que de K.
- Les paires de A "à supprimer" deviennent des paires (σ_i, σ_j) de persistance nulle : $g(\sigma_i) = g(\sigma_j)$
  ($\implies$ g ne peut être injective).
- Si on impose qu'au plus deux simplexes ont la même valeur, alors les paires nulles sont de la forme (σ_i, σ_{i+1}).

Définition

Une paire (σ_i, σ_{i+1}) est dite locale. Nécessairement $\sigma_i \prec \sigma_{i+1}$.

Simplification d'une composante de $\mathcal{P}$

On peut simplifier chaque composante A de $\mathcal{P}$ indépendamment.

- Le nombre de paires de $H_0(\mathcal{K}_f)$ ne dépend que de K.
- Les paires de A "à supprimer" deviennent des paires (σ_i, σ_j) de persistance nulle : $g(\sigma_i) = g(\sigma_j)$
  ($\implies$ g ne peut être injective).
- Si on impose qu'au plus deux simplexes ont la même valeur, alors les paires nulles sont de la forme (σ_i, σ_{i+1}).

Définition

Une paire (σ_i, σ_{i+1}) est dite locale. Nécessairement $\sigma_i \prec \sigma_{i+1}$.

Simplification d'une composante de $\mathcal{P}$

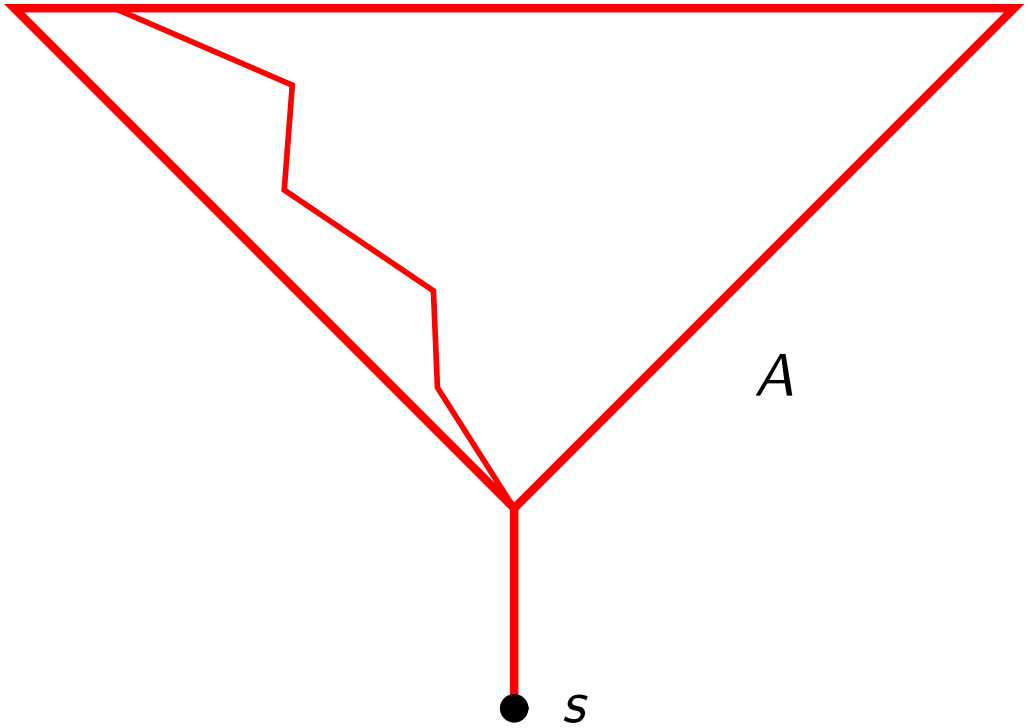
On peut simplifier chaque composante A de $\mathcal{P}$ indépendamment.

- Le nombre de paires de $H_0(\mathcal{K}_f)$ ne dépend que de K.
- Les paires de A "à supprimer" deviennent des paires (σ_i, σ_j) de persistance nulle : $g(\sigma_i) = g(\sigma_j)$
  ($\implies$ g ne peut être injective).
- Si on impose qu'au plus deux simplexes ont la même valeur, alors les paires nulles sont de la forme (σ_i, σ_{i+1}).

Définition

Une paire (σ_i, σ_{i+1}) est dite locale. Nécessairement $\sigma_i \prec \sigma_{i+1}$.

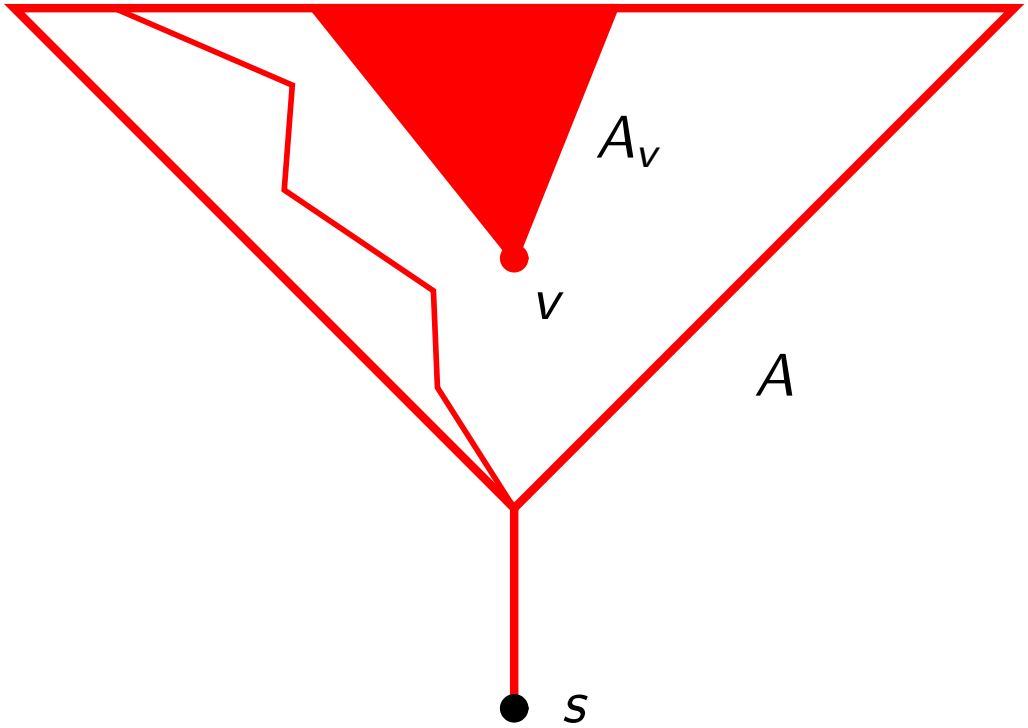
Simplification d'une composante de $\mathcal{P}$



Si toutes les paires sont locales dans la composante A alors f est croissante sur tout chemin issu de s.

Pour obtenir cette condition, on pose $g(v) = \min f(A_v)$ (et $g(e) = g(v)$ si e appariée à v). Le calcul se fait en temps linéaire pour A en l'effeuillant.

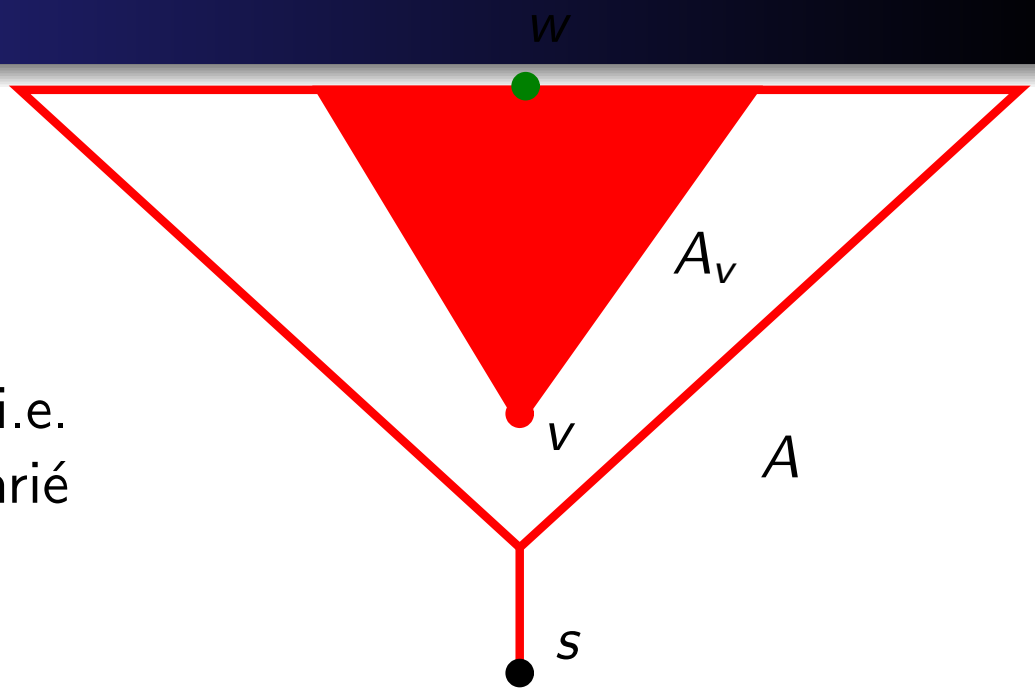
Simplification d'une composante de $\mathcal{P}$



Si toutes les paires sont locales dans la composante A alors f est croissante sur tout chemin issu de s.

Pour obtenir cette condition, on pose $g(v) = \min f(A_v)$ (et $g(e) = g(v)$ si e appariée à v). Le calcul se fait en temps linéaire pour A en l'effeuillant.

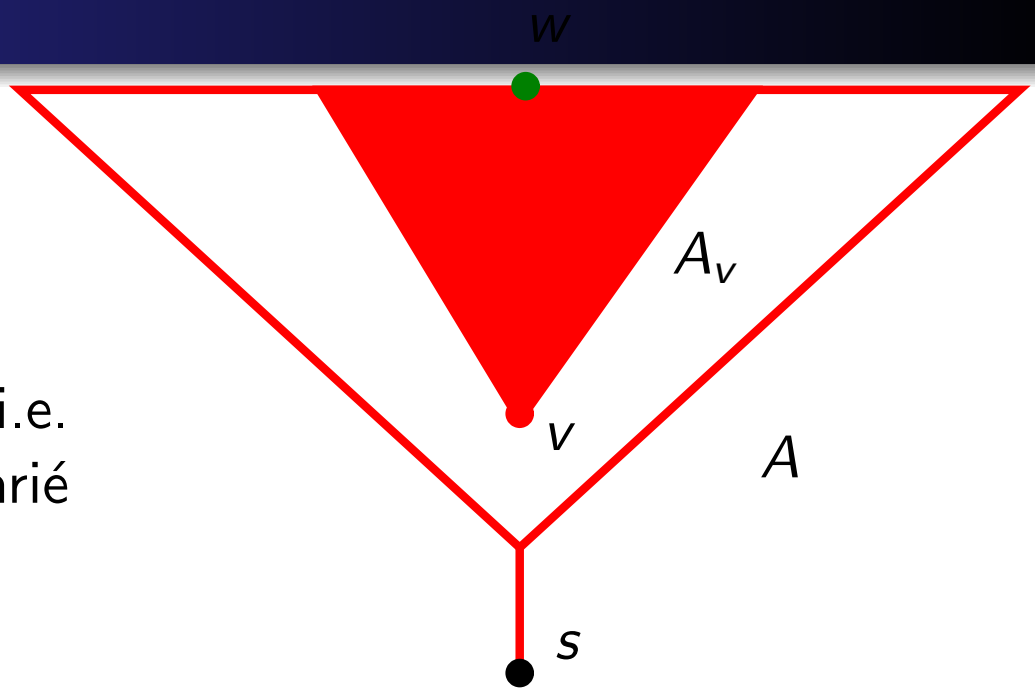
Proximité de f et g



Soient $w := \arg\min_f A_v$, i.e. $g(v) = f(w)$ et $e(w)$ apparié à w (dans A)

- Soit $\overline{[w, e(w)]} := \{$ paires de f incluse dans $[w, e(w)] \}$
- $G_w :=$ composante de w dans $\overline{[w, e(w)]}$. C'est un arbre privé d'un sommet u.
- $G_w \subset A$ car $G_w \subset \overline{[w, e(w)]} \subset \mathcal{P}$ et G_w connexe.
- Donc $v \in G_w$.
- Donc $f(w) < f(v) < f(e(w))$ et $|g(v) - f(v)| = |f(w) - f(v)| < |f(w) - f(e(w))| < \varepsilon$

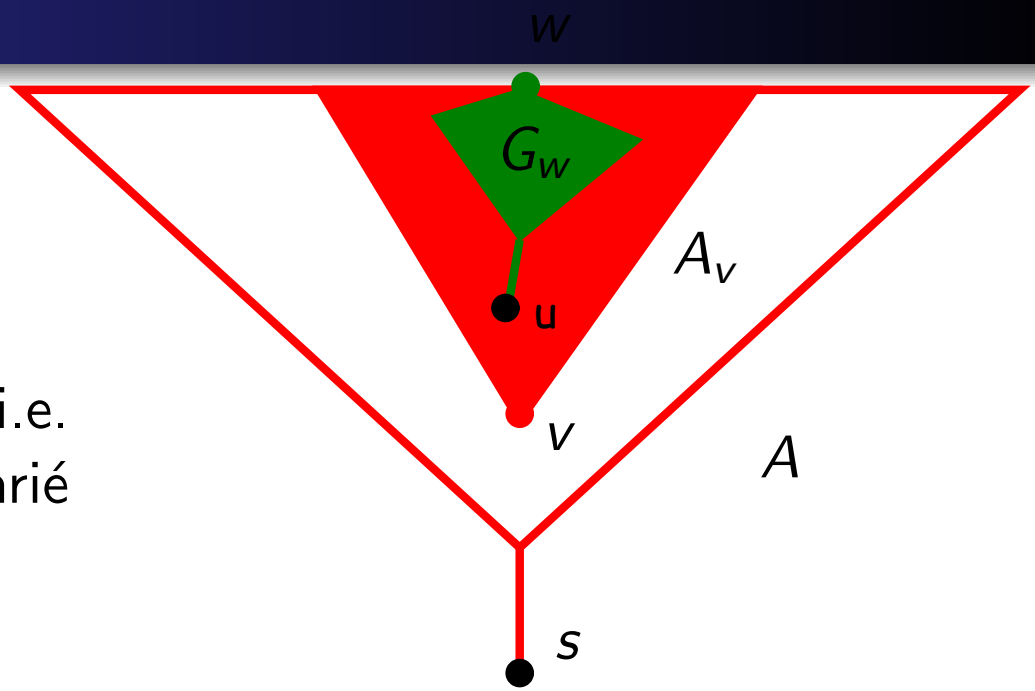
Proximité de f et g



Soient $w := \arg\min_f A_v$, i.e. $g(v) = f(w)$ et $e(w)$ apparié à w (dans A)

- Soit $\overline{[w, e(w)]} := \{$ paires de f incluse dans $[w, e(w)] \}$
- $G_w :=$ composante de w dans $\overline{[w, e(w)]}$. C'est un arbre privé d'un sommet u.
- $G_w \subset A$ car $G_w \subset \overline{[w, e(w)]} \subset \mathcal{P}$ et G_w connexe.
- Donc $v \in G_w$.
- Donc $f(w) < f(v) < f(e(w))$ et $|g(v) - f(v)| = |f(w) - f(v)| < |f(w) - f(e(w))| < \varepsilon$

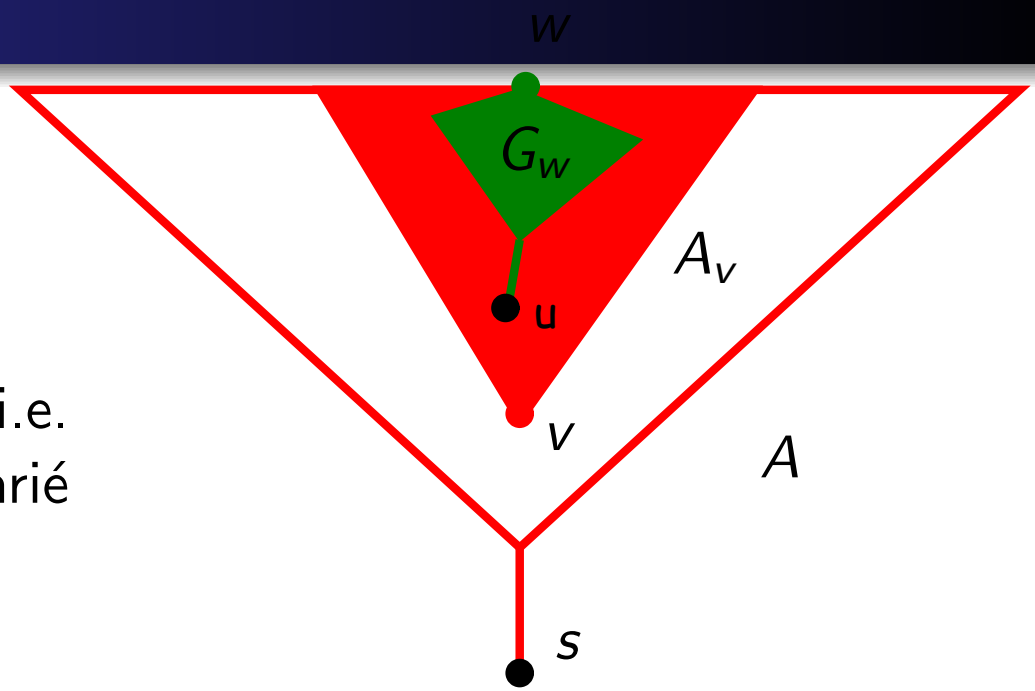
Proximité de f et g



Soient $w := \arg\min_f A_v$, i.e. $g(v) = f(w)$ et $e(w)$ apparié à w (dans A)

- Soit $\overline{[w, e(w)]} := \{$ paires de f incluse dans $[w, e(w)]$ $\}$
- $G_w :=$ composante de w dans $\overline{[w, e(w)]}$. C'est un arbre privé d'un sommet u.
- $G_w \subset A$ car $G_w \subset \overline{[w, e(w)]} \subset \mathcal{P}$ et G_w connexe.
- Donc $v \in G_w$.
- Donc $f(w) < f(v) < f(e(w))$ et $|g(v) - f(v)| = |f(w) - f(v)| < |f(w) - f(e(w))| < \varepsilon$

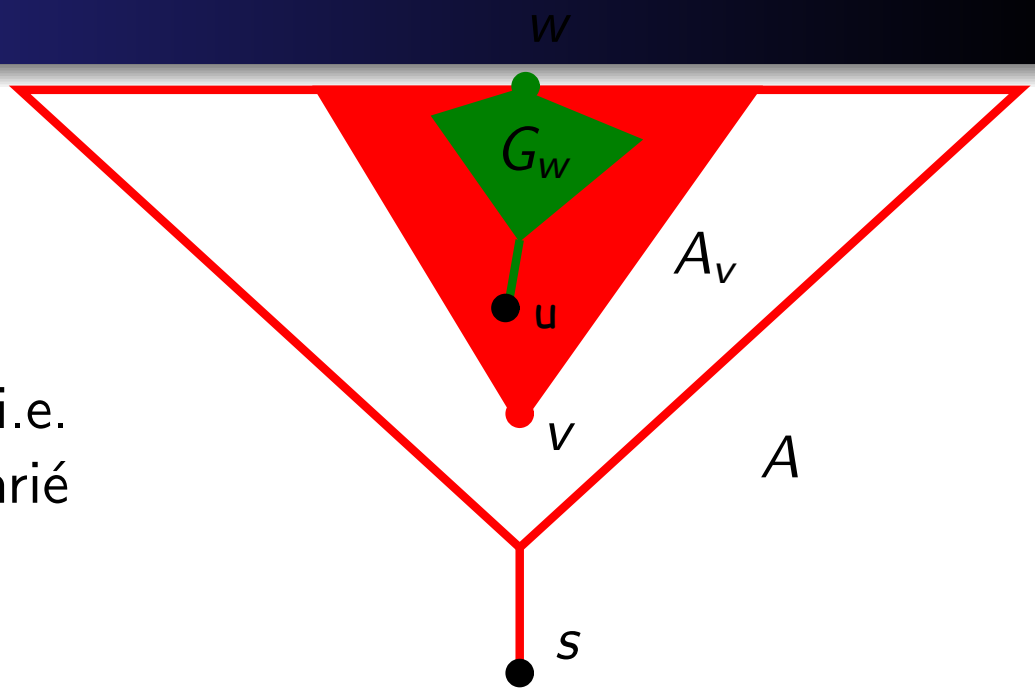
Proximité de f et g



Soient $w := \arg\min_f A_v$, i.e. $g(v) = f(w)$ et $e(w)$ apparié à w (dans A)

- Soit $\overline{[w, e(w)]} := \{$ paires de f incluse dans $[w, e(w)]$ $\}$
- $G_w :=$ composante de w dans $\overline{[w, e(w)]}$. C'est un arbre privé d'un sommet u.
- $G_w \subset A$ car $G_w \subset \overline{[w, e(w)]} \subset \mathcal{P}$ et G_w connexe.
- Donc $v \in G_w$.
- Donc $f(w) < f(v) < f(e(w))$ et $|g(v) - f(v)| = |f(w) - f(v)| < |f(w) - f(e(w))| < \varepsilon$

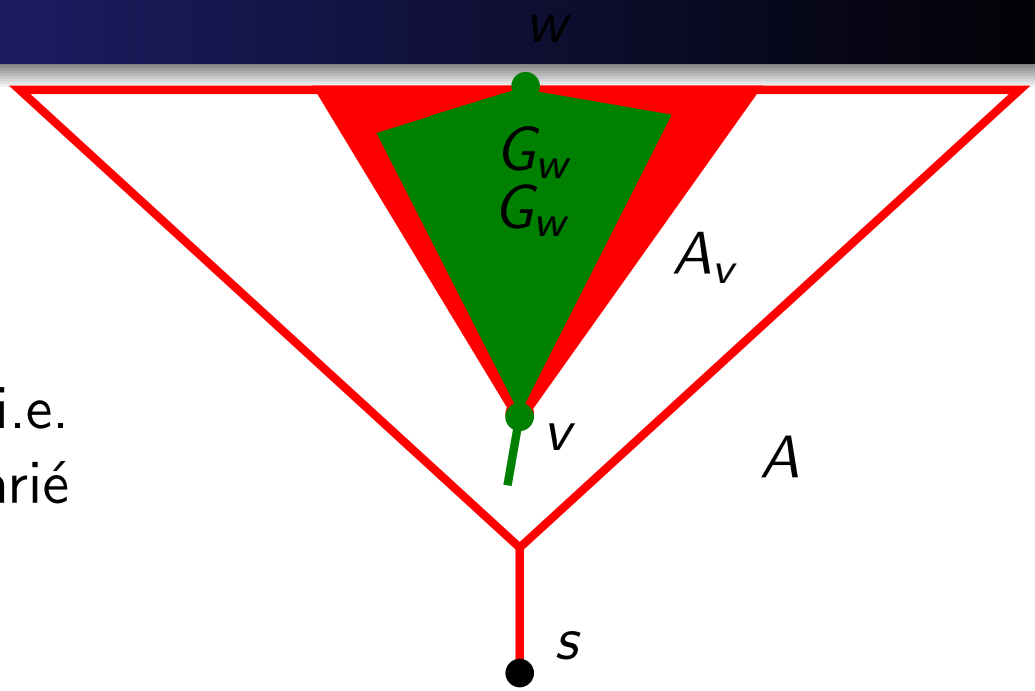
Proximité de f et g



Soient $w := \arg\min_f A_v$, i.e. $g(v) = f(w)$ et $e(w)$ apparié à w (dans A)

- Soit $\overline{[w, e(w)]} := \{$ paires de f incluse dans $[w, e(w)]$ $\}$
- $G_w :=$ composante de w dans $\overline{[w, e(w)]}$. C'est un arbre privé d'un sommet u.
- $G_w \subset A$ car $G_w \subset \overline{[w, e(w)]} \subset \mathcal{P}$ et G_w connexe.
- Si $G_w \subset A_v$ alors $u \in A_v$. Contradiction avec $f(u) < f(w)$.
- Donc $v \in G_w$.
- Donc $f(w) < f(v) < f(e(w))$ et $|g(v) - f(v)| = |f(w) - f(v)| < |f(w) - f(e(w))| < \varepsilon$

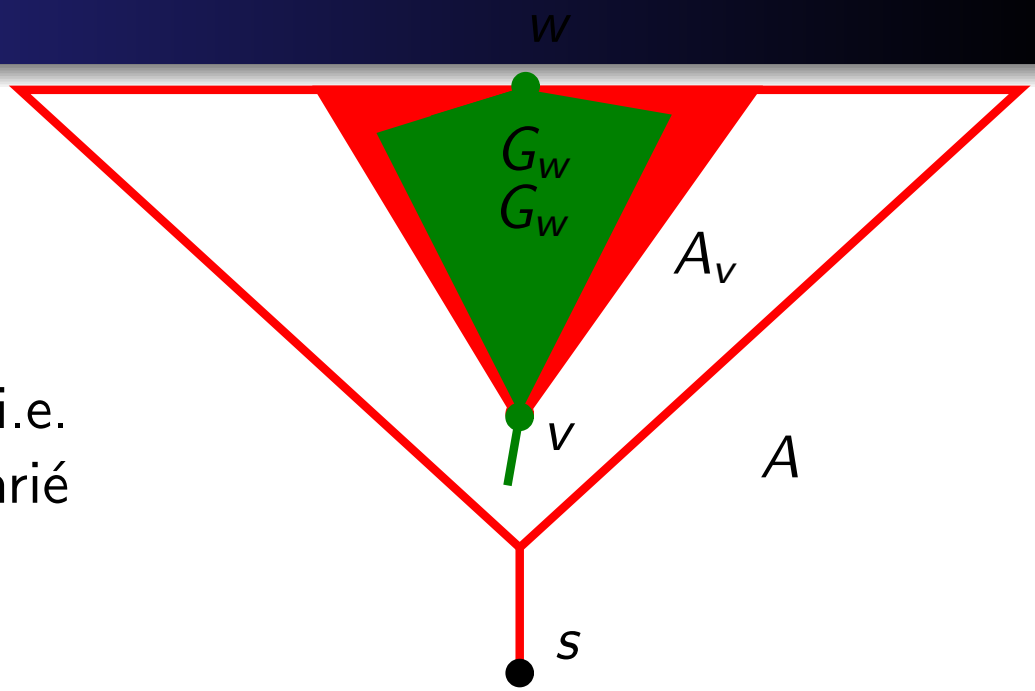
Proximité de f et g



Soient $w := \arg\min_f A_v$, i.e. $g(v) = f(w)$ et $e(w)$ apparié à w (dans A)

- Soit $\overline{[w, e(w)]} := \{$ paires de f incluse dans $[w, e(w)]$ $\}$
- $G_w :=$ composante de w dans $\overline{[w, e(w)]}$. C'est un arbre privé d'un sommet u.
- $G_w \subset A$ car $G_w \subset \overline{[w, e(w)]} \subset \mathcal{P}$ et G_w connexe.
- Donc $v \in G_w$.
- Donc $f(w) < f(v) < f(e(w))$ et $|g(v) - f(v)| = |f(w) - f(v)| < |f(w) - f(e(w))| < \varepsilon$

Proximité de f et g



Soient $w := \arg\min_f A_v$, i.e. $g(v) = f(w)$ et $e(w)$ apparié à w (dans A)

- Soit $\overline{[w, e(w)]} := \{$ paires de f incluse dans $[w, e(w)]$ $\}$
- $G_w :=$ composante de w dans $\overline{[w, e(w)]}$. C'est un arbre privé d'un sommet u.
- $G_w \subset A$ car $G_w \subset \overline{[w, e(w)]} \subset \mathcal{P}$ et G_w connexe.
- Donc $v \in G_w$.
- Donc $f(w) < f(v) < f(e(w))$ et $|g(v) - f(v)| = |f(w) - f(v)| < |f(w) - f(e(w))| < \varepsilon$

Complexité

- Tri des simplexes selon $f \longrightarrow O(n \log n)$
- Calcul du MST et des paires de persistance $\longrightarrow O(n\alpha(n))$
- Calcul de $g \longrightarrow O(n)$

Complexité de la simplification : $O(n \log n)$ dans le model Real-RAM usuel ou $O(n)$ dans un modèle RAM approprié.

Complexité

- Tri des simplexes selon $f \longrightarrow O(n \log n)$
- Calcul du MST et des paires de persistance $\longrightarrow O(n\alpha(n))$
- Calcul de $g \longrightarrow O(n)$

Complexité de la simplification : $O(n \log n)$ dans le model Real-RAM usuel ou $O(n)$ dans un modèle RAM approprié.

Complexité

- Tri des simplexes selon $f \longrightarrow O(n \log n)$
- Calcul du MST et des paires de persistance $\longrightarrow O(n\alpha(n))$
- Calcul de $g \longrightarrow O(n)$

Complexité de la simplification : $O(n \log n)$ dans le model Real-RAM usuel ou $O(n)$ dans un modèle RAM approprié.

Complexité

- Tri des simplexes selon $f \longrightarrow O(n \log n)$
- Calcul du MST et des paires de persistance $\longrightarrow O(n\alpha(n))$
- Calcul de $g \longrightarrow O(n)$

Complexité de la simplification : $O(n \log n)$ dans le model Real-RAM usuel ou $O(n)$ dans un modèle RAM approprié.

FIN